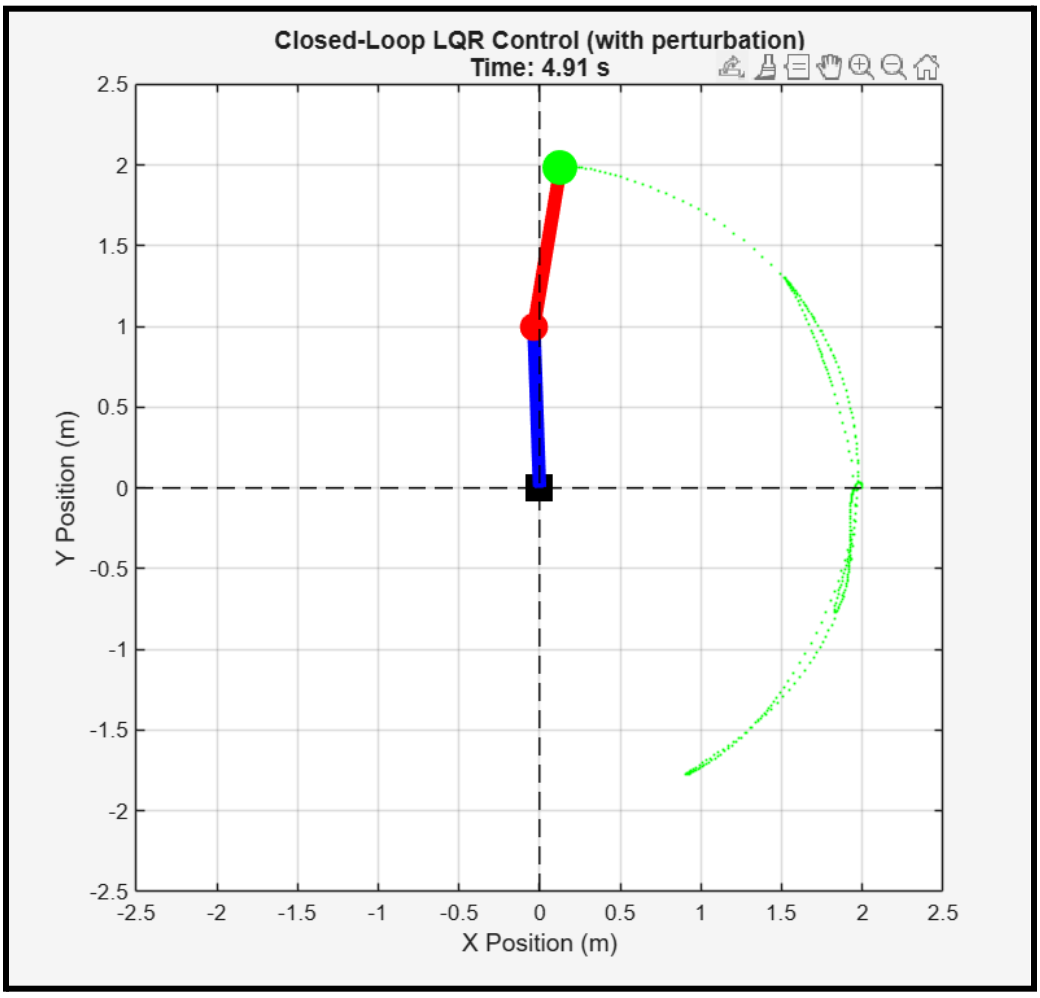
Tracking Controller- TO+LQR

Pendubot



Jash Shah

1 Dec. 2025

ECE 238 - ADV CONTROL DES LAB

Link to Code:

https://github.com/Jash-2000/Adv_Control_Systems/tree/main/Pendubot_Swingup

INTRODUCTION

Trajectory Optimization (TO) and Model Predictive Control (MPC) are powerful methods for controlling nonlinear dynamical systems. This project explores the application of direct collocation–based trajectory optimization using Matthew Kelly's OptimTraj toolbox, and uses MATLAB's "fmincon" solver (Interior Point Method) to generate optimal solutions for a underactuated 2-link pendulum with torque only at the shoulder.The Pendubot Swingup analysis is studied in detail through Parts A, B, and C, including open-loop trajectory optimization and closed-loop LQR stabilization. The goal of this project is to explore trajectory optimization (TO) and closed-loop stabilization via LQR for the Pendubot. The coupling between the links make the system all the more complex to study and control.

