Stabilization of Under-Actuated Spatial 2R Manipulator using Feedback Linearization and LQR

ELL-703: Optimal Control - Learning Project

Ashwath Ram S

2023MEZ8467

Submitted on 08/04/2024

Ashwath Ram S
2023MEZ8467

Contents

1 Introduction

Stabilization of an Inverted pendulum is a widely acknowledged and extensively studied problem in the field of Control theory. Researchers and control engineers have been devising control laws and algorithms to stabilize various iterations of the inverted pendulum mechanism. These solutions have seen significant progress with the introduction of innovative tools and techniques in the realm of control theory, resulting in the problem undergoing multiple evolutionary phases.

The Underactuated Spatial 2R mechanism represents a specific type of inverted pendulum system, comprising a base link connected to a motor. A pendulum link is affixed to the base link through a revolute joint oriented along the base link's longitudinal axis, resulting in a 3-dimensional mechanism. The primary challenge lies in stabilizing the pendulum link along the vertical direction (an unstable equilibrium) solely by controlling the motor's torque.

In practical scenarios, an encoder is typically attached only to the pendulum link to provide feedback on its angular position and velocity. However, for the sake of simplicity in this project, state feedback incorporating both the angular position and velocity of the base link has been employed. This approach aims to streamline the process by focusing solely on controller design, thus bypassing the necessity of developing an estimator (observer) for the system.

The dynamics of the mechanism has been elucidated in Section 2 through the derivation of Newton-Euler equations of each link using DeNOC Algorithm.

The problem is approached using 2 different control strategies where both employ LQR to obtain the feedback gains -

(1) Approximate Linearization at unstable equilibrium and state feedback stabilization of

the linearized system using LQR as given in Section 3.

(2) Exact Linearization via state feedback and stabilization of the input-output linear system using LQR as given in Section 4.

The system is simulated separately using the ode45 solver under both control strategies. The results from each strategy are then compared to analyze their respective performances. This is presented in Section 5.

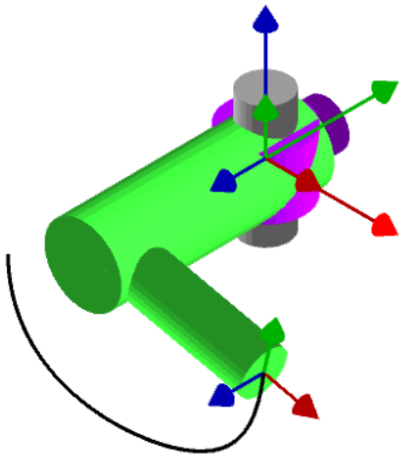

Figure 1: Spatial 2R Modelled in RoboAnalyzer

2 Dynamics of the System

The dynamics of the system is studied using Newton-Euler equations. To derive the equations, a Recursive Newton Euler algorithm using Decoupled Natural Orthogonal Complement(DeNOC) matrices is developed. This algorithm is implemented in MATLAB using the symbolic toolbox to get the final equations relating the change in generalized coordinates(θ_1 and θ_2) and its rates($\dot{\theta_1}$ and $\dot{\theta_2}$) to the input which is the motor torque(τ). Gravity is consid-

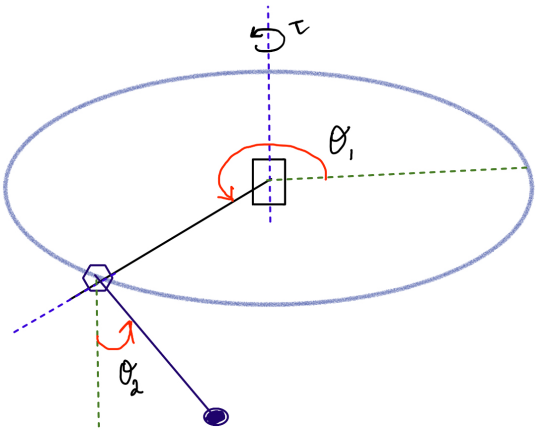

Figure 2: Schematic representation

ered to act normal to the plane containing the base link. For the sake of simplicity, both the base link and pendulum link are assumed to have identical geometry and mass properties. The center of mass is assumed to be located at the midpoint of each link.

The section presents the final equations obtained from the DeNOC Algorithm implemented using MATLAB's Symbolic toolbox. The MATLAB code is provided in the Appendix for reference.

