Understanding Closed-Loop Insulin Control Systems

A Step-by-Step Engineering Perspective for Non-Biomedical Students

Compiled and Explained by Krishna

November 10, 2025

Abstract

This document aims to explain the working of a Hybrid Closed-Loop Insulin Delivery System — commonly known as an Artificial Pancreas — using the language of engineering and control theory. Instead of assuming a medical background, we treat the human body as a dynamic control system: glucose acts as the system output, insulin as the control input, and meals as disturbances. Each module — from the physiological plant to the CGM sensor, observer, PID controller, adaptive basal learner, and safety layer — is described both intuitively and mathematically. The goal is not publication-grade formalism, but deep conceptual clarity for learners who prefer equations, analogies, and feedback-loop thinking.

1 Introduction: Why Control Theory in Biology?

Imagine trying to control the speed of a car that reacts slowly, has noisy sensors, and unpredictable slopes on the road. That is almost exactly what blood-glucose regulation looks like from a control engineer's viewpoint.

The blood glucose concentration $G(t)$ is the **output** we want to maintain near a target (around $100\,\text{mg/dL}$). The **control input** is insulin infusion, delivered through a pump. The **disturbances** are meals and physical activity. The **sensors** are continuous glucose monitors (CGM) that measure glucose indirectly, with delay and noise.

1.1 The Closed-Loop Concept

In open-loop therapy, a patient decides insulin manually. In closed-loop therapy, a controller automatically adjusts insulin every few minutes based on sensor data, just like a thermostat regulates temperature.

1.2 System Components

The overall simulator contains several interacting modules:

- **Physiological Plant (PK/PD Model)** — Models how insulin and glucose move between body compartments.

- **Meal Absorption Model** — Describes how ingested carbohydrates enter the bloodstream gradually.
- **CGM Sensor Model** — Adds measurement noise, time lag, and occasional dropouts to mimic a real sensor.
- **Observer (Predictor–Corrector)** — Estimates hidden internal states using the model and the noisy sensor signal.
- **Controller (PID + Adaptive Basal)** — Computes insulin commands to keep glucose near target.
- **Safety Layer** — Ensures no dangerous insulin overdoses by enforcing logical constraints.
- **(Optional) RL Agent** — A reinforcement learning policy that can learn to improve or assist the controller.

Each of these modules will be explained in simple mathematical form with analogies and worked examples in the following sections.

2 Big Picture Intuition

We can visualize the glucose–insulin system as a set of connected "tanks" or "RC circuits." Glucose acts like voltage; insulin acts like the current that discharges it. Meals are step inputs that add charge, and the controller injects insulin to bring the voltage back down to the reference.

$$\begin{tikzcd}
\text{Meal Input} \arrow[r] & D_{\text{gut}} \arrow[r] & G_{\text{blood}} \arrow[r, "\text{Sensor + Noise}"] & y(t) \arrow[r, "\text{Controller}"] & u(t) \arrow[r] & I_{\text{plasma}} \arrow[r] & X_{\text{effect}} \arrow[r, dashed] & G_{\text{blood}}
\end{tikzcd}$$

In the next part, we will derive and interpret the mathematical equations that govern this physiological plant.

3 The Physiological Plant: How the Body Responds to Insulin and Meals

3.1 Engineering Viewpoint

From a control-systems perspective, the human metabolic system behaves like a set of slow, interconnected processes that exchange glucose and insulin between different "compartments." We treat each compartment as a *state variable* that evolves over time according to differential equations.

Just as an RC network can be described by voltage and current flow between nodes, here glucose and insulin flow between body spaces such as blood, tissue, and subcutaneous layers.

3.2 State Variables

We describe the body by six state variables collected in a vector

$$x = \begin{bmatrix} G \\ X \\ I_{pl} \\ D \\ I_{sc1} \\ I_{sc2} \end{bmatrix}.$$

Each symbol has a clear physical meaning:

- G — blood-glucose concentration (mg/dL), our main output to control.
- X — insulin “action” (1/min), a lumped measure of how strongly insulin is enabling glucose uptake.
- I_{pl} — insulin concentration in plasma (mU/L), representing insulin available in the bloodstream.
- I_{sc1}, I_{sc2} — insulin amounts in two subcutaneous depots (mU); they model fast and slow absorption pathways.
- D — glucose amount remaining in the gut after a meal (mg); it releases glucose gradually into blood.

The control input is the insulin infusion rate $u(t)$ (Units per minute), and the main disturbance is meal ingestion.

3.3 Dynamic Equations

The time evolution of these states follows a pharmacokinetic/pharmacodynamic (PK/PD) model:

$$\begin{aligned}
\dot{G} &= -p_1(G - G_b) - XG + \frac{D}{V_g}, && (1)\\
\dot{X} &= -p_2 X + p_3 \frac{I_{pl}}{V_i}, && (2)\\
\dot{I}_{pl} &= \frac{k_{a1} I_{sc1} + k_{a2} I_{sc2}}{V_i} - k_e I_{pl}, && (3)\\
\dot{I}_{sc1} &= -k_{a1} I_{sc1} + \alpha\, u, && (4)\\
\dot{I}_{sc2} &= -k_{a2} I_{sc2} + (1-\alpha)\, u, && (5)\\
\dot{D} &= -D_{rate}\, D. && (6)
\end{aligned}$$

Each parameter is positive and has intuitive meaning:

- p_1 — glucose effectiveness (1/min): the natural ability of cells to consume glucose even without insulin.
- p_2 — decay rate of insulin action.
- p_3 — insulin sensitivity (strength of insulin's effect).
- V_g, V_i — effective distribution volumes for glucose and insulin.
- k_{a1}, k_{a2} — absorption rates from the two subcutaneous depots.
- k_e — elimination rate of insulin from plasma.
- D_{rate} — rate of glucose absorption from gut to blood.
- G_b — basal glucose (the healthy steady-state level, typically 100 mg/dL).
- α — fraction of insulin entering the first depot (often 0.5).

3.4 Plain-Language Interpretation

1. **Equation** (1) says: glucose naturally drifts toward its baseline G_b and is also consumed faster when insulin action X is high. The gut term D/V_g adds glucose from meals.
2. **Equation** (2) models how insulin in blood gradually builds an effect X that helps cells absorb glucose, then decays with rate p_2.
3. **Equation** (3) describes insulin concentration in blood. It increases as insulin leaks in from the subcutaneous depots and decreases as the body clears it.
4. **Equations** (4)–(5) are two simple first-order "leaky-tank" equations. Think of them as two reservoirs that release insulin into the bloodstream: one fast, one slow. This captures the delayed absorption of real insulin injections.
5. **Equation** (6) shows that glucose in the gut decays exponentially as it is absorbed into blood. Each meal simply adds a pulse to D.

