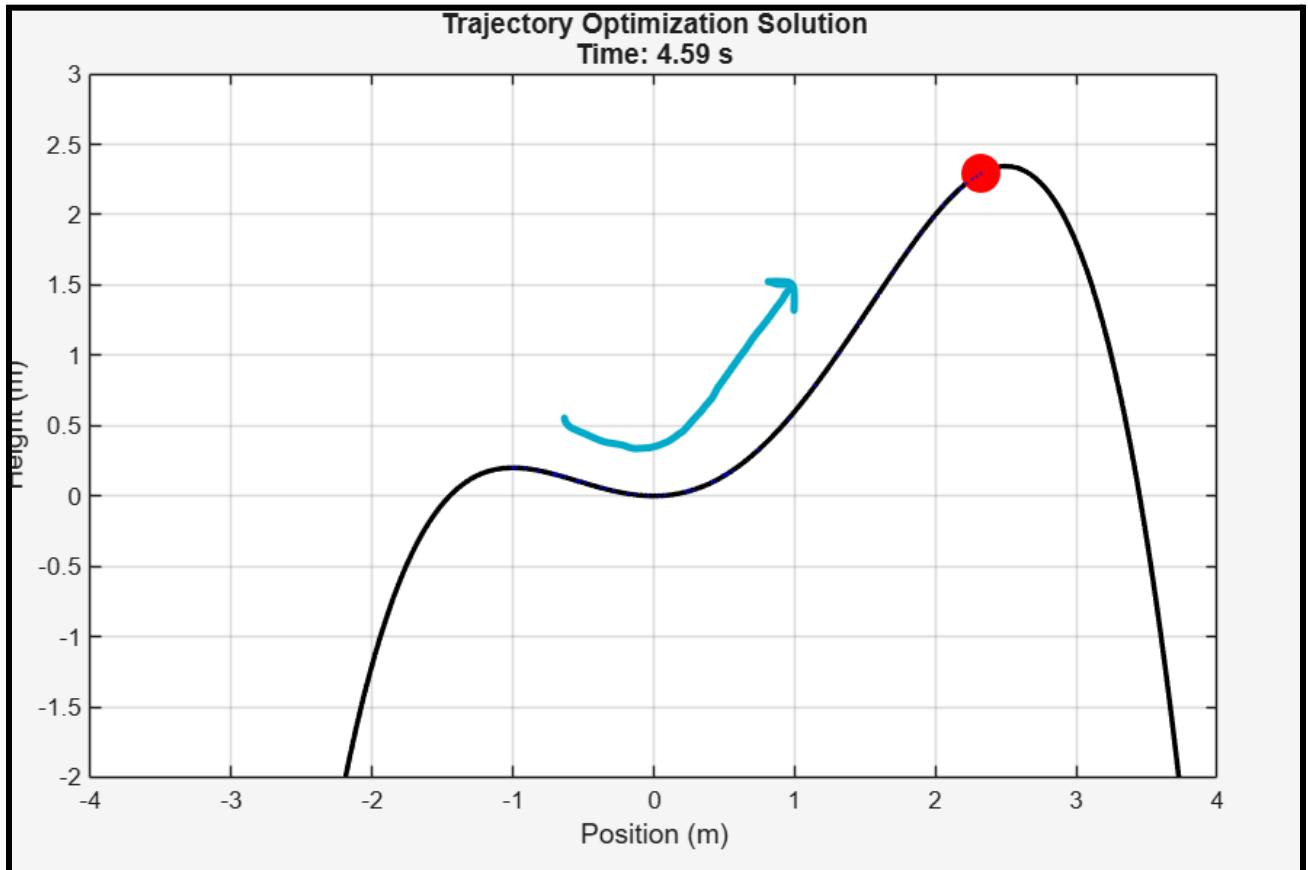


Tracking Controller- TO+LQR

Mountain Car



Jash Shah

1 Dec. 2025
ECE 238 - ADV CONTROL DES LAB

Link to Code: https://github.com/Jash-2000/Adv_Control_Systems/tree/main/Mountain_Car

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 Mountain Car system that is predominantly underactuated due to gravity, requiring dynamic momentum buildup. The Mountain Car system is studied in detail through Parts A, B, and C, including open-loop trajectory optimization and closed-loop LQR stabilization.

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 mountain car problem is a non-linear system where the car has to overcome a hill in presence of gravity and reach its target position. The hill can be varying in the y-axis(i.e. Its height is not constant and is varying with x i.e. horizontal ground movement).

