# Prescribed-time Sliding Mode Control of Inverted Pendulum

Author: Paradesi Reshwanth

February 24, 2025

**Abstract**

This technical report presents the modeling, controller design, and simulation framework for controlling a simple pendulum using advanced sliding mode control strategies. In particular, we derive a Prescribed-Time Super-Twisting Sliding Mode Controller (PT-STSMC) tailored to the pendulum dynamics, such that the tracking error is driven to zero before a user-specified convergence time $T_a$, independent of initial conditions. The derivation is fully developed for the pendulum model, and practical implementation considerations (numerical caps, gain selection, and simulation placeholders) are provided. MATLAB-generated simulation results and robustness experiments are intended to be inserted in the simulation section as figures.

# Contents

# Chapter 1

# Introduction

Sliding Mode Control (SMC) is a robust control framework for systems with matched uncertainties and disturbances. The Super-Twisting Algorithm (STA) is an effective higher-order sliding algorithm that delivers continuous control signals and reduces chattering while preserving robustness. Prescribed-time control augments classical control by guaranteeing convergence within a pre-specified finite time $T_a$, independent of initial conditions. Combining STA with a prescribed-time design yields PT-STSMC: a controller that is continuous (chattering-reduced) and guarantees convergence before a deadline.

This report focuses on:

- deriving PT-STSMC for the simple pendulum model,
- providing a ready-to-implement control law for simulation (MATLAB),
- outlining practical implementation notes for numerical stability and gain selection.

# Chapter 2

# System Modelling

## 2.1 Physical Description

Consider a simple pendulum consisting of a point mass $m$ attached to a rigid, massless rod of length $l$. A torque $\tau$ is applied at the pivot to control the angular position $\theta$ (measured from the vertical). The model includes viscous damping $k\dot{\theta}$.

## 2.2 Equations of Motion

Applying rotational dynamics:

$$ml^2\ddot{\theta} + k\dot{\theta} + mgl\sin(\theta) = \tau. \tag{2.1}$$

For compactness define $I \triangleq ml^2$. Then:

$$I\ddot{\theta} + k\dot{\theta} + mgl\sin\theta = \tau.$$

## 2.3 State-Space Representation

Let $x_1 = \theta$, $x_2 = \dot{\theta}$. The state-space model becomes:

$$\dot{x}_1 = x_2, \tag{2.2}$$

$$\dot{x}_2 = \frac{1}{I}\big(\tau - kx_2 - mgl\sin x_1\big). \tag{2.3}$$

## 2.4 Parameters

Nominal parameters used in simulation :

$$m = 0.5 \text{ kg}, \quad l = 1 \text{ m}, \quad k = 0.5 \text{ Nms}, \quad g = 9.81 \text{ m/s}^2.$$

## 2.5 Illustration

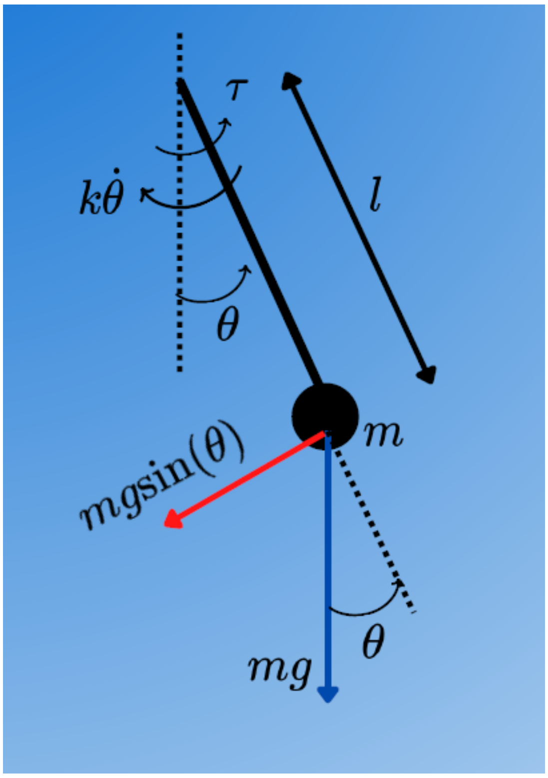


Figure 2.1: Pendulum schematic: torque input $\tau$, damping $k\dot{\theta}$.

