Sliding Mode and LQR based Hybrid Control for Rotary Inverted Pendulum

Harishankar M
B-Tech Electrical Engineering

September 7, 2025

Abstract: The project aims to design and implement a hybrid controller based on LQR and Sliding mode Control algorithms and experimentally demonstrate it on a rotary inverted pendulum

1 Introduction

Sliding Mode Control (SMC) is a widely used nonlinear control method known for its robustness against matched perturbations, theoretically ensuring superior control accuracy. However, in practical applications, the chattering phenomenon prevents the realisation of the ideal sliding mode.

On the other hand, the Linear Quadratic Regulator (LQR) controller is one of the most sought-after state space controllers as it is easier to model and has a straightforward implementation.

In this project we try to implement a hybrid controller which utilises both LQR and sliding mode control(SMC) techniques to better controller performance over the system. The project is based on Updating LQR to SMC

2 Problem Statement

We consider the problem of stabilisation of a system with single input whose linear plant model is given by eqn 1.

$$\frac{d\vec{x}}{dt} = Ax + Bu \tag{1}$$

Eqn 1 is written for $t \geq 0$ and $x(t) \in R^n$ - the system state, $u(t) \in R$ - the control input, $A \in R^{n*n}$ and $B \in R^{n*1}$ are system matrices assumed to be known. We model the project's controller, assuming a mathematical model's unavailability for the system's uncertainties and disturbances.

The whole state $x(\vec{t})$ is assumed to be available (measured or estimated), and the pair A, B is controllable. We assume that the system is already controlled by linear feedback found using LQR principle.

$$u_{lqr} = K_{lqr}x, \qquad K_{lqr} \in R^{1*n} : \tag{2}$$

We assume that K_{lqr} is already well-tuned.

Assumption 1: Assume that the matrix of the closed-loop linear system $A + BK_{lqr}$ is Hurwitz, and it has at least one real eigenvalue $\lambda < 0$ such that the corresponding left eigenvector $\nu \in R^{1*n} \neq 0$:

$$\nu(A + BK_{lqr}) = \lambda\nu \tag{3}$$

and also not orthogonal to B:

$$\nu B \neq 0 \tag{4}$$

The sliding mode control (SMC) steers all solutions of the closed-loop system to a sliding surface $Cx = 0$ in a finite time, , where $C \in R^{1*n}$. Theoretically, it stabilises the origin of the system (1) and completely rejects the matched perturbations.

3 Design of the Controller from Linear Feedback

The sliding mode controller for the linear plant is designed according to the following steps:

1. selecting the sliding surface $Cx = 0$ such that the motion of the system under consideration on this surface is stable (all trajectories converge to zero when time goes to infinity).
2. defining a control law which steers the state of (1) towards the surface $Cx = 0$ in a finite time and ensures that this surface is a positively invariant set of the system.

The sliding-mode control law is given by (for rigorous mathematical definition and analysis, refer to **??**)

$$u_{SM} = K_{\text{nom}}\, x + \gamma(t, x)\, \text{sign}(Cx) \tag{5}$$

where $\gamma : \mathbb{R}\times\mathbb{R}^n \to (-\infty, 0)$ is such that $\inf \gamma(t, x) < 0$,

$$K_{\text{nom}} = -(CB)^{-1}CA$$

the row vector $C \in R^{1\times n}$ is selected such that $CB \neq 0$ and the differential-algebraic equation

$$\begin{cases} \dot{x} = \left(I_n - B(CB)^{-1}C\right) Ax \\ Cx = 0 \end{cases} \tag{6}$$

is globally asymptotically stable. Also the signum function is defined as: $\text{sgn}(\sigma) = -1$ if $\sigma < 0$, $\text{sgn}(\sigma) = 1$ if $\sigma > 0$, and $\text{sgn}(0) = [-1, 1]$.

Since, by Assumption 1, the matrix of the closed-loop linear system $A + BK_{\text{lin}}$ is Hurwitz and has a real eigenvalue $\lambda < 0$, then the corresponding eigenvector $v \in \mathbb{R}^{1\times n}$ defines an invariant manifold $vx = 0$ of the linear closed-loop system. This manifold is the sliding surface of the linear unperturbed system . Therefore, for the sliding-mode control (5), the sliding surface

$$Cx = 0, \quad C = \frac{v}{vB} \tag{7}$$

is selected.