Table 1: Link Parameters

Length of each link	0.1 m
Mass of each link	1 Kg
I_{xx}	0.00333 $Kg \cdot m^2$
I_{yy}	0.00333 $Kg \cdot m^2$
I_{zz}	0.00333 $Kg \cdot m^2$

$$\begin{pmatrix} (1) \quad 0.005\cos(\theta_2)\sigma_1 - 0.0025(2.0\sin(\theta_2)\dot{\theta}_2 - 2.0\cos(\theta_2)\sin(\theta_2)\dot{\theta}_1)\dot{\theta}_2 \\ -1.0(0.0025\cos^2(\theta_2) - 0.02166)\sigma_2 - 1.0\tau \\ (2) \quad 0.4905\sin(\theta_2) + 0.005\cos(\theta_2)\sigma_2 - 0.0025\cos(\theta_2)\sin(\theta_2)\dot{\theta_1}^2 + 0.00583\sigma_1 \end{pmatrix}$$

$= 0$

where

$\sigma_1 = \ddot{\theta}_2$

$\sigma_2 = \ddot{\theta}_1$

These equations are verified by comparing their results with those obtained from RoboAnalyzer, considering the same initial conditions and zero input. Initial condition is taken as $\theta_1 = 0$, $\theta_2 = \pi/2$, $\dot{\theta}_1 = 0$, $\theta_2 \dot{=} 0$ in MATLAB which corresponds to $\theta_1 = 0$, $\theta_2 = 0$, $\dot{\theta}_1 = 0$, $\theta_2 \dot{=} 0$ in RoboAnalyzer.

