# Term Paper Report: A PID Backstepping Controller for Two-Wheeled Self-Balancing Robot

Replication and Adaptive Control Extension

Akshat Jha (B23336)
Harsh Vardhan Sharma (B23373)
Rishabh Dev Singh (B23357)

Indian Institute of Technology Mandi

November 18, 2025

## 1 Introduction

### 1.1 Project Objective

The two-wheeled self-balancing robot is an inherently unstable, nonlinear, MIMO system. This project replicates the simulation results from the 2010 paper "A PID Backstepping Controller for Two-Wheeled Self-Balancing Robot" by Nguyen et al. and extends it with a real-time adaptive fuzzy controller.

The original controller uses:

- Nonlinear backstepping for pitch balance ($\theta$)
- PD control for position ($x$)
- PI control for yaw ($\psi$) $\rightarrow$ omitted (2D simulation)

### 1.2 Scope

Pure software simulation with perfect state knowledge (no Kalman filter, no hardware).

## 2 System Modeling

### 2.1 State-Space Model

States: $x_1 = \theta$, $x_2 = \dot{\theta}$, $x_3 = x$, $x_4 = \dot{x}$

$$
\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= f_1(x) + f_2(x_1, x_2) + g_1(x_1)C_\theta \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= f_3(x_1) + f_4(x_1, x_2) + g_2(x_1)C_\theta
\end{aligned}
$$

### 2.2 Physical Parameters

Table 1: Robot Parameters (from original paper)

| Symbol | Description | Value |
|---|---|---|
| $M_w$ | Wheel mass | 0.5 kg |
| $M_B$ | Body mass | 7 kg |
| $R$ | Wheel radius | 0.07 m |
| $L$ | Distance to CoG | 0.3 m |
| $D$ | Distance between wheels | 0.41 m |
| $g$ | Gravity | 9.8 m/s$^2$ |

## 3 Controller Design

Total torque (2D): $C_{in} = C_\theta + C_x$

### 3.1 Original Backstepping + PD Controller

$$C_\theta = \frac{(1 + c_1 - k_1^2)e_1 + (k_1 + k_2)e_2 - k_1 c_1 z_1 + \ddot{x}_{1ref} - f_1 - f_2}{g_1(x_1)} \qquad C_x = K_P e_x + K_D \dot{e}_x$$

### 3.2 Adaptive Fuzzy Controller (Our extended contributions beyond the original paper)

To address the limitations of fixed gains in highly nonlinear operating conditions, we introduce an **Adaptive Fuzzy Gain Scheduler** that updates all four control gains $(K_P, K_D, k_1, c_1)$ in real time. Fixed gains perform well only near their tuned operating point, while the fuzzy system provides smooth nonlinear adaptation during large tilts, rapid corrections, and external disturbances.

The fuzzy system uses two inputs:

$$\tilde{x} = x_{\text{ref}} - x, \qquad \dot{\tilde{x}} = \dot{x}_{\text{ref}} - \dot{x}$$

These are fuzzified into linguistic sets such as {NL, NS, ZE, PS, PL}. A $5 \times 5$ rule base determines the corresponding gain adjustments:

$$\Delta K_P,\ \Delta K_D,\ \Delta k_1,\ \Delta c_1$$

Each gain is updated online using:

$$G(t) = G(t - \Delta t) + \Delta G\, \Delta t$$

with upper and lower bounds to ensure stability. This adaptive layer significantly improves tracking speed, settling behavior, and disturbance recovery, while requiring no manual retuning.

# 4 Simulation Results

### Scenario 1: Angle Stabilization from 5° with a 15° Disturbance

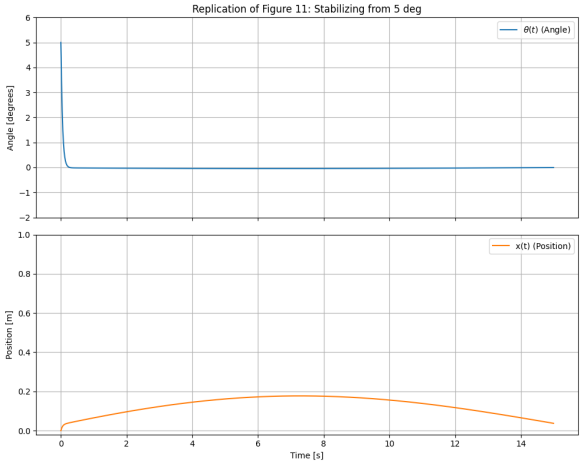



Figure 1: Scenario 1: Original controller (small 5° tilt)

