**UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS**

**Facultad de Ingeniería**


UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS
Acreditación Institucional de Alta Calidad

# Workshop 2

*Dynamical Systems Analysis & Design*

**C**omputer Systems Engineering Program

**Professor:** Carlos Andres Sierra

Kevin Emmanuel Tovar Lizarazo - 20221020068

Jhojan Stiven Aragon Ramirez - 20221020060

may 9, 2025

# Índice

# 1. System Dynamics Analysis:

## 1.1. Mathematical/Simulation Model: Extended Kinematic Bicycle Model

We represent the vehicle by a single track ("bicycle") model, including a slip angle $\beta$. The state is

$$\mathbf{x} = \big(x,\ y,\ \psi,\ v,\ \beta\big),$$

where

- $x, y$ are the Cartesian coordinates of the vehicle's center of mass,
- $\psi$ is the heading (orientation) angle,
- $v$ is the forward speed,
- $\beta$ is the slip (or side–slip) angle at the center of mass.

The control inputs are

$$u = \big(a,\ b\big),$$

where

- $a$ is the longitudinal acceleration,
- $b$ is the rate of change of the slip angle $\beta$ (i.e. how fast the steering effect changes).

The model equations are

$$\dot{x} = v\ \cos\big(\psi + \beta\big), \tag{1}$$
$$\dot{y} = v\ \sin\big(\psi + \beta\big), \tag{2}$$
$$\dot{\psi} = \frac{v}{l_r}\ \sin(\beta), \tag{3}$$
$$\dot{v} = a, \tag{4}$$
$$\dot{\beta} = b. \tag{5}$$

Here $l_r$ is the distance from the rear wheel to the center of mass. Equations (1)–(2) say that the vehicle moves in the direction of its heading plus the slip angle. Equation (3) shows how the heading changes due to slip, and the last two are simply the kinematics of speed and slip as driven by our inputs.

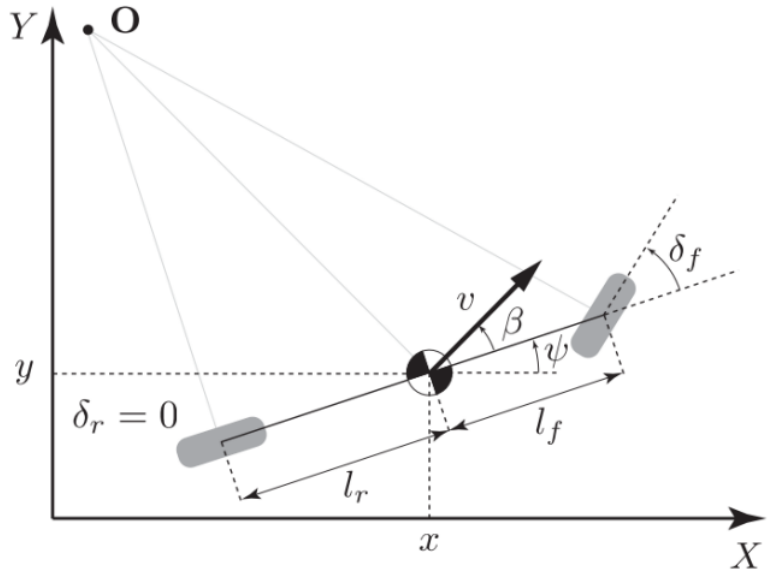



Figura 1: Top-down schematic of the extended kinematic bicycle model. The wheelbase is split into $l_f$ (front) and $l_r$ (rear), and the slip angle $\beta$ at the center of mass is shown. Adapted from Estrada et al., 2021.