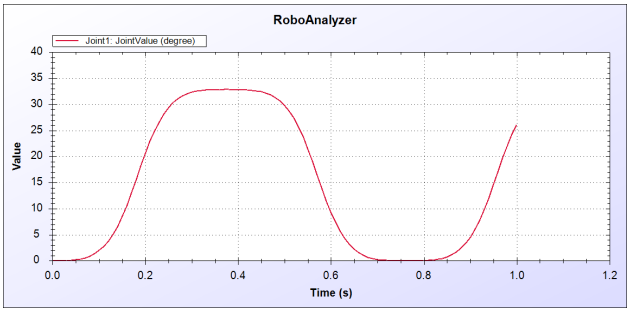

Figure 3: RoboAnalyzer- θ_1 *vs time*

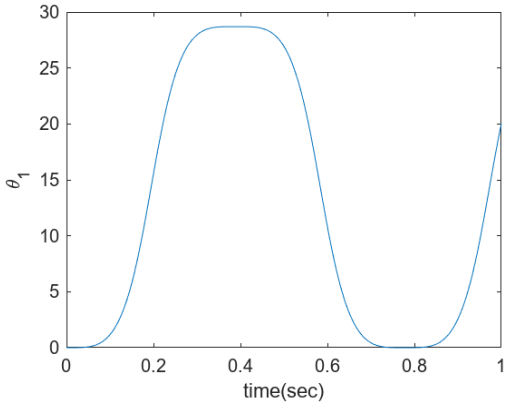

Figure 4: MATLAB- θ_1 *vs time*

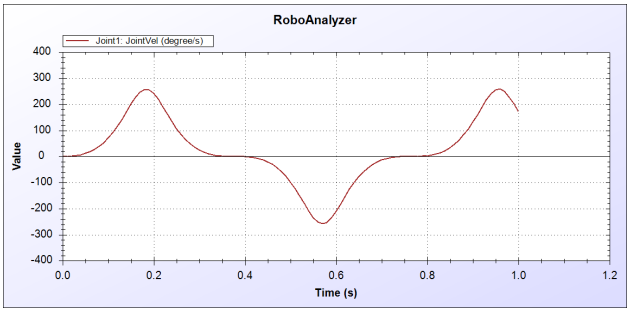

Figure 5: RoboAnalyzer- $\dot{\theta_1}$ *vs time*

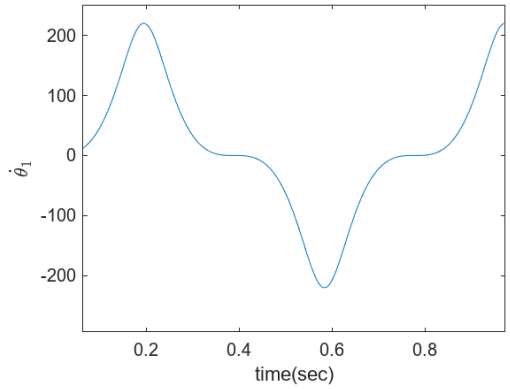

Figure 6: MATLAB- $\dot{\theta_1}$ *vs time*

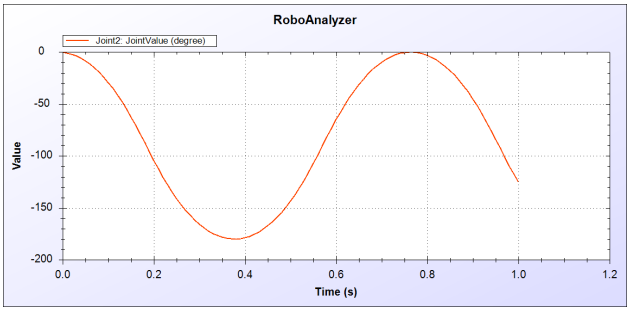

Figure 7: RoboAnalyzer- θ_2 *vs time*

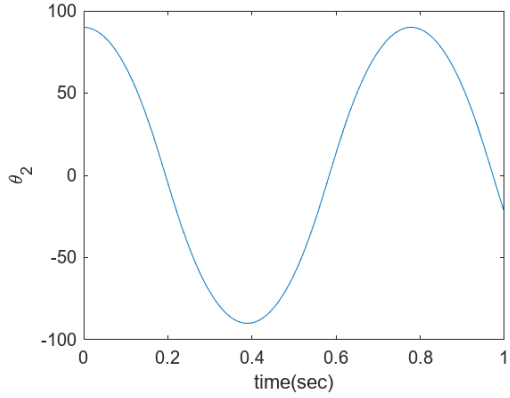

Figure 8: MATLAB- θ_2 *vs time*

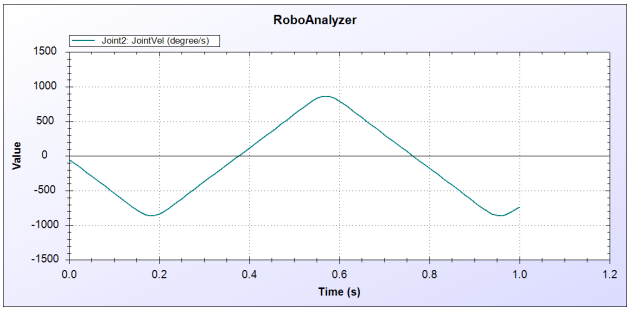

Figure 9: RoboAnalyzer- $\dot{\theta_2}$ *vs time*

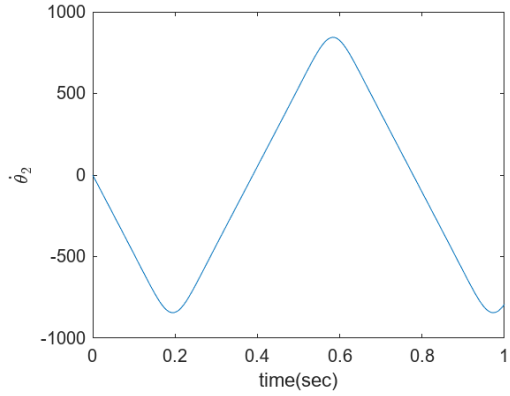

Figure 10: MATLAB- $\dot{\theta_2}$ *vs time*

3 Control Strategy - 1

The state space coordinates consist of the angular positions and angular rates of each revolute joint. y is taken as the state vector where $y_1 = \theta_1$, $y_2 = \dot{\theta}_1$, $y_3 = \theta_2 - \pi$, $y_4 = \dot{\theta}_2$.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

Differentiating this with respect to time and substituting generalized coordinates in terms of state coordinates in the Newton-Euler equations obtained in Section 2, we get the state dynamics as

$$\dot{y} \;=\; f(y) + g(y)\tau$$

$$g(y) = \begin{pmatrix} 0 \\ -\frac{2.915\text{e+}7}{197880.0\cos(y_3)^2 - 631390.0} \\ 0 \\ -\frac{2.5\text{e+}7\ \cos(y_3)}{197880.0\cos(y_3)^2 - 631390.0} \end{pmatrix}$$

$$f(y) =$$

$$\begin{pmatrix} (1) \quad y_2 \\ (2) \quad -\frac{1.0\left(62500.0\,{y_2}^2\sin(y_3)^3-62500.0\,{y_2}^2\,\sin(y_3)+72875.0\,\sigma_1\,y_2\,y_4+145750.0\,{y_4}^2\,\sin(y_3)\right)}{197880.0\sin(y_3)^2+433510.0} \\ +\frac{6.1312\times10^6\,\sigma_1}{197880.0\sin(y_3)^2+433510.0} \\ (3) \quad y_4 \\ (4) \quad -\frac{4.699\text{e}{+7}\,\sin(y_3)+6.1312\text{e}{+6}\sin(y_3)^3+135380.0\,{y_2}^2\,\sigma_1-62500.0\,{y_4}^2\,\sigma_1}{197880.0\cos(y_3)^2-631390.0} \\ -\frac{125000.0\,y_2\,y_4\left(\sin(y_3)-\sin(y_3)^3\right)-31250.0\,{y_2}^2\cos(y_3)^3\,\sin(y_3)}{197880.0\cos(y_3)^2-631390.0} \end{pmatrix}$$

