Deep Reinforcement Learning for Closed-loop Glucose Control

Andrea Deboni, Giovanni Visi

December 2025

This project develops a semi-automated Artificial Pancreas (AP) system utilizing Deep Reinforcement Learning (DRL), specifically using Proximal Policy Optimization (PPO). The system is implemented within the opensource `simglucose` simulation framework, which provides a Python interface to the FDA-approved UVA/Padova Type 1 Diabetes Simulator. The RL-based control algorithm's performances are compared to both Basal-Bolus "manual" method and a PID control approach.

Contents

1 Introduction

1.1 Clinical Context and Motivation

Type 1 Diabetes Mellitus (T1DM) is a chronic autoimmune disorder characterized by the destruction of pancreatic β-cells, resulting in absolute insulin deficiency. Affected individuals require lifelong exogenous insulin therapy to maintain glucose levels in a safe range. The primary therapeutic objective is to keep blood glucose (BG) concentrations within the euglycemic range of 70–180 mg/dL, thereby preventing both acute hypoglycemia,which poses immediate risk of seizure, coma, or death,and chronic hyperglycemia, which leads to progressive microvascular and macrovascular complications including retinopathy, nephropathy, neuropathy, and cardiovascular disease.

The concept of automated insulin delivery, commonly referred to as the Artificial Pancreas (AP),proved to be very promising for diabetes management. An ideal AP system would continuously monitor glucose levels, compute appropriate insulin doses, and actuate delivery through an insulin pump, all without requiring patient intervention. While significant progress has been achieved using classical control methods, fundamental limitations still exist.

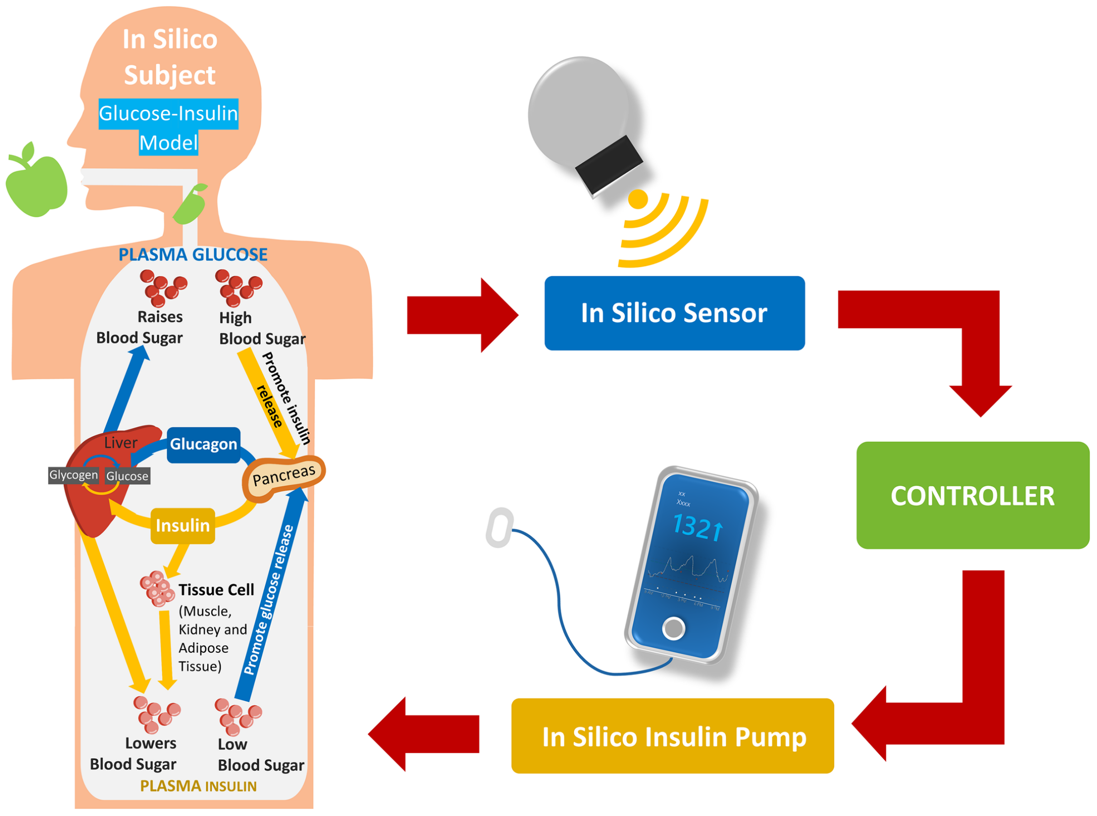

Figure 1: Principal components of the computer simulation environment

1.2 Limitations of Classical Control Approaches

Traditional approaches to automated insulin delivery have predominantly employed Proportional-Integral-Derivative (PID) controllers and Model Predictive Control (MPC). While demonstrating clinical utility, these methods exhibit inherent limitations when applied to glucose regulation:

1. **Reactive Control Limitation:** PID controllers compute control actions (amount of insuline to inject) based on instantaneous error $e(t) = G_{\text{target}} - G(t)$. This fundamentally reactive approach needs that glucose deviations occur before corrective action is initiated. This inevitably results in delayed responses to meal-induced disturbances.
2. **Linear System Assumptions:** The vast majority of real-world medical controllers (like PID controllers) use Linear Time-Invariant (LTI) assumptions. They treat the body like a simple system with a fixed delay and a fixed response rate. However, glucose-insulin metabolism is profoundly non-linear: insulin sensitivity varies with ambient glucose concentration, time of day (circadian rhythms), physical activity, stress, and numerous other factors.
3. **Transport Delays:** Subcutaneous insulin absorption exhibits significant pharmacokinetic delays (onset: 15–30 minutes; peak action: 60–90 minutes; duration: 4–6 hours). These delays challenge reactive controllers that cannot anticipate future glucose trajectories.

1.3 Reinforcement Learning as a Control Paradigm

Reinforcement Learning (RL) offers a fundamentally different approach to the glucose control problem. Rather than explicitly modeling system dynamics and deriving control laws analytically, RL agents learn optimal control policies directly from interaction data through trial-and-error optimization. This data-driven paradigm confers several advantages:

- **Proactive Control:** RL agents can learn to initiate insulin delivery before glucose excursions occur. In our simplified project this is also obtained through informing the RL-model of upcoming meals information.
- **Non-linear Function Approximation:** Deep neural networks employed as policy approximators can represent arbitrarily complex, non-linear control mappings.
- **Multi-objective Optimization:** Reward functions can encode sophisticated clinical objectives beyond simple setpoint tracking, including asymmetric penalties for hypoglycemia versus hyperglycemia.