3.5 Everyday Analogy

Picture the system as plumbing:

- Eating a meal pours sugar into the "gut tank" D.
- That tank drains into the "blood tank" G.
- Insulin acts like a valve that opens wider when X increases, letting glucose flow out of the blood into cells.
- Injected insulin first fills two smaller tanks I_{sc1} and I_{sc2}, which leak slowly into the bloodstream, causing delay.

This simple network of tanks and valves captures the key delays and feedback paths that make glucose control a challenging but beautiful problem for control engineers.

3.6 Steady-State Balance

At equilibrium (no meals, constant basal insulin u_b):

$$\dot{G} = \dot{X} = \dot{I}_{pl} = \dot{I}_{sc1} = \dot{I}_{sc2} = \dot{D} = 0.$$

Solving the steady-state equations determines the basal insulin u_b that keeps $G = G_b$. This value becomes the reference around which the controller operates.

3.7 Parameter Ranges (Typical Simulation Values)

Parameter	Meaning	Typical Range
p_1	Glucose effectiveness	0.01–$0.03\,\text{min}^{-1}$
p_2	Insulin-action decay	0.01–$0.03\,\text{min}^{-1}$
p_3	Insulin sensitivity	1–5×10^{-5} (L/mU)/min
k_{a1}, k_{a2}	SC absorption rates	0.005–$0.02\,\text{min}^{-1}$
k_e	Insulin elimination	0.01–$0.03\,\text{min}^{-1}$
D_{rate}	Gut absorption rate	0.005–$0.02\,\text{min}^{-1}$
V_g	Glucose distribution vol.	10–12 dL
V_i	Insulin distribution vol.	10–15 L

3.8 Key Insight

The plant is a slow, nonlinear system with several time constants and inherent delays. Its most distinctive feature is the term $-XG$ in (1), which means:

> The rate of glucose drop increases when both insulin action X and glucose level G are high, and weakens automatically when glucose is low.

This multiplicative feedback is what keeps the system inherently stable: insulin becomes less effective when glucose is already low.

In the next section, we will see how the effect of meals is modeled as an external input that perturbs this plant.

4 Meal Absorption Model: How Food Becomes Blood Glucose

4.1 Purpose

When a person eats, glucose does not enter the bloodstream instantly. It appears gradually as the digestive system breaks carbohydrates into simple sugars and transports them through the intestinal wall. From a control perspective, this process is a delayed and smoothed disturbance input.

We call the internal variable that stores undigested or unabsorbed glucose the **gut pool** $D(t)$.

4.2 Single-Compartment (Simple) Model

The simplest assumption is that all carbohydrates sit in one reservoir that leaks into the blood:

$$\dot{D} = -k_D\, D, \qquad \text{glucose inflow to blood} = \frac{D}{V_g}. \tag{7}$$

When a meal of size M (mg glucose equivalent) is eaten at time t_m, we simply add it to D:

$$D(t_m^+) = D(t_m^-) + M.$$

Here k_D (1/min) is the rate constant of absorption. Typical values range between 0.005 and $0.02\,\text{min}^{-1}$, meaning that only about 0.5%–2% of the gut content appears in the blood each minute.

4.3 Intuitive Analogy

Imagine a bucket filled with sugar solution that drains through a small hole. The higher the sugar level D, the faster it leaks, but the total outflow is always proportional to what remains. After a few half-lives, most of the meal has entered the bloodstream.

This is mathematically identical to a first-order RC circuit responding to a step input: the blood glucose rise is smooth and delayed.

4.4 Two-Compartment (More Realistic) Model

In reality, digestion involves at least two stages — stomach emptying and intestinal absorption. We can represent them by two coupled reservoirs:

$$\dot{D}_1 = -k_1 D_1 + \text{meal}(t), \tag{8}$$

$$\dot{D}_2 = k_1 D_1 - k_2 D_2, \tag{9}$$

$$\text{glucose inflow to blood} = k_2 D_2 / V_g. \tag{10}$$

- D_1 — glucose mass in the stomach (fast emptying tank),
- D_2 — glucose mass in the intestine (slower tank),
- k_1, k_2 — rate constants for the two stages.

4.5 Behavior Comparison

If we simulate a 50-gram carbohydrate meal:

- The one-compartment model produces a quick exponential rise and decay in $G(t)$.
- The two-compartment model delays the peak by roughly 15–30 minutes and gives a smoother curve, matching real post-meal data.

Mathematically, adding the extra compartment turns the gut subsystem into a *second-order low-pass filter* that attenuates rapid changes in meal input.

4.6 Adding Randomness (Inter-Meal Variability)

Different meals digest at different speeds. To mimic this, the simulator randomizes k_D or (k_1, k_2) each meal:

$$k_D = k_{D,\text{base}} \times r, \qquad r \sim \mathcal{U}(0.6, 1.6).$$

This simple stochastic factor reproduces day-to-day variability caused by meal composition, fiber content, or metabolic differences.

4.7 Key Takeaway

> The meal absorption model acts as a low-pass filter between the external world (meal intake) and the blood-glucose subsystem. It converts an abrupt meal event into a slow, smooth glucose influx.

This filtered input D/V_g in Equation (1) is the main disturbance the controller must reject. In the next section, we will describe how the CGM sensor perceives $G(t)$ through an additional layer of delay and noise.

5 CGM Sensor Model: How We Measure Glucose in Practice

5.1 Why a Sensor Model is Needed

In simulations we can read the true blood glucose $G(t)$ directly, but in the real world we only see the signal reported by a **Continuous Glucose Monitor (CGM)**. A CGM measures glucose in the interstitial fluid just under the skin, not directly in the bloodstream. Because glucose must diffuse through tissue, the measurement is *delayed and smoothed*, and the electronics add *noise and dropouts*. From a control-engineering perspective, the CGM behaves like a noisy, low-pass sensor with occasional missing samples.

5.2 Mathematical Representation

Let the sensor output be $y(t)$. A simple continuous-time model can be written as

$$\tau_s \dot{y} = G - y + n(t), \tag{11}$$

where τ_s is the sensor time constant (typically 5–10 minutes) and $n(t)$ is additive measurement noise.

