Trajectory Optimization for Inverted Double Pendulum with DDP

1st Hao Liu
Department of Robotics
University of Michigan
Ann Arbor, MI, United States
wdliu@umich.edu

2nd Junkai Zhang
Department of Robotics
University of Michigan
Ann Arbor, MI, United States
junkaiz@umich.edu

***Abstract*—Our research project aimed to compare the effectiveness of Differential dynamic programming (DDP) and Model Predictive Path Integral (MPPI) in controlling the inverted double pendulum. We found that both techniques were effective, but DDP outperformed MPPI in terms of tracking accuracy and energy consumption. However, MPPI was faster in computing the control policy than DDP. The choice of optimization technique should depend on the specific requirements of the control application. These findings can be useful in designing control strategies for complex and highly nonlinear systems in various applications such as robotics, aerospace, and transportation.**

***Index Terms*—Inverted double pendulum, Differential dynamic programming, Model Predictive Path Integral, Nonlinear control, Model Predictive Control (MPC)**

I. Introduction

The inverted double pendulum is a challenging mechanical system to control due to its inherent instability and highly nonlinear dynamics. As a result, researchers have developed various advanced optimization techniques to design effective control strategies for this system. Two such techniques are Differential dynamic programming (DDP) and Model Predictive Path Integral (MPPI).

In this research project, we compared the performance of DDP and MPPI in controlling the inverted double pendulum. DDP was implemented in a MPC framework in conjunction with MPPI method.The primary objective of this research was to determine which technique was more effective in controlling the system. Our simulations focused on evaluating the tracking accuracy and computation energy consumption of the two techniques.

The findings of our research can provide insights into the relative merits of these optimization techniques in controlling complex and highly nonlinear systems. This knowledge can be useful in designing control strategies for a range of applications, including robotics, aerospace, and transportation. Therefore, this research can have significant implications for the development of advanced control strategies for complex mechanical systems.

II. Method and Implementation

A. Dynamic Model of Inverted Double Pendulum

Figure. 1 shows the image of an inverted double pendulum system on a cart. This system is a variation of the classic inverted double pendulum problem, where the two rods are mounted on a cart that moves along a horizontal track. This variation adds a degree of freedom to the system, making it a more challenging control problem.

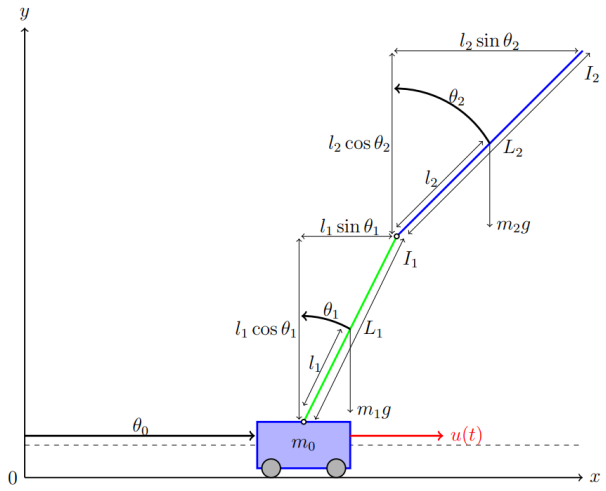

Fig. 1. Dynamic model of inverted double pendulum [1]

In this configuration, the cart is usually considered to be the actuator, as it can move back and forth along the track to control the motion of the system. The two rods are connected to the cart through two revolute joints, which allow them to rotate freely.

The state of the system [1] is defined to be:

$$x = [\theta_0, \dot{\theta}_0, \theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2]^T \tag{1}$$

where θ_0 is the displacement of the cart, θ_1 is the angle between the first rod and vertical direction and θ_2 is the angle between the second rod and the vertical direction.

Derived from Lagrangian mechanics, the dynamic model of the system can be represented as follows:

$$D(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) = Hu \tag{2}$$

where

$$D(\theta) = \begin{bmatrix} m_0 + m_1 + m_2 & (\frac{1}{2}m_1 + m_2)L_1 cos\theta_1 & \frac{1}{2}m_2 L_2 cos\theta_2 \\ (\frac{1}{2}m_1 + m_2)L_1 cos\theta_1 & (\frac{1}{3}m_1 + m_2)L_1^2 & \frac{1}{2}m_2 L_1 L_2 cos(\theta_1 - \theta_2) \\ \frac{1}{2}m_2 L_2 cos\theta_2 & \frac{1}{2}m_2 L_1 L_2 cos(\theta_1 - \theta_2) & \frac{1}{3}m_2 L_2^2 \end{bmatrix}$$