# Chapter 3

# Prescribed-Time Super-Twisting Sliding Mode Control (PT-STSMC)

This chapter provides a step-by-step derivation of the PT-STSMC for the pendulum model (2.1). The derivation covers the shaping functions, time-varying sliding surface, desired sliding dynamics, Super Twisting Algorithm embedding, and the saturated torque law ready for implementation.

## 3.1 Control Objective

Design a control torque $\tau(t)$ such that the tracking error $e(t) = \theta(t) - \theta_{ref}(t)$ satisfies

$$e(t) \to 0 \quad \text{and} \quad \dot{e}(t) \to 0 \quad \text{for all } t \in [0, T_a),$$

i.e., the error is driven to zero before a user-specified convergence time $T_a > 0$, independent of initial conditions.

## 3.2 Prescribed-Time Shaping Function

Prescribed-time shaping function (for $t \in [0, T)$):

$$\phi(t; T, p) \; = \; \left(\frac{T}{T - t}\right)^p, \qquad p > 0, \tag{3.1}$$

with time derivative

$$\dot{\phi}(t; T, p) \; = \; \frac{pT}{(T - t)^{p+1}}. \tag{3.2}$$

Important properties:

- $\phi(t)$ is positive and $\lim_{t \to T^-} \phi(t) = +\infty$.

- $\phi$ grows rapidly as $t$ approaches $T$; implement with numerical caps in simulation.

We will use two shaping functions:

$$\phi_a(t) \triangleq \phi(t; T_a, p_a) \quad \text{(main)}, \qquad \phi_{as}(t) \triangleq \phi(t; T_{as}, p_{as}) \quad \text{(auxiliary)},$$

where $T_{as} \le T_a$ and $p_a, p_{as} > 0$.

## 3.3 Error and Sliding Surface

Tracking error and its derivative

$$e(t) = \theta(t) - \theta_{ref}(t), \qquad \dot{e}(t) = \dot{\theta}(t) - \dot{\theta}_{ref}(t).$$

Choosing a time-varying sliding surface:

$$\sigma(t) \;=\; \dot{e}(t) + \big(c_1 + c_2\phi_a(t)\big)\, e(t), \tag{3.3}$$

with constants $c_1, c_2 > 0$. The time-varying gain $c_2\phi_a(t)$ ensures an increasingly strong correction as $t \to T_a$.

## 3.4 Differentiation of the Sliding Surface

Differentiate $\sigma$:

$$\dot{\sigma} \;=\; \ddot{e} + \big(c_1 + c_2\phi_a(t)\big)\dot{e} + c_2\dot{\phi}_a(t)\, e. \tag{3.4}$$

Using the pendulum dynamics (and $I = ml^2$) we have:

$$\ddot{e} \;=\; \ddot{\theta} - \ddot{\theta}_{ref} \;=\; \frac{1}{I}\big(\tau - k\dot{\theta} - mgl\sin\theta\big) - \ddot{\theta}_{ref}.$$

Substitute into (3.4):

$$\dot{\sigma} \;=\; \frac{1}{I}\big(\tau - k\dot{\theta} - mgl\sin\theta\big) - \ddot{\theta}_{ref} + \big(c_1 + c_2\phi_a\big)\dot{e} + c_2\dot{\phi}_a e. \tag{3.5}$$

## 3.5 Desired Time-Varying Sliding Dynamics

We impose a desired first-order time-varying dynamics for $\sigma$ that contains both a (time-varying) linear stabilizing term and a Super-Twisting corrective term:

$$\dot{\sigma} \;=\; -\big(c_3 + c_4\phi_{as}(t)\big)\sigma - u_{sta}(t), \tag{3.6}$$

where $c_3, c_4 > 0$. The auxiliary shaping $\phi_{as}$ (possibly with $T_{as} < T_a$) allows us to schedule when the STA should act more aggressively.