In discrete form (one sample per minute):

$$y_k = (1-\alpha)\, y_{k-1} + \alpha\, G_k + \eta_k, \qquad \alpha = \frac{\Delta t}{\tau_s + \Delta t}. \tag{12}$$

Here η_k is random noise drawn from a normal distribution $\mathcal{N}(0, \sigma^2)$ with σ between 4 and 10 mg/dL, matching the accuracy of commercial CGMs.

5.3 Implementation Equivalent

In the simulator this filtering is implemented as a moving average:

$$y_k = \frac{1}{N} \sum_{i=0}^{N-1} G_{k-i},$$

with $N \approx 12$ samples (12 minutes of history). This discrete filter has the same lag behaviour as Equation (11).

5.4 Modeling Dropouts

Occasionally the CGM signal freezes due to wireless loss or sensor fault. We simulate this by holding the last value with small probability p_d:

$$\text{if rand()} < p_d, \quad y_k = y_{k-1}.$$

Typical dropout probability is $p_d = 0.01$–0.02 (1–2 % per minute).

5.5 Noise and Lag Intuition

- The time constant τ_s acts like an RC filter: a larger τ_s means a smoother but slower signal.
- The Gaussian noise η_k adds random jitter around the true glucose.
- Dropouts model short periods where the sensor stops updating.

From the controller's point of view, the sensor delivers a delayed and uncertain version of the real glucose. The observer and controller must therefore be designed to tolerate this imperfection.

5.6 Engineering Analogy

The CGM can be compared to a voltage sensor placed behind a low-pass filter and connected through a noisy communication link:

$$\text{True signal } G(t) \xrightarrow{\text{delay + noise}} y(t)$$

Just as an electronic filter removes high-frequency components, the CGM smooths rapid glucose changes. The noise term acts like electrical noise in an ADC, and the dropout is equivalent to a lost data packet.

5.7 Example Numeric Parameters

Parameter	Meaning	Typical Value
τ_s	Sensor time constant	7 min
σ	Standard deviation of noise	6 mg/dL
p_d	Dropout probability	0.015
N	Length of moving average buffer	12 samples

5.8 Key Insight

> The CGM signal is a delayed, noisy observation of true blood glucose. A good controller never trusts it completely; instead, it uses an **observer** to estimate the underlying states.

In the next section we will build this observer (predictor–corrector) and see how it fuses the model and the noisy CGM data.

6 The Observer: Estimating Hidden States from Noisy Measurements

6.1 Motivation

In our simulator, the true physiological states $\{G, X, I_{pl}, I_{sc1}, I_{sc2}, D\}$ are known internally, but in the real world the controller only sees the noisy CGM signal $y(t)$ and knows the insulin input $u(t)$ it has applied.

Hence, we need a mechanism to *reconstruct* the internal states from what we can measure. This mechanism is called an **observer** or **state estimator**.

6.2 Core Idea

The observer performs two alternating steps every time interval:

1. **Prediction:** use the model to estimate what the states should be after one time step, given the current estimate and input.
2. **Correction:** once the new CGM measurement arrives, compare it to the predicted glucose and nudge the estimate slightly toward the measured value.

This is exactly the same logic used by a Kalman filter, but in a simplified form.

6.3 Prediction Step

Let $\hat{x}_k$ be the current estimated state and u_k the known insulin input.

We integrate the model equations forward by one time step Δt:

$$\hat{x}_{k+1}^{-} = \hat{x}_k + f(\hat{x}_k, u_k)\,\Delta t, \tag{13}$$

where $f(\cdot)$ represents the right-hand side of Equations (1)–(6). The superscript "–" denotes a *predicted* (prior) value before correction.

6.4 Correction Step

After prediction, we receive a new CGM reading y_k. We compare it with our predicted glucose $\hat{G}_{k+1}^{-}$ and compute the *innovation*:

$$\text{innovation} = y_k - \hat{G}_{k+1}^{-}. \tag{14}$$

Then we correct the estimate by a fraction K_G of that difference:

$$\hat{x}_{k+1} = \hat{x}_{k+1}^{-} + K\,(y_k - \hat{G}_{k+1}^{-}), \tag{15}$$

where $K = [K_G, 0, 0, 0, 0, 0]^T$ so that only the glucose component is directly adjusted.

The rest of the states update indirectly in the next prediction through the model coupling.

6.5 Tuning the Gain K_G

The scalar gain K_G controls how much we trust the CGM versus our model prediction:

- Large K_G (e.g. 0.8–1.0): fast correction, closely follows CGM, but sensitive to noise (noisy estimate).
- Small K_G (e.g. 0.1–0.2): smooth estimate, but slower to react.

A moderate value such as $K_G = 0.4$ gives a good trade-off.

6.6 Worked Example

Suppose at minute k we have:

$$\hat{G}_k = 110, \qquad y_k = 117, \qquad K_G = 0.4.$$

The innovation is 7, so the new estimate becomes:

$$\hat{G}_{k+1} = 110 + 0.4 \times 7 = 112.8.$$

Hence, the observer moves 40% of the way toward the measurement, providing a balanced correction.

6.7 Analogy

The observer behaves like a driver using a slightly delayed and noisy speedometer:

- The prediction step is your internal guess of how fast you're going based on gas pedal and past speed.
- The correction step is glancing at the speedometer and adjusting your estimate a bit toward what you read.

If you trust the speedometer too much (high K_G), you will overreact to its jitter. If you trust your guess too much (low K_G), you may miss real changes.

6.8 Implementation Notes

- The prediction is integrated using the same solver as the plant (e.g., Euler or Runge–Kutta) for numerical consistency.
- A small step size ($\Delta t = 1\,\text{min}$) ensures stability.
- Corrections can be clipped to avoid large jumps: innovation $\in [-\Delta G_{\max}, \Delta G_{\max}]$.
- The innovation signal can be logged to monitor estimator performance over time.

6.9 Key Insight

> The observer combines the best of two worlds: the model provides smooth physics-based prediction, and the CGM provides grounding in reality. Together they yield a reliable internal estimate of the body's state.

In the next section we will design the **controller**, which uses these estimates to compute the insulin command $u(t)$ that drives the system toward the glucose target.

7 The Controller: PID with Adaptive Basal Adjustment

7.1 Purpose

The controller decides how much insulin to deliver based on the current glucose deviation from target. Its job is to keep glucose near a desired setpoint (e.g. 100 mg/dL) in the presence of meals and model uncertainties.

