



# Insulin Simulator: A Comparison of RL Agents vs. PID Controller

## A Patient-Safety-First Approach for Type 1 Diabetes Management

---

Stefano Lusardi   Elena Ruiz de la Cuesta Castaño   Arahí Fernández Monagas

June 30, 2025

University of Trieste

# Outline

1. Introduction
2. Methodology
3. Results & Conclusion

# Introduction

---

# The Problem & Motivation

## The Challenge: Improving Quality of Life in Type 1 Diabetes

The primary goal is to automate insulin delivery to consistently keep glucose levels within a healthy, non-critical range.

- This project simulates food intake and physiological responses to develop a more intelligent and personalized control system.
- Traditional control systems can be imprecise and often introduce significant risks, especially hypoglycemia.

# Methodology

---

# Project Summary: RL-Based Insulin Control

## System Description

The system operates in discrete 1-minute cycles:

1. **Input:** A meal's carbohydrate level is translated by the `simglucose` library, into a glucose level,  $G(t)$  taking into account the initial glucose level.

# Project Summary: RL-Based Insulin Control

## System Description

The system operates in discrete 1-minute cycles:

1. **Input:** A meal's carbohydrate level is translated by the `simglucose` library, into a glucose level,  $G(t)$  taking into account the initial glucose level.
2. **Action:** The RL agent observes  $G(t)$  and administers an insulin dose.

# Project Summary: RL-Based Insulin Control

## System Description

The system operates in discrete 1-minute cycles:

1. **Input:** A meal's carbohydrate level is translated by the `simglucose` library, into a glucose level,  $G(t)$  taking into account the initial glucose level.
2. **Action:** The RL agent observes  $G(t)$  and administers an insulin dose.
3. **Environment Update:** `simglucose` calculates the new glucose level,  $G(t+1)$ , based on the meal and insulin dose.



# Project Summary: RL-Based Insulin Control

## System Description

The system operates in discrete 1-minute cycles:

1. **Input:** A meal's carbohydrate level is translated by the `simglucose` library, into a glucose level,  $G(t)$  taking into account the initial glucose level.
2. **Action:** The RL agent observes  $G(t)$  and administers an insulin dose.
3. **Environment Update:** `simglucose` calculates the new glucose level,  $G(t+1)$ , based on the meal and insulin dose.
4. **Reward:** The agent is rewarded for staying in the healthy range and penalized for hypoglycemia and hyperglycemia.

# The Simulation Engine: `singlucose`

## What is `singlucose`?

It is a validated, open-source simulator that models the complex dynamics of glucose metabolism in the human body. It serves as the main engine of our simulation environment.

# The Simulation Engine: `simglucose`

## What is `simglucose`?

It is a validated, open-source simulator that models the complex dynamics of glucose metabolism in the human body. It serves as the main engine of our simulation environment.

## Core Components Provided

- **T1DPatient:** Simulates the patient's unique physiology and their body's reaction to insulin and carbohydrates.
- **CGMSensor:** Simulates the continuous glucose monitor (CGM) that provides observations to our agent.
- **InsulinPump:** Simulates the device that administers the insulin doses chosen by our agent.

## Our setting with `simglucose`

- We will focus on a specific patient, `adolescent001`  
e.g. the way he reacts to glucose and insulin
- Meals are randomized in carbs content and time.
- Noise is introduced in  $G(t)$ .

# State Discretization

- 4 glucose levels (more important)
- 3 levels for glucose velocity to be added in the overall state computation.

Bigger positive velocity implies a bigger contribution; negative velocity does not contribute.

# State Discretization

- 4 glucose levels (more important)
- 3 levels for glucose velocity to be added in the overall state computation.

Bigger positive velocity implies a bigger contribution; negative velocity does not contribute.

## Total levels

$$total_{levels} = glucose_{level} * 3 + velocity_{level}$$

# The Learning Process (Reinforcement Learning)

1. **Observation:** The agent sees the current glucose level,  $G(t)$ , and its velocity.
2. **Action:** It chooses an insulin dose from a discrete set of options.
3. **Reward Function:**
  - A Gaussian function rewards values near the target (115 mg/dL).
  - A **heavy penalty** is applied near hypoglycemia ( $<70$  mg/dL) to prioritize safety.

The agent's goal is to maximize its cumulative reward through trial and error.