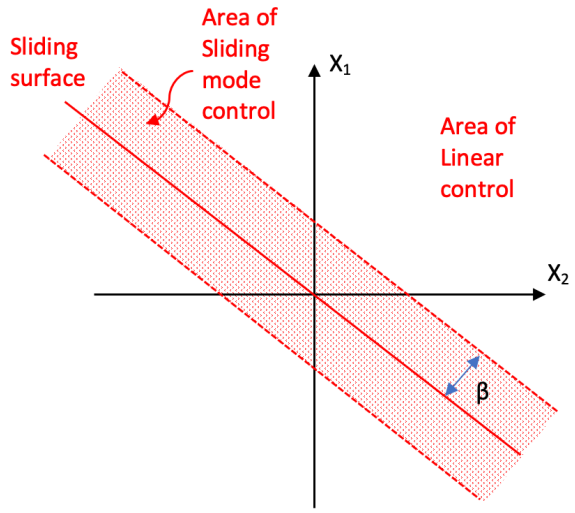

Figure 1: *SMC and LQR combined*

We define the sliding mode control (5) such that

$$u_{SM}(x) = K_{lqr}x \quad \text{for } |Cx| \geq \beta$$

and for $|Cx| < \beta$ the sliding mode algorithm has the form (5), which steers all trajectories of the closed-loop system to the surface $|Cx| = 0$ (β is a scale parameter). To avoid discontinuity of (5) on $|Cx| = \beta$, the function γ is selected as follows.

$$\gamma(t,x) := \tilde{\gamma}(|Cx|) = \lambda \begin{cases} |Cx| & \text{if} \quad |Cx| \geq \beta \\ \beta & \text{if} \quad |Cx| < \beta \end{cases} \tag{8}$$

A full mathematical rigorous definition of the above controller involving stability and finite time convergence is shown in [1].

The sliding mode controllers (5), and (8) can be written as linear feedback with state-dependent gain:

$$u_{SM} = K_{SM}(|Cx|)x \quad \text{for} \quad Cx \neq 0$$

where $K_{SM} : (0, +\infty) \to \mathbb{R}^{1\times n}$ is defined as follows:

$$K_{SM}(\varphi) = K_{\text{nom}} + \frac{\tilde{\gamma}(\varphi)}{\varphi}C, \quad \varphi > 0 \tag{9}$$

The sliding mode control designed by the latter theorem coincides with the linear feedback for $|Cx| \geq \beta$. For β tending to zero the original linear feedback is recovered.

3.1 Chattering Reduction

The classical idea for the chattering reduction of SMC is to replace the sign multifunction with a piecewise-linear saturation function and, next, to tune its width parameter. We have implemented the same as shown below.

As shown above the sliding mode controller (5), (8) can be interpreted as a linear feedback

$$u_{SM} = K_{SM}(|Cx|)x$$

with the state-dependent gain $K_{SM}(|Cx|)$ given by (9), which tends to ∞ as $|Cx| \to 0$. The infinite gain in the linear controller contributes to the chattering phenomenon. The chattering phenomenon is reduced by introducing the saturation function as follows

$$\text{sat}_{\delta,\beta}(\varphi) = \begin{cases} \delta & \text{if} \quad \varphi < \delta \\ \varphi & \text{if} \quad \delta \leq \varphi \leq \beta \\ \beta & \text{if} \quad \varphi > \beta \end{cases} \tag{10}$$

To avoid the infinite gain in the explicit discretization of the sliding mode controller, we re-define it as follows

$$\tilde{u}(x) = K_{SM}\left(\text{sat}_{\delta,\beta}(|Cx|)\right)x \tag{11}$$

where $\beta > 0$ is defined in the formula (8) and $\delta \in (0, \beta)$ is a tuning parameter. Obviously, $\delta = 0$ corresponds to the original sliding-mode controller, but for $\delta = \beta$ it follows that

$$K_{SM}\left(\text{sat}_{\beta,\beta}\left(|Cx_k|\right)\right) = K_{lin}$$

i.e., the proposed controller becomes the linear feedback in the limit case. The values β and δ are carefully selected and tuned to get good results.

4 Experimentation: Rotary Inverted Pendulum

The platform QUBE-Servo 2 of Quanser is used for the experiment, and it is depicted in Fig. 2 and the convention of the sign for the angles θ and α, respectively for the rotary arm and the pendulum arm in Fig 3.