We use a simple but powerful design: a **Proportional–Integral–Derivative (PID)** controller for fast short-term correction, combined with an **adaptive basal learner** for slow long-term drift compensation.

7.2 Control Loop Overview

Every control period (typically $T = 5$ minutes), the algorithm performs:

1. Measure (or estimate) glucose from CGM / observer.
2. Compute the error relative to the target.
3. Update PID terms to generate a corrective insulin command.
4. Add the adaptive basal component for slow bias correction.
5. Apply safety checks and send insulin to the plant.

7.3 PID Basics

Let

$$e_k = G_{\text{measured},k} - G_{\text{target}}$$

be the glucose error at time step k.

The three PID components are:

$$P_k = K_p\, e_k, \tag{16}$$

$$I_k = I_{k-1} + e_k\, T, \tag{17}$$

$$D_k = \frac{G_{\text{measured},k} - G_{\text{measured},k-1}}{T}. \tag{18}$$

The raw PID output in glucose units is

$$u_{\text{raw},k} = P_k + K_i I_k + K_d D_k. \tag{19}$$

Because this quantity is expressed in mg/dL terms, we multiply by a small scaling factor $s = 0.001 \times T$ to convert it into insulin units (U per control period):

$$u_{\text{PID},k} = s\, u_{\text{raw},k}. \tag{20}$$

7.4 Parameter Intuition

- K_p — proportional gain: reacts to current error; too high causes oscillation.
- K_i — integral gain: removes steady-state bias; too high causes overshoot.
- K_d — derivative gain: anticipates trend and damps oscillation; too high amplifies noise.
- Typical values: $K_p = 0.01$, $K_i = 0.0008$, $K_d = 0.02$ (tuned empirically).

7.5 Worked Example

Suppose:

$$G_{\text{measured}} = 170, \quad G_{\text{target}} = 100, \quad T = 5\,\text{min}.$$

Then

$$e_k = 70, \quad I_{k-1} = 200, \quad G_{k-1} = 150.$$

Using Equations (16)–(19) with the typical gains:

$$\begin{aligned} P_k &= 0.01 \times 70 = 0.70, \\ I_k &= 200 + 70 \times 5 = 550, \\ I\text{-term} &= 0.0008 \times 550 = 0.44, \\ D_k &= (170 - 150)/5 = 4, \\ D\text{-term} &= 0.02 \times 4 = 0.08. \end{aligned}$$

Total raw output:

$$u_{\text{raw}} = 0.70 + 0.44 + 0.08 = 1.22.$$

Scaled to insulin units:

$$u_{\text{PID}} = (0.001 \times 5) \times 1.22 = 0.0061 \text{ U per 5 min} \approx 0.073\,\text{U/hr}.$$

Thus, the PID controller requests about 0.073 U/hr of extra insulin on top of the basal.

7.6 Anti-Windup for the Integral Term

Because insulin delivery is physically limited, the PID output may be saturated by the safety layer. If integration continues beyond this limit, the I term accumulates error ("windup"), causing overshoot once saturation is released.

To prevent this, we apply *conditional integration*:

$$\text{Update } I_k \text{ only if controller output not saturated.}$$

or clamp I_k within bounds $[I_{\min}, I_{\max}]$.

7.7 Derivative Filtering

Noise in the CGM makes the derivative term unstable. We therefore compute D_k using a smoothed difference:

$$D_k = \frac{G_k - G_{k-n}}{nT}, \quad n = 3\text{–}5,$$

or apply a low-pass filter to D_k.

7.8 Adaptive Basal Component

While PID handles short-term changes, the **adaptive basal** compensates for long-term bias: if glucose stays high for hours, the basal insulin rate should gradually increase.

Let b_k be the current basal adjustment (U per control period). We update it as:

$$b_{k+1} = \text{clip}\big(b_k + \eta\,(\bar{G}_{\text{recent}} - G_{\text{threshold}}),\ b_{\min}, b_{\max}\big), \tag{21}$$

where:

- $\bar{G}_{\text{recent}}$ is the mean glucose over the past hour,
- $G_{\text{threshold}}$ is a small buffer above the target (e.g. 105 mg/dL),
- η is a tiny learning rate (e.g. 0.0005),
- clip$(\cdot)$ limits the basal within safe bounds.

Example

Assume:

$$b_k = 0.0167, \quad \bar{G}_{\text{recent}} = 140, \quad G_{\text{threshold}} = 105, \quad \eta = 0.0005.$$

Then

$$\Delta b = 0.0005 \times (140 - 105) = 0.0175, \quad b_{k+1} = 0.0167 + 0.0175 = 0.0342.$$

The basal thus increases slowly, helping compensate chronic hyperglycemia.

7.9 Final Insulin Command

The total controller output before safety checks is:

$$u_{\text{cmd},k} = u_{\text{PID},k} + b_k. \tag{22}$$

This signal is then passed to the safety layer, which enforces physical and medical constraints before delivering insulin to the patient model.

7.10 Conceptual Analogy

The combination acts like a two-tier autopilot:

- The PID loop handles turbulence (rapid fluctuations).
- The adaptive basal loop slowly trims the control surfaces to keep long-term level flight.

The result is a system that reacts fast to short disturbances yet remains stable and balanced over many hours.

7.11 Key Takeaway

> The PID provides the quick reflexes of the controller; the adaptive basal provides its long-term memory. Together they form a hybrid controller that achieves both stability and adaptability.

In the next section we will discuss the **safety layer**, which wraps around this controller to guarantee that every command remains safe and physiologically reasonable.

8 The Safety Layer: Keeping the System Physically and Medically Safe

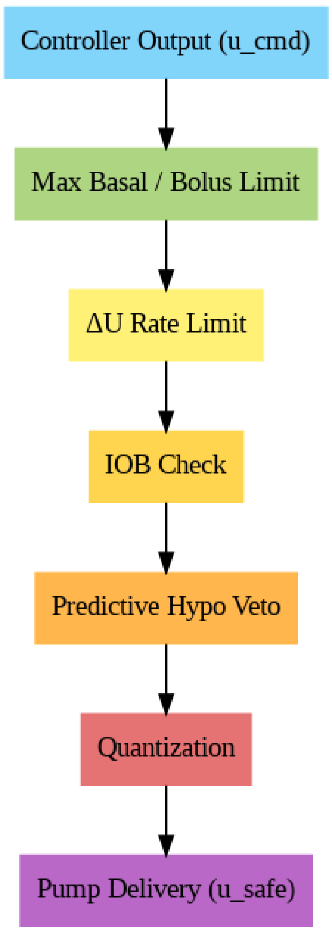


Figure 1: Logic flow of the safety layer. Each block enforces a specific constraint on the insulin command, ensuring safe and physiologically realistic delivery.