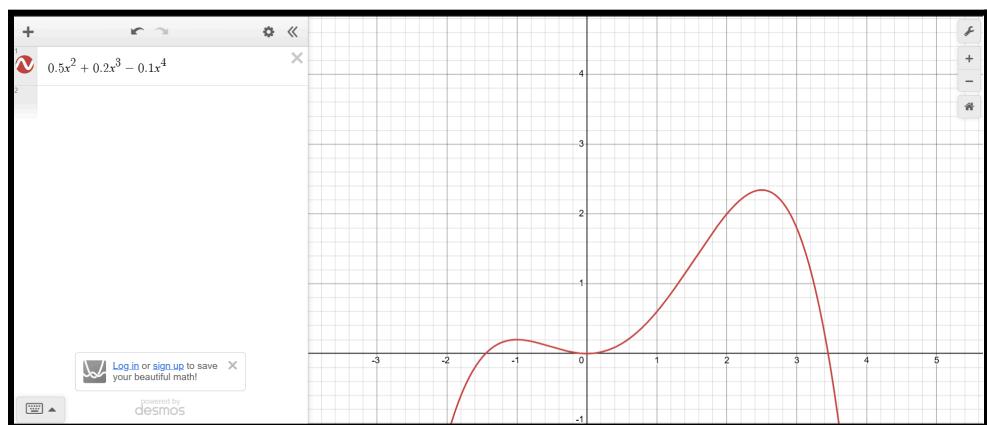
My system is defined with 2 state variables: Position of the cart along the x-axis & The overall velocity of the cart along the slope (as measured by the wheel encoders). The control input is the force being applied to the cart, along the tangential movement direction (basically the input acts on the wheels which supports the tangential movement of the cart on a hilly profile). I have also bound the maximum control effort to be 5N and the problem is to be solved in 10 time steps. Additionally, I constrained the final and the initial velocity to be 0 i.e. the cart starts at rest and stops after reaching its destination.

For simulation, I am starting my car 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 = mountainCarDynamics(t, x, u)
% Inputs:
%   t - time (scalar or array)
%   x - state vector [2 x n] where rows are [position; velocity]
%   u - control input (force) [1 x n]
% Outputs:
%   dx - state derivatives [2 x n] where rows are [velocity; acceleration]
% States:
%   x(1) -- Position of the cart along the x-axis
%   x(2) -- Overall velocity of the cart along the slope (as measured by the wheel
% encoders)
% Mountain car parameters
m = 1.0;           % mass (kg)
g = 9.81;          % gravity (m/s^2)
c = 0.01;          % viscous damping coefficient
pos = x(1,:);
vel = x(2,:);
% Mountain profile: height as a function of position y = 0.5*x^2 - 0.1*x^4 + 0.2*x^3
dy_dx = pos - 0.4*pos.^3 + 0.6*pos.^2;
slope_angle = atan(dy_dx);
dx1 = vel.*cos(slope_angle); % Velocity Component along x-axis
% F = ma along the slope
F_gravity = -m * g * sin(slope_angle);
F_damping = -c * vel;           % viscous damping
F_input = u;                   % user-applied force along slope
% Acceleration (along the slope)
dx2 = (F_gravity + F_damping + F_input) ./ (m);
% Output derivative vector (ensure correct dimensions)
dx = [dx1; dx2];
end
```

The hill profile was generated using a bi-quadratic function that looked cool on DESMOS:



Methodology

1. Trajectory Optimization

Based on the system model, I used the OptiTraj tool box to find the optimal trajectory of the system given the following constraints and boundary conditions: gravity = -9.8, maximum control effort = +/-10 units (This is calculated based on the physics which requires at least 10 units of force to the cart to defy gravity and move forward), number of assumption points = 50. My main path objective is to minimize the control effort. The optimization was solved using MATLAB's fmincon (Interior Point Method)

2. Open-Loop Simulation

The optimal control sequence $u^*(t) \cup \ast(t) \cup^*(t)$ was applied to the dynamic model using:
`[t_ode, x_ode] = ode45(@(t, x) mountainCarDynamics(t, x, uStar(t)), tGrid, x0);`
Comparison was made between: OptimTraj trajectory and ODE45 simulation. The trajectory error was plotted as “ $|x_{ode} - x_{trajopt}|$ ” and other statistics were evaluated.

3. Linearization & LQR Tracking

Linearization is done by evaluating the differentials of A and B matrices for each time-step followed by time dependent LQR (this is not Time-Varying LQR which solves backward iteration of the Riccati equation for the entire path) for gain calculation. The current implementation simply finds the A, B matrix for each time step and applies infinite-horizon, constant-gain LQR using the “lqr(A,B,Q,R)” solver from MATLAB. This assumption works well because we don't have any discontinuities (system is slowly varying) in our optimal trajectory. Formally, this method is called “time-varying LQR with frozen gains”.