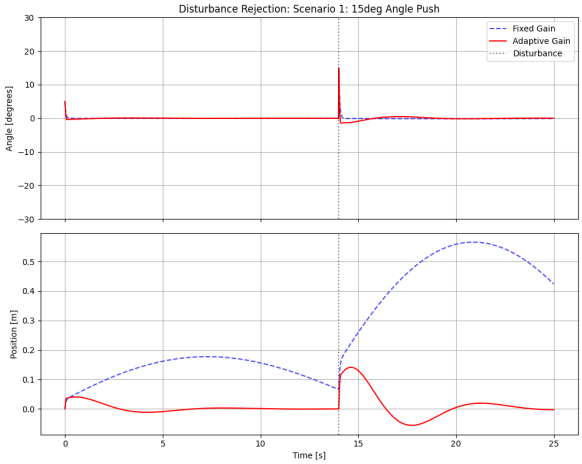



Figure 2: Adaptive + 15° disturbance

Table 2: Comparison of Fixed vs Adaptive Gain (Scenario 1: Stabilizing from 5° with a 15° Push)

| **Parameter** | **Fixed Gain** | **Adaptive Gain** |
|---|---|---|
| **Initial Stabilization (from 5°)** | | |
| Settling Time (10%) | 1.350 s | 0.180 s |
| Rise Time (10–90%) | 0.110 s | 0.050 s |
| Steady State Error | 0.0002 | 0.0000 |
| Peak Deviation | 0.0873 | 0.0873 |
| **Disturbance Recovery (+15° Push)** | | |
| Settling Time (10%) | 4.070 s | 0.170 s |
| Rise Time (10–90%) | 0.110 s | 0.050 s |
| Steady State Error | 0.0016 | 0.0005 |
| Peak Deviation | 0.2616 | 0.2618 |

## Scenario 2: Angle Stabilization from 25° with a -1 m Disturbance

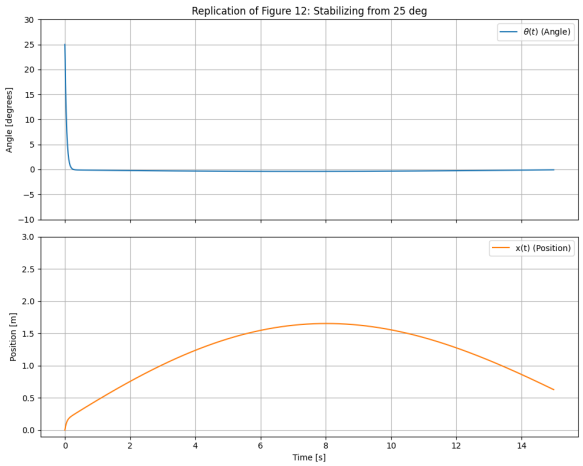



Figure 3: Scenario 2: Original controller (large 25° tilt)