Project Emphasize

- Dynamics modeling
- Optimal control formulation
- Open-loop vs. closed-loop comparison
- Linearization along trajectories
- LQR-based stabilization around the optimized trajectory

Variables and System Definitions

The **Pendubot** is a two-link, underactuated robotic arm consisting of a shoulder link and an elbow link. Torque is applied only at the shoulder joint, while the elbow joint remains passive. The system's configuration is defined by two angular positions: $\theta 1$, the angle of the shoulder link relative to the vertical, and $\theta 2$, the angle of the elbow link relative to the shoulder. The corresponding angular velocities are $\dot{\theta}1$dot and $\dot{\theta}2$dot, which together with the angles form the full **state vector** x=[$\theta 1$;$\theta 2$;$\theta 1$dot;$\theta 2$dot]. The **control input** u is the torque applied at the shoulder joint. The system dynamics are governed by nonlinear equations that depend on the physical parameters: the masses of the shoulder and elbow links (m1,m2), their lengths (l1,l2), the distances from the joints to their centers of mass (lc1,lc2), and their moments of inertia about the centers of mass (I1,I2). Gravity g=9.81 also acts on both links, contributing to the dynamics and requiring the controller to counteract potential energy changes during motion. These parameters are defined in a

single centralized location within the code to facilitate easy adjustments for exploring different physical configurations or experimental setups. The goal of the control strategy is to drive the system from an initial configuration to a desired goal configuration by generating a trajectory of joint angles and velocities, while minimizing a performance cost and ensuring that the passive elbow link moves appropriately under the influence of the shoulder torque. For simulation, I am starting the system from the lowest point of the hill profile with an objective of reaching the topmost point.

The system is defined precisely using the physics in MATLAB as follows:

```
function dx = pendubotDynamics(t, x, u)
% PENDUBOTDYNAMICS Dynamics for a 2-link underactuated robot (Pendubot)
%
% Inputs:
%   t - time (scalar or array)
%   x - state vector [4 x n] where rows are [theta1; theta2; theta1_dot;
theta2_dot]
%       theta1: shoulder angle (actuated joint)
%       theta2: elbow angle (passive joint)
%   u - control input (torque at shoulder) [1 x n]
%
% Outputs:
%   dx - state derivatives [4 x n]
% Physical parameters
m1 = 1.0;       % mass of link 1 (kg)
m2 = 1.0;       % mass of link 2 (kg)
L1 = 1.0;       % length of link 1 (m)
L2 = 1.0;       % length of link 2 (m)
Lc1 = 0.5;      % distance to center of mass of link 1 (m)
Lc2 = 0.5;      % distance to center of mass of link 2 (m)
I1 = m1*L1^2/12;  % moment of inertia of link 1
I2 = m2*L2^2/12;  % moment of inertia of link 2
g = 9.81;       % gravity (m/s^2)
b1 = 0.1;       % damping coefficient at joint 1
b2 = 0.1;       % damping coefficient at joint 2
% Extract states (handle both column vectors and matrices)
theta1 = x(1,:);      % shoulder angle
theta2 = x(2,:);      % elbow angle
theta1_dot = x(3,:);  % shoulder angular velocity
theta2_dot = x(4,:);  % elbow angular velocity
% Compute inertia matrix M(q)
M11 = I1 + I2 + m1*Lc1^2 + m2*(L1^2 + Lc2^2 + 2*L1*Lc2.*cos(theta2));
M12 = I2 + m2*(Lc2^2 + L1*Lc2.*cos(theta2));
M21 = M12;
M22 = I2 + m2*Lc2^2;
% Compute Coriolis/centrifugal matrix C(q,q_dot)
```

```
h = -m2*L1*Lc2.*sin(theta2);
C11 = h.*theta2_dot;
C12 = h.*(theta1_dot + theta2_dot);
C21 = -h.*theta1_dot;
C22 = 0;
% Compute gravity vector G(q)
G1 = (m1*Lc1 + m2*L1)*g.*sin(theta1) + m2*g*Lc2.*sin(theta1 + theta2);
G2 = m2*g*Lc2.*sin(theta1 + theta2);
% Compute damping torques
D1 = b1*theta1_dot;
D2 = b2*theta2_dot;
% Control input (torque applied at joint 1, joint 2 is passive)
tau1 = u;
tau2 = 0;  % No actuation at joint 2
% Compute accelerations using M*q_ddot = tau - C*q_dot - G - D
% Solve: [M11 M12; M21 M22] * [theta1_ddot; theta2_ddot] =
[tau1-C11*th1_dot-C12*th2_dot-G1-D1;
%
tau2-C21*th1_dot-C22*th2_dot-G2-D2]
% Right-hand side
rhs1 = tau1 - C11.*theta1_dot - C12.*theta2_dot - G1 - D1;
rhs2 = tau2 - C21.*theta1_dot - C22.*theta2_dot - G2 - D2;
% Compute determinant of M
det_M = M11.*M22 - M12.*M21;
% Solve for accelerations (using Cramer's rule for 2x2 system)
theta1_ddot = (M22.*rhs1 - M12.*rhs2) ./ det_M;
theta2_ddot = (-M21.*rhs1 + M11.*rhs2) ./ det_M;
% State derivatives
dx = [theta1_dot;
     theta2_dot;
     theta1_ddot;
     theta2_ddot];
end
```