Results and Discussion

It is interesting to note that the trajectory optimization toolbox (due to its algorithm) will generate a trajectory that “supposedly” solves the problem even though it is practically impossible. This is clearly seen when the trajectory is compared with actual trajectory (ODE45) and thus, the LQR gains blow off. For experimental speculations, I first deliberately set the TO gain bounds to a very low number and observe the uncontrollable LQR implementation. Notice, even with LQR I am not able to get non-zero final errors. Then, I calibrated the TO bounds in experiment 2 to get better results:

1. Trajectory Optimization Results
 - a. The optimized trajectory showed that the car must first drive backward to gain enough kinetic energy to climb the hill.
 - b. Control effort is minimized while still meeting final-state constraints.
 - c. Higher force limits produced faster solutions but with greater energy usage.
2. Open-Loop vs ODE45 Simulation
 - a. Small but noticeable deviations occurred, especially in high-curvature parts of the trajectory.
 - b. Errors grew over time due to system sensitivity and discretization of TO.
3. Closed-Loop LQR Tracking
 - a. LQR significantly reduced trajectory error.
 - b. The system remained close to the optimized path even with perturbations or numerical mismatch.
 - c. Closed-loop control successfully eliminated divergence present in the open-loop case.
4. The following plots demonstrate the results generated:
 - a. **IMPRactical** – If I bound the control effort to +/-3 (while solving the TO)



Fig1: Animation final view for the current impractical setting

i. PART A – Trajectory Optimization Results

```
Final time: 5.000 seconds  
Cost (integrated u^2): 40.640
```

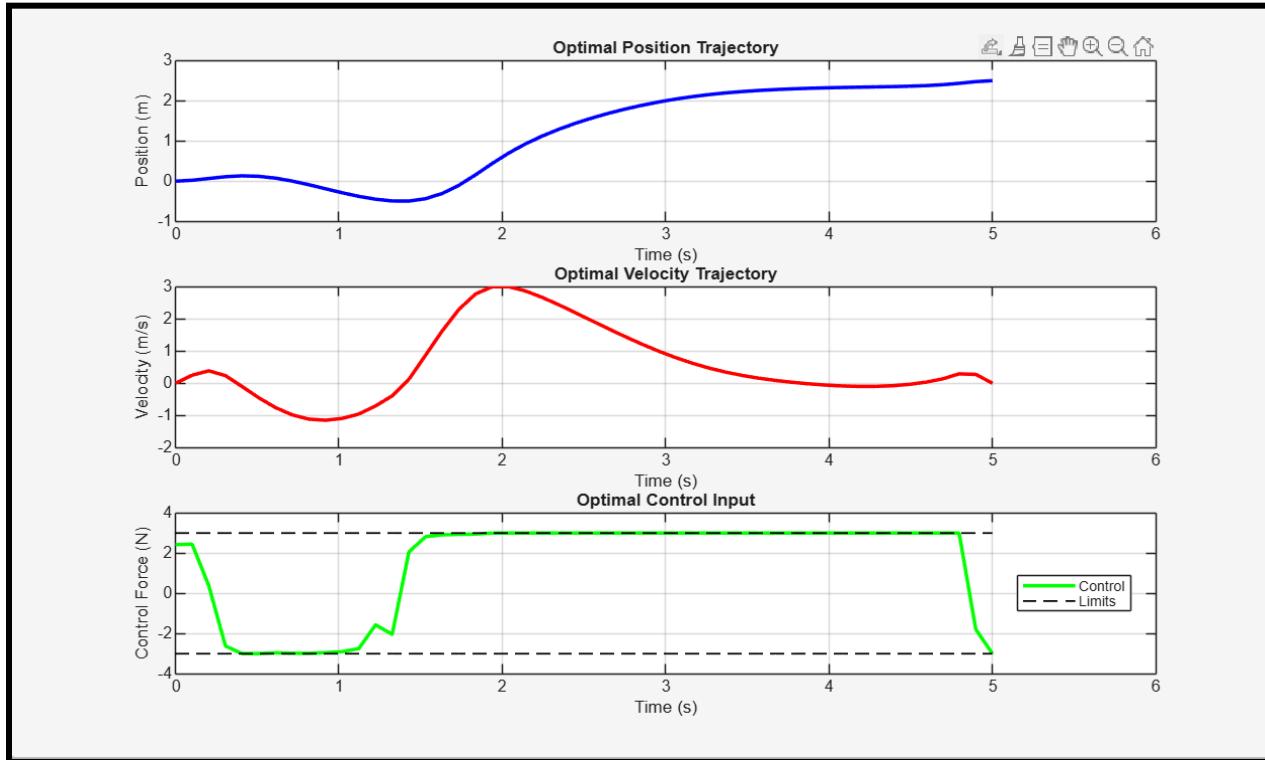


Fig2: Trajectory Optimization Results

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

```
Maximum position error: 6.685974  
Maximum velocity error: 22.831889  
RMS position error: 2.866023  
RMS velocity error: 7.869064
```

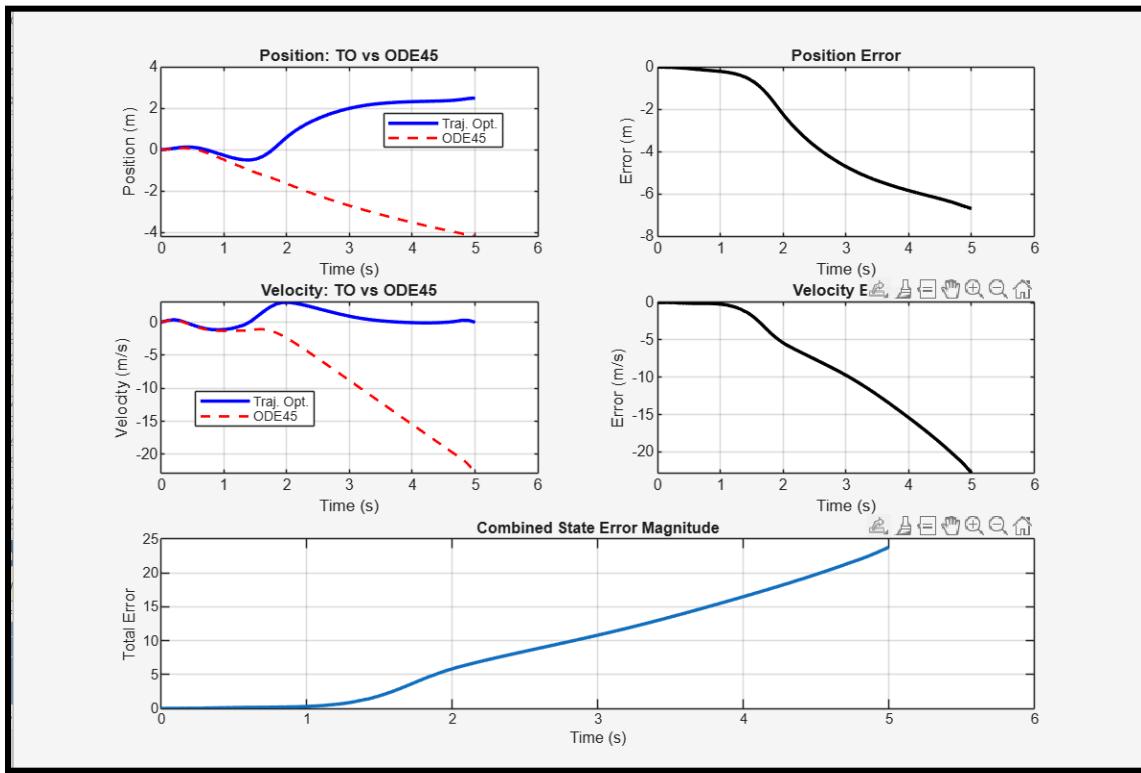


Fig3: Open Loop error analysis for TO path vs ODE45 simulated path