The controllers are realised using the Matlab platform. The linear model of the pendulum system can be expressed in state-space form as:

Figure 2: *Qube Platform*

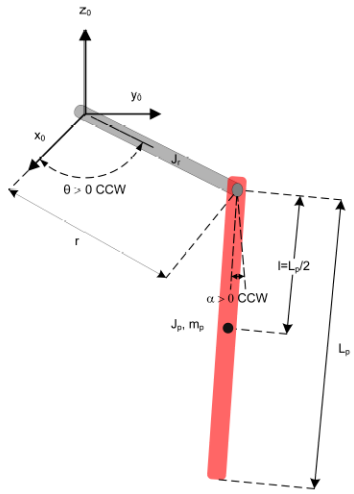

Figure 3: *Angles and lengths associated with the system*

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{23}$$

where $x(t) = (\theta(t), \alpha(t), \dot{\theta}(t), \dot{\alpha}(t))^\top \in \mathbb{R}^4$ is the state vector, $u(t) \in \mathbb{R}$ is the control signal, $A \in \mathbb{R}^{4\times4}$, $B \in \mathbb{R}^{4\times1}$. The matrices of the linear systems are found by solving and linearising the equations of motion of the rotary inverted pendulum given as follows:

$$\left(J_r + J_p \sin\alpha^2\right)\ddot{\theta} + m_p lr\cos\alpha\ddot{\alpha} + 2J_p \sin\alpha\cos\alpha\dot{\theta}\dot{\alpha} \\ m_p lr\sin\alpha\dot{\alpha}^2 = \tau - b_r\dot{\theta} \tag{12}$$

and

$$J_p\ddot{\alpha} + m_p lr\cos\alpha\ddot{\theta} - J_p\sin\alpha\cos\alpha\dot{\theta}^2 \\ +m_p gl\sin\alpha = -b_p\dot{\alpha} \tag{13}$$

where $J_r = m_r r^2/3$ is the moment of inertia of the rotary arm with respect to the pivot (i.e. rotary arm axis of rotation) and $J_p = m_p L_p^2/3$ is the moment of inertia of the pendulum link relative to the pendulum pivot (i.e. axis of rotation of pendulum). The viscous damping acting on the rotary arm and the pendulum link are b_r and b_p, respectively. The applied torque at the base of the rotary arm generated by the servo motor is

$$\tau = \frac{k_m}{R_m}\left(v_m - k_m\dot{\theta}\right) \tag{14}$$

Here v_m, voltage supplied to the motor corresponds to the input to the system ($u(t)$)

Upon solving and linearising the above equation of motion with respect to the states, $x(t) = (\theta(t), \alpha(t), \dot{\theta}(t), \dot{\alpha}(t))^\top \in \mathbb{R}^4$ at the point in state space, $x^*(t) = (0, \pi, 0, 0)$, We get the Values of A and B.(We assume knowledge of all states of the system from output).

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 362.049 & -29.187 & -1.192 \\ 0 & 471.909 & -28.848 & 0 \end{pmatrix},$$

$$\mathrm{B} = \begin{pmatrix} 0 \\ 0 \\ 120.608 \\ 119.205 \end{pmatrix}.$$

The gains of the LQR-based controller are found and as follows

$$K_{\mathrm{lqr}} = [-1, 20.288, -0.978, 1.619]$$

The matrix $A + BK_{\mathrm{lqr}}$ has the following eigenvalues

$$\lambda_1 = -86.434; \lambda_2 = -1.985$$
$$\lambda_3 = -8.713 + 2.062i; \lambda_4 = --8.713 - 2.062i;$$

Two of them are really negative and can be utilised for the design of the SMC-based controller. For the subsequent study we use $\lambda = \lambda_2$ for the computation of the SMC and its comparison with the original linear algorithm.

For the purpose of testing, we look at the regulatory ability of the system; we use two square signals of frequency 0.125Hz and 0.05Hz as a reference and look at their performance. For simulation, we also put the corresponding LQR performance.

4.1 Simulations

Here the orange graph represents the reference signal, blue graph represents the SMC-based controller and the green graph represents the pure LQR controller.