Results and Discussion

1. PART A – Trajectory Optimization Results

```
Optimization complete!
Final time: 4.913 seconds
Control effort: 62.070
Max torque used: 5.353 N-m
```

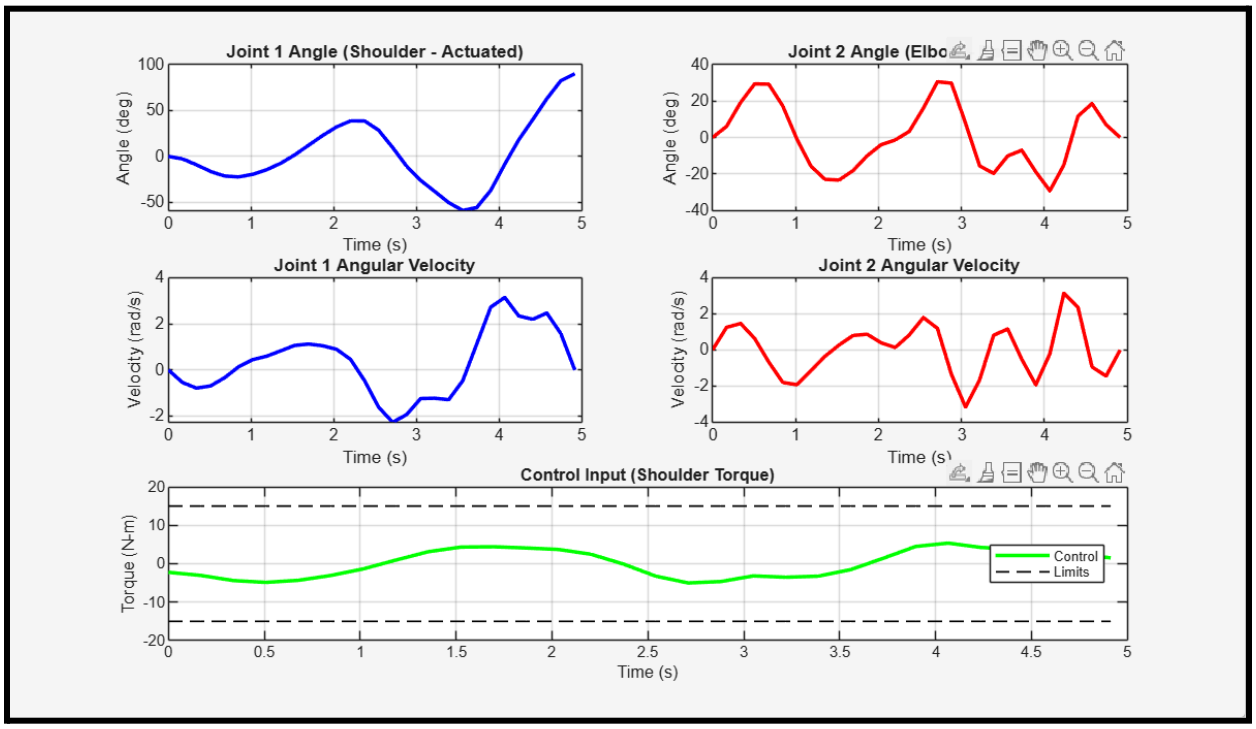


Fig1: Trajectory Optimization Results

2. PART B – ODE45 solver results and errors for open loop control

```
Max angle 1 error: 0.206829 rad (11.850 deg)
Max angle 2 error: 0.447712 rad (25.652 deg)
RMS angle 1 error: 0.066607 rad
RMS angle 2 error: 0.154621 rad
```