iii. PART C – Closed loop solutions with LQR feedback

```
Final position error (open-loop): -6.6860
Final position error (closed-loop): -0.0856
```

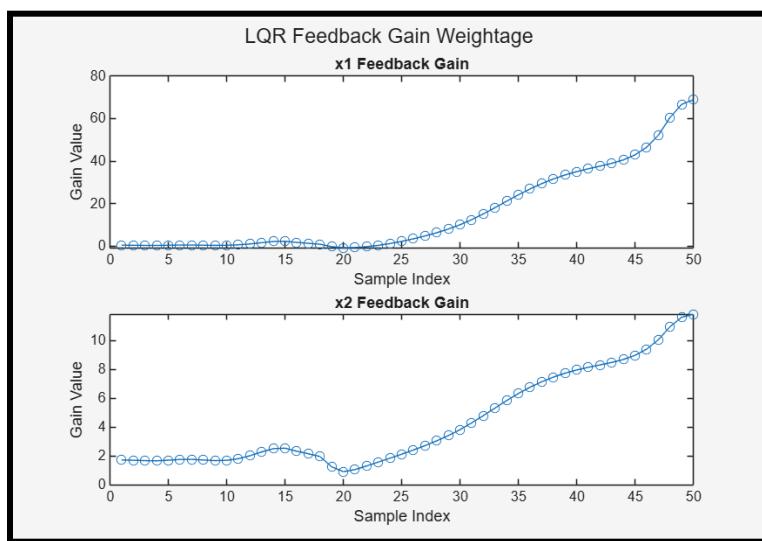


Fig4: LQR Controller Feedback Gains - K matrix

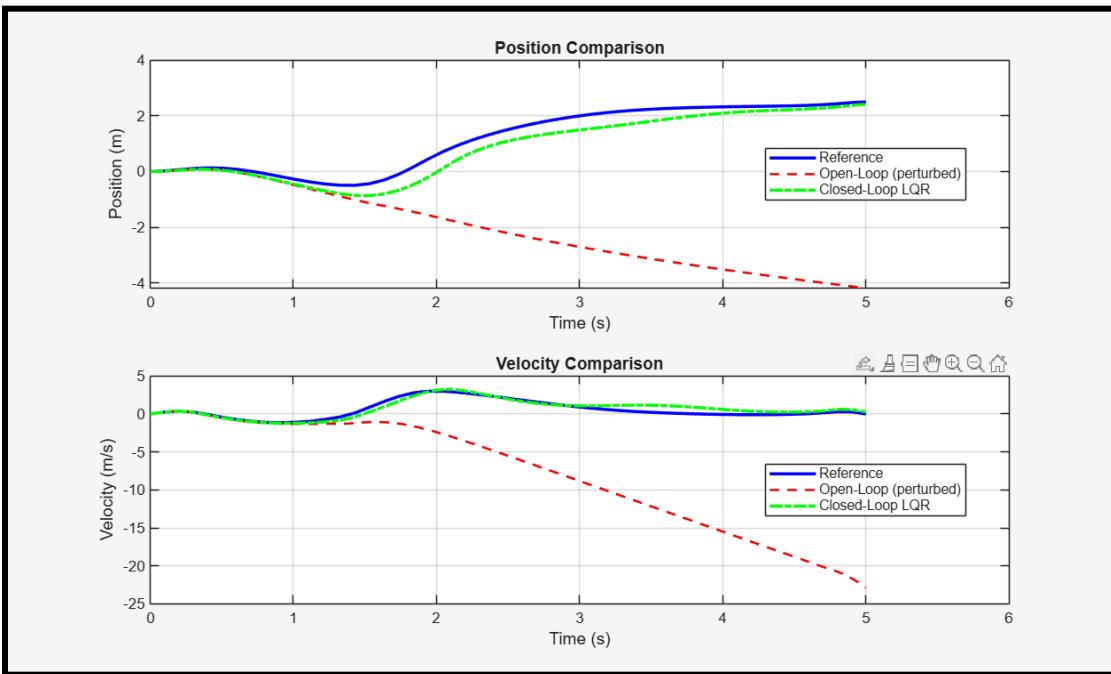


Fig5: State Variables comparison for all three methods

b. PRACTICAL – If I bound the control effort to +/-9 (while solving the TO)

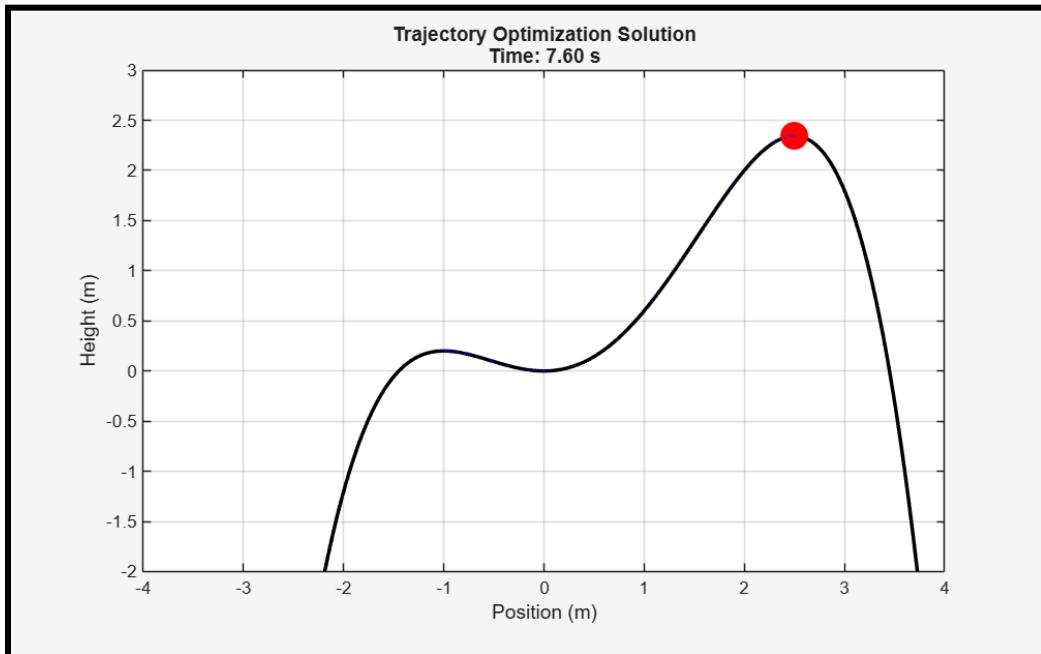


Fig6: Animation final view for the current practical setting

i. PART A – Trajectory Optimization Results

Final time: 8.273 seconds
Cost (integrated u^2): 50.806

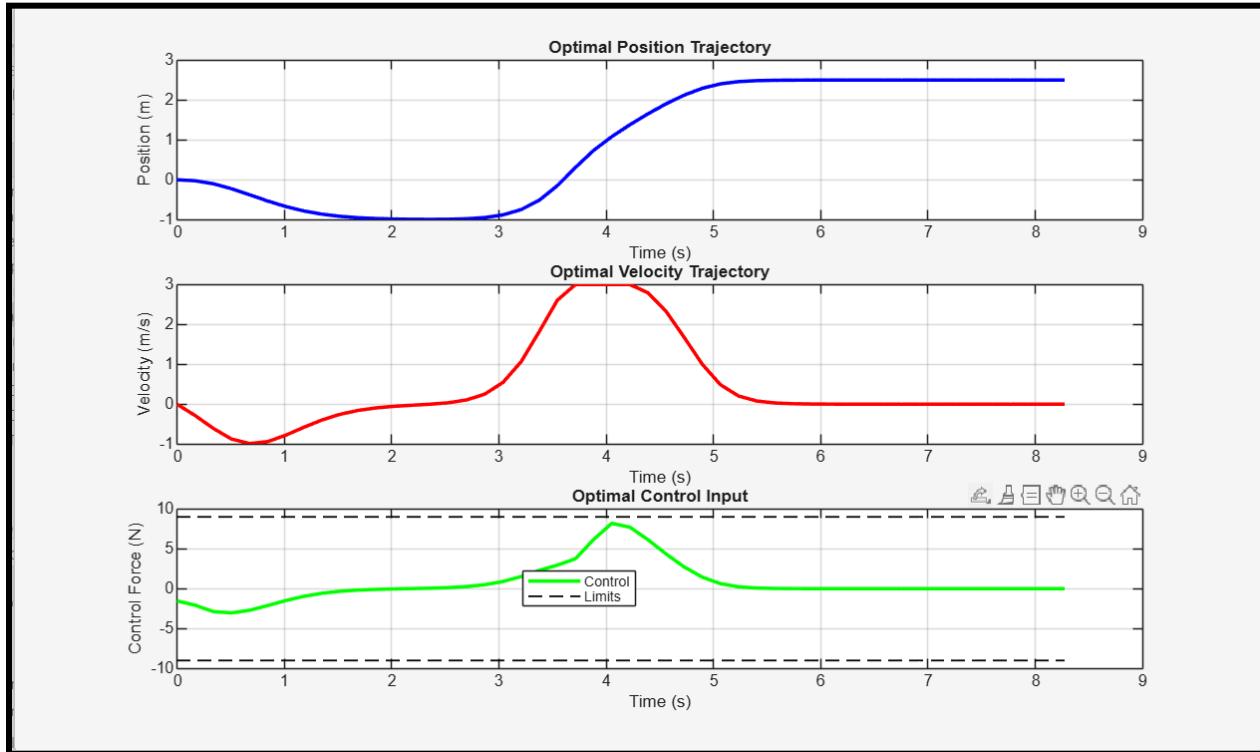