2 The Simulation Environment

2.1 The UVA/Padova Type 1 Diabetes Simulator

This project utilizes the `simglucose` opensource library, a Python implementation of the UVA/Padova 2008 Type 1 Diabetes Simulator. This simulator holds unique regulatory significance as the first and only computational model of glucose-insulin dynamics accepted by the United States Food and Drug Administration (FDA) as a substitute for preclinical animal trials in the evaluation of closed-loop insulin delivery algorithms. The simulator also included models of commercially available insulin pumps and glucose sensors and simulates the errors associated with the selected sensors and pumps while allowing for the definition of different meal protocols for simulations. The FDA acceptance of this simulator for preclinical testing signifies that its mathematical model captures biological complexity and inter-patient variability with sufficient fidelity to predict real-world controller performance. The simulator incorporates:

- **Glucose Kinetics:** A multi-compartment model describing glucose distribution, utilization, and endogenous production.
- **Insulin Kinetics:** Subcutaneous insulin absorption dynamics with realistic pharmacokinetic profiles.
- **Meal Absorption:** Gastrointestinal carbohydrate digestion with physiologically accurate dynamics.
- **Sensor Model:** Continuous Glucose Monitor (CGM) simulation including measurement noise, calibration drift, and interstitial-to-blood glucose transport delays. Because the CGM is in the subcutaneous interstitial zone rather than the intravascular compartment, the reinforcement learning agent must deal with a physiologically-mediated transport lag of approximately 5–15 minutes. This diffusion delay transforms the environment into a Partially Observable Markov Decision Process (POMDP), where the agent's perceived state is a latent representation of the true blood glucose concentration. The primary consequence of this is that the agent may struggle to associate a specific insulin dose with its eventual effect on glucose level. Clinically, this often leads to insulin stacking, where the agent is unaware that blood glucose has already begun to drop and injects redundant insulin doses based on the lagged data that the CGM reads.
- **Pump Model:** Insulin pump actuation with realistic delivery precision and mechanical characteristics.
- **Virtual Patient Cohort:** Parameterized patient models representing adults, adolescents, and children with diverse metabolic characteristics. Each virtual patient is defined by a unique set of Ordinary Differential Equation (ODE) parameters that dictate their specific metabolic responses, such as insulin sensitivity, glucose distribution volume, and endogenous production rates.

    1. Children (10 subjects): Characterized by high variability and higher insulin sensitivity compared to adults.
    2. Adolescents (10 subjects): Generally have lower insulin sensitivity compared to adults.
    3. Adults (10 subjects): Usually the most stable group.

2.2 Patient Selection: `adolescent#003`

Due to time limitations for this project, we had to simplify it in order to reduce the training and testing time of our RL agent. For this reason we specifically selected only one virtual patient profile `adolescent#003` to work with.

Within the patients cohort, adolescents are characterized by a lower insulin sensitivity compared to adults. In particular, patient adolescent#003 exhibits a typical but non-trivial insulin–glucose dynamics profile: insulin response is clearly delayed and more pronounced than in adults, but not as extreme and unstable as in the most challenging children virtual patients.

Preliminary characterization of the adolescent cohort identified adolescent#003 as exhibiting particularly high intra-day variability and sensitivity to meal disturbances. This patient profile serves as a "stress test" for controller design: successful glucose regulation in this challenging case could provide some evidence of algorithmic robustness. A controller capable of stabilizing glucose levels in `adolescent#003` could generalize effectively to more metabolically stable patient profiles. Due to time constraints for this project, this assumption has not been tested extensively.

2.3 Simulation Infrastructure Configuration

The simulation environment was configured with the following specifications:

- **CGM Sensor:** Dexcom model with realistic noise characteristics
- **Insulin Pump:** Insulet Omnipod model with appropriate delivery precision
- **Sample Time:** 3-minute intervals (matching typical CGM reading frequency)
- **Episode Duration:** Configurable (default: 24 hours)

3 RL formulation of glucose control problem

The glucose control problem easily maps to the classical RL framework:

RL Component	Glucose Control Interpretation
Agent	Artificial Pancreas algorithm (PPO neural network)
Environment	Patient metabolic system (UVA/Padova simulator)
State	Glucose and insulin sensor measuring history, meals information
Action	Insulin delivery rate (continuous, U/min)
Reward	Signal based on the risk of glucose levels

3.1 Markov Decision Processes

The standard mathematical formalization of RL problems is the Markov Decision Process (MDP).

Definition 3.1 (Markov Decision Process)**.** A Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where:

- $\mathcal{S}$ is the state space
- $\mathcal{A}$ is the action space
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function, where $P(s'|s, a)$ denotes the probability of transitioning to state s' given current state s and action a
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function
- $\gamma \in [0, 1]$ is the discount factor

The defining characteristic of an MDP is the Markov property:

Definition 3.2 (Markov Property)**.** A state s_t satisfies the Markov property if the future is conditionally independent of the past given the present:

$$\mathbb{P}(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = \mathbb{P}(s_{t+1}|s_t, a_t) \tag{1}$$

Equivalently, the current state s_t contains all information necessary to predict future states and rewards.

3.2 Partially Observable Markov Decision Processes

Definition 3.3 (Partially Observable MDP)**.** A Partially Observable Markov Decision Process (POMDP) extends the MDP framework to situations where the agent cannot directly observe the true state. A POMDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma, \Omega, O)$ where:

- $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ define the underlying MDP
- Ω is the observation space
- $O : \mathcal{S} \times \mathcal{A} \times \Omega \to [0, 1]$ is the observation function, where $O(o|s', a)$ is the probability of observing o after taking action a and transitioning to state s'

4 From POMDP to Approximate MDP: State Augmentation

4.1 The POMDP Nature of Glucose Control

The human glucose-insulin metabolic system is inherently a POMDP. Consider the scenario where a CGM reports a glucose concentration of 120 mg/dL. This single observation is profoundly insufficient for optimal control because:

1. **Unknown Glucose Trajectory:** Is glucose rising, falling, or stable? The optimal insulin dose differs dramatically between these scenarios.
2. **Unobserved Insulin-on-Board (IOB):** How much previously administered insulin remains active in the subcutaneous depot? This latent variable critically influences future glucose dynamics.
3. **Unknown Meal State:** Is carbohydrate absorption ongoing from a recent meal? Are additional carbohydrates going to be ingested soon?
4. **Unobserved Physiological State:** Factors such as physical activity, stress, and circadian phase affect insulin sensitivity but are not directly measured.

Formally, let $\mathbf{x}_t \in \mathcal{X}$ denote the true physiological state (including plasma glucose, interstitial glucose, insulin concentrations in multiple compartments, gut glucose absorption rate, etc.). The CGM observation $o_t = G_t^{\text{CGM}}$ represents a noisy, delayed measurement of only one component of this high-dimensional state:

$$o_t = h(\mathbf{x}_t) + \epsilon_t \tag{2}$$

where $h(\cdot)$ is the observation function and ϵ_t represents measurement noise.

Remark 4.1 (Violation of Markov Property)**.** The raw CGM observation o_t violates the Markov property:

$$\mathbb{P}(\mathbf{x}_{t+1}|o_t, a_t) \neq \mathbb{P}(\mathbf{x}_{t+1}|o_t, a_t, o_{t-1}, a_{t-1}, \ldots) \tag{3}$$

The history of observations and actions contains information that is not captured by the current observation alone and affects the probability of observing a specific state in the next time step.

