



**Budapest University of Technology and Economics**  
Faculty of Transportation and Vehicle Engineering  
Department of Automotive Technologies



# Master thesis

## Reinforcement Learning Based Drift Control

**Author:**  
Ahmet Tikna

**Supervisor:**  
Remeli Viktor

Budapest, 2021.

## SPECIFICATIONS OF MASTER'S THESIS

**Name:** Ahmet Tikna (EMFL5E)  
 Engineer MSc (Full-time)

**Form:** Autonomous Vehicle Control

**Course code:** BMEKOGGM554

**Title:** Reinforcement Learning based drift control

**No.:** GJT-M-G-2021-22

**Accessibility:** public/classified

**Tasks:**

- Literature review: study and summarize different RL paradigms.
- Literature review: study and summarize existing approaches to RL based drift control.
- Literature review: compare and contrast the RL approach with traditional controller-based approaches.
- Re-implement cutting-edge papers' reported results in simulation.
- Develop a method of translating the simulation method to a real-world vehicle.
- Publish the achieved results in a joint paper.

**Supervisor at the Department:**

Viktor Remeli

**Host institution:**

Budapest University of Technology and  
 Economics


**Industrial/External Supervisor:**

Name of Industrial / External Supervisor

**Subjects of Final Examination:**

1. Vehicle Operation - KOGGM174, 4, Basic
2. Automated Driving System – KOGGM707, 5, Differentiated
3. Vehicle Dynamics, Vehicle Mechanics Fundamentals - BMEKOGGM705, BMEKOGGM713, 7, Basic / Differentiated

**Deadline date of Thesis submission:** 14th May 2021.



P.H.  
 Zolt Szalay PhD.  
 associate professor  
 head of department

## Declaration

I, Ahmet Tikna, hereby declare that the present master's thesis was composed by myself and that the work contained here in is my own. I also confirm that I have only used the specified resources. Any part that I have taken from another source, either verbatim or in the same content, but reworded, has been clearly marked with the source indicated.



Ahmet Tikna

## Abstract

In this thesis, deep reinforcement learning algorithms have been applied to autonomous drift control task. The main intention of this research is to develop a controller that performs drifting maneuver around a randomly selected goal point. Some of the key problems which I have dealt with during the development phase are: (i) sample efficiency, (ii) generalization of learned policies.

In the first phase of this research, I built up a simulation environment involving a kinematic bicycle vehicle model which is consistent with the physical characteristic of the model car BMW M2 Competition. This environment is used to generate dataset for supervised learning and then to conduct the training process of deep reinforcement learning algorithms. In this environment, instead of earth-fixed coordinate system, I used body-fixed coordinate system and hence the agent could efficiently learn the generalized (near) optimal policies.

Since reinforcement learning might require many samples to learn useful policies due to the complexity of the high-dimensional environment, I developed transfer learning framework to refine useful policies and to transfer gained knowledge from traditional controller and supervised learning to the reinforcement learning side. Thus, it can be achievable to increase the scalability of the reinforcement learning-based agents with fewer samples and to achieve (near) optimal policies by bringing the advantages of those algorithms and methods together.

## Table of Contents

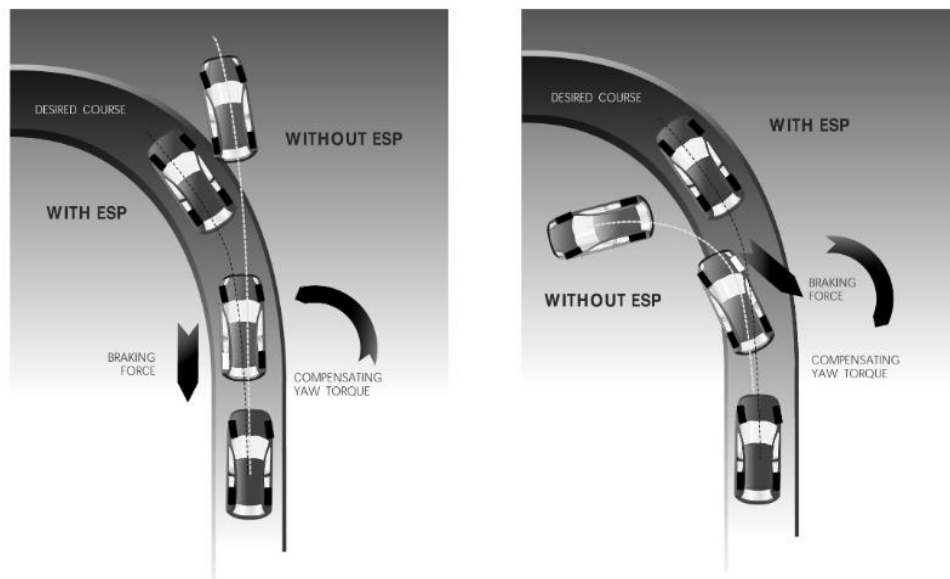
1. Introduction .....	7
1.1. Motivation .....	9
1.2. Related Work .....	10
1.2.1. Traditional Control Approaches .....	10
1.2.2. Learning-Based Control Approaches .....	10
1.3. Objective .....	11
2. Reinforcement Learning .....	12
2.1. Markov Decision Process .....	12
2.2. Trade-Off: Exploration vs Exploitation .....	13
2.3. Value-Based Reinforcement Learning Approach .....	13
2.3.1. Q-Learning .....	14
2.3.2. SARSA .....	15
2.4. Actor-Critic Methods .....	15
2.4.1. Asynchronous Advantage Actor-Critic (A3A) Algorithm .....	16
2.4.2. Soft Actor-Critic (SAC) Algorithm .....	17
2.5. Model-Based Reinforcement Learning Approach .....	18
2.5.1. PiLCO (Probabilistic Inference for Learning Control) Algorithm .....	18
3. Simulation Environment .....	19
3.1. The Bicycle Model .....	19
3.2. Brush Tire Model .....	21
3.2.1. Road Friction Coefficient .....	24
3.3. Model Vehicle .....	24
3.3.1. Vehicle Dimensions .....	25
3.3.2. Center of Gravity .....	25
3.4. Implementation of the Bicycle Model in SIMULINK .....	26
3.4.1. Tire Model .....	28
3.4.2. State Blocks .....	29
3.4.3. Additional Blocks .....	31
3.5. RL Setup for Autonomous Drift Control .....	32
3.5.1. Turning Radius .....	33
3.5.2. The Position of the Vehicle's Turning Center .....	34
3.5.3. Coordinate Transformation .....	35
3.5.4. Encoding Heading Angle For Learning .....	35
4. Transfer Learning Framework .....	38
4.1. MIMO LQR Controller For Stabilizing Vehicle Drifting .....	39
4.2. Autonomous Drifting with Deep Learning .....	44
4.3. Deep Reinforcement Learning .....	48
4.3.1. State Space .....	50
4.3.2. Action Space .....	51
4.3.3. Reward Function .....	51
5. Experiments and Results .....	53
5.1. Actions To Be Learn .....	53
5.2. Implementation .....	54
5.3. Results .....	55
6. Conclusion .....	57

7. References.....	58
8. Table of Figures .....	60
9. List of tables .....	62
10. List of abbreviations and symbols .....	63

## 1. INTRODUCTION

Over the past decades, thousands of serious and fatal accidents have occurred all around the world due to the increase in the number of vehicles on the roads, lack of available road spaces, environmental conditions, and driver errors. According to the 2019 crash data statistics of the National Highway Transportation Safety Administration (NHTSA), there were 36,096 fatalities resulting from road accidents in the United States [1]. One approach for decreasing the number of accidents and enhancing road safety is to equip the vehicles with stability control systems and state-of-the-art collision detection sensors, such as cameras and radars.

The installation of electronic stability control (ESC) system in truck tractors and large buses, manufactured in or after 2012, was established as a new Federal Motor Vehicle Safety Standard by the National Highway Transportation Safety Administration (NHTSA) [2]. The ESC system, also known as the electronic stability program (ESP), was designed to improve driving dynamics and reduce severe and fatal traffic accidents by mitigating the risk of oversteering and understeering situations. The accidents related to the ESC system happen due to the high speed in cornering, slippery surfaces, crash avoidance maneuver, or a combination of them [3].



*Figure 1: Action of ESC understeer and oversteer conditions*  
 [3]

As a core component of the ESC system, the yaw sensor measures the change in the direction of the vehicle's longitudinal axes. When considering the role of ESC in vehicle control, it uses automatic braking of individual wheels and adjusts the engine output to stabilize the vehicle's heading if it detects a significant deviation between the driver's decisions and the dynamic responses of the car.

Indeed, various vehicle control maneuvers, as opposed to the basis of the ESC system, are utilized to exploit the full potential of vehicle dynamics by professional racing drivers. For example, instead of normal cornering, proficient racing drivers execute drift maneuver to take sharp corners quickly by intentionally inducing the saturation of rear tires by using throttle or brakes [4].

Drifting is a cornering technique that dealt with the high side-slip angle  $\beta$ , which results from a large discrepancy between the direction of the vehicle's longitudinal axes and the direction of its velocity vector at the center of gravity, for stabilizing and maintaining the vehicle's control. As depicted in Figure 2, the side-slip angle  $\beta$ , which is defined in equation (1.1), can be computed as the arctangent of the ratio of the vehicle's lateral velocity  $V_y$  at the center of gravity to its longitudinal velocity  $V_x$  at the center of gravity [5].

$$\beta = \arctan \left( \frac{V_y}{V_x} \right) \quad (1.1)$$

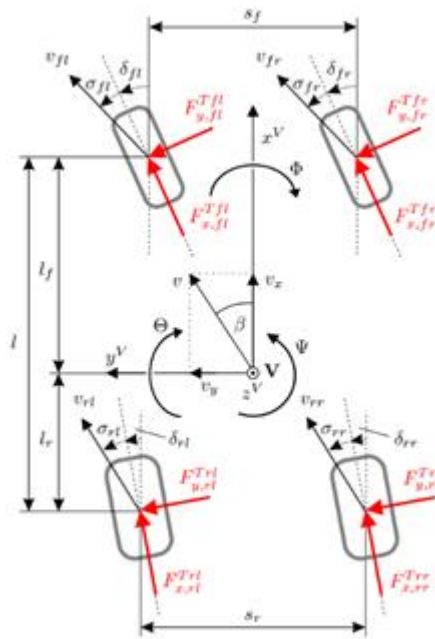


Figure 2: Four-Wheel-Drive (4WD) Model



## 1.1. MOTIVATION

Traditional control approaches are mostly deterministic and require writing down explicitly equations of systems. Since the traditional control approaches use extensive system knowledge but are fixed and rigid for all time, the design of classical controllers can not be easy either since it necessitates a deep understanding of the control tasks expressed in equations or because of the systems' nonlinearity. Traditional control techniques massively exploit prior system knowledge to achieve high-performance and stability. Unlike the traditional control methods, the framework of reinforcement learning provides robust algorithms, where an agent explicitly selects actions and learns policies through trial and error under uncertainty [6]. Rather than writing down explicit equations of tasks, a reward function measuring the performance of the agent step by step is defined by the designer of the controller [6].

Reinforcement Learning has been successfully used to enhance the performance of complex and safety-critical sequential decision-making tasks in robotics, often achieving robust performance. This has led to the launch of several promising projects where a human or a traditional controller can be replaced by a reinforcement learning-based or reinforcement learning-assisted controller. However, in robotics, reinforcement learning might require many samples to learn useful policies due to the complexity of the high-dimensional environment. Traditional control methods can be used to refine initial policies to decrease the number of required samples from the real-world and simulation environments.

A vehicle, which performs drifting maneuvers in a dynamic environment, is a stochastic system since it shows unsteady behavior, and its environment has randomness such as friction coefficient of road and air resistance disturbances. It can also be considered as a sequential decision-making process that requires the precise execution of a series of high-frequency decisions such as throttle and steering [7]. The long-term maneuverability and agility performance of a vehicle, whether controlled by a driver or a controller, determines its drifting quality. Thus, the autonomous drifting task is a well-suited use case for reinforcement learning.

## 1.2. RELATED WORK

There have been a number of studies that focus on development of the drift control techniques to enhance vehicle handling and/ or give vehicles additional maneuverability to improve their trajectory following ability. The published approaches for drift control can be categorized into three categories: (i) traditional control methods, (ii) learning-based methods and (iii) combination of at least two of them.

### 1.2.1. TRADITIONAL CONTROL APPROACHES

The paper [5] of Voser et al. indicates the existence of steady-state drifting equilibria and presents a simple framework to deliberately induce the rear-tire saturation and stabilize the vehicle around the drift equilibria.

Learning from skilled drivers can be applicable for drift control task. Velenis et al. [8] suggest a method using experimental lateral (steering) and longitudinal (throttle and brake torque) data, measured during execution of drifting maneuver by an expert driver using a rear-wheel-drive (RWD) vehicle, for designing steady-state drift controller. They show that, by imitating skilled human drivers, it can be achievable to design a controller stabilizing RWD vehicle drifting with respect to the drifting equilibria.

### 1.2.2. LEARNING-BASED CONTROL APPROACHES

RL-based local methods can not guarantee global optimality for huge dimensional state space. To avoid local minima, Cutler et al. [9] present a framework transferring policies generated by a simple controller to more complex simulation environment. The policies, which a gradient-based learner improves in a more complex environment, are subsequently deployed in the real world for bootstrapping real-world learning. They have drawn inspiring conclusions for real-world learning through bootstrapping that enhances the policies already found in the complex simulation environment.

Cai et al. [7] use actor-critic algorithms to train closed-loop drift controller in CARLA simulator. In this work, the problem is formulated as a trajectory following task at high

speeds. They show how deep reinforcement learning algorithms can perform excellent generalization ability in challenging environments.

### **1.3.OBJECTIVE**

The primary objective of this thesis is the development of a practical transfer learning framework refining purposive initial policies from the multiple-input multiple-output (MIMO) linear quadratic LQ drift stabilizer [10] and transferring them to the deep RL-based vehicle motion controller learning how to move towards the randomly selected target points and execute drifting maneuver around them. For this purpose, the linear quadratic (LQ) controller [10], which generates steering angle and rear drive force as control inputs of three-state bicycle model, was deployed in the partially observable environment to produce efficient datasets. The generated datasets were used for training of a deep neural network model.

In order to reach the desired results, the agent must be able to fulfill following subtasks:

- Point-goal navigation,
- Moving closer to the randomly selected goal point and drifting around it.

## 2. REINFORCEMENT LEARNING

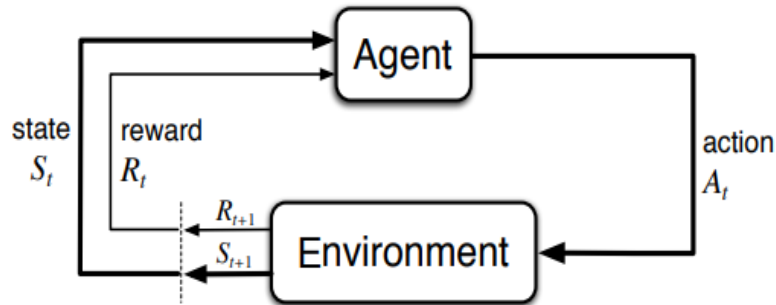
Reinforcement learning is an area of machine learning that deals with the problem of sequential decision making under uncertainty and offers robust algorithms to map situations to actions to maximize cumulative reward. In reinforcement learning, the agent must be capable of comparatively perceiving the state of its environment and taking action to affect the state [11]. The agent is not told which action to select, but needs to discover the dynamic environment and tries to find the optimal actions being more efficient in producing reward under different conditions, through trial and error.

### 2.1. MARKOV DECISION PROCESS

Markov Decision Process (MDP) is a framework that suggests a simple way of formalizing sequential decision-making in a probabilistic environment. At each time-step  $t$ , the agent observes its environment's state  $s_t$  and takes any action that is available in the current state. Based on the consequence of its last action, it receives a scalar reward signal  $R_{t+1}$  and brings itself to a new state  $s_{t+1}$  at time  $t+1$ . The goal of MDP is to train the agent to learn a policy that maximizes the cumulative rewards:

$$\sum_{t=0}^{\infty} \gamma_t R(s_t, a_t, s_{t+1}) \quad (2.1)$$

Where  $\gamma$  represents the discount factor, varying between 0 to 1, that determines how much the agent cares about the immediate reward.



*Figure 3 : Representation of agent-environment interaction in a Markov Decision Process*

## 2.2. TRADE-OFF: EXPLORATION VS EXPLOITATION

A key problem in reinforcement learning is to find an efficient way of striking a balance between exploration and exploitation. When exploiting, the agent tends to choose the greedy actions. However, initially, the agent is required to explore sufficiently all states to collect accurate information for avoiding spinning around local optima.

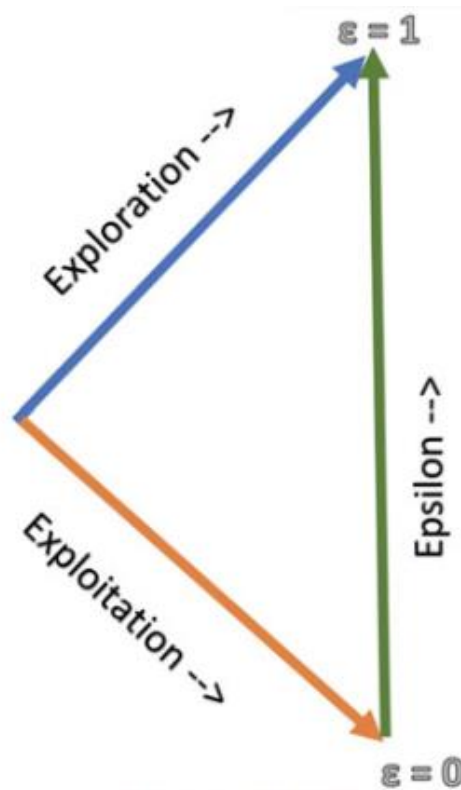


Figure 4 : Exploration-Exploitation dilemma

## 2.3. VALUE-BASED REINFORCEMENT LEARNING APPROACH

According to the value-based approach, a value function approximates the value of any given state or its potential to perform an action when following the policy  $\pi$ . The state-value function  $V^\pi(s)$  and is expected return from state  $s$  following policy  $\pi$ .

$$V^\pi(s) = E_\pi \{ R_t \mid s_t = s \} \quad (2.2)$$

The state-action value function, also known as Q function, is the expected return after selecting an action from state  $s$  with the action  $a$  according to policy  $\pi$ .

$$Q^\pi(s,a) = E_\pi \{ R_t \mid s_t = s, a_t = a \} \quad (2.3)$$

The optimal value function  $V^*(s)$  is a unique function that maximizes the expected value over all possible state:

$$V^*(s) = \max_a Q(s, a) \quad (2.4)$$

$$Q^*(s, a) = \max_\pi Q_\pi(s, a) \quad (2.5)$$

### 2.3.1. Q-LEARNING

Q-learning is a value-based off-policy algorithm that estimates both state value and state-action value to take the best possible action [12]. At each time-step, Q-learning updates the Q function(state-action value) as depicted in equation (2.5):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (2.6)$$

In the equation (2.5),  $\alpha$  represents the learning rate (between 0 and 1) and adjusts the weight of new return in the  $Q$ -value. In addition,  $\gamma$  represents the discount factor (between 0 and 1).