# What Type of RL

We consider the case of observable phenomenon without knowing the model → **TD-learning**



## Reinforcement Learning Agents Critic Only

- Q-learning TD(0)
- Q-learning TD(1)
- SARSA
- Expected SARSA

# Models Evaluated

## Reinforcement Learning Agents

### Critic Only

- Q-learning TD(0)
- Q-learning TD(1)
- SARSA
- Expected SARSA

### Actor Only

- Reinforce

# Models Evaluated

## Reinforcement Learning Agents

### Critic Only

- Q-learning TD(0)
- Q-learning TD(1)
- SARSA
- Expected SARSA

### Actor Only

- Reinforce

## Baseline Controller

- PID (Proportional-Integral-Derivative) Controller

## What We Added

- **Safety rule:** below 120 mg/dL glucose level, insulin is not given.
- **Forced waiting:** if there has just been a shot, there will not be a new shot.

# General Structure of Critic-Based RL Algorithms

---

**Algorithm 1** Critic-Based RL Algorithms

---

**Initialize:**  $Q_0$

**Set:**  $\gamma, \alpha, \varepsilon$

**for** each episode **do**

**Initialize state:**  $s$

**loop**

        Derive  $\pi$  from  $Q_k$  ( $\varepsilon$ -greedy)

        Select action:  $a \sim \pi(\cdot | s)$

        Take action  $a$ , observe:

            - reward  $r$

            - next state  $s'$

        (Optional) select next action  $a'$

        ▷ only for SARSA

        Compute TD error:  $\delta$

        ▷ depends on algorithm

**Update:**

$$Q_{k+1} = Q_k + \alpha \cdot \delta$$

**end loop**

**end for**

---

# RL Agent: Q-learning TD(0) (Single-Step Update)

## Concept

Updates the value of applying an insulin dose  $I_t$  given the glucose state  $s_t$ . It learns using the maximum (most optimal) value of the next state.

## Update Formula

$$Q(s_t, I_t) \leftarrow Q(s_t, I_t) + \alpha [R_{t+1} + \gamma \max_{i'} Q(s_{t+1}, i') - Q(s_t, I_t)]$$

# RL Agent: Q-learning TD(1) (Monte Carlo Update)

## Concept

Updates using the actual return  $G_t$  (the sum of future rewards) obtained at the end of the episode for the state-insulin pair  $(s_t, I_t)$ .

## Update Formula

$$Q(s_t, I_t) \leftarrow Q(s_t, I_t) + \alpha[G_t - Q(s_t, I_t)]$$

# RL Agents: SARSA & Expected SARSA

## SARSA (On-policy)

Updates using the value of the next state-action pair  $(s_{t+1}, I_{t+1})$  that was *actually* chosen by the policy.

$$Q(s_t, I_t) \leftarrow Q(s_t, I_t) + \alpha [R_{t+1} + \gamma Q(s_{t+1}, I_{t+1}) - Q(s_t, I_t)]$$



# RL Agents: SARSA & Expected SARSA

## SARSA (On-policy)

Updates using the value of the next state-insulin pair  $(s_{t+1}, I_{t+1})$  that was *actually* chosen by the policy.

$$Q(s_t, I_t) \leftarrow Q(s_t, I_t) + \alpha [R_{t+1} + \gamma Q(s_{t+1}, I_{t+1}) - Q(s_t, I_t)]$$

## Expected SARSA

Updates using the *expected* value of the next state, averaging over all possible next insulin doses.

$$Q(s_t, I_t) \leftarrow Q(s_t, I_t) + \alpha \left[ R_{t+1} + \gamma \sum_{i'} \pi(i' | s_{t+1}) Q(s_{t+1}, i') \right]$$

---

**Algorithm 2** Reinforce Algorithm

---

**Input:**

$$\pi_{\theta}(i, s) = \frac{e^{\theta_{i,s}}}{\sum_{i'} e^{\theta_{i',s}}}$$

$\theta$

**for** each episode **do**

    generate an episode from  $\pi_{\theta}(i|s)$

**for** each time step (reversed) **do**

$$G_t = R_t + \gamma G_{t+1}$$

$$\theta = \theta + \alpha G_t \nabla \ln[\pi_{\theta}(i|s)]$$

**end for**

**end for**

---

# Baseline: PID Controller

## Concept

Calculates the insulin dose  $I(t)$  based on the error  $e(t)$  between the current glucose  $G(t)$  and the target setpoint  $G_{target}$ .