4.2 State Augmentation Strategy

To apply standard MDP-based RL algorithms, we construct an augmented observation that approximately restores the Markov property. The key insight is that while individual observations are insufficient, a sufficiently rich history of observations and actions can serve as a surrogate for the unobserved latent state.

Definition 4.1 (Augmented State Space)**.** We define the augmented state $s_t \in \mathbb{R}^{27}$ as:

$$s_t = (\mathbf{g}_t, \mathbf{u}_t, \mathbf{c}_t) \tag{4}$$

where:

- $\mathbf{g}_t = (G_{t-k+1}, G_{t-k+2}, \ldots, G_t) \in \mathbb{R}^k$ is the glucose history
- $\mathbf{u}_t = (I_{t-k+1}, I_{t-k+2}, \ldots, I_t) \in \mathbb{R}^k$ is the insulin history
- $\mathbf{c}_t = (C_{0-30}, C_{30-60}, C_{60-120}) \in \mathbb{R}^3$ is the meal announcement vector

with $k = 12$ (corresponding to 36 minutes of history at 3-minute sampling intervals).

4.2.1 Glucose History Vector

The glucose history $\mathbf{g}_t$ enables inference of glucose dynamics:

Proposition 4.1 (Glucose Derivative Estimation)**.** Given the glucose history $\mathbf{g}_t$ with sampling interval Δt, the neural network can implicitly compute:

First derivative (velocity):

$$\left.\frac{dG}{dt}\right|_t \approx \frac{G_t - G_{t-1}}{\Delta t} \tag{5}$$

Second derivative (acceleration):

$$\left.\frac{d^2G}{dt^2}\right|_t \approx \frac{G_t - 2G_{t-1} + G_{t-2}}{\Delta t^2} \tag{6}$$

These derivatives provide critical information about glucose trajectory that would be unavailable from a single observation.

With $k = 12$ samples spanning 36 minutes, the history captures:

- Short-term trends (5–10 minutes)
- Medium-term dynamics (15–30 minutes)
- The onset phase of insulin action (15–30 minutes post-injection)

4.2.2 Insulin History Vector

The insulin history $\mathbf{u}_t$ serves as a surrogate for Insulin-on-Board (IOB):

Definition 4.2 (Insulin-on-Board Approximation)**.** The true IOB at time t is:

$$\mathrm{IOB}(t) = \int_0^t I(\tau) \cdot \kappa(t - \tau)\, d\tau \tag{7}$$

where $I(\tau)$ is the insulin delivery rate at time τ and $\kappa(\cdot)$ is the insulin absorption kernel characterizing subcutaneous pharmacokinetics.

While the exact kernel $\kappa(\cdot)$ is patient-specific and unknown, the insulin history vector provides the neural network with sufficient information to learn an implicit IOB estimate. This is critical for preventing "insulin stacking",the dangerous accumulation of insulin from multiple doses administered before previous doses have been fully absorbed.

4.2.3 Meal Announcement Vector

The meal announcement vector $\mathbf{c}_t$ is an important simplification that is being used in our project because it provides very important information enabling easier and more precise anticipatory control:

$$\mathbf{c}_t = \begin{pmatrix} C_{0-30} \\ C_{30-60} \\ C_{60-120} \end{pmatrix} = \begin{pmatrix} \sum_{m: t \le t_m < t+30} \mathrm{CHO}_m \\ \sum_{m: t+30 \le t_m < t+60} \mathrm{CHO}_m \\ \sum_{m: t+60 \le t_m < t+120} \mathrm{CHO}_m \end{pmatrix} \tag{8}$$

where t_m and CHO_m denote the time and carbohydrate content of meal m, respectively.

This information enables the RL agent to learn how to initiate insulin delivery in anticipation of meals rather than reacting to post-prandial glucose rises. This makes our Artificial Pancreas system just "semi-automatic" because it still needs to get input information about the meals from the patients 2 hours before the meal.

4.2.4 Observation Normalization

Neural network training is facilitated by input normalization to approximately unit scale:

$$s_t^{\text{normalized}} = \left(\frac{\mathbf{g}_t}{200}, \frac{\mathbf{u}_t}{I_{\max}}, \frac{\mathbf{c}_t}{100} \right) \tag{9}$$

Without normalization:

- Glucose values (40–400 mg/dL)
- Insulin values (0–0.05 U/min)
- Learning could be slow and potentially unstable

The chosen normalization factors (200 for glucose, $I_{\max}$ for insulin, 100 for carbohydrates) yield input values approximately in $[0, 2]$, ensuring balanced gradient contributions from all state components.

4.2.5 Approximate Markov Property

Under the assumption that the glucose-insulin system has finite memory (i.e., state at time t depends only on inputs within a bounded time window), the augmented state s_t satisfies the Markov property approximately:

$$\mathbb{P}(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots) \approx \mathbb{P}(s_{t+1}|s_t, a_t) \tag{10}$$

with approximation error decreasing as history length k increases.

5 Definition of RL functions

5.1 Return Function

Definition 5.1 (Return)**.** The return G_t is the cumulative discounted reward from time t:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{11}$$

where $\gamma \in [0, 1]$ is the discount factor.

Remark 5.1 (Interpretation of Discount Factor)**.** The discount factor γ determines the agent's temporal horizon:

- $\gamma = 0$: Myopic agent considering only immediate reward
- $\gamma \to 1$: Far-sighted agent valuing future rewards nearly equally

The effective horizon can be quantified by the "half-life" of reward importance:

$$\gamma^{t_{1/2}} = 0.5 \iff \ln\left(\gamma^{t_{1/2}}\right) = \ln(0.5) \iff t_{1/2} \ln(\gamma) = \ln(0.5) \iff t_{1/2} = \frac{\ln(0.5)}{\ln(\gamma)} \tag{12}$$

For $\gamma = 0.995$ with 3-minute time steps:

$$t_{1/2} = \frac{\ln(0.5)}{\ln(0.995)} \approx 138 \text{ steps} \approx 7 \text{ hours} \tag{13}$$

This horizon is appropriate for glucose control given that insulin action duration spans 4-6 hours.

5.2 State-Value Function

Definition 5.2 (State-Value Function)**.** The state-value function $V^\pi(s)$ is the expected return when starting from state s and following policy π:

$$V^\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s\right] \tag{14}$$

5.3 Action-Value Function

Definition 5.3 (Action-Value Function)**.** The action-value function $Q^\pi(s, a)$ is the expected return when starting from state s, taking action a, and, after, following policy π:

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a] \tag{15}$$

5.4 Advantage Function

Definition 5.4 (Advantage Function)**.** The advantage function $A^\pi(s, a)$ measures how much higher or lower the expected return given by taking action a when in state s is compared to the expected return when in state s and taking actions under policy π:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \tag{16}$$

1. $A^\pi(s, a) > 0 \Rightarrow$ action a is better than average
2. $A^\pi(s, a) < 0 \Rightarrow$ action a is worse than average

6 Action Space Design and Reward Engineering

6.1 Continuous Action Space

The agent outputs a continuous action $a_t \in [-1, 1]$. This bounded output is mapped to insulin delivery rate through an exponential transformation ensuring clinical safety.