---

#### **Algorithm 1 : Q-Learning (off-policy) for estimating $Q$ Function**

---

- 1 : Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\varepsilon > 0$
- 2 : Initialize  $Q(s,a) = 0$ , for all  $s \in S, a \in A$
- 3 : **Repeat until the end of the episode:**
- 4 :    $s_t \leftarrow$  Initial State
- 5 :   Select  $a_t$  based on the exploration strategy (e.g.  $\varepsilon$  – greedy) from  $s_t$
- 6 :   **Loop for each step of episode:**
- 7 :     Take action  $a_t$ , observe  $r_{t+1}, s_{t+1}$
- 8 :     Choose  $a_{t+1}$  based on the exploration strategy from  $s_{t+1}$
- 9 :      $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$
- 10 :     $s_t \leftarrow s_{t+1}$
- 11 :    **if then**  $s ==$  terminal **then**
- 12 :       $Q(s_{t+1}, a_{t+1}) = 0$

```

13 :      end if
14 :      end for

```

---

Q-Learning [Chris Watkins,1989]

### 2.3.2. SARSA

Unlike Q-learning, the SARSA is a value-based on-policy algorithm. As its name suggests, the SARSA works on action works on a  $(state_t, action_t, reward_t, state_{t+1}, action_{t+1})$  tuple to estimate  $Q$  function. The SARSA updates the  $Q$  function based on the consequent state-action  $(s_{t+1}, a_{t+1})$  pair and immediate rewards received by the agent.

---

**Algorithm 2 :** SARSA (on-policy TD control) for estimating  $Q$  Function

---

```

1 : Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\epsilon > 0$ 
2 : Initialize  $Q(s,a)$  arbitrarily for all  $s \in S, a \in A$ 
3 : Repeat until the end of the episode:
4 :    $s_t \leftarrow$  Initial State
5 :   Select  $a_t$  based on the exploration strategy (e.g.  $\epsilon$  – greedy) from  $s_t$ 
6 :   Loop for each step of episode:
7 :     Take action  $a_t$ , observe  $r_{t+1}, s_{t+1}$ 
8 :     Choose  $a_{t+1}$  based on the exploration strategy from  $s_{t+1}$ 
9 :      $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$ 
10 :     $s_t \leftarrow s_{t+1}$ 
11 :     $a_t \leftarrow a_{t+1}$ 
12 :    if then  $s ==$  terminal then
13 :       $Q(s_{t+1}, a_{t+1}) = 0$ 
14 :    end if
15 :  end for

```

---

SARSA [Rummery & Niranjan,1994]

## 2.4.ACTOR-CRITIC METHODS

Actor-critic methods incorporate the benefit of the policy-gradient approach and the value function approach. Based on the current policy, the actor selects an action  $a$  for a given state. Instead of using the value function for action selection, it measures the goodness of the action taken by the actor and to update the policy when it is needed [6], [12]. The value function approximation lead to higher bias and lower variance, while the policy gradient lead to higher variance and lower bias.

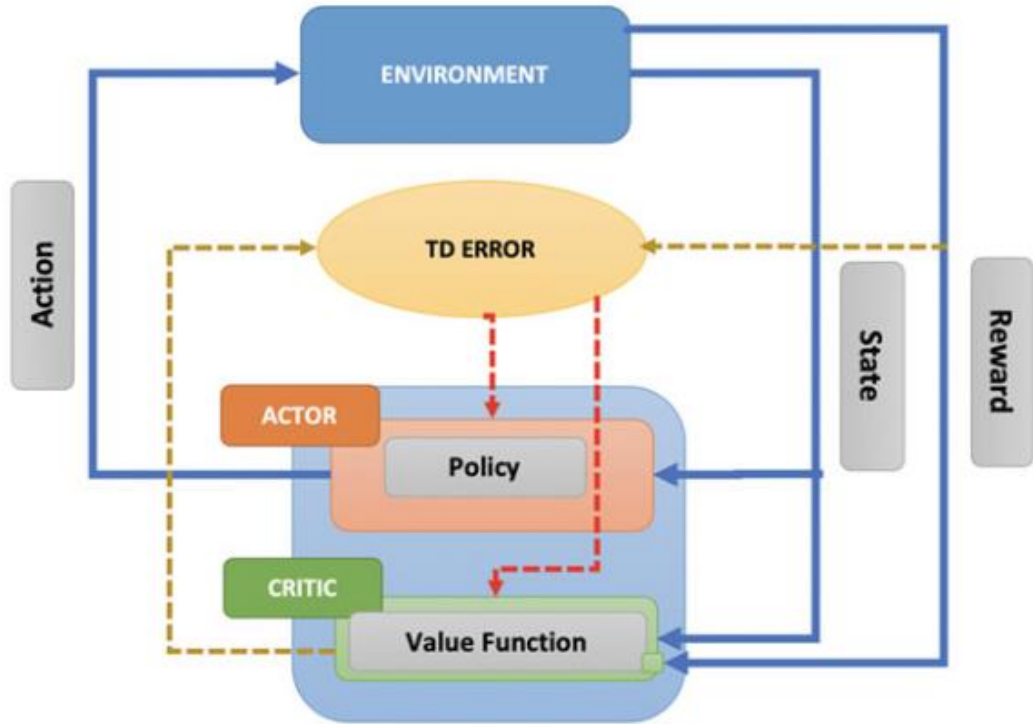


Figure 5: Actor-Critic Architecture : According to critic's estimations, the actor updates the policy.

#### 2.4.1. ASYNCHRONOUS ADVANTAGE ACTOR-CRITIC (A3A) ALGORITHM

The asynchronous advantage actor critic(A3A) algorithm , presented in the paper [13] of Mnih et al., has shown a great success on various continous motor control and navigation problems. The key advantage of the A3A algorithm is that it can train deep neural network policies without massive amounts of computational power requirements [13].

---

##### Algorithm 3: (A3A) Algorithm

---

- 1 : Assume global shared parameter vectors  $\vartheta$  and  $\vartheta^-$ , and  $T = 0$
- 2 : Initialize thread step counter  $t \leftarrow 0$
- 3 : Initialize target network weights  $\vartheta^- \leftarrow \vartheta$
- 4 : Initialize network gradients  $d\vartheta \leftarrow 0$
- 5 : Get initial state  $s$
- 6 : **Repeat**
- 7 :     Take action  $a$  with  $\epsilon$  – greedy policy based on  $Q(s,a; \vartheta)$
- 8 :     Receive new state  $s'$  and reward  $r$



```

9 :    $y = \begin{cases} r & \text{for terminal } s' \\ r + \gamma \max Q(s', a'; \vartheta^-) & \text{for non-terminal } s' \end{cases}$ 
10:   Accumulate gradients wrt  $\vartheta$ :  $d\vartheta \leftarrow d\vartheta + \frac{\partial (y - \vartheta(s', a'; \vartheta))^2}{\partial \vartheta}$ 
11:    $s \leftarrow s'$ 
12:    $T \leftarrow T + 1$  and  $t \leftarrow t + 1$ 
13:   if  $T \bmod I_{target} == 0$  then
14:     update the target network  $\vartheta^- \leftarrow \vartheta$ 
15:   end if
16:   if  $t \bmod I_{AsyncUpdate} == 0$  or  $s$  is terminal then
17:     perform asynchronous update of  $\vartheta$  using  $d\vartheta$ 
18:     clear gradients  $d\vartheta \leftarrow 0$ 
19:   end if
20: until  $T > T_{max}$ 

```

A3A Algorithm [Mnih et al., 2016]

### 2.4.2. SOFT ACTOR-CRITIC (SAC) ALGORITHM

The soft actor-critic (SAC) [14] algorithm was proposed by Haarnoja et al. in 2018. According to the basis of SAC, the actor focuses on simultaneously maximizing expected return and uncertainty in the policy to fulfill the task while mostly acting randomly. In the paper [14], it is claimed that the SAC algorithm reaches robust performance, outperforming other reinforcement learning algorithms, whether on-policy algorithms or off-policy algorithms, in sample-efficiency and asymptotic performance.

---

**Algorithm 4:** Soft Actor-Critic (SAC)
 

---

1 : <b>Input:</b> $\theta_1, \theta_2, \varphi$	<u>Initial parameters</u>
2 : $\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$	<u>Initialize target network weights</u>
3 : $D \leftarrow \emptyset$	<u>Initialize an empty replay pool</u>
4 : <b>for</b> each iteration <b>do</b>	
5 : <b>for</b> each environment step <b>do</b>	
6 : $a_t \sim \pi_\varphi(a_t, s_t)$	<u>Sample action from the policy</u>
7 : $s_t \sim p(s_{t+1}   s_t, a_t)$	<u>Sample transition from the environment</u>
8 : $D \leftarrow D \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$	
9 : <b>end for</b>	
10: <b>for</b> each gradient step <b>do</b>	
11: $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{Q_i} J_Q(Q_i)$ for $i \in \{1, 2\}$	
12: $\varphi \leftarrow \varphi - \lambda_\pi \nabla_\varphi J_Q(\varphi)$	<u>Update policy weights</u>
13: $\alpha \leftarrow \alpha - \lambda \nabla_\alpha J(\alpha)$	

---

```

14:          $\theta_i \leftarrow r \theta_i + (1-r)(Q_i)$  for  $i \in \{1,2\}$ 
15:     end for
16: end for
17: Output:  $\theta_1, \theta_2, \varphi$ 

```

---

Soft actor-critic (SAC) [Haarnoja, 2018]

## 2.5. MODEL-BASED REINFORCEMENT LEARNING APPROACH

### 2.5.1. PiLCO (PROBABILISTIC INFERENCE FOR LEARNING CONTROL)

#### ALGORITHM

The PiLCO [15], proposed by Deisenroth et al., is a model-based policy search algorithm. By eliminating the negative impact of model bias, the PiLCO overcomes the uncertainty in the learning model [9]. Due to its data-efficient property, it speeds up learning process by extracting more information from data. Another advantage of the PiLCO is that it is capable of detecting where additional data is required to enhance the performance of control strategy.

---

#### Algorithm 5: PiLCO (Probabilistic Inference for Learning Control)

---

```

1 : init: Sample controller parameters  $\theta \sim N(0, I)$ . Apply random control signals and
    record data
2 : Repeat
3 :   Learn probabilistic (GP) dynamics model
4 :   Model-based policy search
5 :   Repeat
6 :     Approximate inference for policy evaluation
7 :     Gradient-based policy improvement
8 :     Update parameters  $\theta$ 
9 :   until convergence; return  $\theta^*$ 
10:   Set  $\pi^* \leftarrow \pi(\theta^*)$ 
11:   Apply  $\pi^*$  to system (single trial/episode) and record data.
12: until task learned

```

---

PiLCO [Deisenroth, 2011]

### 3. SIMULATION ENVIRONMENT

This chapter introduces the simulation environment, which provides an opportunity to test different vehicle control techniques under various road surface conditions. The capability of vehicle controllers must be validated before deploying in the real world. Furthermore, evaluating vehicle controllers on actual cars can be very costly. High-fidelity driving simulators accurately approximate the real world and provide a proper environment for verifying the expected vehicle behavior. Therefore, I have built up a simulation environment consisting of the brush tire model and kinematic bicycle model.

#### 3.1. THE BICYCLE MODEL

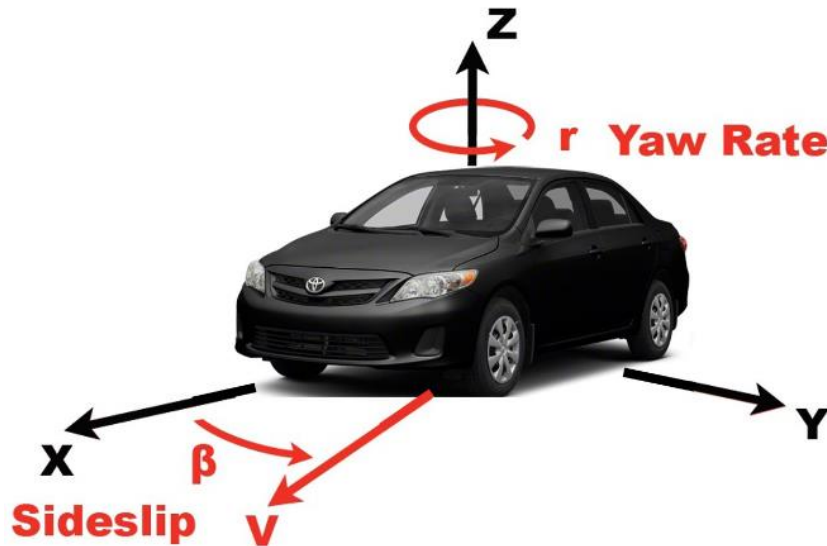


Figure 6: Body-Fixed Coordinate System [16]

The vehicle model was designed as a rigid body in a 2-dimensional world since the number of inputs needed for 2D simulations is smaller than the number of inputs needed for 3D simulations. The key advantage of 2D simulations is that they require much less computational power, and hence it speeds up the training process [17]. Most of the robots represented in the two-dimensional environment have three degrees of

freedom, which refers to rotation about the robot's longitudinal axis (roll), rotation about the robot's lateral axis (pitch), and rotation about the robot's vertical axis (yaw) [18].

In this research, I have used a three-state bicycle model, which assumes that the right and left tires of the vehicle are lumped into one at each axle with twice cornering stiffness, as depicted in Figure 7 [19]. The bicycle model allows two control inputs: steering angle ( $\delta$ ) of front wheels and longitudinal force ( $F_{xR}$ ) at rear wheels. The force balance of the center of gravity in the lateral axis and moment balance along the vertical axis can be used to compute the equations of the lateral and longitudinal motion of the vehicle body [10]:

$$\dot{U}_y = \frac{F_{yF} + F_{yR}}{m} - rU_x \quad (3.1)$$

$$\dot{U}_y = \frac{F_{yF} + F_{yR}}{m} - rU_x \quad (3.2)$$

$$\dot{r} = \frac{aF_{yF} + bF_{yR}}{I_z} \quad (3.3)$$

As shown in Figure 7,  $CG$  is the center of gravity,  $r$  is the yaw-rate,  $F_{xR}$  is the longitudinal force at rear wheels,  $\delta$  is the steering angle of front wheels,  $F_{yF}$  is the front lateral tire force,  $F_{yR}$  is the rear lateral tire force,  $m$  is the vehicle mass,  $\alpha_f$  is the front wheel side-slip angle,  $\alpha_r$  is the rear wheel side-slip angle,  $\beta$  is the vehicle's side-slip angle,  $U$  is a vehicle velocity at  $CG$ ,  $a$  is the distance between the  $CG$  and the front axle,  $b$  is the distance between the  $CG$  and the rear axle.

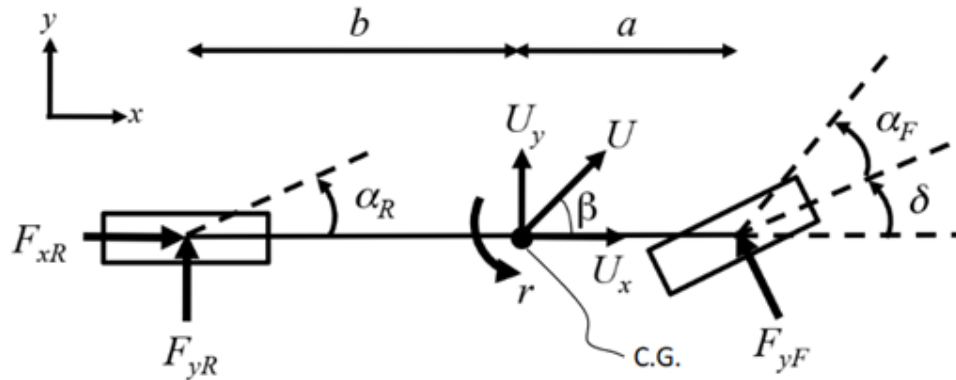


Figure 7: Bicycle Model [19]

The side-slip angle ( $\beta$ ) can be computed as the arctangent of the ratio of the  $U_y$  to the  $U_x$ , as shown in the equation (3.4). Drifting can be characterized by large side-slip angles and oversteering, which occurs when the rear tires are saturated.

$$\beta = \arctan\left(\frac{U_y}{U_x}\right) \approx \frac{U_y}{U_x} \quad (3.4)$$

$$\dot{\beta} = \frac{U_y}{U_x} \quad (3.5)$$

The bicycle model is primarily used to characterize the vehicle's drifting dynamics. Therefore, I decided to set the first state of the bicycle model to side-slip angle. By using the approximation in the equation (3.5), the equation (3.1) can be substituted with the following equation:

$$\dot{\beta} = \frac{F_{yF} + F_{yR}}{m U_x} - r \quad (3.6)$$

### 3.2. BRUSH TIRE MODEL

In this research, the air drag and rolling resistance are ignored since the developed controller primarily intends to operate at modest speeds. The forces, mentioned in section 3.1, are generated by the contact patches between the tires of the car and the road surface. The force generation mechanism on the car relies strongly on the side force and the total available force from friction. Accordingly, the design of a realistic tire model plays a crucial role in the vehicle model's overall accuracy.