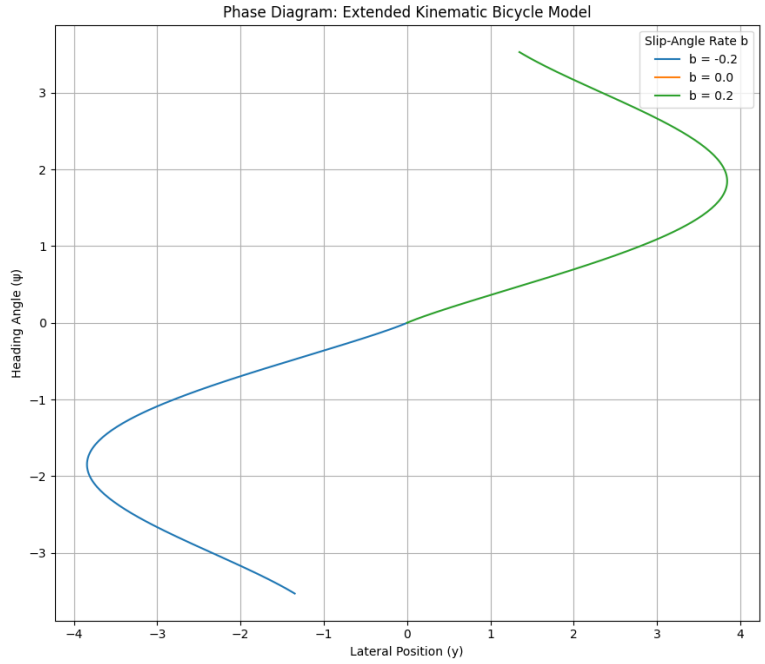
## 1.2. Phase Portraits or Diagrams:

The phase diagram shows how the vehicle's lateral position $y$ and heading angle $\psi$ evolve over time under different steering slip-angle rates $b$. Each curve starts from the origin:

$$x = (x = 0,\ y = 0,\ \psi = 0,\ v = 1,\ \beta = 0),$$

with constant forward speed $v = 1$ and no longitudinal acceleration ($a = 0$).

- **Horizontal axis ($y$)**: lateral displacement of the vehicle's center of mass.
- **Vertical axis ($\psi$)**: heading angle (orientation) of the vehicle.
- **Different curves**: each uses a different constant input $b = \dot{\beta}$, so the slip angle $\beta$ changes at different rates.
- When $b > 0$, $\beta$ increases, the vehicle turns right and $\psi$ grows as $y$ moves to positive values.
- When $b < 0$, $\beta$ decreases, the vehicle turns left and $\psi$ decreases as $y$ moves to negative values.




- When $b = 0$, the slip angle is constant and the vehicle goes straight ($\psi = 0$, $y = 0$).

This simple phase portrait illustrates how different steering inputs $b$ cause distinct trajectories in the $(y, \psi)$ state space of the extended bicycle model.

# 2. Feedback Loop Refinement:

## 2.1. Enhanced Control Mechanisms:

Finally, an intelligent episode termination criterion is recommended: if the expected return drops significantly over several consecutive steps, the environment may reset early to prevent inefficient rollouts in non-promising trajectories. All these refinements are derived directly from the feedback loop structure described in the original document and aim to enhance self-regulation by making the state-action-reward-policy cycle more responsive and precise.

To refine the feedback loop in this project(autonomous driving system), we propose mechanisms to improve the reinforcement learning efficiency and the agent's self-regulation capabilities.

First, we will implement a technique of reward shaping adaptation, this allowing the reward function to dynamically adjust based on recent performance metric-such as penalizing increased oscillation or granting higher reward for maintaining a stable trajectory. This is based on the criteria of define previews rewards, such as, progress along the track, off-road penalties, collisions, and unnecessary braking, as described in the functional specifications. besides, we will propose to increase by doubling the number of LiDAR rays (from 14 to 28) and incorporate additional data like drift angle or a Z-axis accelerometer, enabling the agent to gain a richer perception of it's environment and make more informed decisions.

To avoid then all sensor have the same weigh in all moments, and attention-based neural module is proposed dynamically assigning weights to each sensory input depending on the driving context, thereby focusing the learning on the most relevant information.And the same way, for stabilize the penalty signals and avoid abrupt shifts in policy, a PID(Proportional-Integral-Derivative) controller can be applied over the error signals-especially effective for tuning penalties related to crashes of off-track deviations.

Finally, we recommended incorporate the intelligent episode termination criterion: if the expected return drops significantly over several consecutive steps, the environment may reset early to prevent inefficient roll outs in non-promising trajectories. all this propose then we have are derived from the feedback loop structure described in the workshop 1 and aim to enhance self-regulation by making the state-action-reward-policy cycle more responsive and precise.