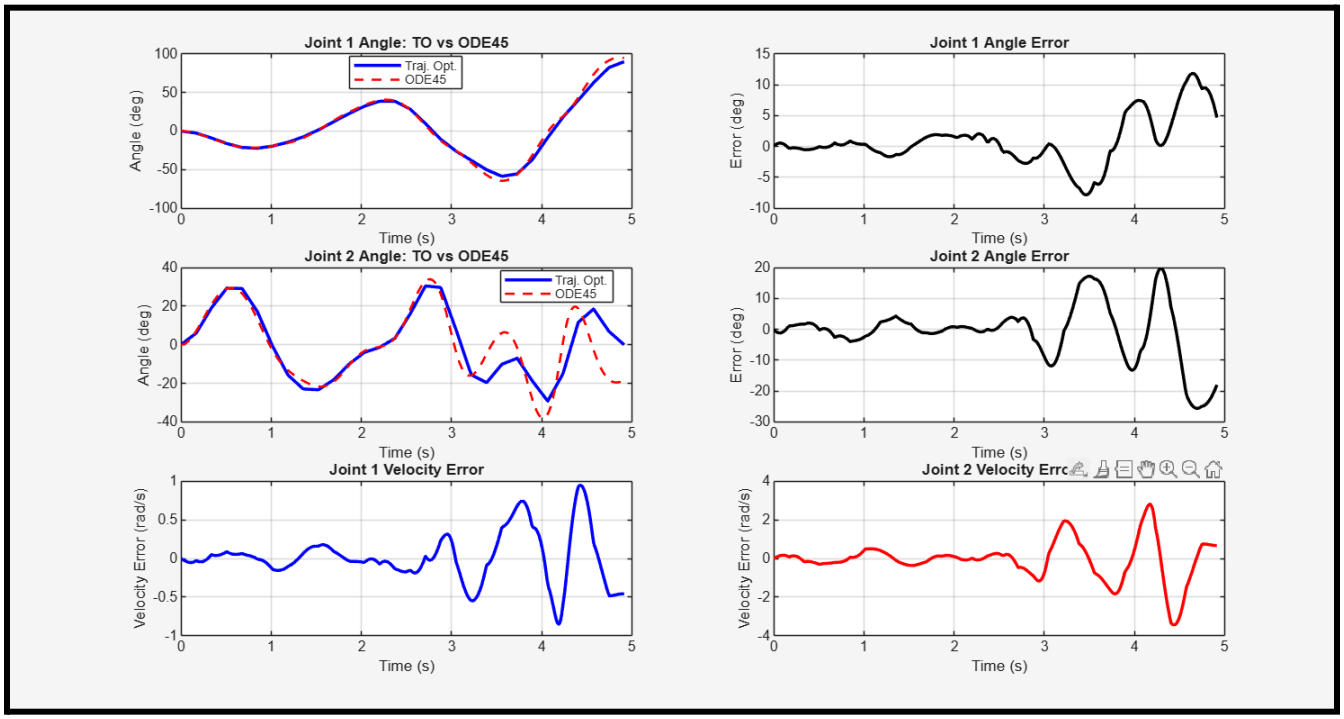


Fig2: Open Loop error analysis for TO path vs ODE45 simulated path

3. PART C – Closed loop solutions with LQR feedback

```
Final angle error (open-loop): 0.3250 rad (18.62 deg)
Final angle error (closed-loop): 0.2094 rad (12.00 deg)
```