The brush tire model has been implemented and utilized for computing the lateral forces acting on front tire. As depicted in Figure 8, the brush tire model has three main parts: (i) carcass, (ii) ring, and (iii) elastic brushes.

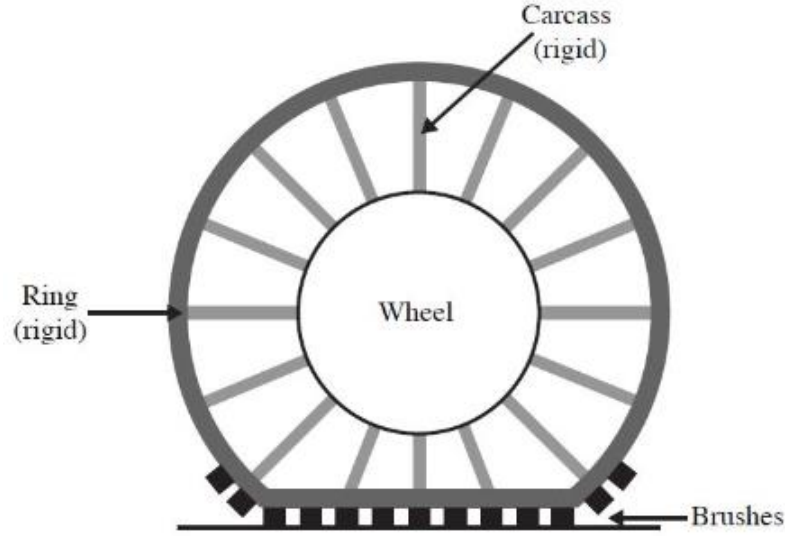


Figure 8: The Parts of Brush Tire Model [19]

The lateral force ( $F_Y$ ) acting on contact patches can be written as the following, where  $\alpha$  is the slip angle.

$$F_Y = \begin{cases} -C_\alpha \tan \alpha + \frac{C_\alpha^2}{3\mu F_z} |\tan \alpha| \tan \alpha - \frac{C_\alpha^3}{27\mu^2 F_z^2} \tan^3 \alpha & |\alpha| \leq \alpha_{sl} \\ -\mu F_z \operatorname{sgn} \alpha & |\alpha| > \alpha_{sl} \end{cases} \quad (3.7)$$

To compute the tire lateral forces, all that is required are the front and rear lateral tire slip values, which can be calculated by using the following equations:

$$\alpha_F = \operatorname{atan} \frac{U_y + ar}{U_x} - \delta \approx \arctan \left( \beta + \frac{a}{U_x} r \right) \quad (3.8)$$

$$\alpha_R = \operatorname{atan} \frac{U_y + br}{U_x} \approx \arctan \left( \beta + \frac{b}{U_x} r \right) \quad (3.9)$$

I use a more generic method in which the tire forces are coupled in order to obey friction limitations, which is called the friction circle shown in Figure 9. According to the friction circle method, the limit of lateral force can be written as:

$$F_y^{max} = \xi \mu F_z \quad (3.10)$$

$$\xi = \frac{\sqrt{(\mu F_z)^2 - F_x^2}}{\mu F_z} \quad (3.11)$$

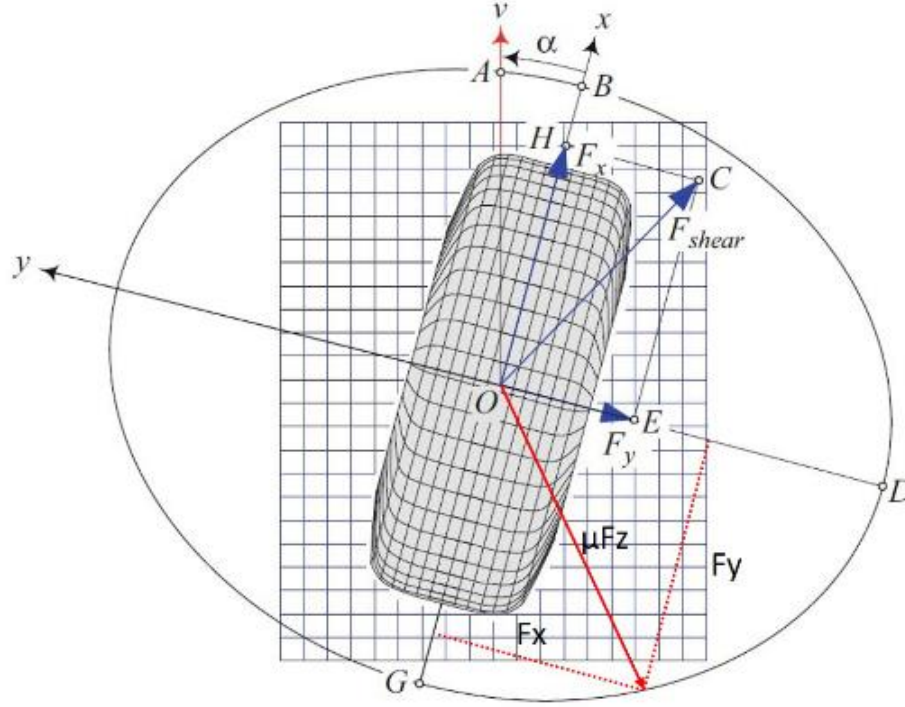


Figure 9: Diagram of the Friction Circle [20]

The equation (3.7) can be modified to calculate the lateral and longitudinal forces for rear tire:

$$F_Y = \begin{cases} -C_\alpha \tan \alpha + \frac{C_\alpha^2}{3\xi\mu F_z} |\tan \alpha| \tan \alpha - \frac{C_\alpha^3}{27\xi^2\mu^2 F_z^2} \tan^3 \alpha & |\alpha| \leq \alpha_{sl} \\ -\xi\mu F_z \operatorname{sgn} \alpha & |\alpha| > \alpha_{sl} \end{cases} \quad (3.12a)$$

$$\alpha_{sl} = \arctan \frac{3\xi\mu F_z}{C_\alpha} \quad (3.12b)$$

### 3.2.1. ROAD FRICTION COEFFICIENT

The road friction coefficient used in the bicycle model has been set to 0.9 since I assume that the vehicle is running on the dry asphalt road condition in both the training and test scenarios.

*Table 1- Road Friction Coefficient Range [22]*

Road Category	Friction Coefficient Range $\mu_{max}$
Macro rough/micro rough surface (draining mixes, bituminous concretes)	0.5 – 0.9
Macro smooth/micro rough surface (fine mixes)	0.4 – 0.8
Macrorough / microsmooth surface (rolled aggregates)	0.2 – 0.3
Macrosmooth / microsmooth surface(flushing asphalt)	0.1 – 0.2

### 3.3.MODEL VEHICLE

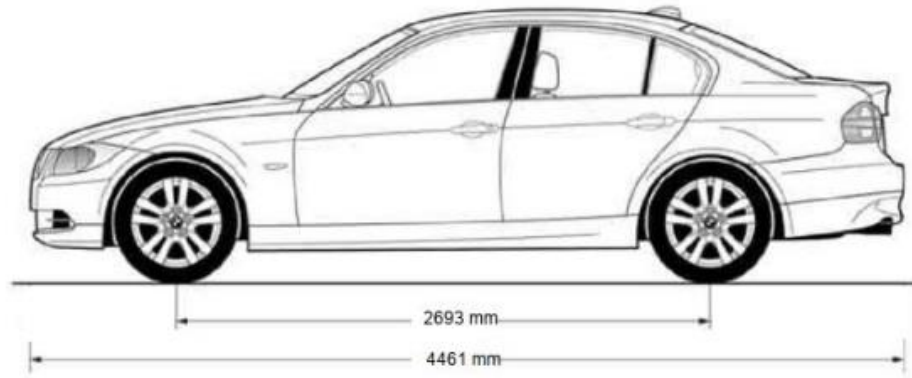


*Figure 10: BMW M2 Competition [20]*

I decided to use the BMW M2 Competition as the model vehicle for this research. The parameters of bicycle model are updated by taking into account the physical characteristics of the model car.



### 3.3.1. VEHICLE DIMENSIONS



*Figure 11: Wheelbase of Model Car [21]*

As depicted in Figure 11, the model vehicle's wheelbase, which refers to the horizontal distance between the center of the front and rear axles, is 2.693 m. This parameter will be used to find the model car's center of gravity.

### 3.3.2. CENTER OF GRAVITY

The measured wheel loads, which are given in Table 2, will be used to calculate the *CG* of the model car. The total loads on the front and rear axles are 930 and 875 kg, respectively.

*Table 2 – Wheel Loads*

<b>2 Persons inside [kg]</b>				
RL	FL	FR	RR	SUM
485	435	495	390	1805

The distance between the *CG* of the vehicle and its front axle can be calculated by using the following equation:

$$F_f * 0 + F_r * \ell - F_g * a = 0 \quad (3.10)$$

In a similar way, the distance between the *CG* of the vehicle and its rear axle can be calculated by using the following equation:

$$F_r * 0 + F_f * \ell - F_g * b = 0 \quad (3.11)$$

where the  $F_g$  is the force acting on the vehicle through its *CG*,  $\ell$  is the wheelbase of the car, the  $F_f$  and  $F_r$  are the forces acting on the car's front and rear axles.

### 3.4. IMPLEMENTATION OF THE BICYCLE MODEL IN SIMULINK

In this section, the three-state bicycle model will be implemented with the brush tire model. The equations defined in Sections (3.1) and (3.2) regarding the bicycle model can now be transferred into the SIMULINK environment.

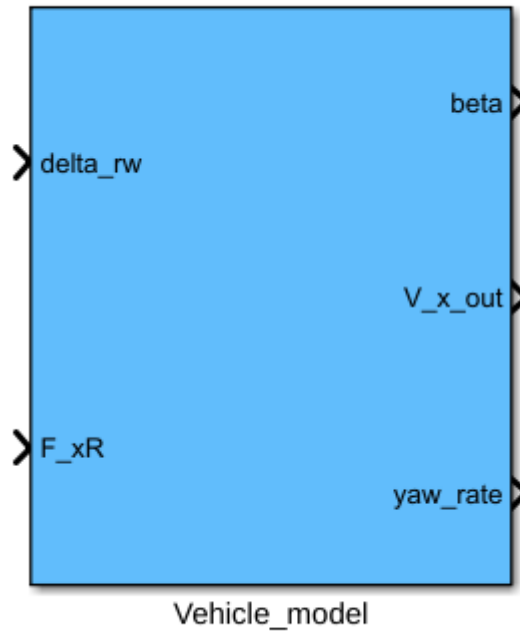


Figure 12: Three-state Dynamic Bicycle Model

The variables used for the dynamic bicycle model are given in Table 3. The parameters  $a$ ,  $b$ , and  $I_z$  were computed based on the equation (3.10) and (3.11).

Table 3 - Vehicle Parameters

Symbol	Parameter	Value
$m$	Vehicle Mass	1805 kg
$I_z$	Yaw Inertia	1634.8 kg.m <sup>2</sup>
$a$	Distance from the front axle center line and the CG	1.3055 m
$b$	Distance from the rear axle center line and the CG	1.3875 m
$C_{\alpha F}$	Front wheel cornering stiffness	300000 N/rad
$C_{\alpha R}$	Rear wheel cornering stiffness	500000 N/rad
$\mu$	Road friction coefficient	0.9
$\delta_{max}$	Maximum steering angle	0.62 rad

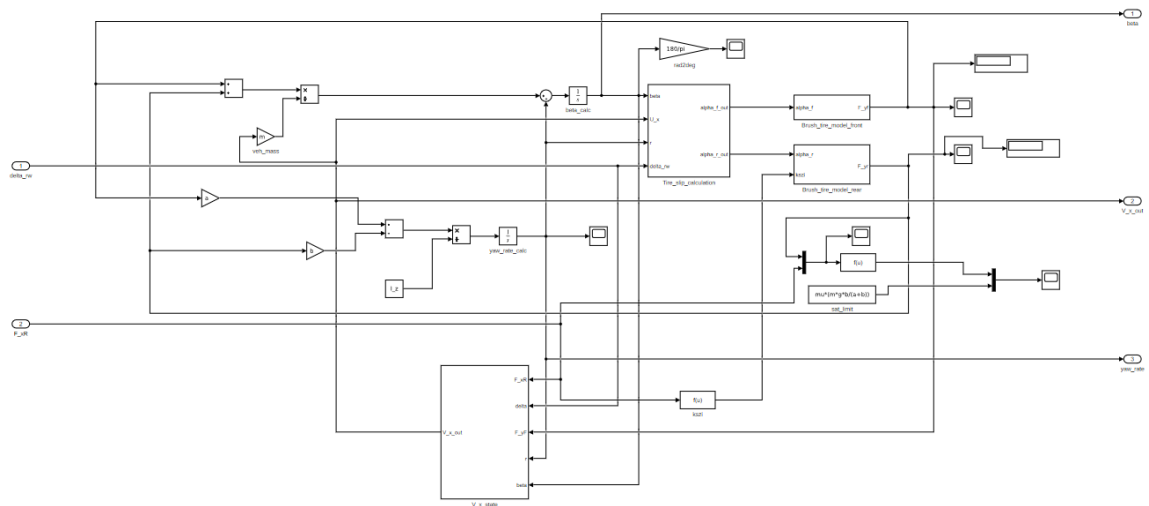


Figure 13: The General Structure of the Dynamic Bicycle Model

### 3.4.1. TIRE MODEL

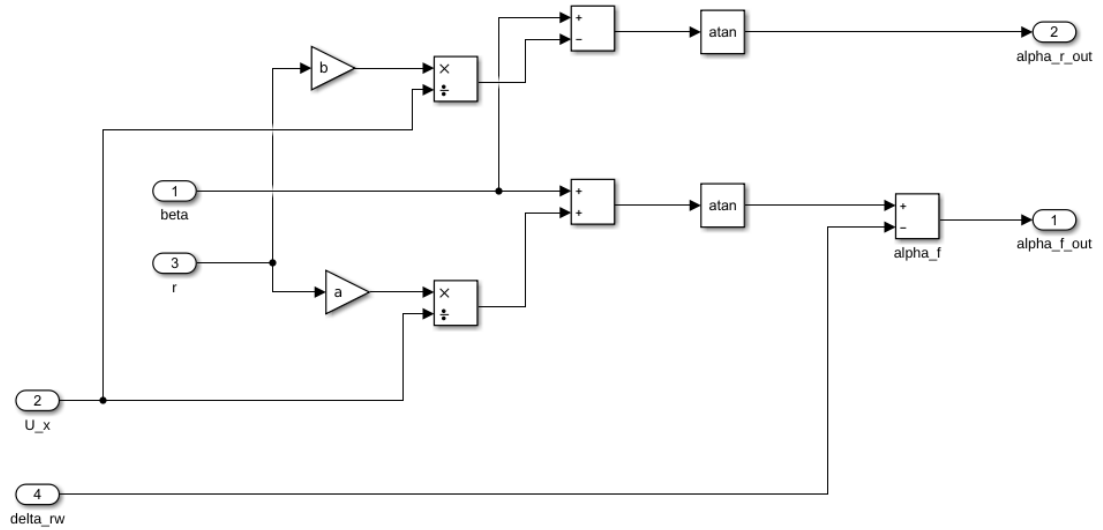


Figure 14: Structure of the block “Tire Slip Calculation”

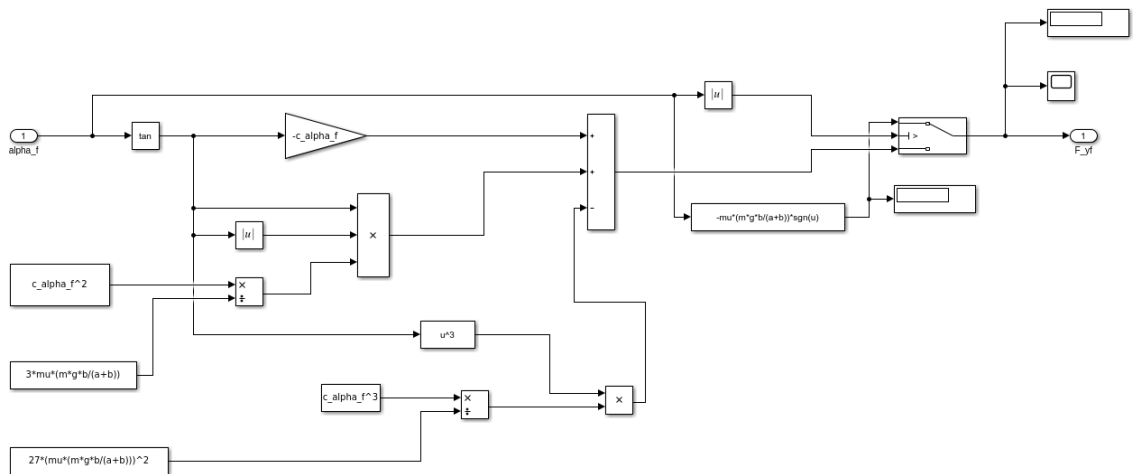


Figure 15: Structure of the block “Brush Tire Model – Front Wheel”