where

$$\sigma_1 = \sin\left(2.0\,y_3 + 6.2832\right)$$

As mentioned in the introduction, The first control strategy entails linearizing the system around the vertical position of the pendulum using Taylor's approximation method. Thus we will linearize it about $y = \mathbf{0}$, $\tau = 0$. On linearizing it, we will obtain the state space matrices(A & B) using the relation $A = \left.\frac{\partial f(y)}{\partial y}\right|_{y=\mathbf{0},\tau=0}$ and $B = \left.\frac{\partial g(y)}{\partial y}\right|_{y=\mathbf{0},\tau=0}$. System dynamics is now given by $[\dot{y} = Ay + B\tau]$. We will get A & B as

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 28.286 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 108.39 & 0 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 \\ 67.241 \\ 0 \\ 57.668 \end{pmatrix}.$$

Subsequently, LQR (Linear Quadratic Regulator) is applied to derive the State feedback gain matrix for the linear state space model. Considering $Q = \begin{pmatrix} \frac{1}{10} & 0 & 0 & 0 \\ 0 & 10000 & 0 & 0 \\ 0 & 0 & 10000 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $R = 100000$. We get gain matrix as $K = \begin{pmatrix} -0.001 & -0.3165 & 10.249 & 1.07 \end{pmatrix}$ and close loop $poles = \begin{pmatrix} -0.0031623 & -8.1642 & -9.7246 & -22.533 \end{pmatrix}$.
The MATLAB code for state space representation and LQR implementation can be found in the Appendix.

4 Control Strategy - 2

For the second strategy, we will be using a Non-linear Diffeomorphism(State space transformation) to linearize the input-output relation using state feedback. Unlike Taylor's approximation, the dynamics of the system is not lost in this case as we are eliminating the non-linear part of the dynamics using state feedback. As long as the equilibrium lies in the transformation(meaning when transformed coordinates becomes 0, y becomes 0 which is desired), we can remove the non-linear part of the system using state feedback and then use the exactly linearized system to determine the Gain matrix. We then use LQR to determine the gains for stabilizing the linearized system.
However, the downside of this method is the presence of zero dynamics. Zero dynamics is the dynamics of the system when the output is identically 0. Basically, it is the dynamics of the states whose non-linearities cannot be cancelled out using state-feedback. We must ensure that the zero dynamics of the system is stable, else any gain value would not stabilize

the system. The new state vector is taken as

$$z = \begin{bmatrix} z_1 \\ z_2 \\ \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} y_3 \\ y_4 \\ y_1 \\ y_2\ \cos\left(y_3\right) - \frac{5\,y_4}{2} \end{bmatrix}.$$

We can see that $z = 0 \implies y = 0$, thus Transformation contains the origin. Now the dynamics of the system based on the new state coordinates would be

$$\dot{z} = \begin{pmatrix} (\dot{z_1} =) & y_4 \\ (\dot{z_2} =) & -\frac{25.0\,\sigma_2}{\sigma_1} \\ (\dot{\eta_1} =) & y_2 \\ (\dot{\eta_2} =) & \frac{62.5\,\sigma_2}{\sigma_1} - y_2\,y_4\,\sin\left(y_3\right) \\ & -\frac{250.0\,\cos(y_3)\left(116600.0\,\tau + 24525.0\,\sigma_3 - 583.0\,{y_4}^2\,\sin(y_3) + 250.0\,{y_2}^2\left(\sin(y_3) - 1.0\sin(y_3)^3\right)\right)}{\sigma_1} \end{pmatrix}$$

where

$$\sigma_1 = \left(197880.0\cos\left(y_3\right)^2 - 631390.0\right)$$

$$\begin{aligned}\sigma_2 = \big(&2.1248\text{e+}6\,\sin\left(y_3\right) + 5415.0\,{y_2}^2\,\sigma_3 - 2500.0\,{y_4}^2\,\sigma_3 \\ &+ 245250.0\,\sin\left(y_3\right)\,\left(\sin\left(y_3\right)^2 - 1.0\right) + 1.0\text{e+}6\,\tau\,\cos\left(y_3\right)\big) \\ &- 1250.0\,{y_2}^2\cos\left(y_3\right)^3\,\sin\left(y_3\right) - 5000.0\,y_2\,y_4\left(\sin\left(y_3\right) - 1.0\sin\left(y_3\right)^3\right)\end{aligned}$$