and

$$C(\theta,\dot{\theta}) = \begin{bmatrix} 0 & -(\frac{1}{2}m_1 + m_2)L_1 sin\theta_1\dot{\theta_1} & -\frac{1}{2}m_2 L_2 sin\theta_2\dot{\theta_2} \\ 0 & 0 & \frac{1}{2}m_2 L_1 L_2 sin(\theta_1-\theta_2)\dot{\theta_2} \\ 0 & -\frac{1}{2}m_2 L_1 L_2 sin(\theta_1-\theta_2)\dot{\theta_1} & 0 \end{bmatrix}$$

$$G(\theta) = \begin{bmatrix} 0 \\ -\frac{1}{2}(m_1+m_2)L_1 g sin\theta_1 \\ -\frac{1}{2}m_2 g L_2 sin\theta_2 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The state space system can be written as:

$$\dot{x} = \begin{bmatrix} 0 & I \\ 0 & -D^{-1}C \end{bmatrix} x + \begin{bmatrix} 0 \\ -D^{-1}G \end{bmatrix} + \begin{bmatrix} 0 \\ D^{-1}H \end{bmatrix} u \quad (3)$$

In the case of the inverted double pendulum on a cart, the control target is to stabilize the system in the upright position while keeping the cart stationary at a desired position on the track. This is a challenging control problem that can be approached using state-feedback control laws or optimal control methods. In this project, we are using MPPI and DDP.

B. MPPI

MPPI is a stochastic control algorithm that uses the path integral formulation to solve non-linear, non-convex optimal control problems [2]. The path integral formulation represents the control policy as a probability distribution over all possible control trajectories, rather than a deterministic function. The control policy is optimized by minimizing a cost function that integrates over the entire trajectory, subject to the system dynamics and any constraints.

MPPI uses a sample-based approach to estimate the optimal control policy. Control actions are generated by sampling random perturbations around a nominal control trajectory, which are propagated forward through the system dynamics to obtain a set of state trajectories. The cost associated with each state trajectory is evaluated using a user-defined cost function, and the optimal control policy is computed over a finite time horizon using a recursive update formula. Despite some limitations, MPPI has shown promising results in various control and robotics applications, making it a valuable tool for solving complex control problems.

C. DDP

DDP was initially introduced by David Mayne [3] in 1965 as one of the earliest trajectory optimization methods in the field of optimal control. This method is an extension of Dynamic Programming, which focuses on optimizing around a nominal trajectory by utilizing second-order Taylor approximations, instead of optimizing over the entire state space. By repeatedly implementing this process, DDP allows us to obtain local solutions for non-linear trajectory optimization problems. Despite the simplicity of DDP, Despite its apparent simplicity, DDP is an extremely powerful method, which has recently been simplified into iLQR by Tassa et al [4]. The DDP implemented in this project is based on the work by Houghton et al [5]. This section only gives brief overview of the derivation, equation used and procedure. Interested readers are referred to works such as [5], [6] for a more thorough derivation and analysis of the convergence.

The dicrete-time trajectory optimization problem is defined as following:

$$\mathcal{J}(\mathbf{U}) = \min_{\mathbf{U}} \sum_{t=1}^{T-1} \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t) + \phi(\mathbf{x}_T) \quad (4)$$

subject to

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t) \quad (5)$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is the state and $\mathbf{u}_t \in \mathbb{R}^m$ is the control at time t. $\mathbf{X} := \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ and $\mathbf{U} := \{\mathbf{u}_1, \ldots, \mathbf{u}_{T-1}\}$ are sequences of state and control over finite control horizon $\mathbf{T} \in \mathbb{N}^+$. The running cost and terminal cost are defined as $\mathcal{L}$ and ϕ in Eq. (4) respectively. To solve this optimization problem, DDP runs backward pass and forward pass iteratively, until the cost for the trajectory converges. To begin the algorithm, an initial state trajectory is calculated based on initial state and a random control sequence. Then for backward pass part, small correction sequence on control sequence is calculated based on the equations below. The forward pass updates control sequence and state sequence accordingly based on corrections in backward pass, then the new sequences are sent into the backward pass for the next iteration. Pseudocode for DDP is summarized in Algorithm 1 to give reader a clearer understanding.