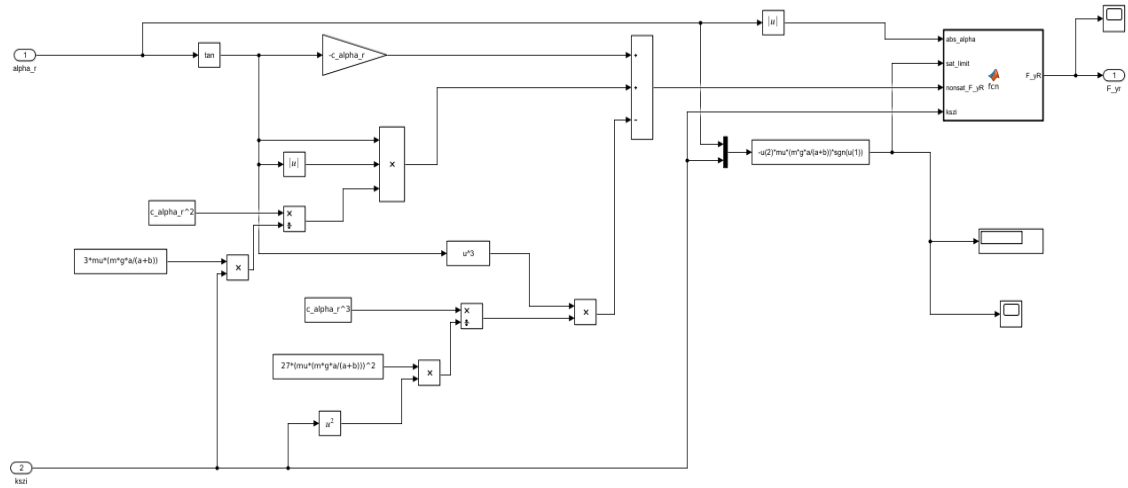


Figure 16: Structure of the block “Brush Tire Model – Rear Wheel”

### 3.4.2. STATE BLOCKS

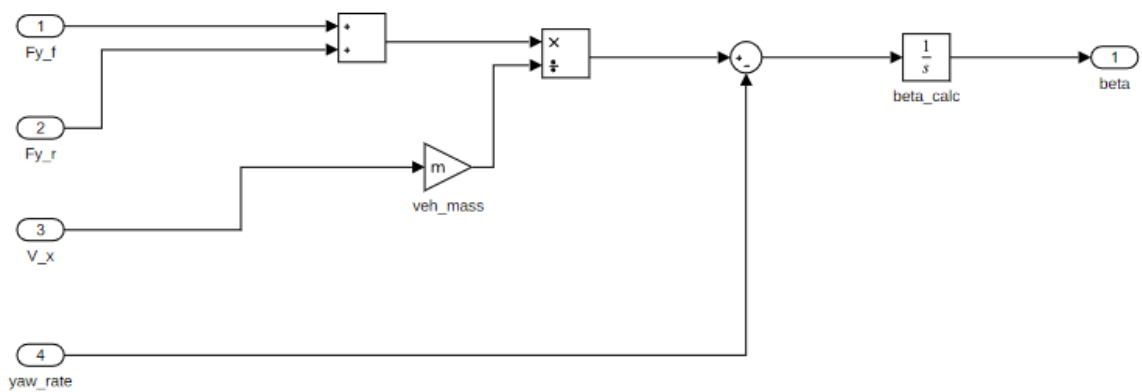


Figure 17: Structure of the block "Side-slip Angle"

As illustrated in Figure 18, the integrator block was included in the model to calculate the vehicle's instantaneous longitudinal velocity. For different starting scenarios, the initial value of the integrator block can be set to different values.

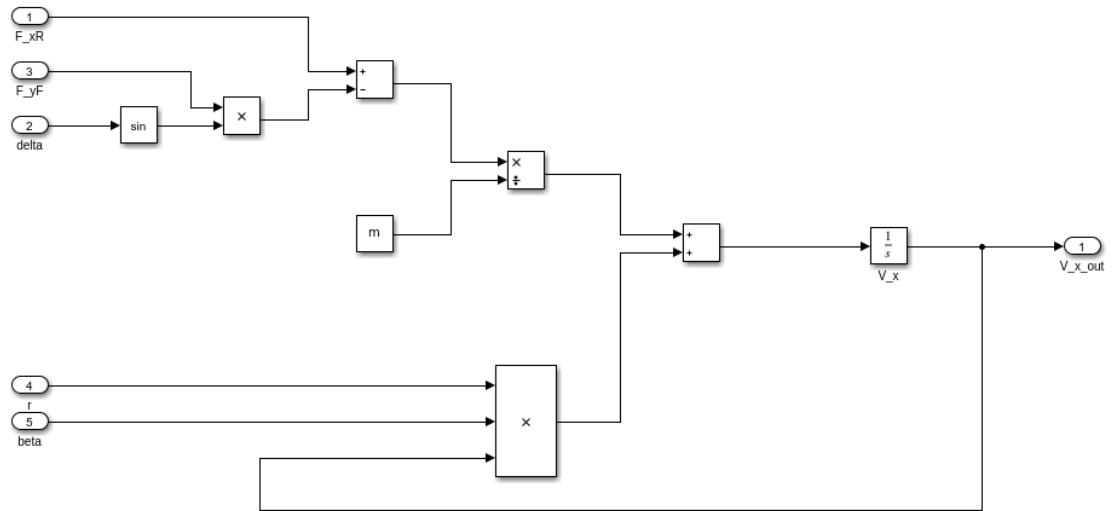


Figure 18: Structure of the block “Longitudinal Velocity”

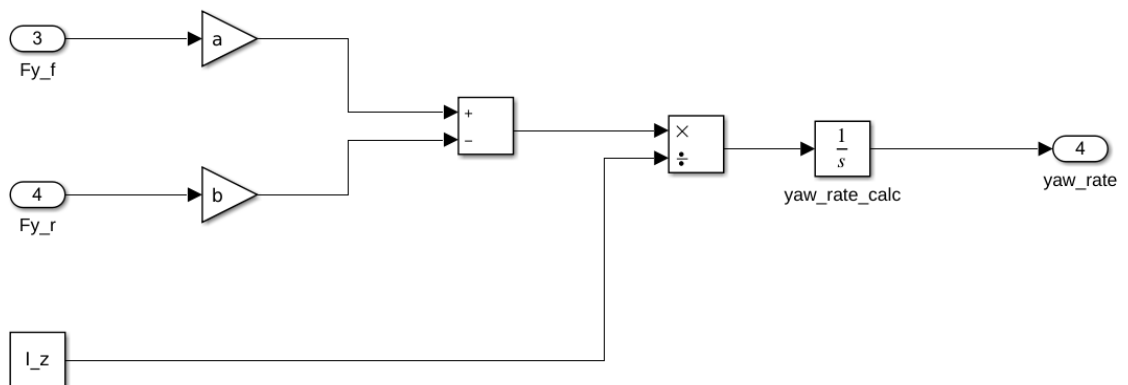


Figure 19: Structure of the block “Yaw-Rate”

### 3.4.3. ADDITIONAL BLOCKS

In this model, the integrator block is used to compute the heading angle of the vehicle body. The initial heading direction of the car can be adjusted by changing this block's initial value.

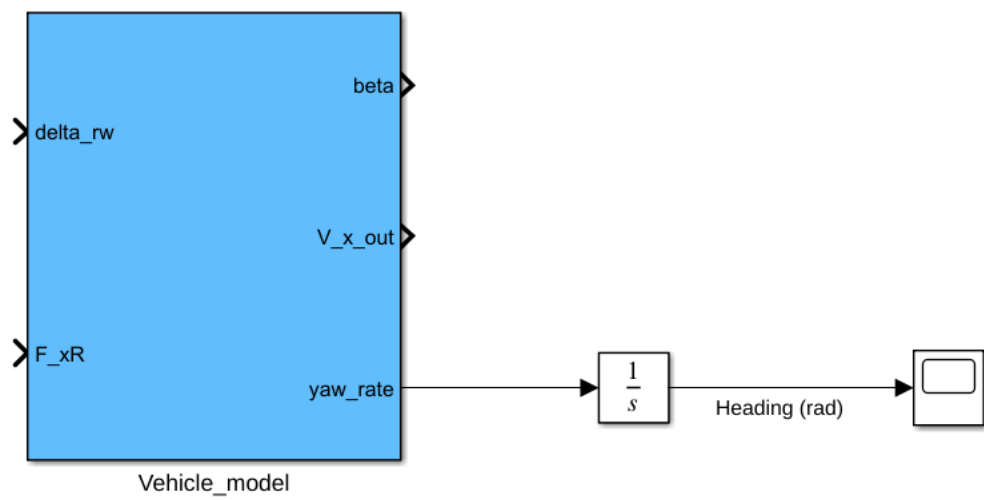


Figure 20: Heading angle

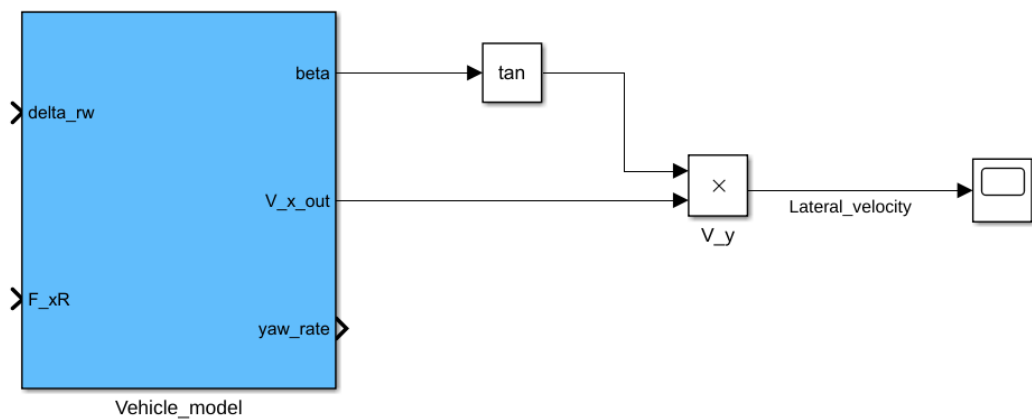


Figure 21: Lateral velocity

Figure 22 below illustrates how the vehicle's actual position can be computed from its yaw rate, side-slip angle, and the side velocities in the direction of its longitudinal and lateral axes. Furthermore, the trajectory followed by the vehicle can be plotted by using "XY Graph" block.

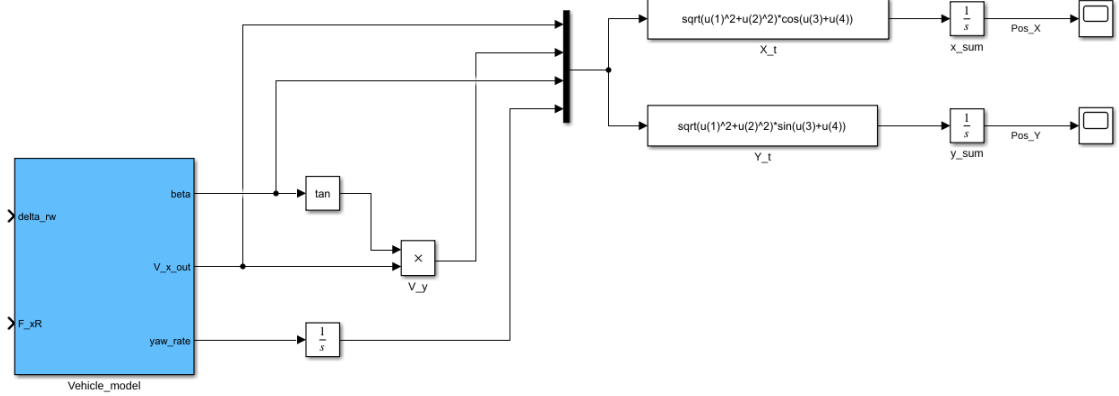


Figure 22: Calculation for the exact vehicle position

### 3.5. RL SETUP FOR AUTONOMOUS DRIFT CONTROL

In order to reach the desired results, the agent must be able to fulfill following subtasks:

- Point-goal navigation,
- Moving closer to the randomly selected goal point and drifting around it.

For this purpose, I selected two symmetric equilibria points based on the drifting equilibria analysis. In this way, the agent will learn how to drift in both clockwise and counter-clockwise directions. The drifting performance of the agent is evaluated with respect to the selected equilibria points.

In this section, some additional features, which are specific to the autonomous drifting problem, will be added to the previously introduced dynamic bicycle model: (i) the turning radius, (ii) the position of the vehicle's turning center, (iii) coordinate transformation from



the earth-fixed coordinate system to the body-fixed coordinate system, (iv) the encoding of heading angle, and (v) the maximum rate-of-change limit on the input signal for enhancement of the smoothness of the vehicle behavior.

### 3.5.1. TURNING RADIUS

One of the crucial steps to design a proper RL-based controller for the autonomous drift control task is to precisely know the length of the turning radius. With this knowledge, the agent can abstract some beneficial features regarding the drifting maneuver. As [24] proposed, I use the value of the rotation rate of the vehicle velocity vector  $\dot{\psi}$  to calculate the turning radius of the vehicle. It can be written as:

$$\dot{\psi} = \dot{\beta} + r \quad (3.13)$$

The length of the turning radius can be derived from the equation (3.13):

$$\kappa_{ref} = \frac{\dot{\psi}}{V_{total}} \quad (3.14)$$

$$\kappa_{ref} = \frac{1}{\mathcal{R}} \quad (3.15)$$

where the  $\mathcal{R}$  is the turning radius [24].

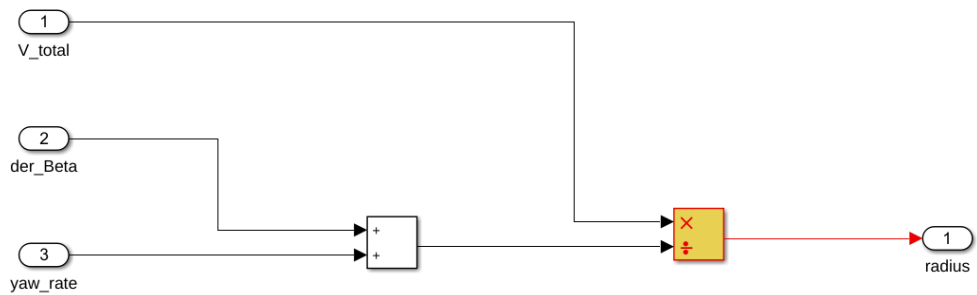


Figure 23: Structure of the block “Turning Radius”

### 3.5.2. THE POSITION OF THE VEHICLE'S TURNING CENTER

The turning radius described in the previous section can be used to find the position of the turning center of the car. The vehicle's instantaneous center of rotation can be written as following, where the  $y_{CoR}$  and  $x_{CoR}$  are the coordinates of the car's center of rotation:

$$x_{CoR} = x_{actual} + \cos[\psi - \text{sgn}(r) * \frac{\pi}{2} - \pi] * R \quad (3.16)$$

$$y_{CoR} = y_{actual} + \sin[\psi - \text{sgn}(r) * \frac{\pi}{2} - \pi] * R \quad (3.17)$$

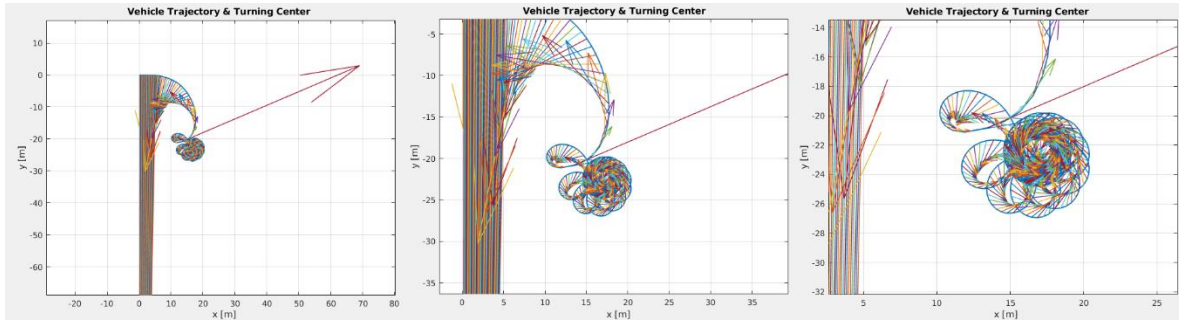


Figure 24: The vehicle's instantaneous center of rotation

As depicted in Figure 24, when the steering angle gets very small values, the turning center converges to the plus or minus infinity. Since it is not wise to use such input values for the neural networks, I set a limit for the turning radius. Thus, the turning radius can not exceed the threshold value, which is 10000.

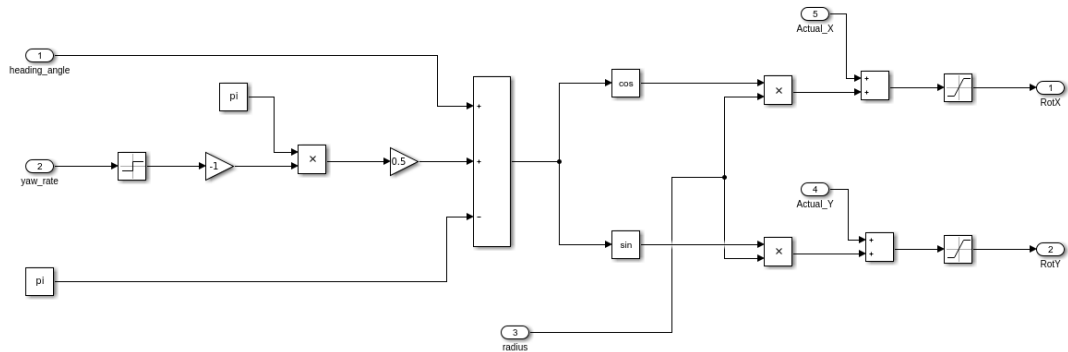


Figure 25: Structure of the block "Center of Rotation"

### 3.5.3. COORDINATE TRANSFORMATION

The body-fixed coordinate system is fixed to the vehicle; hence the orientation and the position of the vehicle can not be identified. The *CG* of the car is considered as the origin, and x-axes is assigned to longitudinal axes of the car [24] [25]. According to the earth-fixed coordinate system, the positive x-axis points east, and the positive y-axis points north. I compare the pros and cons of both coordinate systems. All outputs of the bicycle model are not described with respect to the same coordinate system. Before using them as the features of a neural network, the model outputs must be described with respect to the selected coordinate system. This allows the RL-agent to refine more effective policies for the continuous control domain.

Rather than transferring the vehicle's side velocities from the body-fixed coordinate system to the earth-fixed coordinate system, I transfer all other vectors and positions from the earth-fixed coordinate system to the body-fixed coordinate system. This will also strengthen the agent's generalization ability.