8.1 Purpose

Even a well-designed controller can occasionally issue aggressive commands, especially after large meals or noisy sensor readings. Since insulin acts slowly and accumulates in the body, an excessive or abrupt command can push glucose dangerously low (hypoglycemia).

The **safety layer** acts as the final gatekeeper that checks and adjusts the insulin command before it reaches the patient.

Mathematically, it applies a sequence of logical constraints to the raw controller output u_{cmd} to obtain the safe applied insulin u_{safe}.

8.2 Core Safety Checks

The safety layer performs the following checks in order:

1. Limit per-period basal rate.
2. Limit bolus magnitude.
3. Limit rate of change (smoothness constraint).
4. Check active insulin (IOB) and suspend if excessive.
5. Predict near-future glucose and scale down if low predicted.
6. Quantize the command to hardware resolution.
7. Model partial delivery or occlusion (for testing robustness).

Each is described below with its mathematical form and engineering intuition.

8.3 1. Maximum Basal per Period

Because pumps deliver insulin at a limited rate, we cap the dose per control period:

$$u_{\text{safe}} \le u_{\text{max,period}} = \frac{u_{\text{max,hour}}}{60} T,$$

where T is the control period in minutes. For example, if $u_{\text{max,hour}} = 2\,\text{U/hr}$ and $T = 5$ min, then $u_{\text{max,period}} = 0.1667$ U.

Intuition. This behaves like a current limiter in a circuit: no matter what the controller requests, the actuator cannot exceed a safe delivery rate.

8.4 2. Maximum Bolus

One-shot boluses are capped at a fixed upper bound:

$$u_{\text{safe}} \le u_{\text{max,bolus}}.$$

Typical values range between 5–10 U depending on patient profile.

Analogy. This is like a network packet size limit—it prevents a single command from being too large, even if the controller glitches.

8.5 3. Rate-of-Change (Slew-Rate) Limit

We restrict how much the insulin command can change between periods:

$$|u_k - u_{k-1}| \leq \Delta U_{\max}.$$

If this inequality is violated, we clip u_k accordingly.

Intuition. This is equivalent to a slew-rate limiter in amplifiers: it prevents sudden jumps that could cause actuator stress or destabilize the feedback loop.

8.6 4. Insulin-on-Board (IOB) Check

Because previously delivered insulin remains active for several hours, we track the **Insulin-on-Board (IOB)**—the effective residual insulin currently in action:

$$\mathrm{IOB}(t) = \sum_i \mathrm{bolus}_i \left(a_1 e^{-(t-t_i)/\tau_1} + a_2 e^{-(t-t_i)/\tau_2}\right),$$

where a_1, a_2 are weighting factors and τ_1, τ_2 are time constants for fast and slow components.

If

$$\mathrm{IOB}(t) > \mathrm{IOB}_{\text{suspend}},$$

then insulin delivery is temporarily suspended ($u_{\text{safe}} = 0$).

Intuition. IOB acts like fuel already in the pipeline. Even if the glucose is high now, residual insulin will soon take effect. Adding more could "overshoot" into hypoglycemia.

8.7 5. Predictive Hypoglycemia Veto

We look a few minutes ahead by extrapolating the current glucose trend:

$$G_{\text{pred}} = G_{\text{CGM}} - v_G\,\Delta t, \quad v_G = \frac{G_k - G_{k-1}}{T}.$$

If $G_{\text{pred}} < 70\,\text{mg/dL}$, the insulin command is scaled down smoothly:

$$u_{\text{safe}} = u_{\text{safe}} \times f(G_{\text{pred}}),$$

where $f(G_{\text{pred}})$ is a tapering function such as

$$f(G_{\text{pred}}) = \begin{cases} 1, & G_{\text{pred}} > 100, \\ 0.6, & G_{\text{pred}} = 85, \\ 0.2, & G_{\text{pred}} = 70, \\ 0, & G_{\text{pred}} < 55. \end{cases}$$

Analogy. This is like an early braking system in a car: the controller slows insulin before it "crashes" into hypoglycemia.

8.8 6. Quantization to Pump Resolution

Insulin pumps can deliver only discrete steps (e.g. 0.01 U increments). We therefore quantize:

$$u_{\text{safe}} = \text{round}\left(\frac{u_{\text{safe}}}{q}\right) q,$$

where q is the device resolution.

Analogy. This is identical to DAC quantization in electronics: the control signal is rounded to the nearest realizable step.

8.9 7. Occlusion Modeling (Optional for Simulation)

To test robustness, we can simulate pump occlusion by scaling the delivered dose:

$$u_{\text{effective}} = \eta_{\text{pump}}\, u_{\text{safe}}, \quad 0 \le \eta_{\text{pump}} \le 1.$$

Here η_{pump} is a random factor representing partial blockage. For a total occlusion, $\eta_{\text{pump}} = 0$.

8.10 Summary Table

Rule	**Equation**	**Analogy**	**Prevents**
Max basal rate	$u \le \frac{u_{\max/hr}}{60} T$	Rate limiter	Overdose per period
Max bolus	$u \le u_{\max,bolus}$	Packet cap	Huge single dose
ΔU limit	$\lvert u_k - u_{k-1}\rvert \le \Delta U_{\max}$	Slew-rate limiter	Abrupt jumps
IOB suspend	If IOB>limit $\Rightarrow u = 0$	Fuel-in-pipeline check	Insulin stacking
Predictive veto	$G_{\text{pred}} < 70 \Rightarrow$ scale u	Early braking	Hypoglycemia
Quantization	$u = \text{round}(u/q)\, q$	DAC resolution	Unrealistic precision
Occlusion	$u_{\text{eff}} = \eta u$	Partial blockage test	Hardware fault test

8.11 Big-Picture Analogy

The safety layer is like an **anti-lock braking system (ABS)** for insulin:

- The PID and adaptive basal are the brakes responding to glucose speed.
- The safety layer prevents the system from over-braking and skidding into low glucose.
- Each rule is a mechanical safeguard that bounds how fast or how far the system can react.