$$\sigma_3 = \sin\left(2.0\,y_3 + 2.0\,\pi\right)$$

We have taken our output as y_3 which is the angle made by the pendulum with respect to vertical. Now if our output(y_3) becomes identically zero, then we can see that y_4 becomes 0 and $\dot{y_2}$ becomes 0, which means $y_2(t) = y_2(0)$, thus $\dot{y_1} = y_2(0)$. This implies once we get the desired output $\theta_1(t) = \dot{\theta_1}(0)t + \theta_1(0)$. From this, we can see that θ_1 will not come to zero even after reaching the desired output, it will always be linearly changing with time(pole at 0). Therefore our zero dynamics is not completely stable, it is only marginally stable. **The future scope of this project would be to identify a Non-linear state transformation so that the zero dynamics becomes completely stable with negative close loop poles.**

Now focusing on the controllable states, we can see that dynamics of z_1 and z_2 can be linearized by cancelling out the non-linearities using state feedback. If we take $\tau = \frac{1}{\gamma(y)} \cdot (v - \delta(y))$ where

$$\delta(y) = -\frac{4.699\text{e+}7 \sin(y_3) + 6.1312\text{e+}6 \sin(y_3)^3 + 135380.0\, {y_2}^2 \sin(2.0\, y_3 + 6.2832)}{197880.0 \cos(y_3)^2 - 631390.0} - \frac{62500.0\, {y_4}^2 \sin(2.0\, y_3 + 6.2832) - 125000.0\, y_2\, y_4 \left(\sin(y_3) - \sin(y_3)^3\right) - 31250.0\, {y_2}^2 \cos(y_3)^3 \sin(y_3)}{197880.0 \cos(y_3)^2 - 631390.0}$$

$$\gamma(y) = -\frac{2.5\text{e+}7 \cos(y_3)}{197880.0 \cos(y_3)^2 - 631390.0}$$

Our dynamic equations become

$$\dot{z}_1 = z_2$$

$$\dot{z}_2 = v$$

$$\dot{\eta}_1 = y_2$$

$$\dot{\eta}_2 = \frac{2.3495\text{e+}8\ \sin(y_3) + 3.0656\text{e+}7\ \sin^3(y_3) + 676880.0\, y_2^2\, \sigma_1 - 312500.0\, y_4^2\, \sigma_1}{395750.0\ \cos^2(y_3) - 1.2628\text{e+}6}$$
$$+ \frac{-1.2262\text{e+}7\, \sigma_1\ \cos(y_3) + 6.67\text{e+}7\, \tau\ \cos(y_3) + 145750.0\, y_4^2\ \sin(2.0\, y_3) + 242030.0\, y_2\, y_4\ \sin(y_3)}{395750.0\ \cos^2(y_3) - 1.2628\text{e+}6}$$
$$+ \frac{-281250.0\, y_2^2\ \cos^3(y_3)\ \sin(y_3) + 1.0208\text{e+}6\, y_2\, y_4\ \sin^3(y_3) + 145750.0\, y_2\, y_4\, \sigma_1\ \cos(y_3)}{395750.0\ \cos^2(y_3) - 1.2628\text{e+}6}$$

Where:

$$\sigma_1 = \sin\left(2.0\, y_3 + 6.2832\right)$$

We can see that there is a linear mapping between output(which is $y_3 = z_1$) and input(in this case v but actual input is $\tau = \frac{1}{\gamma(y)} \cdot (v - \delta(y))$). The system is in controllable canonical form with respect to z_1 and z_2. Thus $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ and $B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Using LQR to find gains for the normalized state coordinates(transformed coordinates), considering $Q = \begin{pmatrix} 10000 & 0 \\ 0 & 10000 \end{pmatrix}$ and $R = 100000$, we get gain matrix as $K = \begin{pmatrix} 0.31623 & 0.85584 \end{pmatrix}$ and closed loop $poles = \begin{pmatrix} -0.42792 + 0.36485\,\mathrm{i} & -0.42792 - 0.36485\,\mathrm{i} \end{pmatrix}$. The MATLAB code for Feedback linearization and LQR of transformed system can be found in Appendix.