## Control Formula

First, the error is defined as:  $e(t) = G(t) - G_{target}$

Then, the insulin dose  $I(t)$  is calculated:

$$I(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

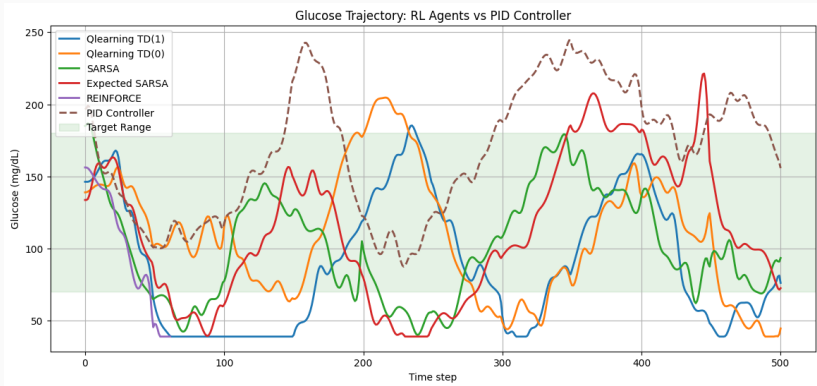
# Notes on PID

- RL should capture better the highly non-linear dynamics of the system.
- RL is continuously learning.
- PID has static parameters.
- PID can be good in short term control and it also does not need a training.

## Results & Conclusion

---

# Comparative Results: Glucose Trajectory



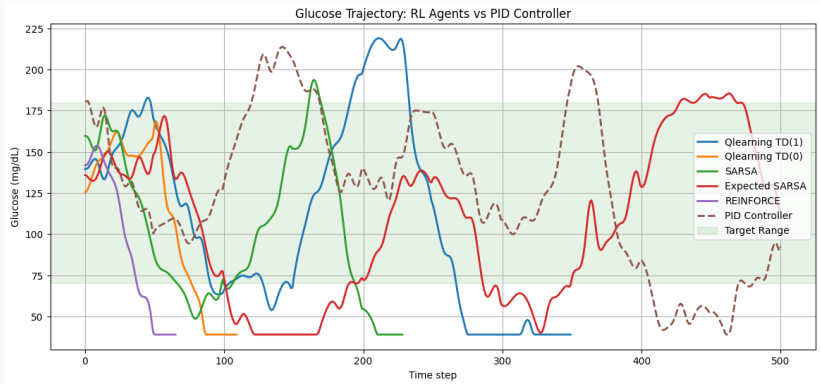
**Figure 1:** RL Agents vs PID Controller Glucose Levels

# Comparative Results: Insulin Dosing Strategy



**Figure 2:** Dosing strategies reveal the behavior behind the glucose outcomes.

## But also...



**Figure 3:** In a lot of scenarios the (almost) only technique that allows to reach the end of the simulation is the PID.



## Discussion: Shifting the Evaluation Criterion

While RL agents are better at tracking the target range, a different criterion is more important for clinical application.

### The Clinical Priority

**Hypoglycemia** (low glucose) is an acute and far more severe danger than controllable **Hyperglycemia** (high glucose).

### Our Primary Goal

The system must, above all, avoid inducing dangerous low-glucose events.

# Final Conclusion

## Main Takeaway

When prioritizing patient safety, the **PID Controller** is the more responsible and superior option.

- **RL Agents' Risk:** All RL agents, with their aggressive dosing, caused episodes of critically low glucose. This is an unacceptable failure mode.
- **PID Controller's Safety:** Its passive, predictable strategy **never** causes a dangerous hypoglycemic event, even if it means tolerating manageable hyperglycemia.
- **Final Verdict:** It is clinically preferable to manage controllable hyperglycemia than to risk a potentially fatal hypoglycemic event.

## Potential Developments

- **Going multi-agent** with the second agent being the agent meal. This should better avoid the cases of hypoglycaemia.
- **Inject insulin before meals** scheduling a less random meal scenario and letting the agent know about the incoming meal.
- **Use semi-gradient SARSA and Actor-Critic** to explore the continuous case.
- **Consider natural gradient** for Reinforce algorithm.

# References



OpenAI.

## **Chatgpt.**

<https://chat.openai.com>, 2025.

Large language model.



Jinyu Xie.

## **simglucose: A type-1 diabetes simulator.**

<https://github.com/jxx123/simglucose>, 2019.

GitHub repository.

# Questions?

Thank you.