## 3.6 Super-Twisting Correction Term

Selecting the continuous Super-Twisting structure for $u_{sta}$:

$$u_{sta}(t) = \lambda\sqrt{|\sigma|}\,\mathrm{sign}(\sigma) + w(t), \tag{3.7}$$

$$\dot{w}(t) = W\,\mathrm{sign}(\sigma), \qquad W > 0,\ \lambda > 0. \tag{3.8}$$

$u_{sta}$ provides a continuous corrective action that compensates for uncertainties and eliminates chattering typical of first-order SMC.

## 3.7 Solving for the Control Torque $\tau$

Equate (3.5) and (3.6). Rearranged:

$$\frac{1}{I}\big(\tau - k\dot{\theta} - mgl\sin\theta\big) - \ddot{\theta}_{ref} + \big(c_1 + c_2\phi_a\big)\dot{e} + c_2\dot{\phi}_a e = -\big(c_3 + c_4\phi_{as}\big)\sigma - u_{sta}. \tag{3.9}$$

Multiply both sides by $I$ and solve for $\tau$:

$$\tau = k\dot{\theta} + mgl\sin\theta + I\Big(\ddot{\theta}_{ref} - \big(c_1 + c_2\phi_a\big)\dot{e} - c_2\dot{\phi}_a e - \big(c_3 + c_4\phi_{as}\big)\sigma - u_{sta}\Big). \tag{3.10}$$

## 3.8 Final Controller Law

Insert $\sigma$ from (3.3) and $u_{sta}$ from (3.7) into (3.10) to produce an implementable torque command:

$$\sigma = \dot{e} + \big(c_1 + c_2\phi_a(t)\big)e, \tag{3.11}$$

$$u_{sta} = \lambda\sqrt{|\sigma|}\,\mathrm{sign}(\sigma) + w, \quad \dot{w} = W\,\mathrm{sign}(\sigma), \tag{3.12}$$

$$\begin{aligned} \tau = k\dot{\theta} + mgl\sin\theta + I\Big(&\ddot{\theta}_{ref} - \big(c_1 + c_2\phi_a\big)\dot{e} - c_2\dot{\phi}_a e \\ &- \big(c_3 + c_4\phi_{as}\big)\sigma - u_{sta}\Big). \end{aligned} \tag{3.13}$$

This expression is ready to implement in MATLAB. In discrete-time simulation use an ODE integrator or forward Euler with sufficiently small timestep.

# Chapter 4

# Simulation Results

This chapter contains placeholders for the MATLAB-generated plots. Replace the boxed placeholders by `\includegraphics{<file>}` with your figure filenames.

## 4.1 Simulation Setup

Simulation parameters (example):

$$T_{\text{sim}} = 10 \text{ s}, \quad \Delta t = 1 \times 10^{-3} \text{ s}, \quad T_a = 1 \text{ s}, \quad T_{as} = 0.5 \text{ s}.$$

Reference: $\theta_{ref} = 85°$ for $t \in [0, 5)$ s, then $\theta_{ref} = 170°$ for $t \geq 5$ s.