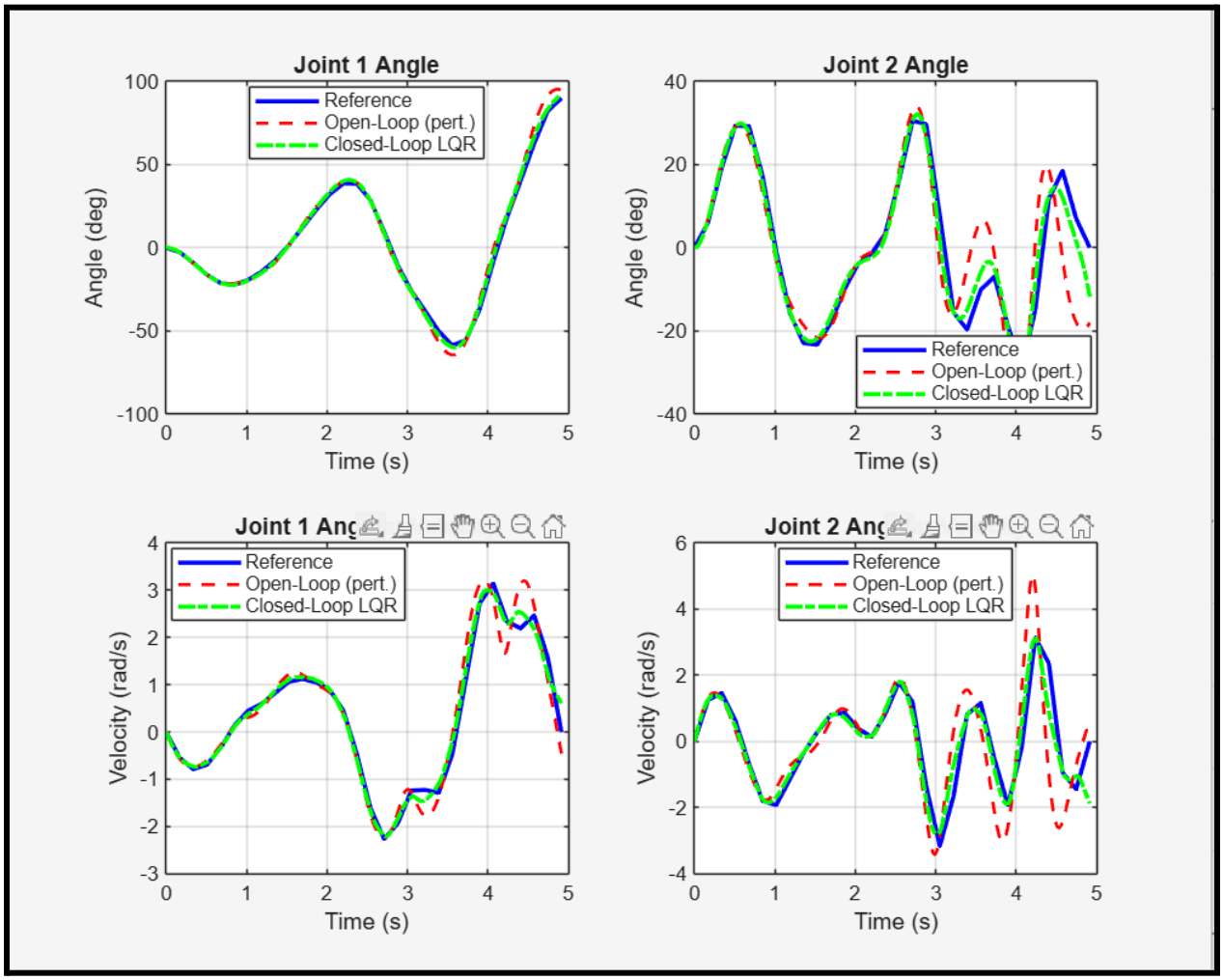


Fig3: State Variables comparison for all three methods

Conclusion

- Pendubot trajectory optimization successfully generated a feasible swing-up trajectory.
- Open-loop simulation shows small deviations due to nonlinear dynamics.
- Time-varying LQR stabilizes the trajectory effectively, demonstrating the power of combining trajectory optimization with linear feedback.

REFERENCES

1. S. Kajita et al., "Biped walking pattern generation by using preview control of zero-moment point," 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Taipei, Taiwan, 2003, pp. 1620-1626 vol.2, doi: 10.1109/ROBOT.2003.1241826. keywords: {Legged locomotion;Foot;Weight control;Humanoid robots;Centralized control;Industrial control;Servomechanisms;Control theory;Spirals;Control systems},
2. Katayama, Tohru & OHKI, TAKAHIRA & INOUE, TOSHIO & KATO, TOMOYUKI. (1985). Design of an optimal controller for a discrete-time system subject to previewable demand. International Journal of Control - INT J CONTR. 41. 677-699. 10.1080/0020718508961156.
3. Katayama, Sotaro & Murooka, Masaki & Tazaki, Yuichi. (2023). Model predictive control of legged and humanoid robots: models and algorithms. Advanced Robotics. 37. 1-18. 10.1080/01691864.2023.2168134.
4. Kelly, Matthew. “trajectoryOptimizationTutorials.” Trajectory optimization, 2016. https://www.matthewpeterkelly.com/tutorials/trajectoryOptimization/index.html.