### 3.5.4. ENCODING HEADING ANGLE FOR LEARNING

The core idea for this section is to make the vehicle navigate to the goal point. The most prevalent issue regarding this task is needed to be solved is to adjust the control inputs according to the heading direction. The agent must minimize the difference between the vehicle's heading angle and the desired heading angle to move towards the goal point. In body-fixed coordinate system, the longitudinal axis of the vehicle becomes x-axis. Therefore, the angle between the actual heading and the desired heading can be written as:

$$\psi_{diff} = \arctan\left(\frac{y_{goal}}{x_{goal}}\right) \quad (3.18)$$

where  $x_{goal}$  and  $y_{goal}$  are the coordinates of the goal point.

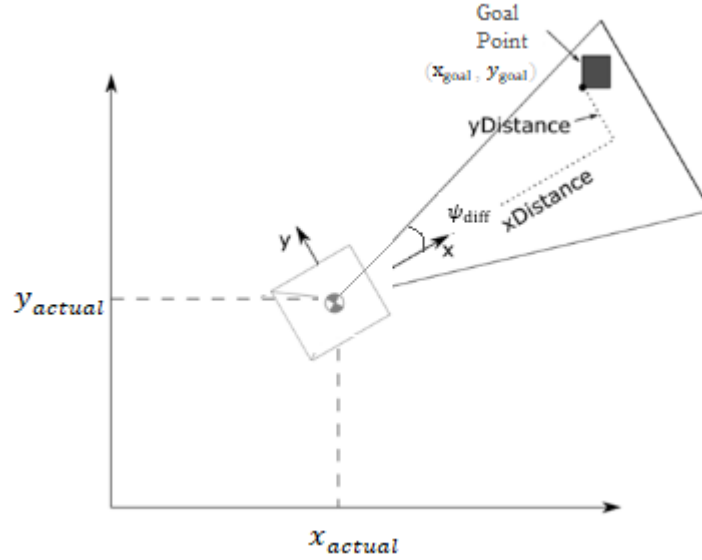


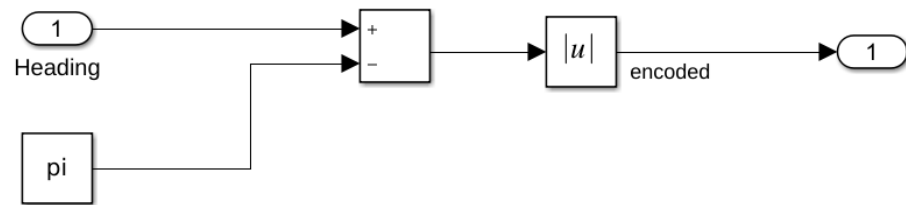
Figure 26: Point-goal navigation [27]

In MATLAB, the range of  $\text{atan}$  and  $\text{atan2}$  built-in functions return results limited to the intervals  $[-\pi/2, \pi/2]$  and  $[-\pi, \pi]$ , respectively. In both case, the  $\psi_{diff}$  tends to reach its maximum or minimum while the vehicle is heading towards the goal point. When the goal point stays right behind the goal point, the  $\psi_{diff}$  shows a tendency to converge to 0. That can be confusing for the Deep RL-agent consisting of neural networks. Thus, this puts an unnecessary burden on the learner. To avoid this problem, I encode the angle  $\psi_{diff}$  to effectively utilize it. The encoded parameter becomes:

$$\psi_{enc} = |\psi_{diff} - \pi| \quad (3.19)$$

The boundry of the encoded angle:  $\psi_{diff} \in [0, 2\pi]$

Unlike the  $\psi_{diff}$ , the  $\psi_{enc}$  always increases while the vehicle is turning its longitudinal axis toward the goal point regardless of its rotation direction. This provides proper understanding of the vehicle motion control.

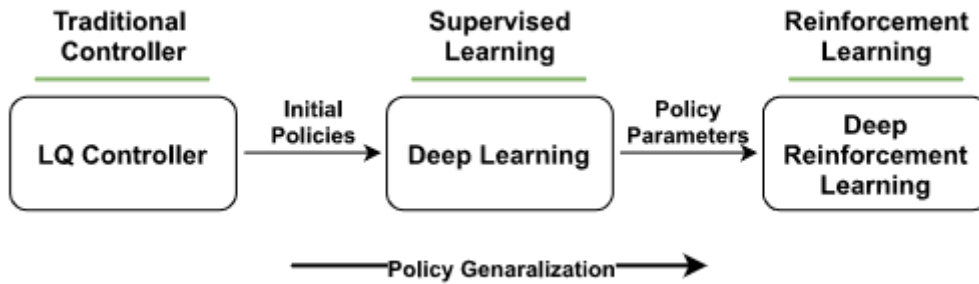


*Figure 27: Structure of the block “Heading-Encoding”*

## 4. TRANSFER LEARNING FRAMEWORK

As a breakthrough approach, traditional RL was primarily created for simple tasks and has provided robust solutions to simple tasks with low dimensional state space but struggled when dealing with very complicated domains [28]. In recent years, an integrated framework, in which the Deep Neural Networks are used as function approximators, has been designed to cope with high dimensional problems [29]. Deep Reinforcement Learning (DRL) is the name given to this hybrid approach. Deep Reinforcement Learning has shown superior performance in areas such as computer games and robotics.

Despite its outstanding achievement, the fact that DRL framework requires a high number of samples and training time is a big disadvantage. Obtaining adequate training samples can be difficult in the partially observable environments. Furthermore, the diversity of the samples determines the quality of the learner. In the area of supervised learning, transfer learning approaches have been intensively researched [30]. However, Transfer Learning for the RL tasks has recently drawn the attention of researchers in the artificial intelligence community [31].



*Figure 28: General Structure of the Transfer Learning Framework designed for this research*

In this section, I will introduce the Transfer learning framework, which I have designed for bootstrapping the RL-based autonomous drift controller. As shown in Figure 28, it consists of 3 subcomponents: (i) LQ controller, (ii) Neural Network, and (iii) Soft Actor-Critic: off-policy actor-critic Deep RL algorithm.

#### 4.1. MIMO LQR CONTROLLER FOR STABILIZING VEHICLE DRIFTING

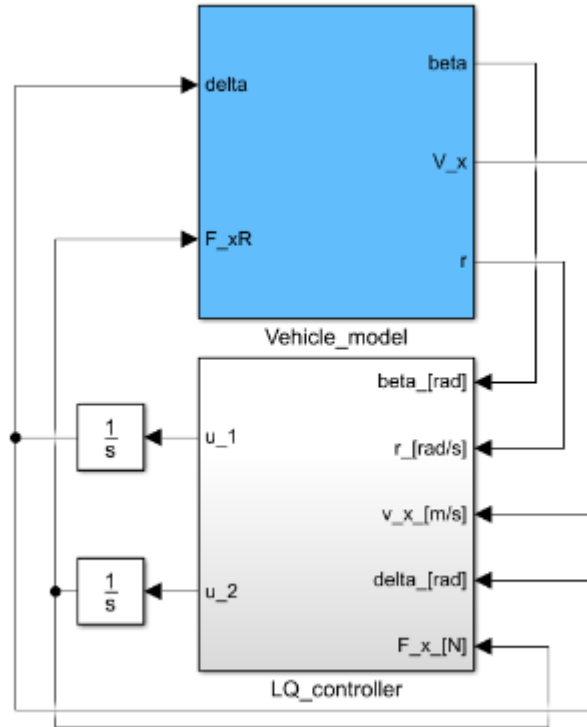


Figure 29 – Linear Quadratic Drift Controller

In this project, the MIMO LQR controller [10], proposed by Á. Bárdos et al., is used to stabilize circular drifting by generating control signals for a three-state bicycle vehicle model shown in Figure 29. The controller intends to stabilize the vehicle around the equilibrium points, which are calculated based on the vehicle system dynamics. The controller generates control signals to individually manipulate the steering angle of the vehicle at front wheels and the longitudinal drive force at rear wheels. Several simulations were conducted to increase the diversity of the dataset under different scenarios, in which the vehicle starts the motion with different initial longitudinal velocities under various road friction conditions.

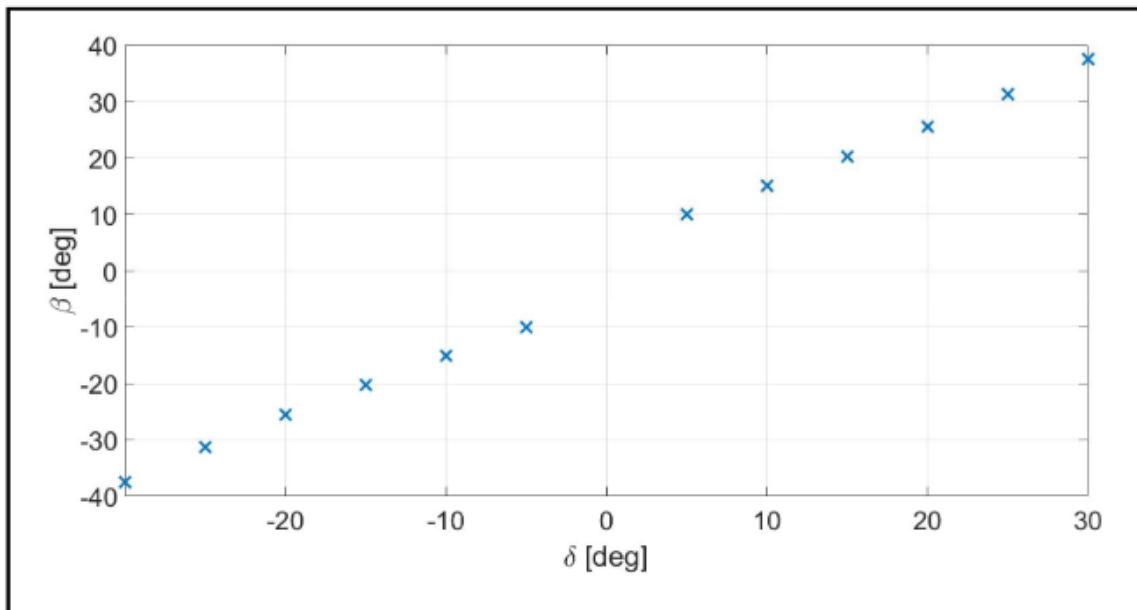


Figure 30: Drifting Equilibria ( Side-slip vs. Road-wheel angle at  $V_x = 10$  m/s) [10]

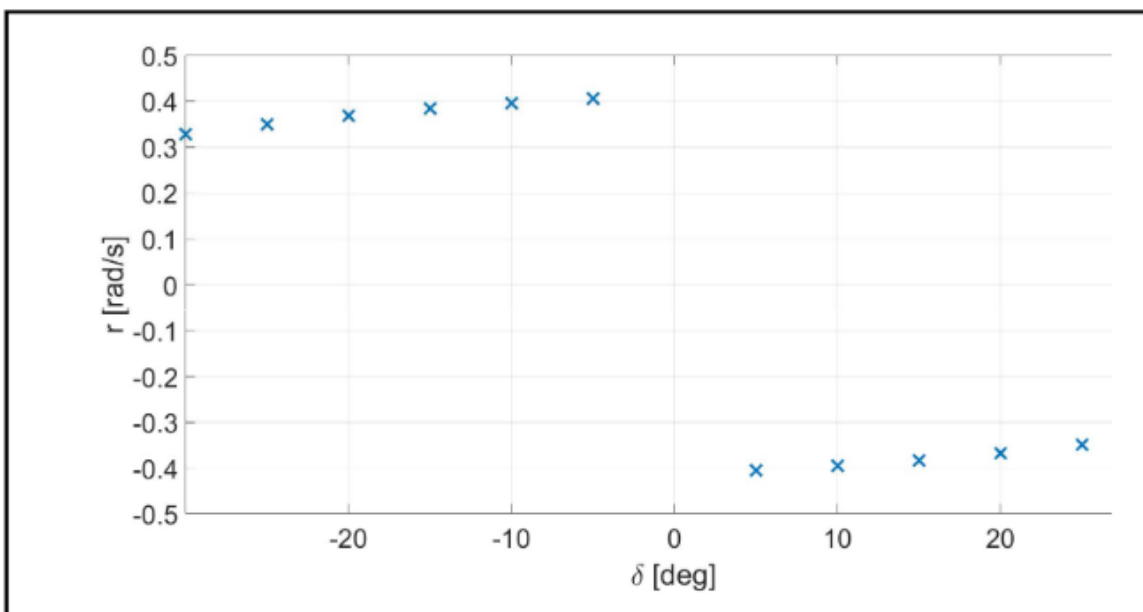


Figure 31: Drifting Equilibria ( Yaw rate vs. Steering angle at  $V_x = 10$  m/s) [10]



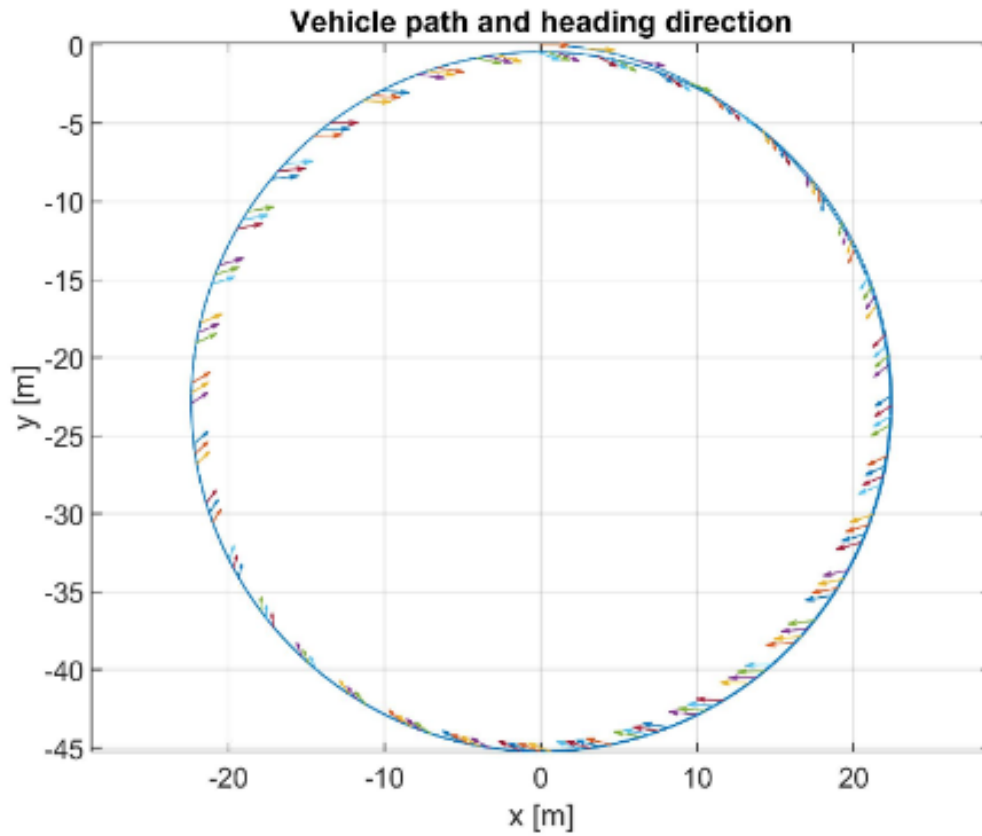


Figure 32: Clockwise drifting at  $\beta = 0.65$  rad and  $V_x = 8$  m/s [10]

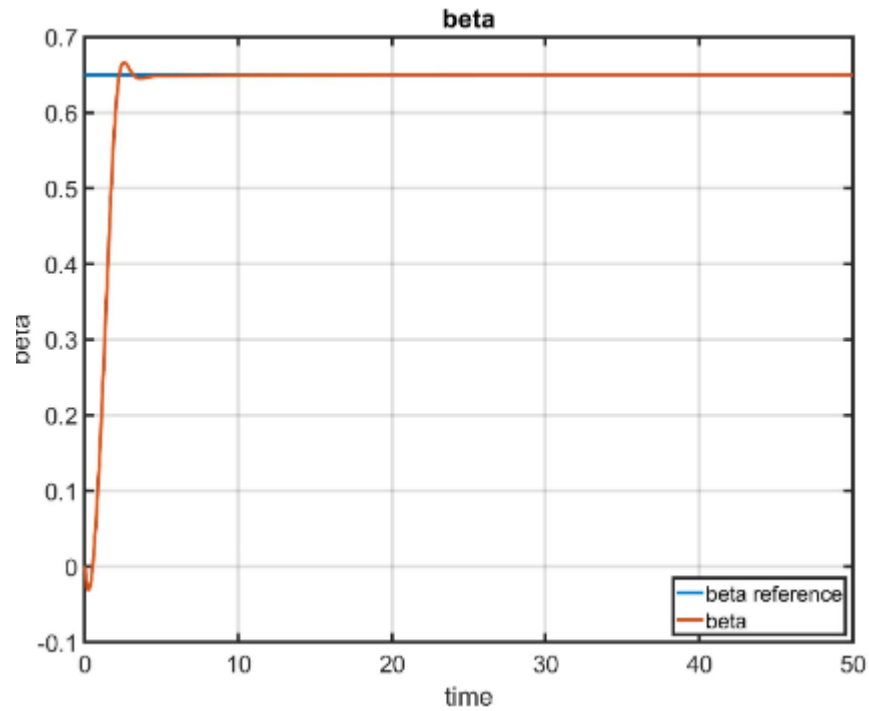


Figure 33: The LQR controller stabilizes the vehicle drifting at  $\beta = 0.65$  [10]