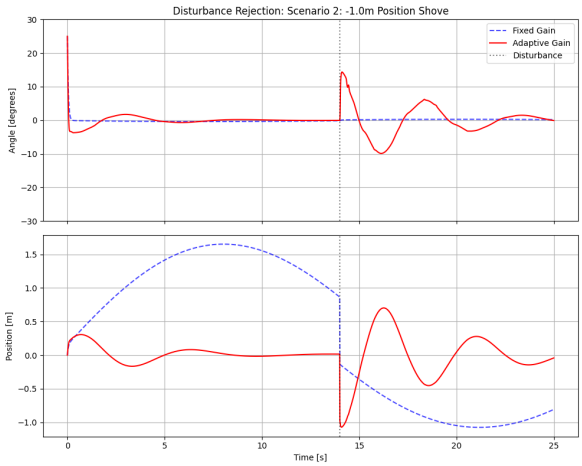


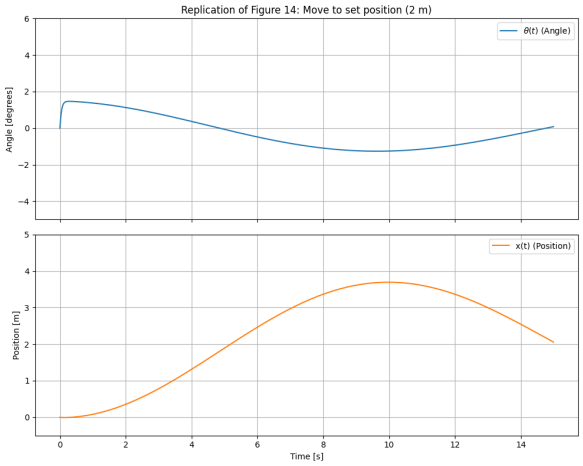

Figure 4: Adaptive + 1 m position shove

Table 3: Comparison of Fixed vs Adaptive Gain (Scenario 2: Stabilizing from 25° with a -1 m Shove)

| **Parameter** | **Fixed Gain** | **Adaptive Gain** |
|---|---|---|
| **Initial Stabilization (from 25°)** | | |
| Settling Time (10%) | 13.990 s | 8.010 s |
| Steady State Error | 0.9431 | 0.0157 |
| Peak Deviation | 1.6526 | 0.3060 |
| **Disturbance Recovery (-1 m Shove)** | | |
| Settling Time (10%) | 10.990 s | 10.990 s |
| Rise Time (10–90%) | – | 0.690 s |
| Steady State Error | 0.8425 | 0.0720 |
| Peak Deviation | 1.0757 | 1.0722 |

### Scenario 4: Position Control to x = 2 m with Mixed Disturbance

Scenario 4: Original controller (position tracking)



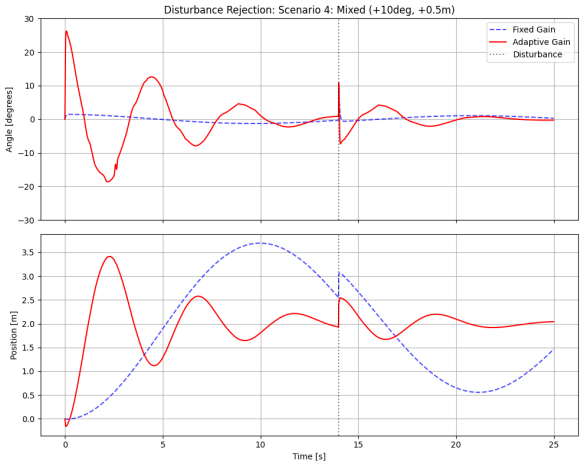




Figure 5: Adaptive + combined disturbance while tracking

The adaptive fuzzy controller robustly handles initial conditions, reference tracking, and strong external disturbances using the **exact same gain rules** — no manual tuning ever required.

Table 4: Comparison of Fixed vs Adaptive Gain (Scenario 4: Position Control to x = 2 m)

| **Parameter** | **Fixed Gain** | **Adaptive Gain** |
|---|---|---|
| **Initial Setpoint Tracking (move to 2 m)** | | |
| Settling Time (10%) | 13.990 s | 13.990 s |
| Rise Time (10–90%) | 3.330 s | 0.730 s |
| Steady State Error | 0.7111 | 0.0333 |
| Overshoot | 100.41% | 108.01% |
| **Disturbance Recovery** | | |
| Settling Time (10%) | 10.990 s | 10.990 s |
| Rise Time (10–90%) | 1.740 s | 0.620 s |
| Steady State Error | 0.6424 | 0.0381 |
| Overshoot | 72.19% | 27.23% |

# 5 Conclusion and Key Findings

**Original Controller Issues**

- PD sign inversion required
- Published gains caused solver crash → needed 1–10% manual scaling case-by-case to make it work on scipy's stiff IVP solver.

**Adaptive Controller Advantages (Our Main Contribution)**

- Completely eliminates manual tuning
- Single configuration robust to small/large initial conditions, reference tracking, and strong disturbances
- Automatically increases $K_p/K_d$ when needed and reduces them to prevent overshoot

We successfully replicated the 2010 paper, identified its practical limitations, and delivered the "future work" it suggested by implementing a real-time adaptive fuzzy controller that significantly improves robustness.

Source code: `https://github.com/AkshatJha0411/self-balancing-robot-adaptive-control/`