5 Results

The initial condition is taken as $\theta_1 = 0, \theta_2 = 3, \dot{\theta}_1 = 0, \theta_2 \doteq 0$. As we can see, θ_2 approaches pi much faster in strategy-1 than in Strategy-2, but the steady state error is lesser in Strategy-2 than in Strategy-1. We can also see that the motor rotates at a much faster rate in strategy-2 than in strategy-1, this could be useful when pendulum is very far from target destination, but when it is closer, it is less power efficient than strategy-1. The peak angular velocity

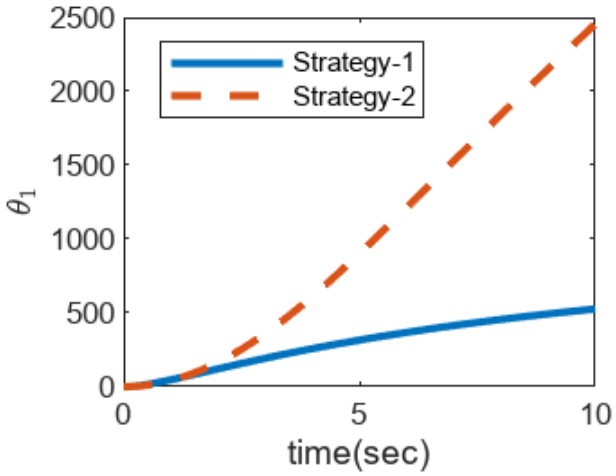

Figure 11: θ_1 *vs time*

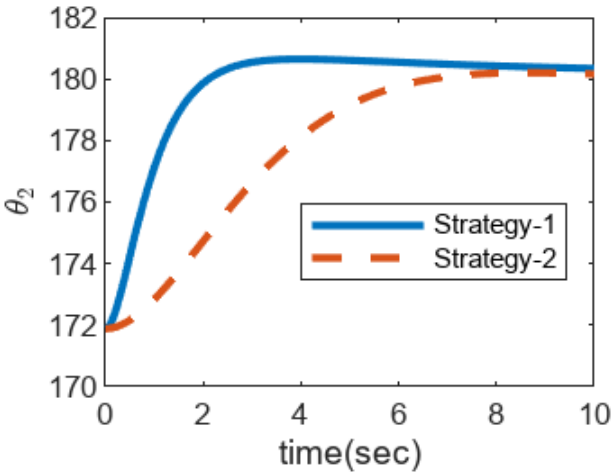

Figure 12: θ_2 *vs time*

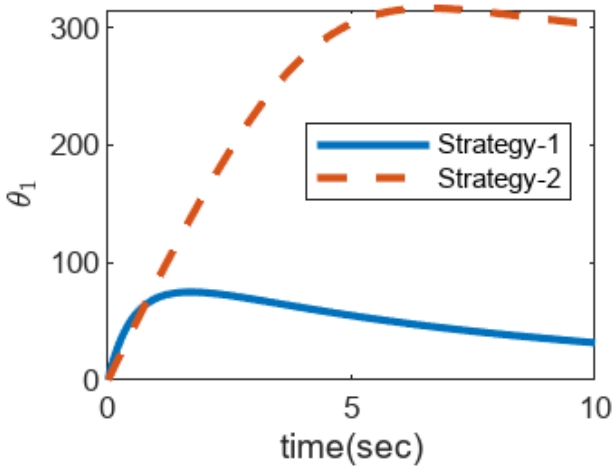

Figure 13: $\dot{\theta_1}$ *vs time*

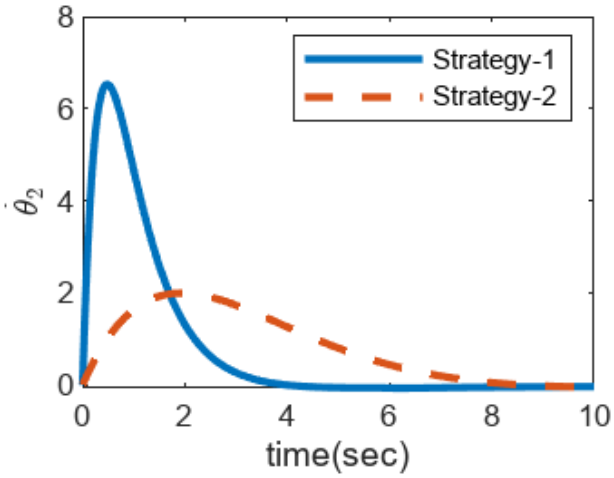

Figure 14: $\dot{\theta_2}$ *vs time*

of pendulum link is much lower in Strategy-2 than in Strategy-1. We can also add integral control to Strategy-1 and see whether it reduces the steady state error and come in par with strategy-2.