Definition 6.1 (Exponential Action Mapping)**.** The insulin delivery rate $I(t)$ in U/min is computed from the raw action a_t as:

$$I(t) = I_{\max} \cdot \exp(\eta \cdot (a_t - 1)) \tag{17}$$

where $I_{\max} = 0.05$ U/min (maximum delivery rate) and $\eta = 4.0$ (scaling parameter).

Proposition 6.1 (Properties of Exponential Mapping)**.** The exponential mapping has the following properties:

Action a_t	Calculation	Insulin Rate (U/min)
-1	$0.05 \cdot e^{4(-2)} = 0.05 \cdot e^{-8}$	≈ 0.000017 (near-zero)
0	$0.05 \cdot e^{4(-1)} = 0.05 \cdot e^{-4}$	≈ 0.00092
0.5	$0.05 \cdot e^{4(-0.5)} = 0.05 \cdot e^{-2}$	≈ 0.0068
1	$0.05 \cdot e^{4(0)} = 0.05 \cdot e^{0}$	$= 0.05$ (maximum)

Remark 6.1 (Design Rationale)**.** The exponential mapping provides:

1. **Safety:** $a_t = -1$ yields near-zero insulin (effective basal suspension for hypoglycemia prevention)
2. **Bounded Output:** $a_t = 1$ yields maximum safe rate
3. **Fine Control at Low Doses:** The exponential curve concentrates resolution where precision matters most (basal modulation)

6.2 Reward Function Design

6.2.1 The Magni Risk Index

Both reward functions are built upon the Blood Glucose Risk Index (BGRI), a clinically validated metric developed by Kovatchev et al.

Definition 6.2 (Magni Risk Index)**.** The risk index $J(G)$ for glucose value G (mg/dL) is computed as:

$$f(G) = 1.509 \cdot \left[(\ln G)^{1.084} - 5.381\right] \tag{18}$$

$$J(G) = 10 \cdot [f(G)]^2 \tag{19}$$

The formulation penalizes hypoglycemia more heavily than hyperglycemia for equivalent absolute deviations, reflecting the immediate danger of low blood glucose.

Representative values:

Glucose (mg/dL)	Risk Index
50	22.7 (severe hypoglycemia)
70	5.8 (hypoglycemia)
100	0.5 (optimal)
150	2.1 (elevated)
200	6.3 (hyperglycemia)
300	17.2 (severe hyperglycemia)

6.2.2 Reward Function

The reward function directly utilizes the Magni Risk Index:

$$R_{\text{paper}}(G_{t+1}) = \begin{cases} -15.0 & \text{if } G_{t+1} \leq 39 \text{ mg/dL} \\ -\dfrac{J(G_{t+1})}{100} & \text{otherwise} \end{cases} \tag{20}$$

This reward function gives severe penalty for hypoglycemia and it is always non-positive, so there is no explicit reward for good glucose control.

7 Policy Gradient Methods

7.1 Limitations of Value-Based Methods

Value-based methods such as Q-learning and DQN learn the optimal action-value function $Q^*(s, a)$ and derive a policy by selecting actions with maximum Q-value. While powerful, these methods have fundamental limitations for continuous control:

1. **Discrete Action Requirement:** DQN requires enumeration of all possible actions. For continuous insulin delivery, discretization loses important precision for fine insulin modulation.

2. **Deterministic Policies:** Value-based methods use some stochasticity during training but ultimately produce deterministic policies. In a partially observable environment, some stochasticity in the final policy can bring to better control performances on average.

7.2 Policy Parameterization

Policy-based methods directly parameterize the policy $\pi_\theta(a|s)$ with parameters θ (neural network weights) and optimize these parameters to maximize expected return.

Definition 7.1 (Gaussian Policy)**.** For continuous action spaces, the standard parameterization is a Gaussian policy:

$$\pi_\theta(a|s) = \mathcal{N}\left(\mu_\theta(s), \sigma_\theta(s)^2\right) = \frac{1}{\sigma_\theta(s)\sqrt{2\pi}} \exp\left(-\frac{(a-\mu_\theta(s))^2}{2\sigma_\theta(s)^2}\right) \tag{21}$$

where the mean $\mu_\theta(s)$ and standard deviation $\sigma_\theta(s)$ are outputs of a neural network with parameters θ.

7.3 The Policy Gradient Objective

The objective is to maximize expected return:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right] = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] \tag{22}$$

where $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ is a trajectory sampled under policy π_θ.

7.4 The Policy Gradient Theorem

Theorem 7.1 (Policy Gradient Theorem)**.** The gradient of the objective $J(\theta)$ with respect to policy parameters θ is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \hat{A}_t\right] \tag{23}$$

where $\hat{A}_t$ is an estimate of the advantage function at time t.

Remark 7.1 (Intuitive Interpretation)**.** The policy gradient has an intuitive interpretation:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\underbrace{\nabla_\theta \log \pi_\theta(a_t|s_t)}_{\text{direction to increase } \mathbb{P}(a_t)} \cdot \underbrace{\hat{A}_t}_{\text{how good or bad was } a_t\text{?}}\right] \tag{24}$$

- $\nabla_\theta \log \pi_\theta(a_t|s_t)$: This vector points in the direction of the parameter space that maximizes the likelihood of choosing action a_t in state s_t.
  - Note that $\nabla \log \pi = \frac{\nabla \pi}{\pi}$. Dividing by π normalizes the update. Without the log, frequently occurring actions would dominate the gradient updates regardless of their quality.
- $\hat{A}_t$: The Advantage function represents the *quality* of the action relative to the average performance in that state.
  - If $\hat{A}_t > 0$: The action yielded a better-than-average return. We "push" θ in the direction of the gradient to **increase** the probability of taking a_t again.

  - If $\hat{A}_t < 0$: The action yielded a worse-than-average return. The scalar is negative, which reverses the gradient direction, **decreasing** the probability of taking a_t.

- $\mathbb{E}_{\tau \sim \pi_\theta}$: Because the environment and policy are stochastic, we cannot calculate the gradient from a single step. We must average over sampled trajectories (τ).

7.5 Problems with Vanilla Policy Gradient

The vanilla policy gradient algorithm (REINFORCE) suffers from:

1. **Sample Inefficiency:** Each batch of data is used once and discarded.Discarding data after a single use limits the amount of learning that can occur from each experience. Collecting new data can be computationally expensive and time-consuming, especially in complex environments or real-world scenarios.

2. **No Stability Guarantees:**A single "bad update" during training doesn't just make the training take longer; it can permanently damage the agent's ability to ever find a safe final policy. The policy is likely to get "stuck" in suboptimal solutions.

In medical applications, the lack of stability guarantees is particularly concerning.