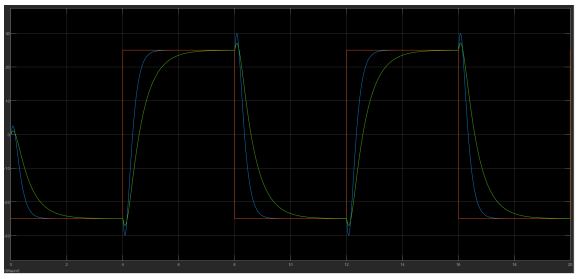

Figure 4: *Rotary angle (θ) for 0.125Hz Square wave*

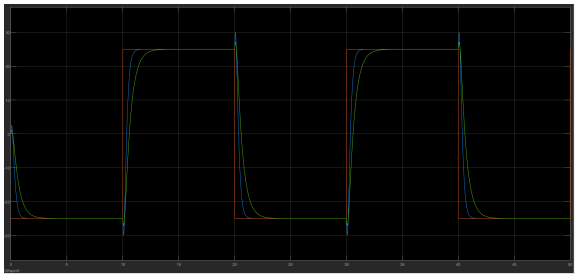

Figure 5: *Rotary angle (θ) for 0.05Hz Square wave*

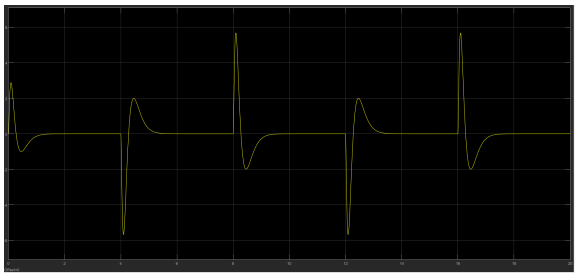

Figure 6: *Pendulum angle(α) for 0.125Hz Square wave*

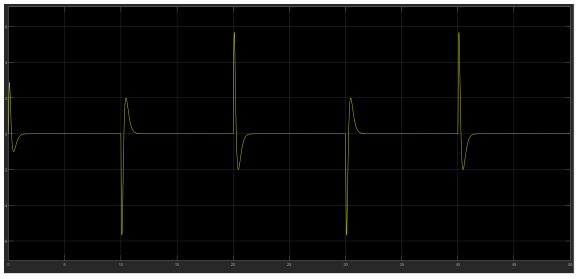

Figure 7: *Pendulum angle(α) for 0.125Hz Square wave*

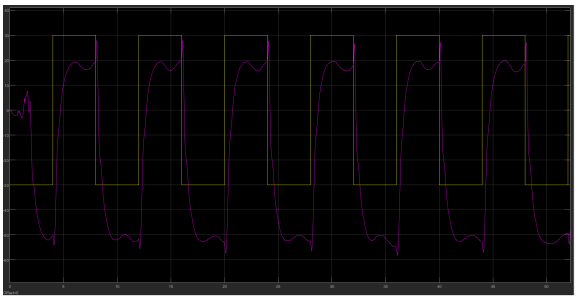

Figure 8: *Rotary angle (θ) for 0.125Hz Square wave*

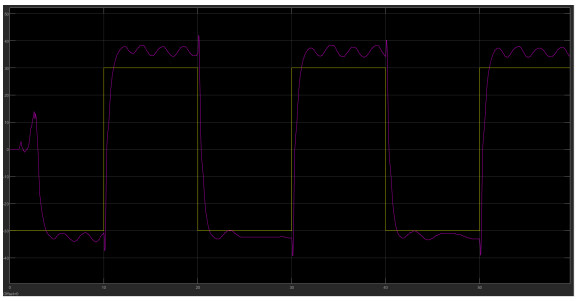

Figure 9: *Rotary angle (θ) for 0.05Hz Square wave*

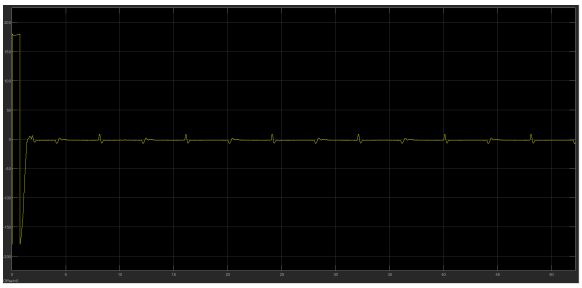

Figure 10: *Pendulum angle(α) for 0.125Hz Square wave*

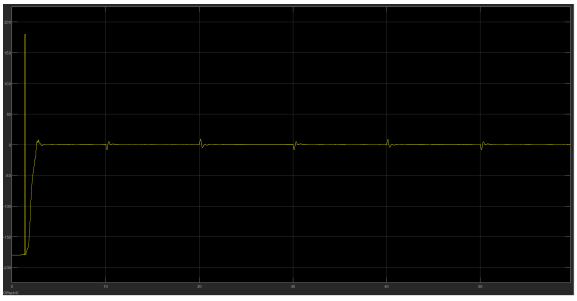

Figure 11: *Pendulum angle(α) for 0.125Hz Square wave*

4.2 Implementation

Her, the yellow graph represents the reference signal, and the violet graph represents the SMC-based controller results.