## 4.2 Plots

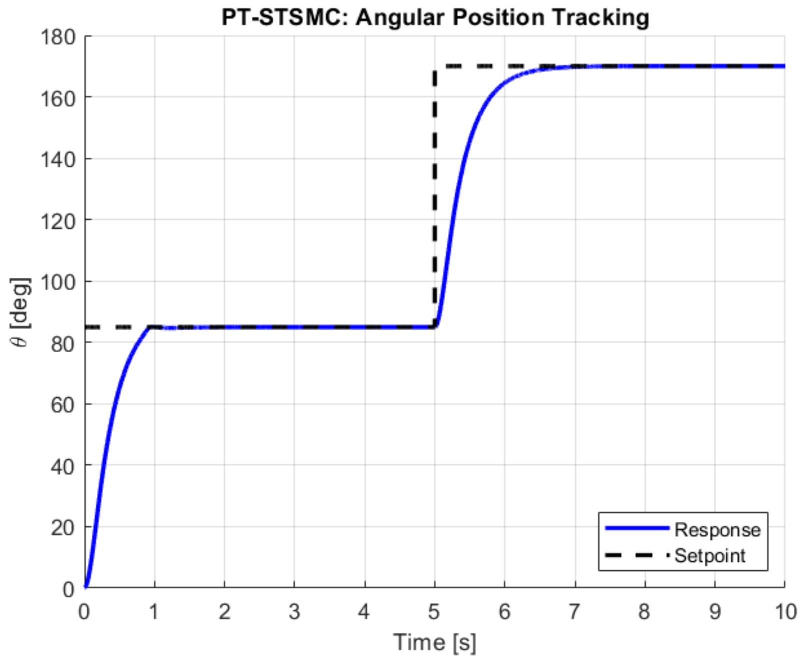


Figure 4.1: Tracking performance: $\theta(t)$ vs $\theta_{ref}(t)$.

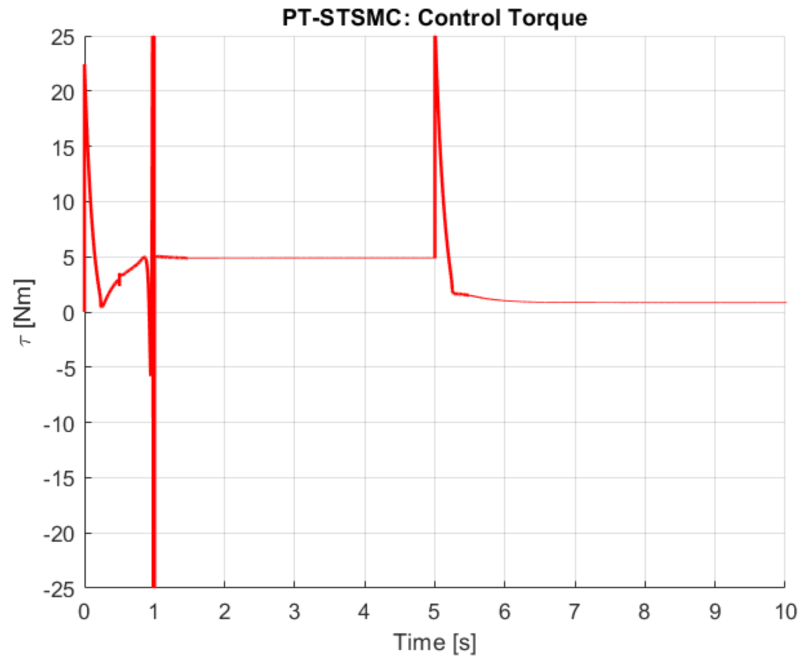


Figure 4.2: Control torque $\tau(t)$.

# Chapter 5

# Conclusion

## 5.1 Conclusion

We derived a Prescribed-Time Super-Twisting Sliding Mode Controller (PT-STSMC) for the pendulum and provided a ready-to-implement torque law. The design blends aggressive time-varying gains that enforce a deadline with the continuous STA corrective action to reduce chattering.