8 Trust Region Methods and Proximal Policy Optimization Algorithms

8.1 Trust Region Optimization

The fundamental problem with vanilla policy gradient is the absence of control over policy change magnitude. Trust region methods address this by constraining updates to a "neighborhood".

Definition 8.1 (Trust Region Policy Optimization (TRPO))**.** TRPO formulates policy optimization as a constrained optimization problem:

$$\max_{\theta} \mathbb{E}_{s,a \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \hat{A}^{\pi_{\theta_{\text{old}}}}(s,a) \right] \tag{25}$$

subject to:

$$\mathbb{E}_s \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot|s) \| \pi_\theta(\cdot|s) \right) \right] \leq \delta \tag{26}$$

where D_{KL} is the Kullback-Leibler divergence.

While TRPO provides theoretical guarantees, its practical implementation is computationally expensive and complex.

8.2 PPO - Proximal Policy Optimization

Proximal Policy Optimization (PPO) is an on-policy, actor-critic deep reinforcement learning algorithm that optimizes a stochastic policy by maximizing a clipped surrogate objective function. Instead of relying on complex optimization methods to define a "trust region" for safe updates, PPO enforces stability through a simpler constraint that limits

how much the policy can change in a single update step.
PPO is classified as an Actor-Critic algorithm because it trains two distinct neural network components simultaneously: an Actor (the policy π) that decides which action to take, and a Critic (the value function V) that evaluates the quality of the current state. The key benefit of this structure is that the Critic provides a baseline; rather than learning from raw, noisy rewards, the Actor learns from the Advantage,the difference between the actual outcome and the Critic's prediction,which allows the agent to reinforce actions that perform "better than expected," significantly reducing variance and stabilizing the learning process.

8.3 PPO Clipped Objective

The inventors of PPO aimed at using a more inexpensive method to compute an approximation of divergence between the two policies states π_θ and $\pi_{\theta_{old}}$. Proximal Policy Optimization achieves similar stability to TRPO through a simpler mechanism: clipping the objective function itself. PPO's clipping mechanism acts as a safety net, preventing the policy from making overly drastic changes based on limited or potentially noisy data. This leads to a more robust and reliable learning process.
PPO-clip updates policies via

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{s,a\sim\pi_{\theta_k}}[L(s,a,\theta_k,\theta)] \tag{27}$$

typically taking multiple steps of SGD to maximize the objective.

To define the objective L, we first define the probability ratio $r_t(\theta)$ between the new policy and the old policy:

$$r_t(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} \tag{28}$$

- $r_t(\theta) = 1$: Action equally likely under both policies
- $r_t(\theta) > 1$: The action is more likely under new policy
- $r_t(\theta) < 1$: The action is less likely under new policy

The objective L is given by:

$$L(s,a,\theta_k,\theta) = \min\left(r_t(\theta)A^{\pi_{\theta_k}}(s,a), \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A^{\pi_{\theta_k}}(s,a)\right) \tag{29}$$

in which ϵ is a (small) hyperparameter which intuitively represents how much away the new policy is "allowed" to differ from the old.

This objective function can be easily simplified to the following one :

$$L(s,a,\theta_k,\theta) = \min\left(r_t(\theta)A^{\pi_{\theta_k}}(s,a), g(\epsilon, A^{\pi_{\theta_k}}(s,a))\right) \tag{30}$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0 \end{cases} \tag{31}$$

Thanks to the clipping operation, the probability ratio $r_t(\theta)$ is supposed to remain within the interval $[(1-\epsilon),(1+\epsilon)]$ even after multiple epochs of weight updates performed on the same data. Thereby, the goal of avoiding destructively large weight updates is supposed to be achieved

Case 1: Advantage is Positive ($A > 0$)

Suppose the advantage for that state-action pair is positive. In this case, its contribution to the objective reduces to:

$$L(s, a, \theta_k, \theta) = \min(r_t(\theta), 1 + \epsilon) A^{\pi_{\theta_k}}(s, a) \tag{32}$$

Because the advantage is positive, the objective will increase if the action becomes more likely,that is, if $r_t(\theta)$ increases. But the **min** in this term puts a limit to how much the objective can increase.

Once $r_t(\theta) > (1 + \epsilon)$, the min comes into play and this term hits a ceiling of $(1 + \epsilon)A^{\pi_{\theta_k}}(s, a)$. Thus, the objective function does not benefit too much when the new policy is very "far" from the old one.

Case 2: Advantage is Negative ($A < 0$)

Suppose the advantage for that state-action pair is negative. In this case, its contribution to the objective reduces to:

$$L(s, a, \theta_k, \theta) = \min(r_t(\theta) A^{\pi_{\theta_k}}(s, a), (1 - \epsilon) A^{\pi_{\theta_k}}(s, a)) = \max(r_t(\theta), (1 - \epsilon)) A^{\pi_{\theta_k}}(s, a) \tag{33}$$

Because the advantage is negative, the objective will increase if the action becomes less likely,that is, if $r_t(\theta)$ decreases. But the **max** in this expression puts a limit to how much the objective can increase.

Once $r_t(\theta) < (1-\epsilon)$, the max "activates" and this term hits a ceiling of $(1-\epsilon)A^{\pi_{\theta_k}}(s, a)$. Thus, again: the objective function does not have a big benefit when the new policy is very far from the old one.

Case 3: Recovery (Undoing a "Wrong" Update)

This is the case where the policy has been updated in the "wrong" direction. For example, the advantage is positive ($A > 0$), but the new policy makes the action *less* likely ($r_t(\theta) < 1$).

- Since $r_t(\theta) < 1$, it is well below the clipping threshold of $1 + \epsilon$.
- The min operator selects the unclipped value: $r_t(\theta)A^{\pi_{\theta_k}}(s, a)$.

This "aggressively" brings the objective function to lower values pushing the algorithm to undo the mistaken policy update.
PPO only limits the update when the policy is moving *too far in the correct direction*, but it does not limit the policy from recovering when it is performing worse than the old policy.

8.4 Complete PPO Objective function

In practice, PPO combines the clipped policy objective with two additional terms:

$$\boxed{L^{PPO} = \mathbb{E}_t \left[L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right]} \tag{34}$$

The coefficients c_1 and c_2 are positive and control the regularization strength.

8.4.1 Component 1: Policy Loss (Clipped Objective)

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{35}$$

8.4.2 Component 2: Value Function Loss

$$L^{\text{VF}}(\theta) = \mathbb{E}_t \left[\left(V_\theta(s_t) - V_t^{\text{target}} \right)^2 \right] \tag{36}$$

The value network is trained to predict expected returns, providing baselines for advantage estimation. This network is trained to minimize the squared error between its predictions and the actual returns. V_t^{target} is usually the discounted return-to-go.

8.4.3 Component 3: Entropy Bonus

To encourage exploration and prevent premature convergence, PPO adds an entropy loss:

$$S[\pi_\theta](s) = -\sum_a \pi_\theta(a|s) \log \pi_\theta(a|s) \tag{37}$$

Higher entropy encourages exploration by preventing premature convergence to deterministic policies.