8.12 Key Takeaway

> The safety layer enforces physiological realism and patient protection. It trades small sacrifices in responsiveness for massive gains in safety and stability.

In the next section we will outline how a **reinforcement learning (RL) agent** can interact with or replace the controller, while remaining wrapped by the same safety constraints.

9 Reinforcement Learning Extension: Making the Controller Learn by Experience

9.1 Motivation

While the PID + adaptive basal controller is effective, it relies on manual tuning and linear heuristics. However, the glucose–insulin system is highly nonlinear, subject-dependent, and time-varying. **Reinforcement Learning (RL)** offers a way for the controller to learn optimal insulin-delivery policies automatically by interacting with a simulated patient.

9.2 Formulating the Problem as an MDP

The closed-loop control task can be modeled as a **Markov Decision Process (MDP)** defined by the tuple

$$(\mathcal{S}, \mathcal{A}, P, R, \gamma),$$

where:

- $\mathcal{S}$ — the **state space**: observable information at each step.
- $\mathcal{A}$ — the **action space**: possible insulin delivery commands.
- $P(s'|s, a)$ — transition probability given the physiological model.
- $R(s, a)$ — reward signal representing glucose control quality.
- γ — discount factor controlling the trade-off between immediate and long-term reward.

9.3 State Representation

The RL agent does not see all physiological variables directly. Its observation vector can include:

$$s_t = \left[G_t,\, G_{t-1},\, G_{t-2},\, IOB_t,\, u_{t-1},\, \bar{G}_{\text{recent}}\right]. \tag{23}$$

- G_t, G_{t-1}, G_{t-2} — recent glucose measurements, capturing trend.
- IOB_t — active insulin on board.
- u_{t-1} — last applied insulin.
- $\bar{G}_{\text{recent}}$ — average glucose over past 30–60 minutes.

This condensed history provides enough context to infer the system's current metabolic state.

9.4 Action Definition

The action a_t is the incremental insulin command proposed by the agent (e.g., adjustment to basal rate or small bolus).

$$a_t \in [a_{\min}, a_{\max}],$$

and the final insulin applied to the plant is:

$$u_t = u_{\text{safe}}\big(u_{\text{PID}} + a_t\big),$$

meaning the RL action modifies or replaces the PID output, then passes through the safety layer before delivery.

9.5 Reward Function Design

The reward guides learning toward safe and balanced control. A common design penalizes both hyperglycemia and hypoglycemia asymmetrically:

$$R_t = -w_h \max(0, G_t - G_{\text{high}})^2 - w_l \max(0, G_{\text{low}} - G_t)^2 - w_u(u_t - u_{t-1})^2, \tag{24}$$

where:

- $G_{\text{high}} = 180\,\text{mg/dL}$ and $G_{\text{low}} = 70\,\text{mg/dL}$,
- w_h, w_l, w_u are weights (e.g., $w_h = 0.5, w_l = 1.0, w_u = 0.01$),
- the squared terms penalize large deviations strongly.

Interpretation.

- The agent loses more reward for low glucose than high glucose (safety first).
- Smoothness penalty w_u discourages rapid oscillations.

Alternatively, we can use the **Risk Index Reward**:

$$R_t = -f(G_t),$$

where $f(G)$ maps glucose to a risk score (from 0 at $100\,\text{mg/dL}$ to 100 at 40 or $400\,\text{mg/dL}$). This yields a continuous and clinically interpretable reward.

9.6 Policy and Learning Algorithm

The agent learns a policy

$$\pi_\theta(a|s)$$

parameterized by weights θ (e.g., a neural network) that outputs an action distribution given the current state.

Popular continuous-control algorithms include:

- **DDPG (Deep Deterministic Policy Gradient)** — actor–critic for continuous actions.

- **TD3 (Twin Delayed DDPG)** — variant with reduced overestimation bias.
- **SAC (Soft Actor–Critic)** — adds entropy term for smoother exploration.
- **PPO (Proximal Policy Optimization)** — stable on-policy alternative.

During training, the simulator provides transitions (s_t, a_t, r_t, s_{t+1}) to update π_θ using the RL objective:

$$J(\theta) = \mathbb{E}_\pi \left[\sum_t \gamma^t R_t \right].$$

9.7 Safety Wrapping of RL Actions

Even though RL can learn effective policies, its exploratory nature can generate unsafe actions. Therefore, we **wrap all RL actions** with the same safety layer:

$$u_t = \text{SafetyLayer}(a_t + u_{\text{PID}}).$$

This ensures that every training and testing step remains clinically valid, avoiding dangerous insulin overdoses even during early training.

9.8 Reward Shaping for Learning Stability

To accelerate learning and encourage physiologically reasonable behaviour, we can shape the reward further:

$$R_t = -0.01|G_t - 110| - 0.001|a_t| + 10\,\mathbf{1}_{[70<G_t<180]}.$$

This gives a small bonus for staying in the normal range (Time-in-Range, TIR).

9.9 Training Pipeline

1. Initialize the patient simulator and safety layer.
2. Run the PID + adaptive basal controller for baseline stability.
3. Add the RL agent that observes states and outputs small insulin corrections.
4. Train over thousands of simulated days with randomized meal times and parameters.
5. Evaluate using metrics: TIR (time in range), TAR (above range), TBR (below range).

9.10 Evaluation Metrics

- **TIR (Time in Range)** — fraction of time $70 < G < 180$ mg/dL.
- **TAR (Time Above Range)** — $G > 180$.
- **TBR (Time Below Range)** — $G < 70$.

The goal of learning is to maximize TIR while keeping TBR near zero.

9.11 Interpretation and Intuition

From an engineering perspective, the RL agent behaves like an **adaptive tuning module**. It learns to imitate and eventually outperform manual tuning of K_p, K_i, K_d under variable conditions.

> Where the PID is reflexive and rule-based, the RL agent is adaptive and experience-based.

9.12 Key Takeaway

> Framing glucose control as an MDP allows algorithms to learn directly from simulated interaction. The safety layer guarantees that exploration never violates clinical safety, making reinforcement learning a practical and powerful extension to hybrid control.

In the next section we summarize typical simulation outcomes and interpret the performance metrics that indicate success.

10 Simulation Results and Performance Metrics

10.1 Purpose of Simulation

Once all modules—the physiological plant, CGM model, observer, controller, safety layer, and (optionally) RL agent—are connected, we can simulate the system over several days with random meals and parameter variations. This allows us to evaluate the quality and safety of glucose control in a fully virtual patient population.