On applying integral control with gain=7, we can see in Figures 15 and 16 that the steady state error in strategy-1 decreases substantially as compared to the results in the absence of integral control. However there is not much decrease in the peak angular velocity with the

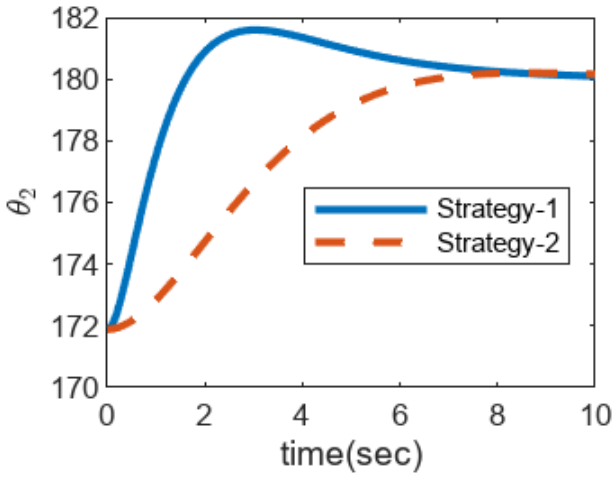

Figure 15: θ_2 *vs time with integral control*

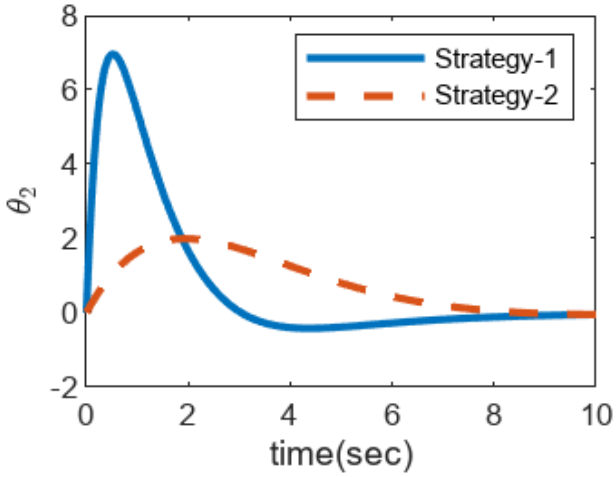

Figure 16: $\dot{\theta}_2$ *vs time with integral control*

introduction of integral control.

Animation for Strategy 1

Animation for Strategy 2

6 Appendix

GitHub Link for the Code

6.1 Recursive Newton-Euler equations using DeNOC in MATLAB

```
syms t real;

syms theta1(t) theta2(t);

syms tau;

syms I1x I1y I1z l1 m1 d1 real;

syms I2x I2y I2z l2 m2 d2 real;
```

```
syms g real;

syms ang real;

assumeAlso(theta1(t),'real');

assumeAlso(theta2(t),'real');


global K_gain;

global K_gain_fb;

global I_fn h_fn del_fn gam_fn;

global Tau_t;


g_f = [0;0;-g];


Id = eye(3);

Id_1 = eye(6);

ZO = zeros(3,3);

ZO_1 = zeros(6,6);

ZO_c = zeros(6,1);

ZO_c2 = zeros(3,1);


Rz = [cos(ang),-sin(ang),0;sin(ang),cos(ang),0;0,0,1];

Ry = [cos(ang),0,sin(ang);0,1,0;-sin(ang),0,cos(ang)];


T2_1 = subs(Rz,ang,theta1);

T3_1 = T2_1*subs(Ry,ang,pi/2);
```

```
T4_1 = T3_1*subs(Rz,ang,theta2);



I1c = [I1x,0,0;0,I1y,0;0,0,I1z];

I2c = [I2x,0,0;0,I2y,0;0,0,I2z];


I1 = I1c + ((m1*d1*d1)*Id) - (m1*([d1,0,0]'*[d1,0,0]));

I2 = I2c + ((m2*d2*d2)*Id) - (m2*([d2,0,0]'*[d2,0,0]));

d_1 = T2_1*[d1,0,0]';

d_2 = T4_1*[d2,0,0]';


e1 = [0,0,1]';

e2 = [0,0,1]';

p1 = [e1;ZO_c2];

%p1 = [e1;cross(e1,d_1)];

p2 = [T3_1*e2;ZO_c2];

%p2 = [T3_1*e2;cross(T3_1*e2,d_2)];

a2_1 = T2_1*[-l1,0,0]';

a3_2 = T4_1*[-l2,0,0]';

A21 = [Id ZO;vec_cross(a2_1) Id];

Nl = [Id_1,ZO_1;A21,Id_1];

Nd = [p1,ZO_c;ZO_c,p2];

I1_f = T2_1*I1*(T2_1');

I2_f = T4_1*I2*(T4_1');
```

```
M1 = [I1_f,m1*vec_cross(d_1);-m1*vec_cross(d_1),m1*Id];

M2 = [I2_f,m2*vec_cross(d_2);-m2*vec_cross(d_2),m2*Id];

M = [M1,ZO_1;ZO_1,M2];

ang1 = formula(diff([0;0;theta1],t));

ang2 = ang1 + (T4_1*formula(diff([0;0;theta2],t)));

%ang2 = (T3_1*formula(diff([0;0;theta2],t)));

w1 = [vec_cross(ang1),ZO;ZO,vec_cross(ang1)];

w2 = [vec_cross(ang2),ZO;ZO,vec_cross(ang2)];

W = [w1,ZO_1;ZO_1,w2];

E1 = [Id,ZO;ZO,ZO];

E = [E1,ZO_1;ZO_1,E1];

N = Nl*Nd;

N_d = diff(N,t);

joint_r = [diff(theta1,t);diff(theta2,t)];

twist = N*joint_r;

twist_d = diff(twist,t);


%Wrench

W1_e = [cross(d_1,m1*g_f);m1*g_f];

W2_e = [cross(d_2,m2*g_f);m2*g_f];

Wrench = [W1_e;W2_e];

ext_f = [tau;0];


eq = ((M*twist_d) + (W*M*E*twist) - Wrench);
```

```
f_eq = simplify(N'*eq);


I = simplify(N'*M*N);

h = simplify((N'*M*N_d*joint_r) + (N'*W*M*E*twist) - (N'*Wrench));


% Theta double dot

%simplify(I\(ext_f - h));

I_1 = [I1x I1y I1z l1 m1 d1];

I_2 = [I2x I2y I2z l2 m2 d2];

I = (subs(I,I_1,[0.00333 0.00333 0.00333 0.1 1 0.05]));

I = (subs(I,I_2,[0.00333 0.00333 0.00333 0.1 1 0.05]));

h = (subs(h,I_1,[0.00333 0.00333 0.00333 0.1 1 0.05]));

h = (subs(h,I_2,[0.00333 0.00333 0.00333 0.1 1 0.05]));

I = (subs(I,g,9.81));

h = (subs(h,g,9.81));

theta_d_d = formula(vpa(simplify(I\(ext_f - h))));


q_d_d = diff(joint_r,t);

vpa(simplify((I*q_d_d)+(h)-(ext_f)));
```