## 2.2. Stability and Convergence:

To ensure the autonomous driving agent behaves in a stable and convergent way, we define the following criteria:

# English

One could check stability and convergence both by analysis and by experiments to make sure the self–driving agent's behavior stays within limits during long training and eventually follows a repeatable policy in the `CarRacing-v0` environment. For example, one could view the discounted Bellman update as a contraction ($\gamma < 1$) and show that the maximum error

$$\|Q_{k+1} - Q^*\|_\infty$$

gets smaller by at least a factor of $\gamma$ each time. Another way is to define a Lyapunov–style function $V(s) \geq 0$ whose expected value drops at every step, so the agent's

paths cannot run away.

In practice, these ideas should be supported by simple checks:

- Plot the average return over 100 episodes and say it has converged when the slope stays below 1 % for several windows.
- Watch the policy entropy level off, showing exploration has settled.
- Check the variance of the TD error until it falls under a chosen threshold across different seeds.

Clear stopping rules help make this process clear. For example, one could stop training when:

- the average return is over 700 for twenty episodes in a row,
- the collision rate stays below 2 %,
- an Augmented Dickey–Fuller test on the reward series rejects a unit–root hypothesis ($p < 0{,}05$).

# 3. Iterative Design Outline:

## 3.1. Proposed Enhancements to Project Architecture:

In workshop 1 we already established in section 3.1 selection of frameworks for reinforcement learning and the staged roadmap presented in Table 1 Timeline for progressing from basic Q-learning to advanced DQN techniques, and the project plan should be refined along three complementary dimensions.

- **First**: For the data structures required a evolution from a tabular Q-table to a Deep Q-Network (DQN) supported by a prioritized experience-replay buffer and a periodically updated target network; these mechanisms are recognized in the specialized literature for mitigating sample-inefficiency and stabilizing temporal-difference updates, and they align directly with the "DQN extensions" foreseen for Week 7 on wards, these allows the agent to approximate the action-value function as a smooth mapping in parameter space, thus generalising to unseen states and exploiting correlations across sensor features.
- **Second**: we need to increase by sequencing the introduction of Double DQN, Dueling DQN, and ultimately Actor-Critic methods such as Proximal Policy Optimization (PPO), thereby addressing over-estimation bias, refining value–advantage decomposition, and enabling smooth continuous-action control.

- **Third**: we gonna extends the software stack: while Gymnasium and Stable-Baselines3 remain the principle core, the adoption of PyTorch Lightning will facilitate modular checkpoint management, and Ray RLlib will provide the distributed-training capabilities required for large-scale hyper-parameter sweeps. These modifications make more coherent the objective then we need find, evidence-based extension of the existing schedule and equip the agent with the data handling, algorithmic nuance, and infrastructural support necessary for advanced dynamic behaviour.

## 3.2. Experimental Validation Strategy:

The dynamic extensions then we have proposed will be validate through a protocol of three layer.

- **First**: parameter sweeps will be carried out on the CarRacing-v0 environment to examine the agent's sensitivity to key hyper parameters such as learning rate, replay-buffer capacity and target-network update interval.
- **Second**: controlled randomness will be ensured by repeating every experiment under ten distinct random seeds, averaging performance metrics (mean episodic return and standard deviation) across seeds will expose variance attributable solely to stochastic effects and thus provide statistically reliable comparisons.
- **Third**: scenario variation will be introduced by evaluating each trained policy on three procedurally generated track profiles of increasing difficulty—Standard, Chicane-Heavy and Obstacle-Rich—thereby testing generalisation beyond the original training distribution. Together, these layers satisfy the methodological rigour implied by the Week 7+ in the workshop "DQN extensions" milestone of the current roadmap and yield a replicable basis for assessing advanced dynamic behaviours.

# Referencias

Estrada, M., Li, S., & Cai, X. (2021). Feedback Linearization of Car Dynamics for Racing via Reinforcement Learning. *arXiv preprint arXiv:2110.10441*. https://arxiv.org/abs/2110.10441