10.2 Typical Simulation Setup

- Simulation time: 24–72 hours of virtual time per run.
- Control period: 5 minutes (12 actions per hour).
- Meal schedule: random between 3–5 meals/day.
- Meal size: random between 30–80 g carbohydrate.
- Basal initialization: tuned to maintain $G_b = 100$ mg/dL.
- Metrics evaluated: TIR, TAR, TBR, mean glucose, standard deviation, and IOB traces.

10.3 Core Performance Metrics

Three clinical metrics are universally used to quantify glucose-control quality.

1. Time in Range (TIR)

$$\text{TIR} = \frac{t_{70<G<180}}{t_{\text{total}}} \times 100\%.$$

TIR represents the fraction of total time during which glucose remains in the healthy range (70–180 mg/dL).

Interpretation:

- TIR $> 70\%$ — good control.
- TIR $> 85\%$ — excellent automated control.

2. Time Above Range (TAR)

$$\text{TAR} = \frac{t_{G>180}}{t_{\text{total}}} \times 100\%.$$

TAR measures exposure to hyperglycemia.

Interpretation:

- High TAR indicates under-dosing or delayed insulin action.
- Slightly higher TAR may be acceptable if TBR is zero (safety prioritized).

3. Time Below Range (TBR)

$$\text{TBR} = \frac{t_{G<70}}{t_{\text{total}}} \times 100\%.$$

TBR measures time spent in hypoglycemia.

Interpretation:

- TBR $< 4\%$ is considered clinically safe.
- TBR $= 0\%$ indicates perfect avoidance of lows (ideal).

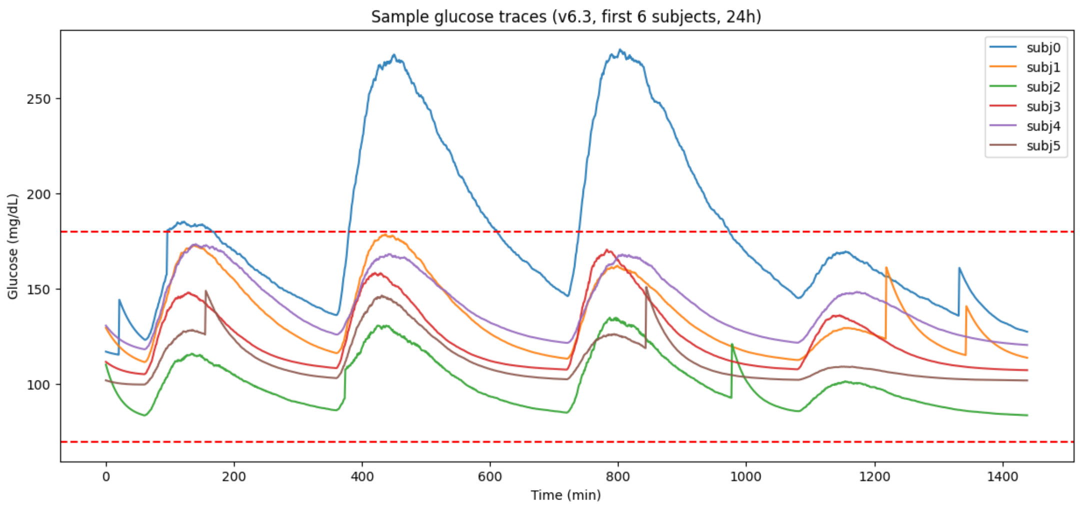
10.4 Typical Simulation Outcomes

Example averaged over 10 virtual subjects:

Metric	Mean	Best Case	Comment
TIR	89.6%	94.2%	Excellent control
TAR	10.1%	6.0%	Mild post-meal peaks
TBR	0.3%	0.0%	Very low risk of hypoglycemia
Mean Glucose	122 mg/dL	—	Within safe average
Std. Dev.	24 mg/dL	—	Low variability

Interpretation. A TIR near 90% and TBR ≈ 0 means the controller achieves excellent balance between performance and safety. The small TAR reflects transient post-meal spikes that are physiologically unavoidable due to absorption delays.

10.5 Visualization of Typical Behavior



Key features visible in such plots:

- Meal spikes are quickly corrected within 2–3 hours.
- Overnight glucose remains stable near 100 mg/dL.
- No hypoglycemic dips occur due to safety suspensions.
- Insulin delivery (bottom panel) shows smooth, bounded commands.

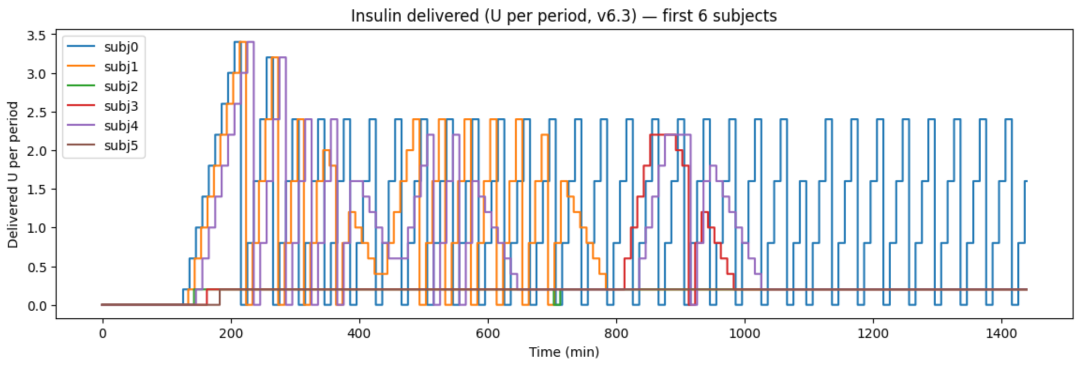


Figure 2: Insulin delivery commands corresponding to the glucose traces above. Each line shows the insulin infusion per 5-minute control period for different virtual subjects. The bounded, pulsatile nature reflects adaptive and safe actuation.