8.5 Generalized Advantage Estimation (GAE)

The theoretical exact value the Advantage is unknown, we need to estimate it in order to use it in the clipped PPO objective function. The Advantage estimation involves a bias-variance tradeoff:

- **Monte Carlo** (high variance, no bias): Uses actual returns

$$\hat{A}_t^{\text{MC}} = \sum_{k=0}^{T-t} \gamma^k r_{t+k} - V(s_t) \tag{38}$$

- **TD(0)** (low variance, high bias): Uses one-step bootstrapping

$$\hat{A}_t^{\text{TD}} = r_t + \gamma V(s_{t+1}) - V(s_t) = \delta_t \tag{39}$$

Definition 8.2 (Generalized Advantage Estimation)**.** GAE interpolates between these extremes using parameter λ:

$$\hat{A}_t^{\text{GAE}(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{40}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD error.

Remark 8.1 (Interpretation of λ)**.**

- $\lambda = 0$: Pure TD(0), low variance but potentially biased
- $\lambda = 1$: Pure Monte Carlo, unbiased but high variance
- $\lambda = 0.95$ (our choice)

9 Training and Testing

9.1 Training Configuration

The training is executed using a vectorized PPO implementation that runs four parallel simulations to efficiently gather 2 million steps of glucose control data.
Meal schedules and carbohydrate intakes are randomized for every episode, forcing the agent to learn adaptable control strategies rather than memorizing a specific daily pattern. As the agent interacts with these constantly shifting scenarios, it updates its neural network policy to maximize the reward function, while a separate evaluation process periodically tests its performance on unseen scenarios to save the optimal model.

9.2 Hyperparameter Analysis and Justification

9.2.1 Learning Rate ($\alpha = 3 \times 10^{-4}$)

Function. Step size for gradient descent updates using the Adam optimizer. Parameters are updated according to

$$\theta \leftarrow \theta - \alpha \nabla_\theta L.$$

Justification. The value 3×10^{-4} is a widely used and empirically stable default for Adam in neural network training. Smaller values (e.g., 10^{-5}) result in slow but stable learning, while larger values (e.g., 10^{-2}) may cause oscillations or divergence. For continuous control tasks, particularly in medical contexts, stability is prioritized over rapid convergence.

9.2.2 Discount Factor ($\gamma = 0.995$)

Function. Determines the relative importance of future versus immediate rewards.
Justification. Rapid-acting insulin exhibits the following pharmacodynamic profile:

- Onset: 15–30 minutes
- Peak action: 60–90 minutes
- Duration: 4–6 hours

With a simulation time step of 3 minutes, a discount factor of $\gamma = 0.995$ corresponds to an effective planning horizon of approximately 7 hours, closely matching insulin activity duration. Lower values of γ would cause the agent to undervalue delayed consequences of insulin delivery.

9.2.3 Clip Range ($\epsilon = 0.2$)

Function. Limits the maximum change in action probabilities per policy update. The probability ratio r_t is clipped to the interval $[1 - \epsilon, 1 + \epsilon] = [0.8, 1.2]$.
Justification. In medical applications, the clip range acts as a safety mechanism. A value of $\epsilon = 0.2$ prevents abrupt policy changes, even in the presence of anomalous data (e.g., unusual patient responses), ensuring conservative and stable learning.

9.2.4 Rollout Length ($n_{\textbf{steps}} = 2048$)

Function. Number of environment steps collected per environment before each policy update.

Calculation. Using 4 parallel environments:

- Total transitions per update: $4 \times 2048 = 8192$
- Simulated time: 8192×3 minutes ≈ 17 days

Justification. The rollout batch must capture a diverse range of physiological scenarios (meals, fasting periods, daytime and nighttime conditions) to ensure stable gradient estimation. A simulated duration of approximately 17 days provides sufficient coverage of typical glucose dynamics.

9.2.5 Mini-batch Size (batch_size = 256)

Function. Number of transitions used per gradient descent step.

Calculation. Given 8192 total transitions per update:

$$\frac{8192}{256} = 32$$

gradient steps are performed per epoch.

Justification. This choice balances gradient accuracy (which favors larger batch sizes) against computational efficiency and stochasticity, promoting stable yet efficient training.

9.2.6 GAE Parameter ($\lambda = 0.95$)

Function. Controls the bias–variance tradeoff in Generalized Advantage Estimation (GAE).

Justification. A value of $\lambda = 0.95$ is commonly used in PPO and provides a favorable compromise between low-variance advantage estimates and acceptable bias, resulting in smoother and more reliable policy updates.

9.2.7 Entropy Coefficient ($c_2 = 0.01$)

Function. Weights the entropy bonus term in the policy objective, encouraging exploration.

Justification. In medical control tasks, exploration must be carefully constrained. A coefficient of 0.01 promotes sufficient exploration to discover effective strategies while avoiding unsafe or erratic behavior.

9.2.8 Value Function Coefficient ($c_1 = 0.5$)

Function. Weights the value function loss relative to the policy loss.

Justification. The value function provides a baseline for advantage estimation and should support, rather than dominate, policy learning. Setting $c_1 = 0.5$ ensures accurate value estimation without overpowering the policy objective.

9.2.9 Maximum Gradient Norm (**max_grad_norm** $= 0.5$)

Function. Clips the gradient norm to prevent exploding gradients.

Justification. Extreme physiological states (e.g., glucose levels exceeding 500 mg/dL) may generate large loss values and gradients. Gradient clipping ensures bounded parameter updates, improving training stability and robustness.

9.2.10 Neural Network Architecture

Configuration. Separate policy and value networks with identical architectures:

$$27 \rightarrow 128 \rightarrow 128 \rightarrow 64 \rightarrow 1.$$

Activation Function. ReLU (Rectified Linear Unit).

Justification. This architecture provides sufficient representational capacity to model non-linear glucose–insulin dynamics while limiting overfitting and maintaining computational efficiency.