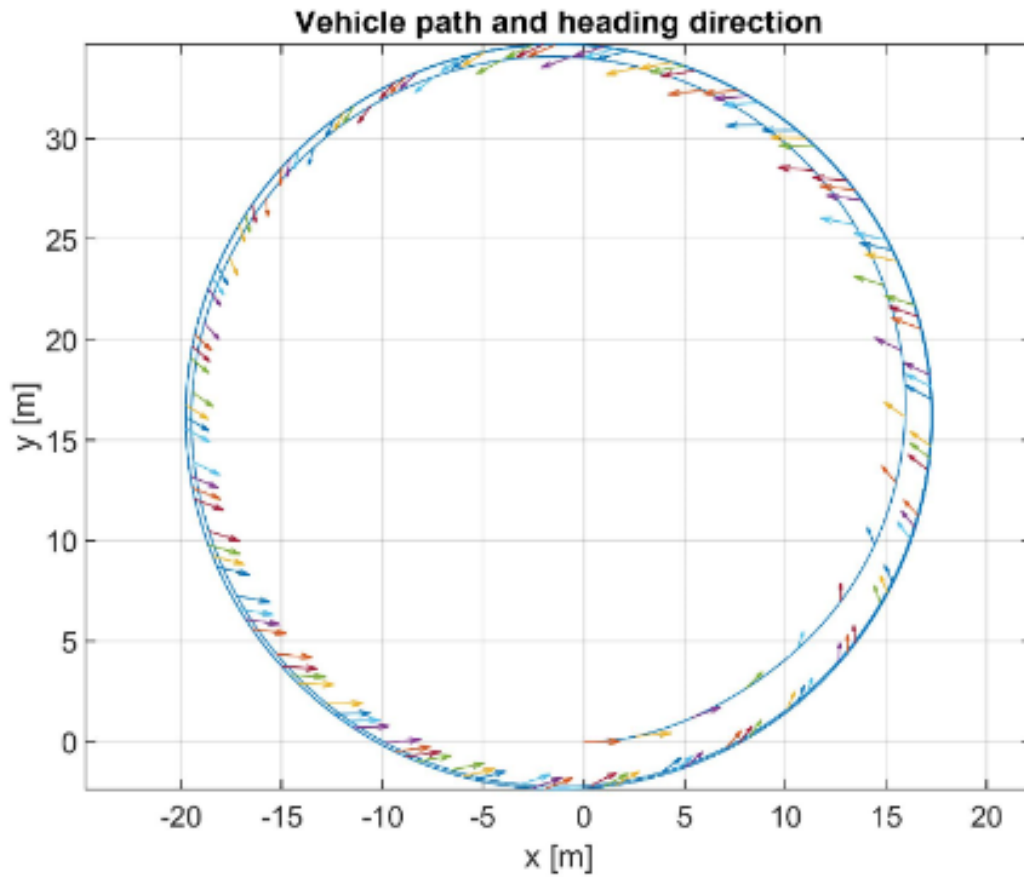


Figure 34: Counter clockwise drifting at  $\beta = -0.7$  rad and  $V_x = 7$  m/s [10]

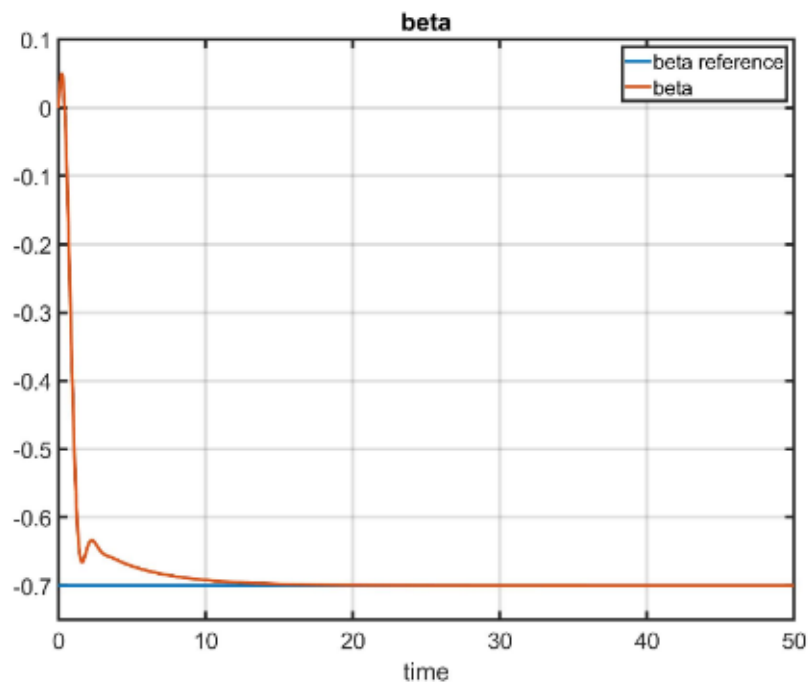


Figure 35: The LQR controller stabilizes the vehicle drifting at  $\beta = -0.7$  [10]

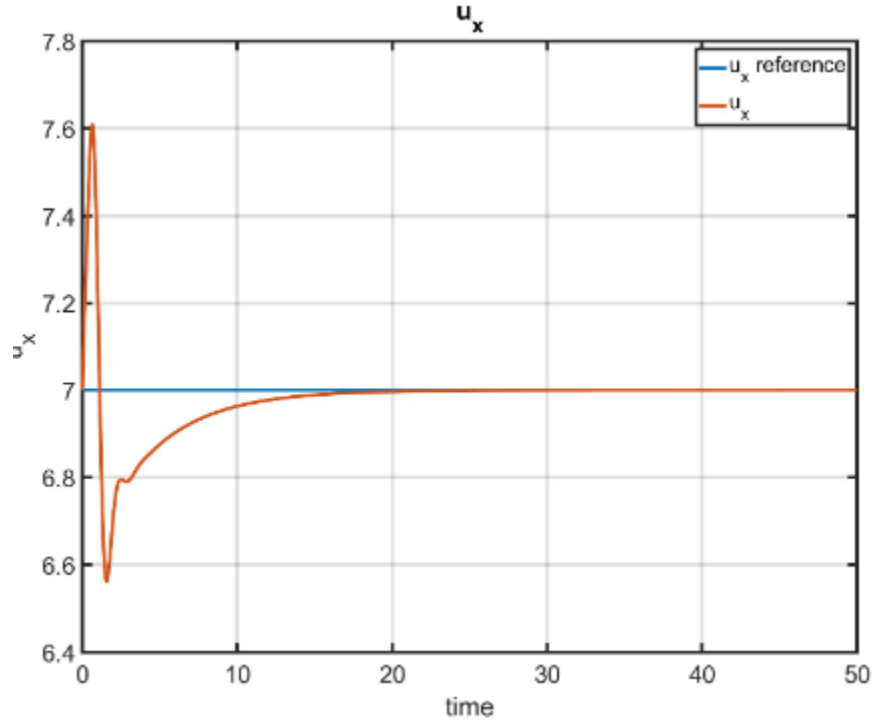


Figure 36: The LQR controller stabilizes vehicle's longitudinal velocity [10]

As shown in the simulation results, the LQR controller is able to stabilize the vehicle on a circular path regardless of whether it rotates clockwise or counterclockwise. Based on the conducted simulation, I generate the dataset, which involves the vehicle's yaw rate( $r$ ), longitudinal velocity( $V_x$ ), lateral velocity( $V_y$ ), heading angle( $\psi$ ), and sideslip angle( $\beta$ ). This dataset was used only to make the learner abstract features regarding the dynamics of the drifting maneuver. The learner fulfills the point-goal navigation task without any prior knowledge or bootstrapping.

## 4.2. AUTONOMOUS DRIFTING WITH DEEP LEARNING

In this part, the control problem is formulated as drifting on a circular path task around the different points without goal-point navigation. Another method that I applied to generate policies for bootstrapping is Deep Learning (DL). A deep neural network (DNN) with the four hidden layers was trained with the dataset generated by using the LQR controller. The first and second hidden layers are fully-connected layers with 16 neurons. The third and fourth hidden layers are fully-connected layers with 8 neurons. The output size of the deep neural network is 2. The DNN takes 5 inputs, which are the yaw rate( $r$ ), longitudinal velocity( $V_x$ ), lateral velocity( $V_y$ ), heading angle( $\psi$ ), and side-slip angle( $\beta$ ). The softmax activation function is applied to all hidden layers.

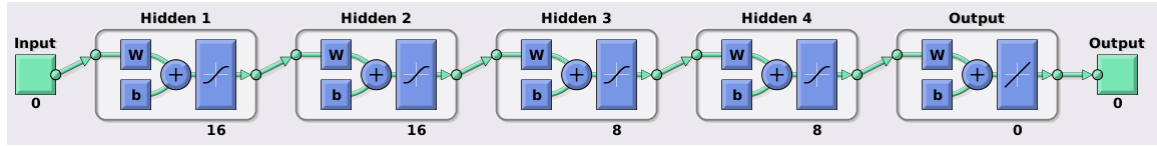
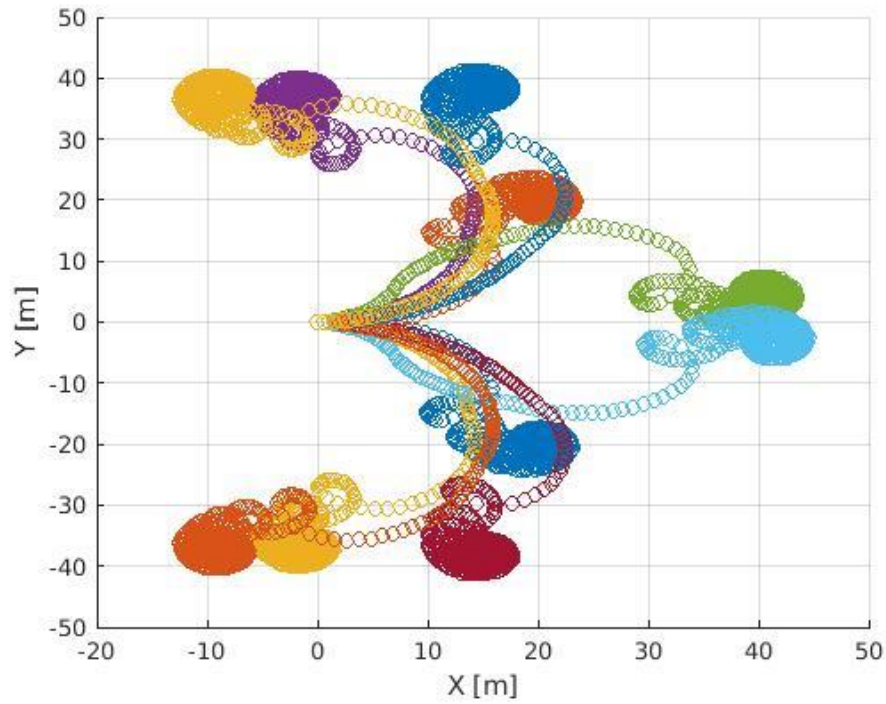


Figure 37 – Architecture of Deep Neural Network

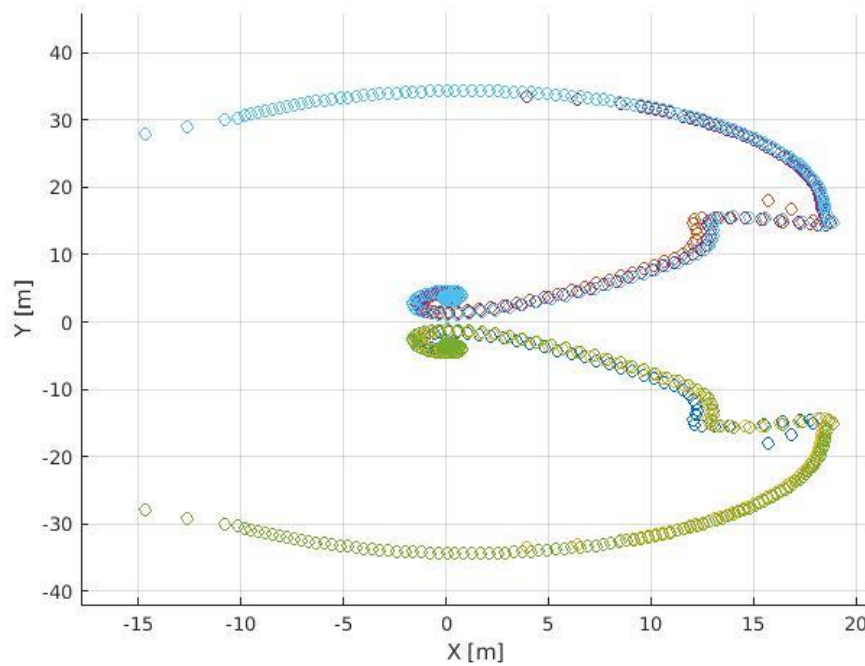
The neural network is activated at random times to conduct different scenarios in different areas. This will enable the RL agent to learn more general policies. The DNN is capable of stabilizing the vehicle around the preselected equilibria points mentioned in the section 3.5. The Figure 40, Figure 41, Figure 42, Figure 43, and Figure 44 demonstrate the simulation results acquired from the implementation of the DNN on drift control. The parameters of the target equilibriums are given in Table 4.

Table 4: Equilibrium points for development of RL-based controller

Parameter	Equilibrium Point-1 (Clockwise)	Equilibrium Point-2 (Counter Clockwise)
$\beta_{eq}$	0.6735 rad	-0.6735 rad
$r_{eq}$	1.436 rad/s	-1.436 rad/s
$V_{x_{eq}}$	4.15 m/s	4.15 m/s
$\delta_{eq}$	-0.12 rad	0.12 rad
$R_{eq}$	3.66 m	3.66 m



*Figure 38: Scenerios for Different Goal Points (Earth-fixed coordinate system)*



*Figure 39: Scenerios for Different Goal Points (Body-fixed coordinate system)*

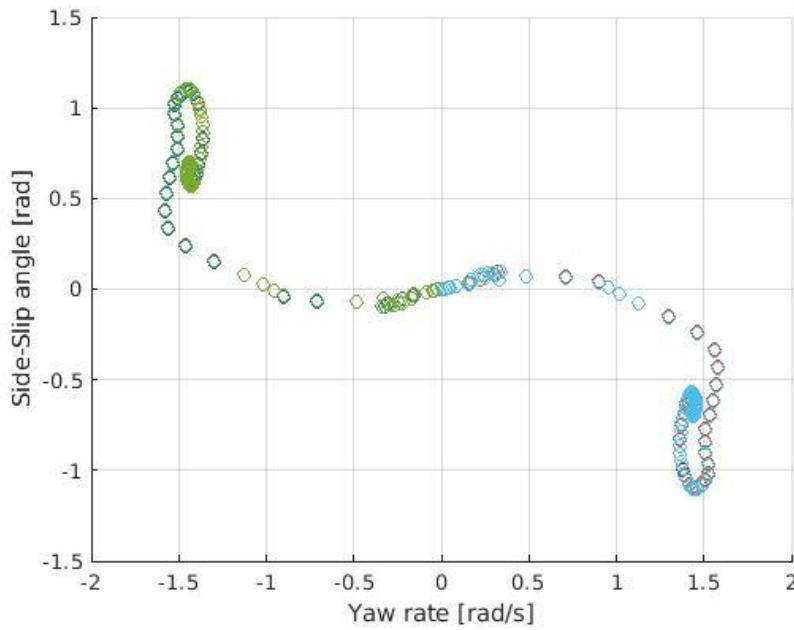


Figure 40: Side-slip angle ( $\beta$ ) versus Yaw Rate ( $r$ ) at

$U_x = 4.15 \text{ m/s}$

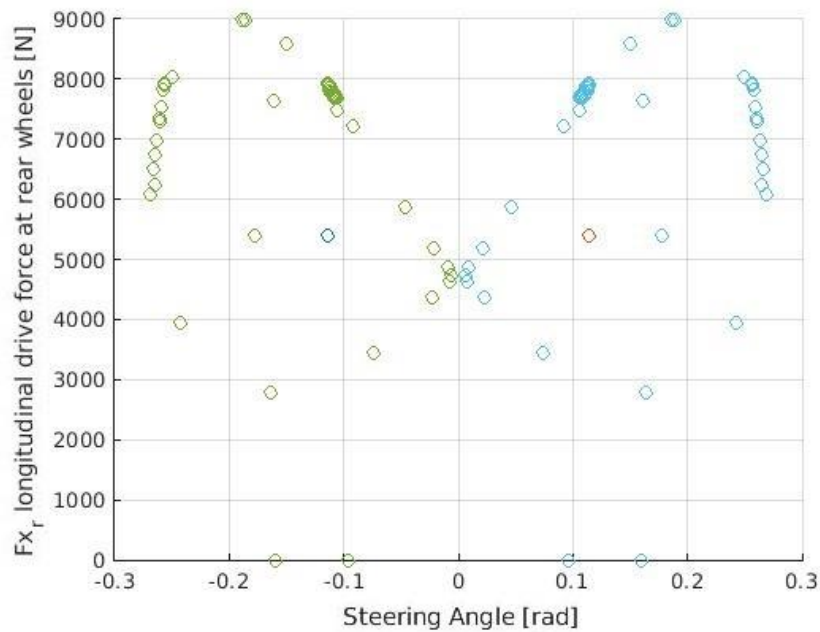


Figure 41: Longitudinal Drive Force at rear wheels ( $F_{x_r}$ )

versus Steering Angle ( $\delta$ ) at  $U_x = 4.15 \text{ m/s}$

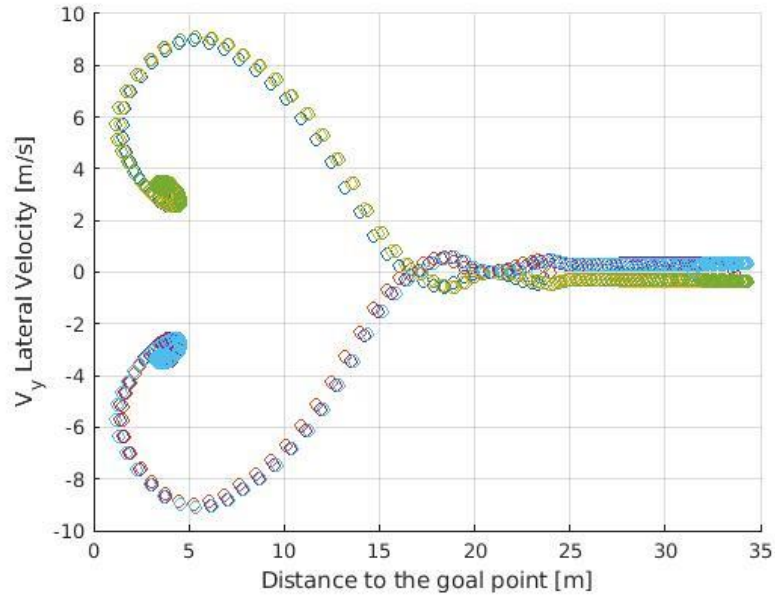


Figure 42: Lateral Velocity ( $V_y$ ) versus Distance to the goal point where the desired turning radius  $R_d = 3.66 \text{ m}$