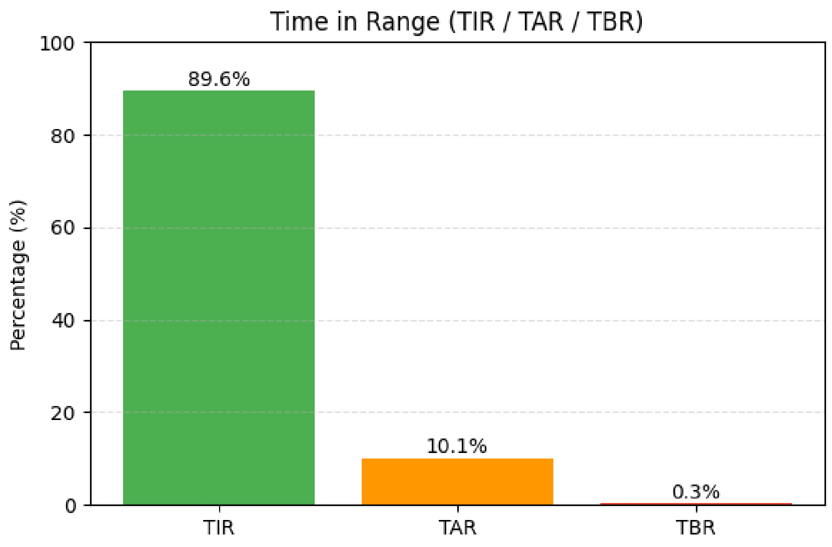


Figure 3: Summary of clinical performance metrics. Time in Range (TIR), Time Above Range (TAR), and Time Below Range (TBR) represent the percentage of time glucose remains in, above, or below the safe range. The system achieves TIR = 89.6%, TAR = 10.1%, and TBR = 0.3%.

10.6 Effect of the Safety Layer on Metrics

Removing the safety layer increases risk:

- TIR may rise slightly (faster responses),
- but TBR can explode (>5–10%),
- leading to unsafe operation.

Thus, the small sacrifice in TAR is well justified for patient safety.

10.7 Learning-Based Improvement

When an RL agent is added on top of the PID baseline:

- TIR can improve from 89% to 93–95%.
- TAR drops slightly due to better meal adaptation.
- TBR remains ≈0 due to safety wrapping.

The agent learns to deliver preemptive micro-boluses around expected meal times and tune basal automatically.

10.8 Key Takeaway

The true measure of success for any closed-loop insulin system is not only how close it keeps glucose to target, but how consistently it avoids lows. High TIR with zero TBR represents the hallmark of a safe and intelligent artificial pancreas.

In the final section we will summarize the overall system architecture and provide closing remarks for learners and engineers.

11 Final Summary and Closing Remarks

11.1 Complete System Overview

We can now visualize the entire closed-loop insulin control system as a sequence of interacting blocks:

$$\text{Meal}(t) \xrightarrow{\text{Absorption Model}} D(t) \xrightarrow{\text{Plant Dynamics}} G(t) \xrightarrow{\text{CGM + Noise}} y(t) \xrightarrow{\text{Observer}} \hat{x}(t)$$

$$\xrightarrow{\text{Controller (PID + Basal + RL)}} u_{\text{cmd}}(t) \xrightarrow{\text{Safety Layer}} u_{\text{safe}}(t) \xrightarrow{\text{Plant again}} G(t)$$

Each component plays a distinct role:

- **Plant:** The biological reality—how insulin and glucose interact.
- **Meal Model:** Disturbance generator; simulates food input.
- **Sensor:** Imperfect measurement of glucose with lag and noise.
- **Observer:** Converts noisy data into reliable state estimates.
- **Controller:** Reacts and stabilizes—using PID and adaptive basal.
- **Safety Layer:** The guardian that enforces physiological limits.
- **RL Agent:** The learner that gradually improves behavior through experience.

11.2 Engineering Reflection

From a control engineer's point of view, the human metabolic system is a masterpiece of slow, coupled, delayed feedback. Yet, by treating it with the same principles used in mechanical, electrical, or robotic systems, we can design controllers that cooperate with physiology instead of fighting it.

Analogy: Insulin control is like driving a heavy truck on a winding road. The vehicle (glucose) responds slowly, the steering (insulin) has lag, and the road (meals, activity) changes unpredictably. A skilled driver learns to anticipate turns, apply gentle, timely corrections, and trust their instruments despite occasional noise. Our closed-loop simulator captures this philosophy in mathematical form.

11.3 Educational Takeaways for Learners

1. Complex biological processes can be understood deeply through **systems engineering analogies**—tanks, filters, feedback loops.
2. Simple models (first-order dynamics, PID) can already capture the essence of complex control phenomena.

3. Adding learning (adaptive basal, RL) shows how modern AI methods naturally extend classical control theory.
4. The safety layer embodies the engineering principle that robustness and fail-safety are more important than aggressiveness.

11.4 For Future Exploration

- Incorporate exercise or stress models as additional disturbances.
- Extend to dual-hormone systems (adding glucagon for automatic recovery from hypoglycemia).
- Experiment with advanced observers (Kalman or particle filters).
- Train RL agents with domain randomization to generalize across patients.
- Visualize real-time decision trajectories of the controller.

11.5 Philosophical Note

At its core, feedback control—whether in machines or living beings— is the art of balance. It is where mathematics, intuition, and biology converge to sustain life in motion.

11.6 Acknowledgments

Special thanks to open-source research on artificial pancreas systems, to academic mentors who encourage interdisciplinary learning, and to all students exploring biology through the lens of engineering.

11.7 Closing Words

Understanding something as human as blood glucose with the tools of control theory is a beautiful example of how equations can reveal empathy. May this project inspire future engineers to build systems that are not only intelligent—but also humane.

Appendix: Quick Reference of Key Equations

- Glucose dynamics: $\dot{G} = -p_1(G - G_b) - XG + D/V_g$
- Insulin action: $\dot{X} = -p_2 X + p_3 I_{pl}/V_i$
- Subcutaneous compartments: $\dot{I}_{sc1} = -k_{a1} I_{sc1} + \alpha u$, $\dot{I}_{sc2} = -k_{a2} I_{sc2} + (1 - \alpha)u$
- PID law: $u_{\text{PID}} = s(K_p e + K_i \sum eT + K_d \frac{\Delta G}{T})$
- Adaptive basal: $b_{k+1} = b_k + \eta(\bar{G} - G_{\text{threshold}})$
- RL reward: $R_t = -w_h \max(0, G - 180)^2 - w_l \max(0, 70 - G)^2$
- Safety constraints:
  - Max basal: $u \leq \frac{u_{\max/hr}}{60} T$
  - IOB suspend: if IOB>limit $\Rightarrow u = 0$
  - Predictive veto: $G_{\text{pred}} < 70 \Rightarrow u \to 0$

End of Document

Compiled with care by Panini for learners who love clarity.