9.3 Evaluation Protocol

9.3.1 Fixed Scenario Comparison

For fair comparison, all controllers were evaluated on an identical fixed 24-hour scenario:

Meal	Time	Carbohydrates (g)	Description
Breakfast	08:00	40	Light morning meal
Lunch	13:00	80	Standard midday meal
Dinner	20:00	60	Evening meal

Table 1: Fixed evaluation scenario

9.3.2 Baseline Controllers

Three controllers were compared:

1. Basal-Bolus Controller:

- Constant basal rate based on patient parameters
- Meal boluses calculated from insulin-to-carbohydrate ratio
- Correction boluses based on glucose deviation

2. PID Controller:

- Target glucose: 140 mg/dL
- Proportional gain: $K_p = 1.0 \times 10^{-4}$
- Integral gain: $K_i = 1.0 \times 10^{-7}$
- Derivative gain: $K_d = 3.9 \times 10^{-3}$

3. PPO Agent:

- Loaded from best checkpoint saved during the training process

9.4 Clinical Metrics

9.4.1 Time in Range (TIR)

$$\text{TIR} = \frac{\sum_{t=1}^{T} \mathbb{I}_{[70,180]}(G_t)}{T} \times 100\% \tag{41}$$

9.4.2 Time Below Range (TBR)

$$\text{TBR} = \frac{\sum_{t=1}^{T} \mathbb{I}_{(-\infty,70)}(G_t)}{T} \times 100\% \tag{42}$$

9.4.3 Time Above Range (TAR)

$$\text{TAR} = \frac{\sum_{t=1}^{T} \mathbb{I}_{(180,\infty)}(G_t)}{T} \times 100\% \tag{43}$$

9.4.4 Coefficient of Variation (CV)

It is a measure of blood glucose level variability

$$\text{CV} = \frac{\sigma_G}{\bar{G}} \times 100\% \tag{44}$$

9.5 Robustness Testing

Beyond standard evaluation on a fixed scenario, some limited robustness testing was conducted using challenging scenarios designed to stress-test the controller:

9.5.1 High Carbohydrate Scenario

Meal	Time	Carbohydrates (g)
Breakfast	08:00	80
Lunch	13:00	110
Dinner	20:00	90

Tests maximum insulin delivery capacity and post-prandial control under extreme carbohydrate loads.

9.5.2 Missed Lunch Scenario

Meal	Time	Carbohydrates (g)
Breakfast	08:00	40
Lunch	—	0 (skipped)
Dinner	20:00	60

Tests hypoglycemia prevention when expected carbohydrates do not arrive.

9.5.3 Late Dinner Scenario

Meal	Time	Carbohydrates (g)
Breakfast	08:00	40
Lunch	13:00	60
Dinner	22:30	70

Tests overnight insulin adjustment.

9.5.4 Random Chaos Scenario

- 4 meals at random times between 06:00 and 22:00
- Random carbohydrate amounts: 20–100g per meal

Tests generalization to completely unpredictable patterns.