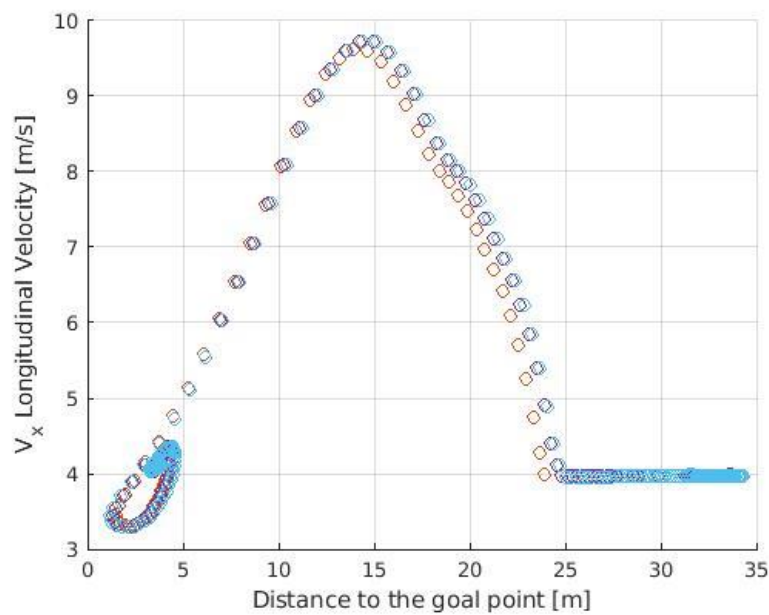
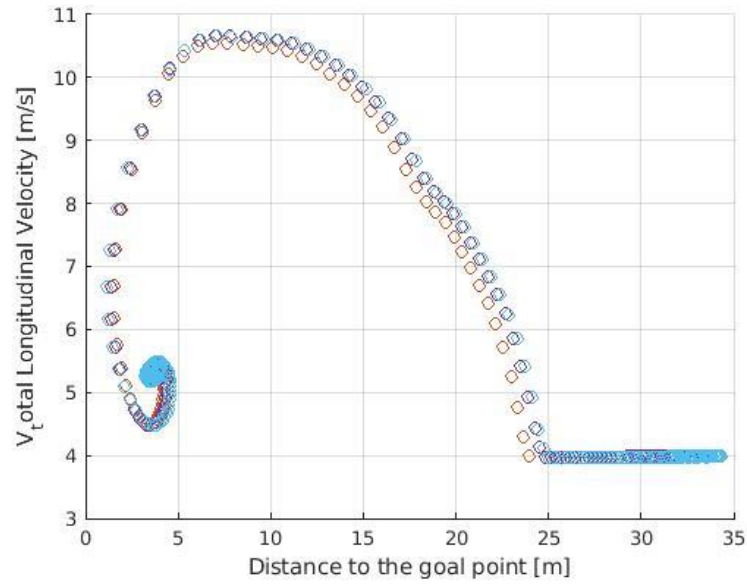


Figure 43: Longitudinal Velocity ( $V_x$ ) versus Distance to the goal point where the desired turning radius  $R_d = 3.66 \text{ m}$



*Figure 44: Total Velocity ( $V_{total}$ ) versus Distance to the goal point where the desired turning radius  $R_d = 3.66m$*

### 4.3. DEEP REINFORCEMENT LEARNING

One of my main concerns in developing a Transfer Learning framework for bootstrapping is to combine the strengths of Reinforcement Learning, Deep Learning, and the traditional control method. In this research, I use the Soft Actor-Critic (SAC) algorithm, which aims at maximizing the trade-off between the cumulative reward and the uncertainty of the policy. The initial policies already refined and strengthened by the DNN were transferred from DNN to the SAC algorithm through Transfer Learning.

A SAC agent's actor produces mean and standard deviation outputs. At the beginning, the actor picks an unrestricted action arbitrarily from a Gaussian distribution of state-action pairs. To calculate the entropy of the policy for a given state-space, the SAC agent utilizes an unbounded probability distribution. If the action-space of the SAC agent is restricted, *tanh* and *scaling* processes are applied to the restricted actions as shown in Figure 45 [32].



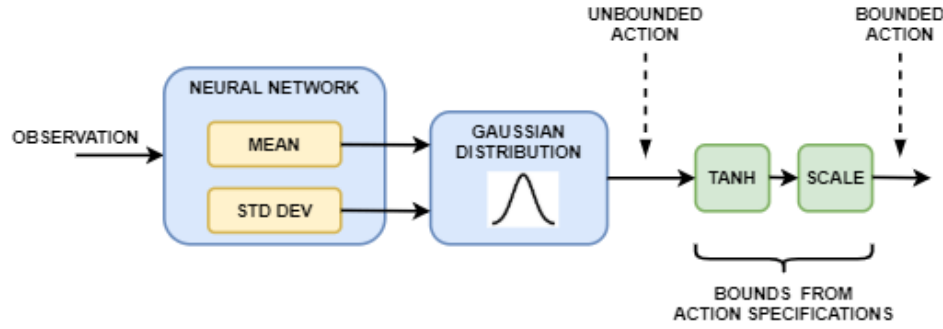


Figure 45: Learning mechanism of SAC agent [32]

The neural network with the same architecture of the below model was trained with the dataset generated by the DNN. The DNN model's first two outputs in the mean layer represent the steering and longitudinal drive force control signals. In the dataset, all labels for the last two outputs of the model were set to 1 so that the entropies of all action-state pairs become equal. This makes the critic part of the agent tune the actor part based on the reward function.

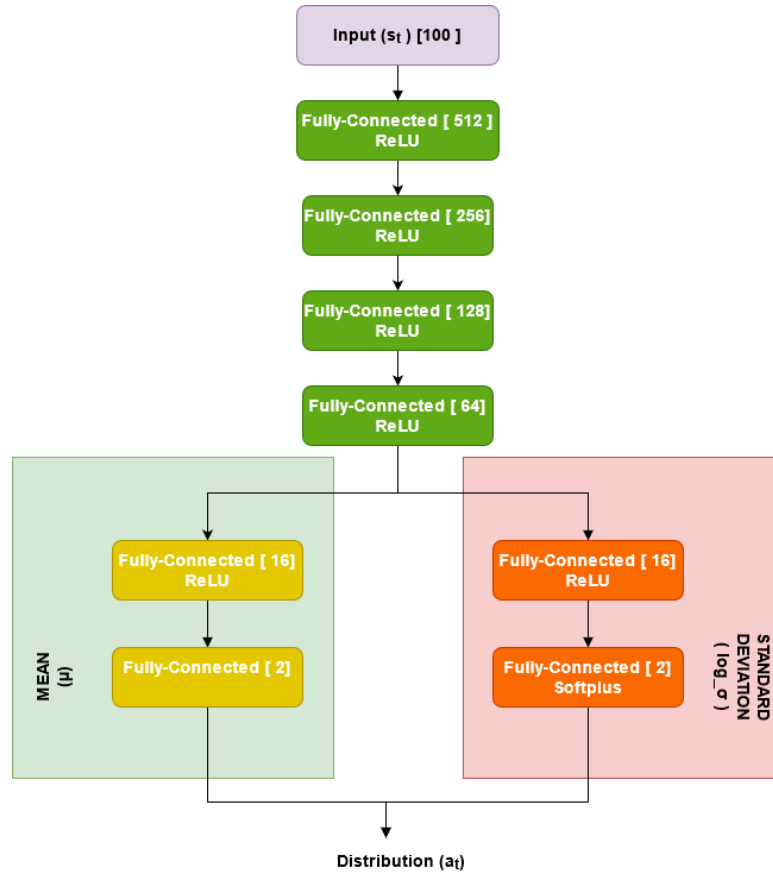


Figure 46: Structure of the SAC agent

### 4.3.1. STATE SPACE

The RL-agent must obtain all necessary data regarding the current state of the environment to complete its task successfully. The state space variables that have been decided to be significant for the autonomous drift control task is listed below:

#### State space variables:

- The last 7 waypoints which the vehicle followed are given as (X,Y) coordinates with respect to the body-fixed coordinate system,
- The last 7 turning center points are given as (X,Y) coordinates with respect to the body-fixed coordinate system,
- The last 7 steering and longitudinal drive force(normalized) control commands,
- The last 7 values of the desired heading angle's sine and cosine,
- The last 7 values of the side velocities (  $V_x$ ,  $V_y$  ),
- The last 7 values of the yaw rate and side-slip angle,
- Distance to the goal point ( $d$ ),
- Distance to the turning center ( $d_{CoR}$ )
- Turning radius ( $R$ ),
- Derivative of the distance to the goal point ( $\dot{d}$ ),
- Derivative of the distance to the goal point ( $\dot{d}_{CoR}$ ),
- Derivative of the longitudinal velocity ( $V_x$ ),
- Derivative of the lateral velocity ( $V_y$ ),
- Derivative of the yaw rate ( $\dot{r}$ ),
- Derivative of the side-slip angle ( $\dot{\beta}$ ),
- Error of the side-slip angle for two symmetric equilibria points (  $\beta_{eq1}$ ,  $\beta_{eq2}$  ),
- Error of the yaw rate for two symmetric equilibria points (  $r_{eq1}$ ,  $r_{eq2}$  ),
- Error of the total velocity (  $e_v$  ),
- Normalized value of the cumulative distance,
- Encoded angle (  $\psi_{enc}$  )

### 4.3.2. ACTION SPACE

The steering angle ( $\delta$ ) and longitudinal drive force ( $F_{x_r}$ ) are limited to the ranges of  $[-0.31, 0.31]$  and  $[0, 9000]$ . As [7] suggested, the smoothing block has been implemented and deployed into the bicycle model.

$$\mathcal{A} = \{ \delta, F_{x_r} \}$$

### 4.3.3. REWARD FUNCTION

I have designed various reward functions during my thesis. In this part, I will present the most effective one. The reward function consists of two components, which are  $r_{drift}$  and  $r_{route}$ . Additionally, the training process is interrupted if the vehicle turn left when the goal point located the vehicle's right at the beginning or vice versa. The reward function can be written as the following:

$$\beta_{eq1} = |\beta - 0.6735| \quad (4.1)$$

$$\beta_{eq2} = |\beta + 0.6735| \quad (4.2)$$

$$r_{eq1} = |\beta + 1.584| \quad (4.3)$$

$$r_{eq2} = |\beta - 1.584| \quad (4.4)$$

$$e_v = |V - 5.25| \quad (4.5)$$

$$r_{\text{drift}} = \begin{cases} \frac{2.68}{1 + |\beta_{eq1}|} * \frac{6.336}{1 + |r_{eq1}|} * \frac{6}{1 + |\dot{\beta}|^2} * \frac{5}{1 + |e_v|} & \text{if } r < 0 \\ \frac{2.68}{1 + |\beta_{eq2}|} * \frac{6.336}{1 + |r_{eq2}|} * \frac{6}{1 + |\dot{\beta}|^2} * \frac{5}{1 + |e_v|} & \text{otherwise} \end{cases} \quad (4.6)$$

$$r_{\text{route}} = |d - 3.66|^2 \quad (4.7)$$

$$r_{\text{punishment}} = -10000 \quad \text{if } \text{sign}(y_{t=0}) \neq \text{sign}(\delta) \quad (4.8)$$

The  $r_t$  becomes:

$$r_t = r_{\text{route}} + r_{\text{drift}} \quad (4.8)$$

## 5. EXPERIMENTS AND RESULTS

In this section, I discuss how the RL can be effectively applied to complicated vehicle motion control tasks. The RL-based controller can individually manipulate the throttle and steering angle in the range of  $[0 \text{ N}, 9000 \text{ N}]$  and  $[-0.31 \text{ rad}, 0.31 \text{ rad}]$  respectively. The results of the evaluations in the simulator are presented in Section 5.3

### 5.1. ACTIONS TO BE LEARN

To perform drifting maneuver, the vehicle must reach high values of side-slip angle while turning. However, it must not exceed certain limits since it can increase the danger of vehicle instability which can result in rollover.

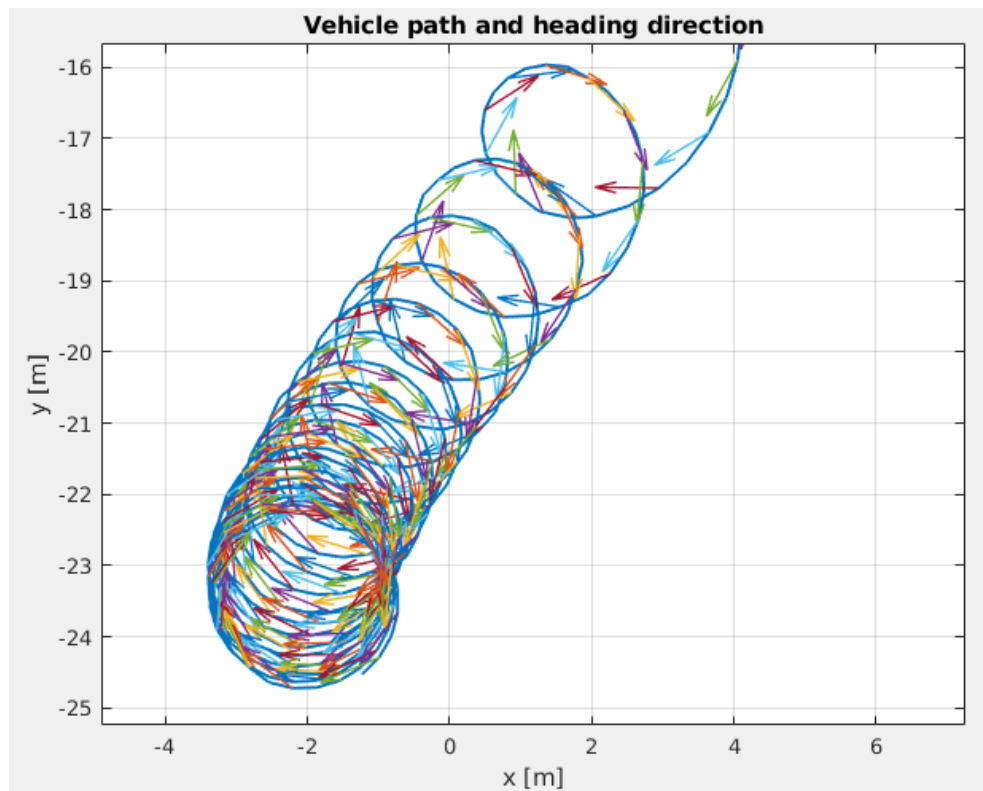


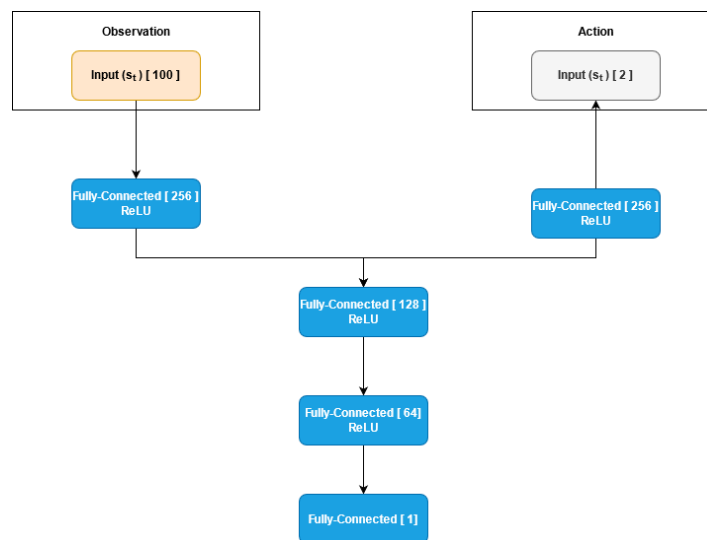
Figure 47 – Trajectory of drifting vehicle

## 5.2. IMPLEMENTATION

The SAC agent shown in Figure 46 has been implemented in this work. The structure of the critic network is shown in . The hyperparameters used in the training of the SAC agent are given in Table 5:

*Table 5: SAC Hyperparameters*

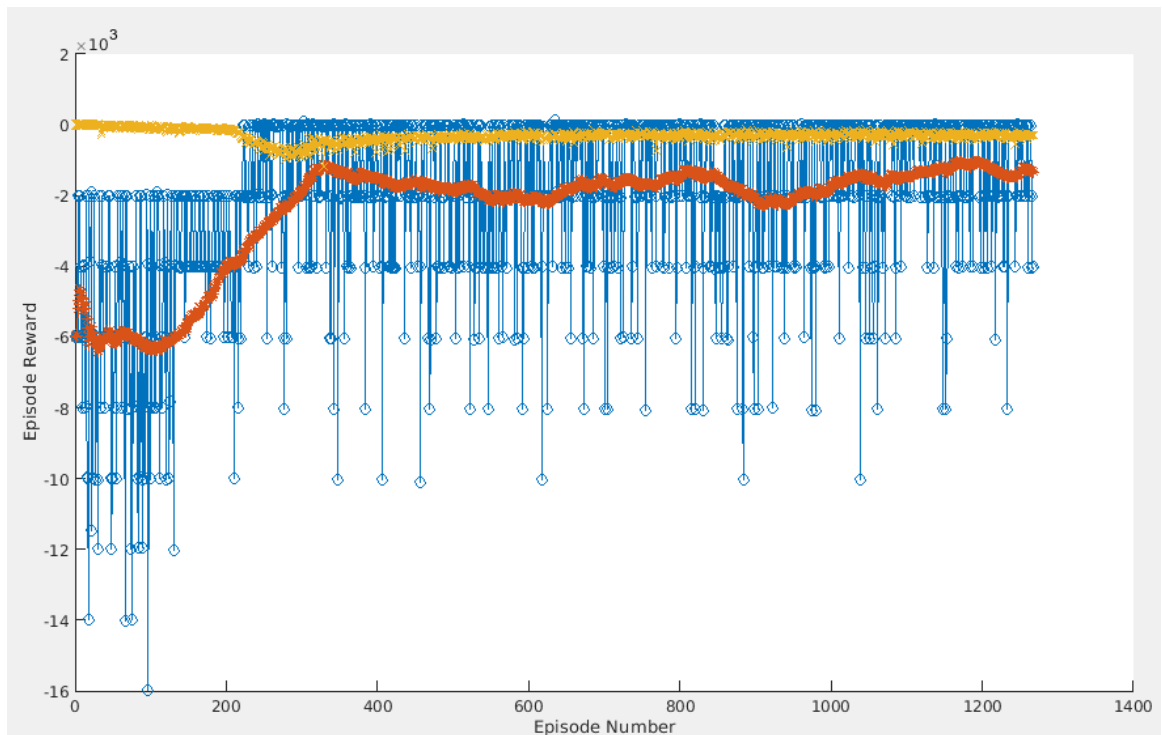
Hyperparameters	Value
Action Space / Size	Continuous / 2
State Space	100
Actor Learning Rate	0.001
Critic Learning Rate	0.005
Max Episodes	5000
TargetSmoothFactor	0.001
MiniBatchSize	512
NumWarmStartSteps	512
ExperienceBufferLength	1e6
DiscountFactor	0.99