6.2 State Space Representation

```
syms y1 y2 y3 y4 real;

% y1 = theta1

% y2 = diff(theta1,t)
```

```
% y3 = theta2 - pi

% y4 = diff(theta2,t)


y1_d = y2;

y2_d = subs(theta_d_d(1),[diff(theta1,t),diff(theta2,t)],[y2,y4]);

y2_d = subs(y2_d,[theta1,theta2],[y1,y3+pi]);

y3_d = y4;

y4_d = subs(theta_d_d(2),[diff(theta1,t),diff(theta2,t)],[y2,y4]);

y4_d = subs(y4_d,[theta1,theta2],[y1,y3+pi]);


y = [y1;y2;y3;y4];

yd_d = [y1_d;y2_d;y3_d;y4_d];


%Non linear affine system : Y_dot = F(Y) + tau*G(Y)

g_s = diff(yd_d,tau);

f_s = simplify(yd_d - (tau*g_s));
```

6.3 Linear approximation and LQR

```
%Linearinzing about (y,u) at (0,0)

digits(5);

A_y = [diff(yd_d,y1),diff(yd_d,y2),diff(yd_d,y3),diff(yd_d,y4)];

A = simplify(subs(A_y,[y1,y2,y3,y4,tau],[0,0,0,0,0]));

B_y = diff(yd_d,tau);

B = simplify(subs(B_y,[y1,y2,y3,y4,tau],[0,0,0,0,0]));
```

```
%LQR Cost
Q = [0.1,0,0,0;0,10000,0,0;0,0,10000,0;0,0,0,1];
R = 100000;
[K_gain,S,lqrpoles] = lqr(double(A),double(B),Q,R);
```

6.4 Feedback Linearization

```
%Feedback lineariztion
syms v;


z1 = y3;
z2 = y4;
eta1 = y1;
eta2 = (y2*cos(y3))-(2.5*y4);
norm_diff = jacobian([z1;z2;eta1;eta2],y)*yd_d;
gama_y = simplify(diff(norm_diff(2),tau));
delta_y = simplify(norm_diff(2) - (tau*gama_y));
A_fb = [0,1;0,0]; B_fb = [0;1];
Q_fb = [10000,0;0,10000]; R_fb = 100000;
[K_gain_fb,S_fb,lqrpoles_fb] = lqr(double(A_fb),double(B_fb),Q_fb,R_fb);
K_gain_fb = [K_gain_fb(1),K_gain_fb(2),0,0];
```