10 Results

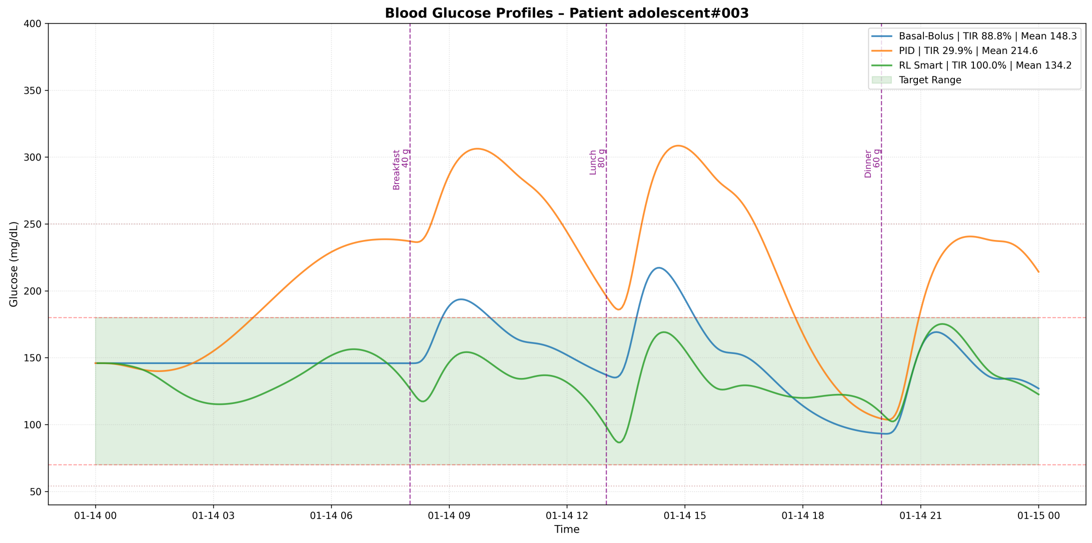

Figure 2: Control Performances Timeseries Comparison

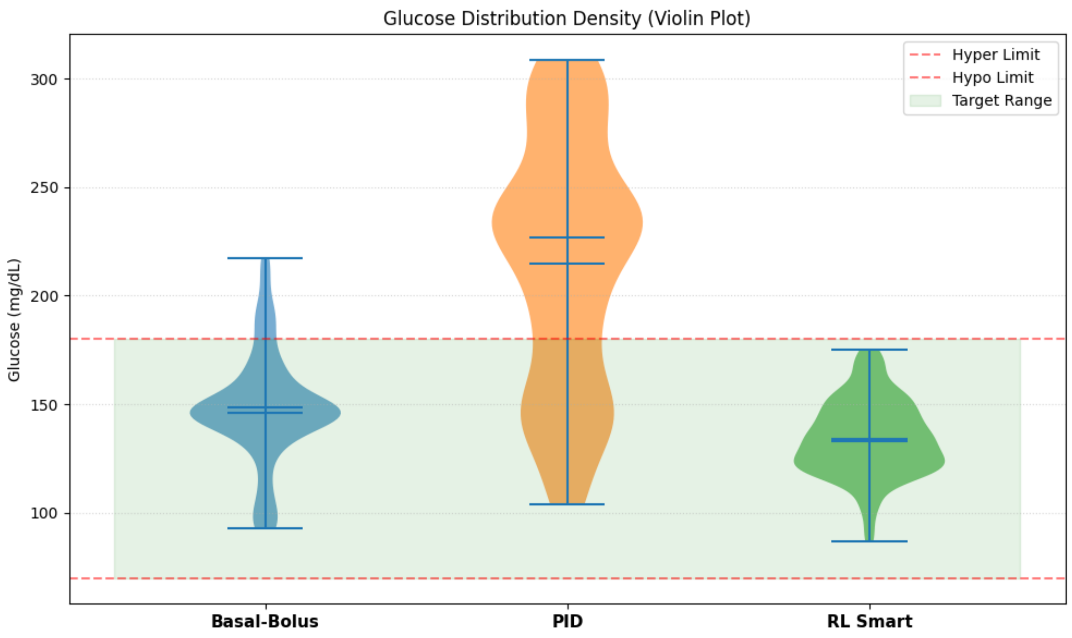

Figure 3: Glucose Distribution Density comparison

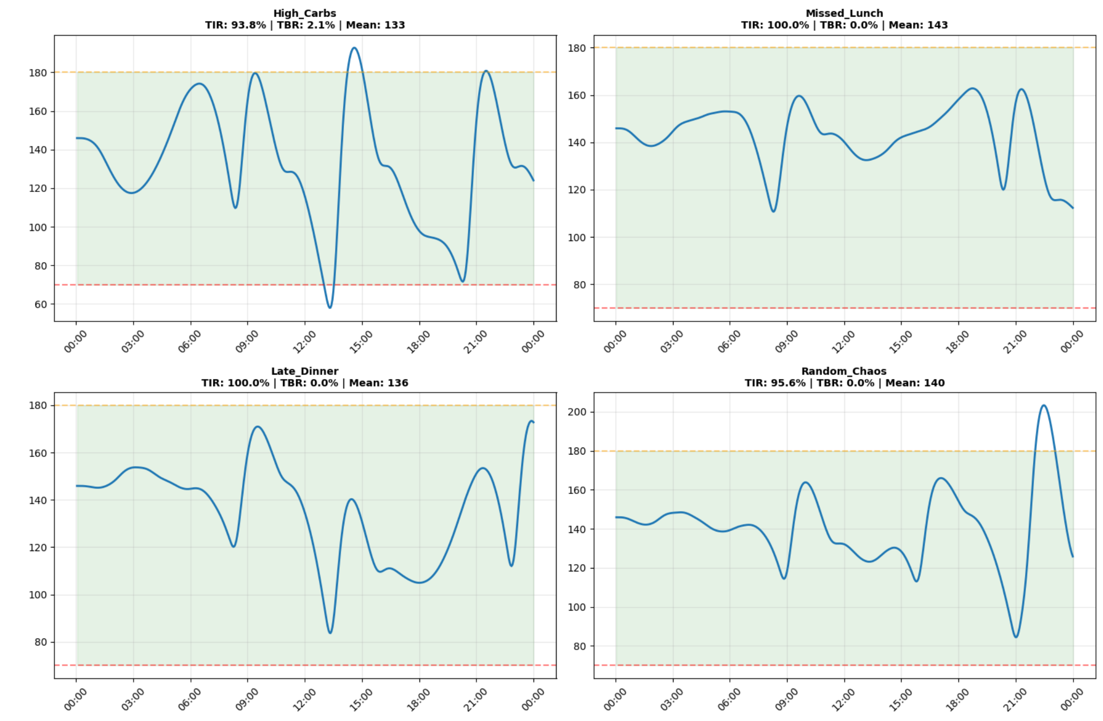

Figure 4: Robustness Test in different scenarios

10.1 Controller Comparison

Analysis of glucose trajectories revealed distinct behavioral patterns:

Basal-Bolus Controller:

- Post-prandial spikes
- Fixed bolus timing causes absorption mismatch
- Stable overnight control due to steady basal rate

PID Controller:

- Significant response lag to meal disturbances
- Oscillatory behavior: high peaks followed by aggressive correction and undershoot
- Derivative term provides limited anticipation capability

It is important to notice that the PID parameters used for testing have been found in the literature and have not been extensively tuned. It is likely that more stable control can be achieved after a better calibration of PID gains.

PPO Agent:

- Clear anticipatory behavior: insulin delivery increased *before* glucose rise
- Less pronounced post-prandial curves compared to other controllers
- Achieved the most precise glucose control among all controllers

The RL agent's anticipatory capability are thanks to :

1. **Meal Announcement Feature:** State includes future carbohydrates, providing advance warning
2. **Learned Causal Model:** Neural network learns the relationship between carbohydrate intake and subsequent glucose rise
3. **High Discount Factor:** With $\gamma = 0.995$, the agent optimizes for long-term outcomes

11 Conclusion and Future Directions

The DRL agent's capacity for anticipatory control enables it to overcome the physiological delays that fundamentally limit reactive controllers. The combination of:

- PPO's clipping mechanism for stable learning
- Appropriate discount factor matching insulin pharmacokinetics
- State augmentation providing information on future meals and glucose/insuline history

produces a controller capable of managing the challenging `adolescent#003` patient profile with clinically satisfactory performance metrics.

11.1 Simplifying assumptions and limitations

It is important to clearly state the main simplifications present in this work. Due to time constraints and our very limited experience in this field, we decided to develop this project as just a "first step" towards possibly implementing a more complex and "realistic" control algorithm. The main limiting simplifications are the following:

- **Single Patient Training:** The current model is patient-specific; generalization across patients requires further investigation..
- **Simulation-Only Validation:** Clinical trials with real patients are necessary before deployment.
- **Limited Disturbance Modeling:** Physical activity, stress, and illness effects are not explicitly modeled. In our project meals represent the only disturbance affecting glucose levels.
- **Upcoming meals announcement** The RL agent is informed of meals happening in the following 120 minutes. This undoubtedly makes it easier for the agent to learn how to proactively take action and inject insulin before glucose levels start rising after a meal.
- **Information about carbs intake** The observation state is augmented in order to inform the RL agent of the quantities of carbs that are going to be ingested in the following 120 minutes.

In particular, the information about upcoming meals and carbs intake represents the main reason why our Artificial Pancreas system cannot be considered fully automatic since it would still need the patient to manually insert information about upcoming meals. This makes the whole system vulnerable to human error and negligence.
Moreover, the control performances of the system were tested on a very low number of scenarios with limited variability. Therefore, there is still a lack of sufficient proof about the robustness of the control system.

11.2 Future Directions

Possible directions for future research include:

- **Multi-Patient Training:** Training on diverse virtual patient populations for improved generalization
- **Meal Uncertainty Handling:** Random or absent meal announcements during training
- **Sensor Noise Injection:** More realistic CGM simulation including failures
- **Transfer Learning:** Pre-training on simulation with fine-tuning on real patient data
- **Constrained Reinforcement Learning:** Hard safety constraints on insulin delivery rates
- **Interpretability:** Analysis of learned policies to understand agent decision-making

References

1. Miguel Tejedor, Ashenafi Zebene Woldaregay, Fred Godtliebsen, *"Reinforcement learning application in diabetes blood glucose control: A systematic review"*, Artificial Intelligence in Medicine, Volume 104, 2020, 101836,ISSN 0933-3657,

2. Chirath Hettiarachchi, Nicolo Malagutti, Christopher J. Nolan, Hanna Suominen, Elena Daskalaki, *"G2P2C — A modular reinforcement learning algorithm for glucose control by glucose prediction and planning in Type 1 Diabetes"*,Biomedical Signal Processing and Control,Volume 90,2024,105839,ISSN 1746-8094,

3. C. Hettiarachchi, N. Malagutti, C. Nolan, E. Daskalaki and H. Suominen, *"A Reinforcement Learning Based System for Blood Glucose Control without Carbohydrate Estimation in Type 1 Diabetes: In Silico Validation"* 2022 44th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)

4. Viroonluecha P, Egea-Lopez E, Santa J. *"Evaluation of blood glucose level control in type 1 diabetic patients using deep reinforcement learning"* 2022 Sep 13

5. Cobelli C, Kovatchev B. *"Developing the UVA/Padova Type 1 Diabetes Simulator: Modeling, Validation, Refinements, and Utility"* J Diabetes Sci Technol. 2023

6. Man CD, Micheletto F, Lv D, Breton M, Kovatchev B, Cobelli C. *"The UVA/-PADOVA Type 1 Diabetes Simulator: New Features."* J Diabetes Sci Technol. 2014

7. Schulman, John et al. *"Proximal Policy Optimization Algorithms."* ArXiv abs/1707.06347 (2017)