5 Results and Conclusion

It can be understood from the simulation and implementation that the SMC-based controller gave a good performance in terms of settling time and is better than the classical LQR(simulations) based controller.

6 Resources

```
%% Motor
% Resistance
Rm = 8.4;
% Current-torque (N-m/A)
kt = 0.042;
% Back-emf constant (V-s/rad)
km = 0.042;
%
%% Rotary Arm
% Mass (kg)
mr = 0.095;
% Total length (m)
r = 0.085;
% Moment of inertia about pivot (kg-m^2)
Jr = mr*r^2/3;
% Equivalent Viscous Damping Coefficient
    (N-m-s/rad)
br = 1e-3; % damping tuned heuristically to
    match QUBE-Sero 2 response
%
%% Pendulum Link
% Mass (kg)
mp = 0.024;
% Total length (m)
Lp = 0.129;
% Pendulum center of mass (m)
l = Lp/2;
% Moment of inertia about pivot (kg-m^2)
Jp = mp*Lp^2/3;
% Equivalent Viscous Damping Coefficient
    (N-m-s/rad)
bp = 5e-5;
% Gravity Constant
g = 9.81;

beta = 1

delta = 1

syms x_1 x_2 x_3 x_4 vm;

a_1 = Jr + (Jp*sin((x_2*x_2)));
a_2 = mp * l* r*cos(x_2);

b_1 = ((km*vm)/(Rm))  +
    (mp*l*r*(sin(x_2))*x_4*x_4) -
    (2*Jp*(sin(x_2))*(cos(x_2))*x_3*x_4) -
    (br*x_3) - (   (km*km*x_3)/Rm);

a_4 = Jp;
a_3 = mp*l*r*cos(x_2);
b_2 = (Jp*(sin(x_2))*(cos(x_2))*x_3*x_3) -
    (mp*g*l*sin(x_2)) - (bp*x_4 );

X = [a_1,a_2;a_3,a_4];
Y = [b_1;b_2];

D = linsolve(X,Y);

f_1 = x_3;
f_2 = x_4;
f_3 = D(1);
f_4 = D(2);


J =
    jacobian([f_1,f_2,f_3,f_4],[x_1,x_2,x_3,x_4,vm]);

x_1_val = 0;
x_2_val = pi;
x_3_val = 0;
x_4_val = 0;


J_subs = vpa(subs(J,[x_1, x_2, x_3, x_4,vm],
    [x_1_val, x_2_val, x_3_val,
    x_4_val,0]),4);

A = double(J_subs(:,1:4));

B = double(J_subs(:,5));

Q_1 =
    [2.5,0,0,0,;0,2,0,0;0,0,0.5,0;0,0,0,0.5;]

R_1= 2.5 ;


K_lin=lqr(A,B,Q_1,R_1);%
C = [1,0,0,0;0,1,0,0];

D = [0,0,0,0];


Closed_loop_sys = (A-(B*K_lin));

[eig_vec,eig_val] =
    eig(transpose(Closed_loop_sys));

% On the sliding surface the dynamics becomes

sliding_surface =
    transpose(eig_vec(:,2))/(transpose(eig_vec(:,2))*B);

dynamics = A - (B*sliding_surface*A);
eig_va = eig(dynamics);

K_nom = -(sliding_surface*A);
```

Listing 1: *MATLAB code*

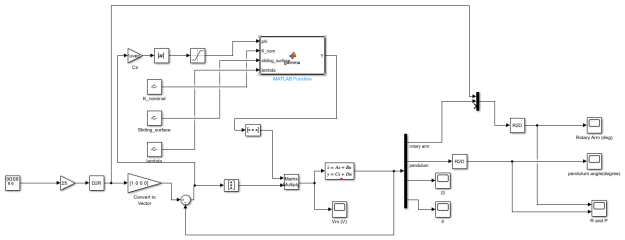

Figure 12: *Simulink Model*

```
function Y =
    gamma(phi,K_nom,sliding_surface,lambda)
    % Calculate the absolute value of Cx

    b = 1;


    % Check if the absolute value is greater
        than or equal to b
    if phi >= b
        % If yes, use the function
            gtilde(absCx)
        X = lambda*sliding_surface;
    else
        % If not, assign the value b
        X= (lambda*b*sliding_surface)/phi;
    end

    Y = -1*transpose(K_nom+ X);

end
```

Listing 2: *Simulink Function code*

References

[1] Gabriele Perozzi et al. “Upgrading linear to sliding mode feedback algorithm for a digital controller”. In: *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE. 2021, pp. 2059–2064.