Fig7: Trajectory Optimization Results

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

Maximum position error: 3.326277
Maximum velocity error: 34.304546
RMS position error: 0.948447
RMS velocity error: 9.033128

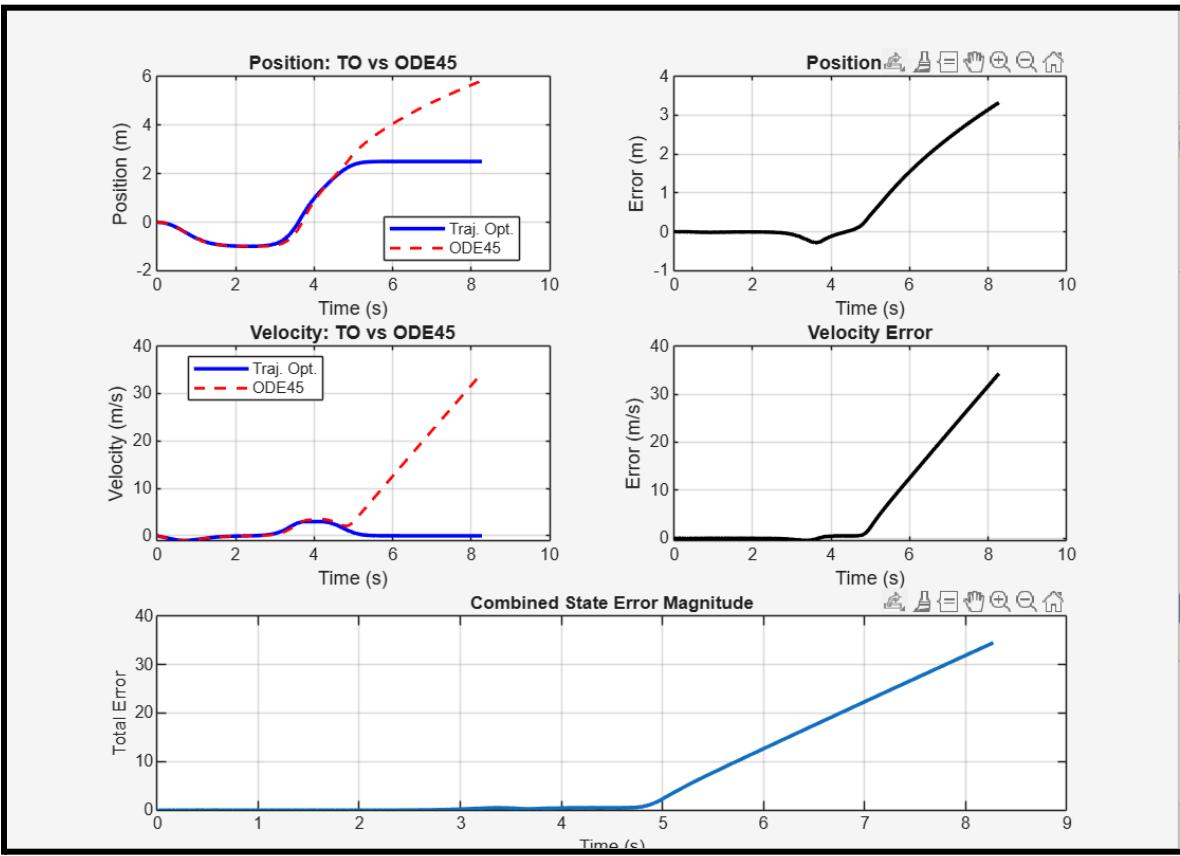


Fig8: Open Loop error analysis for TO path vs ODE45 simulated path

iii. PART C – Closed loop solutions with LQR feedback

```
Final position error (open-loop): 3.3263
Final position error (closed-loop): -0.0000
```

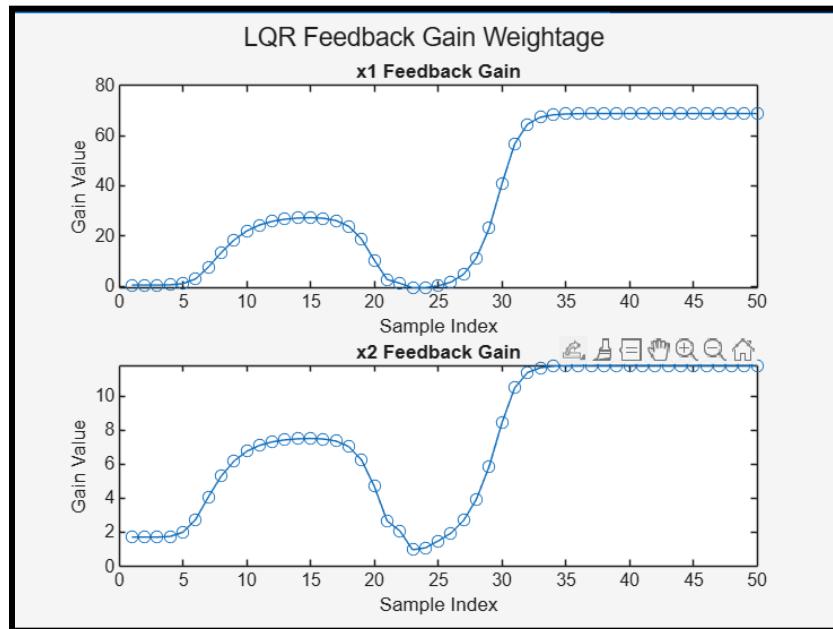


Fig9: LQR Controller Feedback Gains - K matrix

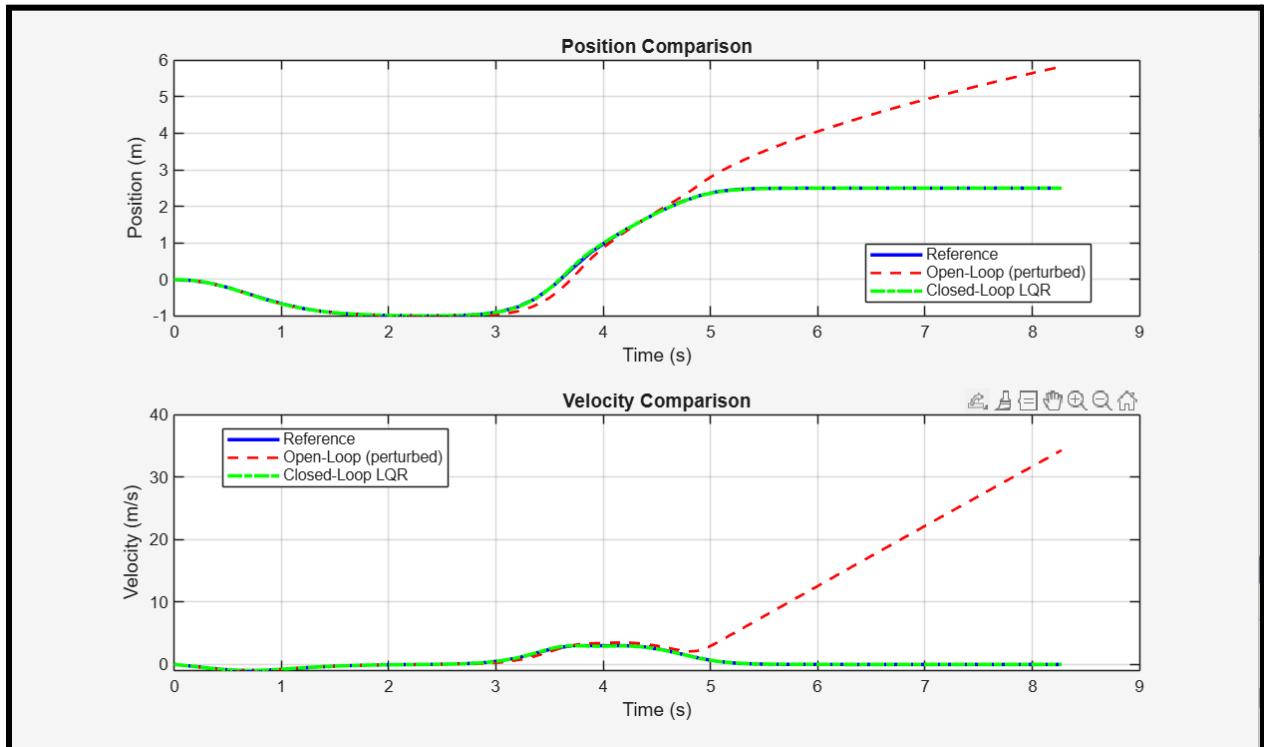


Fig10: State Variables comparison for all three methods

Conclusion

This project demonstrated the effectiveness of trajectory optimization combined with time-varying LQR stabilization for nonlinear systems such as Mountain Car. The TO-generated open-loop trajectory provides a powerful initial solution but lacks robustness when executed directly. Linearizing along the trajectory and applying an LQR controller significantly improves tracking performance, validating the hypothesis that closed-loop control is essential for practical implementation of trajectory-optimized paths. It is essential to set physics-defined practical bounds before solving TO to avoid catastrophic explosion of the feedback controller gains in the tracking controller.

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>.