## MATLAB Code

```matlab
%% 1. WORKSPACE INITIALIZATION
clear; clc; close all;

%% 2. PARAMETER DEFINITIONS
% --- Plant Parameters ---
p.l = 1; p.k = 0.5; p.g = 9.81; p.m = 0.5;
I = p.m*p.l^2; % Moment of inertia

% --- PT-SMC Gains ---
c.c1 = 2.0; c.c2 = 1.0;
c.c3 = 3.0; c.c4 = 1.0;

% --- Super-Twisting Algorithm (STA) Gains ---
c.lambda = 10.0; % STA Proportional Gain
c.W = 5.0;       % STA Integral Gain

% --- Prescribed Times ---
T_as = 0.5;     % Prescribed time for sliding surface to converge (s)
T_a  = 1;       % Prescribed time for states (error) to converge (s)

% --- Simulation Parameters ---
T_sim = 10; dt = 1e-3; % Increased simulation time to 10s
```

```matlab
N = round(T_sim/dt); t = (0:N-1)'*dt;

% --- Time-Varying Setpoint Definition ---
ref1_deg = 85;
ref2_deg = 170;

% --- Pre-allocation and Initialization ---
theta = zeros(N,1); theta_dot = zeros(N,1); tau = zeros(N,1);
setpoint_log_deg = zeros(N,1); % To log the setpoint for plotting
w = 0; % STA integral term initialization

%% 3. SIMULATION EXECUTION
for k = 2:N
    % --- Time-Varying Setpoint ---
    if t(k) < 5
        setpoint_deg = ref1_deg;
    else
        setpoint_deg = ref2_deg;
    end
    setpoint = setpoint_deg * pi/180;
    setpoint_log_deg(k) = setpoint_deg; % Log for plotting

    % --- State Errors ---
    e  = setpoint - theta(k-1);
    ed = 0 - theta_dot(k-1); % Reference velocity is zero

    % --- Time-Varying Gains Calculation (Using Original Functions) ---
    phi_a     = phi_fun(t(k), 0, T_a, 1);
    phi_a_dot = phi_dot_fun(t(k), 0, T_a, 1);
    phi_as    = phi_fun(t(k), 0, T_as, 1);

    % --- Sliding Surface ---
    S = ed + (c.c1 + c.c2*phi_a)*e;

    % --- Super-Twisting Algorithm ---
    u_sta = c.lambda * sqrt(abs(S)) * sign(S) + w;
    w_dot = c.W * sign(S);
    w = w + dt * w_dot; % Integrate w

    % --- Desired Sliding Variable Dynamics ---
    Sdot_des = -(c.c3 + c.c4*phi_as)*S - u_sta;

    % --- Equivalent Control Law Calculation ---
    other_plant_terms = p.k*theta_dot(k-1) + p.m*p.g*p.l*sin(theta(k-1));
    Sdot_deriv_terms = (c.c1 + c.c2*phi_a)*ed + c.c2*phi_a_dot*e;

    tau(k) = other_plant_terms - I * (Sdot_des - Sdot_deriv_terms);
```

```

    % --- Apply Actuator Limits ---
    tau(k) = max(-25, min(25, tau(k)));

    % --- Integrate Plant Dynamics ---
    theta_ddot  = (tau(k) - other_plant_terms) / I;
    theta_dot(k) = theta_dot(k-1) + dt*theta_ddot;
    theta(k)    = theta(k-1)    + dt*theta_dot(k); % Semi-implicit Euler
end
% Ensure first value of setpoint log is correct for plotting
setpoint_log_deg(1) = ref1_deg;

%% 4. PLOTTING RESULTS
figure('Name','PT-STSMC - Angular Position');
hold on; grid on;
plot(t, theta*180/pi, 'b', 'LineWidth', 2);
plot(t, setpoint_log_deg, 'k--', 'LineWidth', 2, 'DisplayName', 'Setpoint');
xlabel('Time [s]');
ylabel('\theta [deg]');
title('PT-STSMC: Angular Position Tracking');
legend('Response', 'Setpoint', 'Location', 'SouthEast');

figure('Name','PT-STSMC - Control Torque');
hold on; grid on;
plot(t, tau, 'r', 'LineWidth', 1.5);
xlabel('Time [s]');
ylabel('\tau [Nm]');
title('PT-STSMC: Control Torque');

%% 5. PRESCRIBED-TIME HELPER FUNCTIONS
function out = phi_fun(t,t0,T,p)
    if t>=t0 && t<t0+T
        out = mu_dot_fun(t0,T,t,p)/(mu_fun(t0,T,t,p));
    else
        out = p/T;
    end
end

function out = mu_fun(t0,T,t,p)
    if t>=t0 && t<t0+T
        out = (T/(t0+T-t))^p;
    else
        out = 1;
    end
end

function out = phi_dot_fun(t,t0,T,p)
```

```
    if t>=t0 && t<t0+T
        out = (p/T^2)*(mu_fun(t0,T,t,p)^(2+(1/p)));
    else
        out = 0;
    end
end

function out = mu_dot_fun(t0,T,t,p)
    if t>=t0 && t<t0+T
        out = (p/T)*(mu_fun(t0,T,t,p)^(1+(1/p)));
    else
        out = 0;
    end
end
```

Listing 1: Full MATLAB Code for PT-STSMC Simulation