Derivation of backward step is briefly shown in the following: a value function representing cost-to-go from a state $\mathbf{x_i}$ in state sequence $\mathbf{U}$ is first defined as:

$$V(\mathbf{x}_i) := \min_{\mathbf{U}_i} \mathcal{J}(\mathbf{x}_i, \mathbf{U}_i), \quad (6)$$

which can be solved recursively based on Bellman's principle of optimality using this equation:

$$V(\mathbf{x}_i) = \min_{\mathbf{u}_i} [\underbrace{\mathcal{L}(\mathbf{x}_i, \mathbf{u}_i) + V(\mathbf{x}_{i+1})}_{Q(\mathbf{x}_i, \mathbf{u}_i)}]. \quad (7)$$

Then we approximate the $\mathbf{Q}$ around some nominal trajectory or initial guess using 2nd order Taylor expansion. By taking the higher order term as zero, we can get the expression for action changes:

$$\delta\mathbf{u}_i^* = \mathbf{k}_i + \mathbf{K}_i \delta\mathbf{x}_i \quad (8)$$

where:

$$\begin{aligned} \mathbf{k}_i &:= -(Q_i^{uu})^{-1} Q_i^u \\ \mathbf{K}_i &:= -(Q_i^{uu})^{-1} Q_i^{ux} \end{aligned} \quad (9)$$

where

$$\begin{aligned} Q_i^x &= \mathcal{L}_i^x + (\mathbf{f}_i^x)^\top V_{i+1}^x, \\ Q_i^u &= \mathcal{L}_i^u + (\mathbf{f}_i^u)^\top V_{i+1}^x, \\ Q_i^{xx} &= \mathcal{L}_i^{xx} + (\mathbf{f}_i^x)^\top V_{i+1}^x \mathbf{1}_i^x + V_{i+1}^x \cdot \mathbf{f}_i^{xx}, \\ Q_i^{ux} &= \mathcal{L}_i^{ux} + (\mathbf{f}_i^u)^\top V_{i+1}^x \mathbf{f}_i^x + V_{i+1}^x \cdot \mathbf{f}_i^{ux}, \\ Q_i^{uu} &= \mathcal{L}_i^{uu} + (\mathbf{f}_i^u)^\top V_{i+1}^{xx} \mathbf{f}_i^u + V_{i+1}^x \cdot \mathbf{f}_i^{uu}. \end{aligned} \quad (10)$$

Algorithm 1 : Differential Dynamic Programming

Require: Nominal state and control trajectory $\overline{x}_t$, $\overline{x}_t$
  while *not converged* **do**
    // Backward Pass
    $V_T^0 \leftarrow \phi_T^0$
    $V_T^x \leftarrow \phi_T^x$
    $V_T^{xx} \leftarrow \phi_T^{xx}$
    for $t = T-1$ to 1 **do**
      Calculate $Q_t^x, Q_t^u, Q_t^{xx}, Q_t^{ux}, Q_t^{uu}$ based on Eq. 12
      if Q_{uu} *not invertible* **then**
        Increase regularization parameters μ_1 and μ_2
      end if
    end for
    Calculate gains $\mathbf{K}_t$ and $\mathbf{k}_t$ based on Eq. 9
    Update V_t^x and V_t^{xx} based on Eq. 13
  end while
  // End Backward Pass
  Decrease regularization parameters μ_1 and μ_2
  // Line search
  while *Cost not decreased* **do**
    // Forward Pass
    $x_1 \leftarrow x_1$
    for $t = 1$ to $T-1$ **do**
      $\delta x_t \leftarrow x_t - \overline{x}_t$
      $u_t \leftarrow \overline{u}_t + \epsilon k_t + K_t \delta x_t$ from Eq. 11
      $x_{t+1} \leftarrow f(x_t, u_t)$
      // End Forward Pass
      $\epsilon \leftarrow \rho\epsilon$ // Decrease step size
    end for
    // End Line Search
    $\overline{x}_t \leftarrow x_t$
    $\overline{u}_t \leftarrow u_t$
  end while
  return $\overline{u}_t$

Moreover, line search technique and regularization is implemented to help the convergence [4]. Therefore, following equations are used to update control sequence:

$$\delta \mathbf{u}_i^* = \epsilon \mathbf{k}_i + \mathbf{K}_i \delta \mathbf{x}_i \tag{11}$$

where k_i and K_i are defined same as Eq. 9.

$$\begin{aligned}
Q_i^x &= \mathcal{L}_i^x + (\mathbf{f}_i^x)^\top V_{i+1}^x, \\
Q_i^u &= \mathcal{L}_i^u + (\mathbf{f}_i^u)^\top V_{i+1}^x, \\
Q_i^{xx} &= \mathcal{L}_i^{xx} + (\mathbf{f}_i^x)^\top \left(V_{i+1}^{xx} + \mu_1 \mathbf{I}\right) \mathbf{f}_i^x, \\
Q_i^{xu} &= \mathcal{L}_i^{xu} + (\mathbf{f}_i^x)^\top \left(V_{i+1}^{xx} + \mu_1 \mathbf{I}\right) \mathbf{f}_i^u, \\
Q_i^{uu} &= \mathcal{L}_i^{uu} + (\mathbf{f}_i^u)^\top \left(V_{i+1}^{xx} + \mu_1 \mathbf{I}\right) \mathbf{f}_i^u + \mu_2 \mathbf{I}
\end{aligned} \tag{12}$$