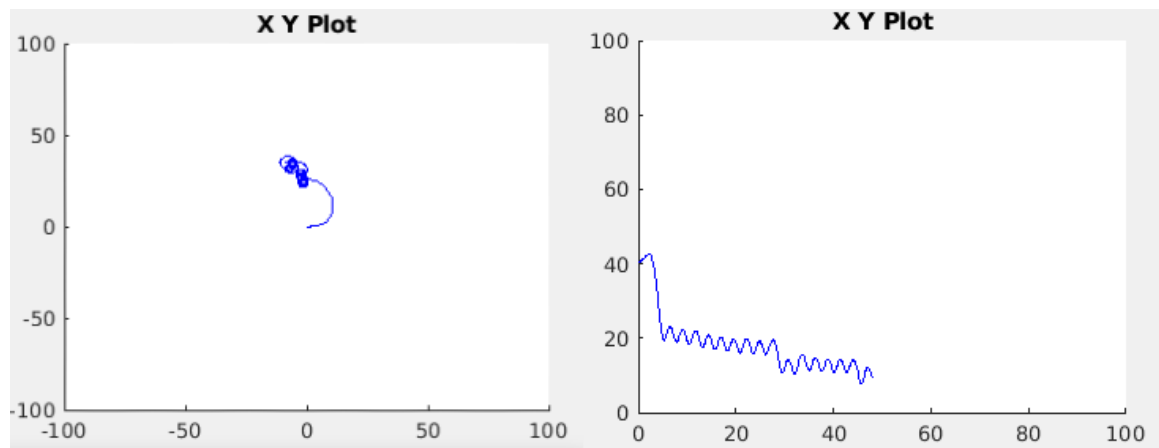
*Figure 48: Structure of the Critic Network*

### 5.3. RESULTS

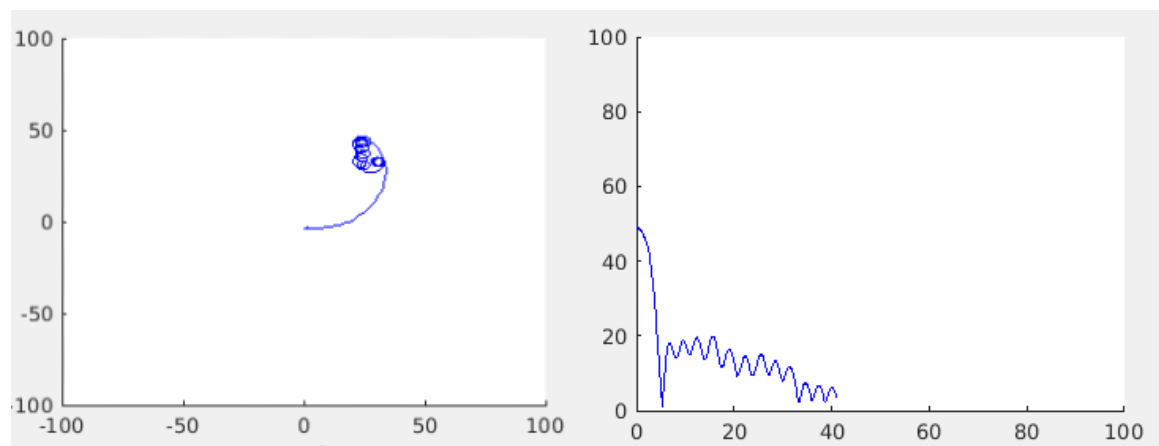
As depicted in Figure 49, the SAC agent has trained for 1300 episodes. In the starting phase of the training, it tends to get punishment reward. After 400 episodes, the agent successfully fulfill both point-goal navigation and drifting tasks.



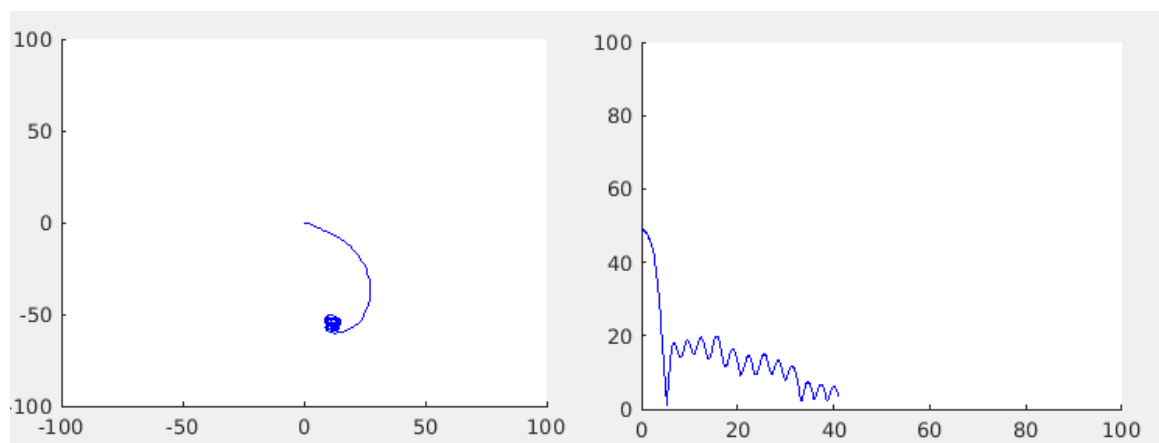
**Figure 49 – Records of rewards**



**Figure 50 – Vehicle Trajectory versus Distance to goal point**



**Figure 51 – Vehicle Trajectory versus Distance to goal point**



**Figure 52 – Vehicle Trajectory versus Distance to goal point**



## 6. CONCLUSION

In this thesis, deep reinforcement learning algorithms have been used to develop a controller that is able to execute the drifting maneuver around a randomly selected target point. Some of the key problems which I have dealt with during the development phase are: (i) sample efficiency, (ii) generalization of learned policies.

In the first phase of this research, I built up a simulation environment involving a kinematic bicycle vehicle model which is consistent with the physical characteristic of the model car BMW M2 Competition. This environment is used to generate dataset for supervised learning and then to conduct the training process of deep reinforcement learning algorithms. In this environment, instead of earth-fixed coordinate system, I used body-fixed coordinate system and hence the agent could efficiently learn the generalized (near) optimal policies.

Since reinforcement learning might require many samples to learn useful policies due to the complexity of the high-dimensional environment, I developed transfer learning framework to refine useful policies and to transfer gained knowledge from traditional controller and supervised learning to the reinforcement learning side. Thus, it can be achievable to increase the scalability of the reinforcement learning-based agents with fewer samples and to achieve (near) optimal policies by bringing the advantages of those algorithms and methods together.

## 7. REFERENCES

- [1] N. U.S. Department of Transportation, "Traffic Safety Facts," Washington DC, USA, 2020.
- [2] N. U.S. Department of Transportation, "Federal Motor Vehicle Safety Standards; Electronic Stability Control Systems for Heavy Vehicles," Washington DC, USA, 2012.
- [3] R. Sferco, Y. Page, J.-Y. Le Coz and P. A. Fay, "Potential Effectiveness of Electronic Stability Program (ESP) - What European Field Studies Tell Us," pp. 2-6, 2001.
- [4] I. Zubov, I. Afanasyev, A. Gabdullin, R. Mustafin and I. Shimchik, "Autonomous Drifting Control in 3D Car Racing Simulator," pp. 1-2, 2018.
- [5] C. Voser, R. Hindiyeh and G. J., "Analysis and control of high sideslip manoeuvres," p. 1, 2010.
- [6] J. Kober, J. Bagnell and J. Peters, "Reinforcement Learning in Robotics: A Survey," *The International Journal of Robotics Research*, p. 1, 2013.
- [7] P. Cai, X. Mei, T. Lei, S. Yuxiang and M. Liu, "High-Speed Autonomous Drifting With Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, p. 1, 2020.
- [8] E. Velenis, D. Katourakis, E. Frazzoli, P. Tsiotras and R. Happee, "Stabilization of steady-state drifting for a RWD vehicle," 2010.
- [9] M. Cutler and J. P. How, "Autonomous Drifting using Simulation-Aided Reinforcement Learning," *IEEE International Conference on Robotics and Automation*, p. 1, 2016.
- [10] Á. Bárdos, Á. Domina, Z. Szalay, V. Tihanyi and L. Palkovics, "MIMO controller design for stabilizing vehicle," pp. 1-6, 2019.
- [11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2018.
- [12] M. Sewak, *Deep Reinforcement Learning : Frontiers of Artificial Intelligence*, Springer, 2019.
- [13] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," pp. 1-3, 2016.
- [14] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel and S. Levine, "Soft Actor-Critic Algorithms and Applications," pp. 1-2, 2018.
- [15] M. Deisenroth and C. Rasmussen, "PILCO: A Model-Based and Data-Efficient Approach to Policy Search," pp. 1-2, 2011.
- [16] C. E. Beal, "APPLICATIONS OF MODEL PREDICTIVE CONTROL TO VEHICLE DYNAMICS FOR ACTIVE SAFETY AND STABILITY," *PhD Thesis, Stanford University*, 2011.
- [17] G. R., "Applying Deep Reinforcement Learning," *Master Thesis, Universität Hamburg*, 2019.
- [18] S. Vatsa, "VEHICLE DYNAMICS MODELING FOR AUTONOMOUS DRIFTING," *Master's Thesis, Michigan Technological University*, 2020.
- [19] R. Y. Hindiyeh, "Dynamics and Control of Drifting in Automobiles," *Ph.D. dissertation, Stanford University*, 2013.
- [20] R. N. Jazar, *Advanced vehicle dynamics*, Springer, 2019.
- [21] Michelin, "The Tyre, Grip," *Société de Technologie Michelin: Clermont-Ferrand, France*, 2001.
- [22] M. Robinson. [Online]. Available: <https://50to70.com/2018/07/31/worship-bmw-m2-competition/>.
- [23] "BMW AG," 2021. [Online]. Available: <https://www.bmw.com.ph/en/all-models/m-series/m2-coupe/2019/bmw-2-series-coupe-m-automobiles-technical-data.html>.
- [24] J. Y. Goh, T. Goel and C. Gerdes, "A Controller for Automated Drifting Along Complex Trajectories," *AVEC'18*, pp. 1-5, 2018.
- [25] K. E. Bekris and A. A. Argyros, "Angle-Based Methods for Mobile Robot Navigation:," *IEEE-International Conference on Robotics & Automation*, 2004.
- [26] S. Datta, O. Maksymets, J. Hoffman, S. Lee, D. Batra and D. Parikh, "Integrating Egocentric Localization for More," pp. 1-3, 2020.
- [27] "ETSI," 12 2019. [Online]. Available: [https://www.etsi.org/deliver/etsi\\_tr/103500\\_103599/103562/02.01.01\\_60/tr\\_103562v020101p.pdf](https://www.etsi.org/deliver/etsi_tr/103500_103599/103562/02.01.01_60/tr_103562v020101p.pdf).
- [28] Z. Zhu, K. Lin and J. Zhou, "Transfer Learning in Deep Reinforcement Learning: A Survey," p. 1, 2021.

- [29] D. Rastogi, "Delft Center for Systems and Control Deep Reinforcement Learning for Bipedal Robots," *Master's Thesis, Delft University of Technology*, 2017.
- [30] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, p. 1345–1359, 2009.
- [31] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains: A Survey," *Journal of Machine Learning Research 10 (2009)*, pp. 1-2, 2009.
- [32] "Mathworks," 2021. [Online]. Available: <https://www.mathworks.com/help/reinforcement-learning/ref/rlsacagent.html>.
- [33] F. Conte, "EXPANDING THE BRUSH TIRE MODEL FOR ENERGY STUDIES," *Master's thesis, KTH Royal Institute of Technology*, 2014.

## 8. TABLE OF FIGURES

Figure 1: Action of ESC understeer and oversteer conditions [3] .....	7
Figure 2: Four-Wheel-Drive (4WD) Model.....	8
Figure 3 : Representation of agent-environment interaction in a Markov Decision Process.....	12
Figure 4 : Exploration-Exploitation dilemma .....	13
Figure 5: Actor-Critic Architecture : According to critic's estimations, the actor updates the policy.....	16
Figure 6: Body-Fixed Coordinate System [16] .....	19
Figure 7: Bicycle Model [19] .....	20
Figure 8: The Parts of Brush Tire Model [19] .....	22
Figure 9: Diagram of the Friction Circle [20] .....	23
Figure 10: BMW M2 Competition [20] .....	24
Figure 11: Wheelbase of Model Car [21] .....	25
Figure 12: Three-state Dynamic Bicycle Model .....	26
Figure 13: The General Structure of the Dynamic Bicycle Model .....	27
Figure 14: Structure of the block "Tire Slip Calculation" .....	28
Figure 15: Structure of the block "Brush Tire Model – Front Wheel" .....	28
Figure 16: Structure of the block "Brush Tire Model – Rear Wheel" .....	29
Figure 17: Structure of the block "Side-slip Angle" .....	29
Figure 18: Structure of the block "Longitudinal Velocity" .....	30
Figure 19: Structure of the block "Yaw-Rate" .....	30
Figure 20: Heading angle .....	31
Figure 21: Lateral velocity .....	31
Figure 22: Calculation for the exact vehicle position .....	32
Figure 23: Structure of the block "Turning Radius" .....	33
Figure 24: The vehicle's instantaneous center of rotation .....	34
Figure 25: Structure of the block "Center of Rotation" .....	34
Figure 26: Point-goal navigation [27] .....	36
Figure 27: Structure of the block "Heading-Encoding" .....	37
Figure 28: General Structure of the Transfer Learning Framework designed for this research .....	38
Figure 29 – Linear Quadratic Drift Controller.....	39
Figure 30: Drifting Equilibria ( Side-slip vs. Road-wheel angle at $V_x = 10$ m/s) [10] .....	40
Figure 31: Drifting Equilibria ( Yaw rate vs. Steering .....	40
Figure 32: Clockwise drifting at $\beta = 0.65$ rad and .....	41
Figure 33: The LQR controller stabilizes the vehicle drifting at $\beta = 0.65$ [10] .....	41
Figure 34: Counter clockwise drifting at $\beta = -0.7$ rad and .....	42
Figure 35: The LQR controller stabilizes the vehicle drifting at $\beta = -0.7$ [10] .....	42
Figure 36: The LQR controller stabilizes vehicle's longitudinal velocity [10] .....	43
Figure 37 – Architecture of Deep Neural Network.....	44
Figure 38: Scenarios for Different Goal Points (Earth-fixed coordinate system).....	45
Figure 39: Scenarios for Different Goal Points (Body-fixed coordinate system).....	45
Figure 40: Side-slip angle ( $\beta$ ) versus Yaw Rate ( $r$ ) at.....	46
Figure 41: Longitudinal Drive Force at rear wheels ( $F_{x_r}$ ) versus Steering Angle ( $\delta$ ) at $U_x = 4.15$ m/s.....	46

Figure 42: Lateral Velocity ( $V_y$ ) versus Distance to the goal point where the desired turning radius $R_d = 3.66$ m .....	47
Figure 43: Longitudinal Velocity ( $V_x$ ) versus Distance to the goal point where the desired turning radius $R_d = 3.66$ m .....	47
Figure 44: Total Velocity ( $V_{total}$ ) versus Distance to the goal point where the desired turning radius $R_d = 3.66$ m .....	48
Figure 45: Learning mechanism of SAC agent [32] .....	49
Figure 46: Structure of the SAC agent .....	49
Figure 47 – Trajectory of drifting vehicle .....	53
Figure 48: Structure of the Critic Network .....	54
Figure 49 – Records of rewards .....	55
Figure 50 – Vehicle Trajectory versus Distance to goal point .....	56
Figure 51 – Vehicle Trajectory versus Distance to goal point .....	56
Figure 52 – Vehicle Trajectory versus Distance to goal point .....	56

## 9. LIST OF TABLES

Table 1- Road Friction Coefficient Range [22] .....	24
Table 2 – Wheel Loads .....	25
Table 3 - Vehicle Parameters .....	27
Table 4: Equilibrium points for development of RL-based controller .....	44
Table 5: SAC Hyperparameters .....	54

## 10. LIST OF ABBREVIATIONS AND SYMBOLS

NHTSA	-	National Highway Transportation Safety Administration
ESC	-	Electronic Stability Control
ESP	-	Electronic Stability Program
4WD	-	4 Wheel Vehicle
LQR	-	Linear Quadratic Regulator
MIMO		Multiple Input, Multiple Output
RWD	-	Rear-Wheel-Drive
SAC	-	Soft Actor-Critic
MDP	-	Markov Decision Process
A3A		Asynchronous Advantage Actor-Critic