Then V_x and V_{xx} are updated based on:

$$\begin{aligned}
V_i^x &= Q_i^x + \mathbf{K}_i^\top Q_i^{uu} \mathbf{k}_i + \mathbf{K}_i^\top Q_i^u + (Q_i^{ux})^\top \mathbf{k}_i, \\
V_i^{xx} &= Q_i^{xx} + \mathbf{K}_i^\top Q_i^{uu} \mathbf{K}_i + \mathbf{K}_i^\top Q_i^{ux} + (Q_i^{ux})^\top \mathbf{K}_i.
\end{aligned} \tag{13}$$

D. MPC

Model Predictive Control (MPC) is a popular optimal control technique that is widely used for real-time control of nonlinear systems due to its effectiveness [7]. The primary objective of MPC is to achieve a stable closed-loop control law by computing and solving a discrete-time optimal control problem for a given planning time horizon $H << T$. The computational advantages of MPC over optimizing for the full time horizon T lie in the time complexity of the optimal control solver, which usually grows linearly (or worse) with the length of the planning horizon.

At each time instant t, the MPC approach involves solving for the optimal controls $u_t, u_{t+1}, \ldots, u_{t+H-1}$ for the horizon $t, t+1, \ldots, t+H$, executing the first control u_t, observing the state x_{t+1}, and repeating this process for the next horizon $t+1, t+2, \ldots, t+1+H$, executing the control u_{t+1}, and observing x_{t+2}, until the task is completed up to the terminal time T.

III. Experiment and Result

In this section, we present the experimental results obtained from the swing-up task of a double pendulum on a cart using DDP and MPPI controllers. We compare the performance of both controllers under different initial conditions.

A. Experiment Setup

1) Swing-up Task: The swing-up task involves bringing the pendulum from a hanging position to the upright position and maintaining it there. The initial states for the swing-up task are chosen randomly within a predefined range as mentioned above.

2) Initial States: We conducted experiments on a simulated double pendulum on a cart system with self-defined analytic dynamics. The system was controlled using either DDP or MPPI controllers. To compare the performance of the controllers under different conditions, the initial states of the system were varied across three different sets. These sets included relatively stable (upper-right) to highly unstable states (downwards), represented by the following initial states with configuration defined in figure. 1:

$$\begin{aligned}
x_0^1 &= [0, 0, 45°, 0, 45°, 0]^T \\
x_0^2 &= [0, 0, 90°, 0, 90°, 0]^T \\
x_0^3 &= [0, 0, 180°, 0, 180°, 0]^T
\end{aligned} \tag{14}$$

These initial states were selected to provide a comprehensive evaluation of the controller's performance in various scenarios and to demonstrate the controller's ability to swing-up and stabilize the double pendulum from highly unstable positions.

3) Goal Error Definition: Since our focus is on the swing-up task rather than the cart position control, we define the error as a measure of the deviation from the desired double pendulum state. Specifically, we convert the angle difference to the range $[-\pi, \pi]$ to ensure that the error metric is well-defined. To account for the fact that certain elements of the

state vector are more important than others, we weight the error terms using a diagonal matrix with the following scaling factors: $[0.0, 0.0, 1, 0.1, 1, 0.1]$. The overall error $error_i$ at time step i is then computed as the Euclidean norm of the weighted error vector. By using this error definition, we are able to focus on the swing-up task and quantify the performance of the controllers in achieving the desired double pendulum state.

B. Result

1) Computing Efficiency Comparison: Considering the real-time requirements of control scenarios and the relatively poor computational efficiency of DDP during initial iterations before convergence, we compared the average iteration times required for each step between DDP and MPPI. We used a horizon of 10 for both DDP and MPPI, and set the maximum iteration number to 50 for DDP and the sample number to 100 for MPPI. We compared the average iteration times required for the first five steps and reaching the goal during the entire process of swinging up from a natural hanging position to a vertical standing position using the two controllers. We performed 10 runs for each controller.

As shown in table I, for MPPI, the average iteration time for the first five steps and the total average iteration time were both around 0.8 seconds. In contrast, the average iteration time for the first five steps for DDP was 7.5 seconds, and the total average iteration time was 0.58 seconds. These results indicate that MPPI has a faster convergence rate and is more suitable for real-time control scenarios. However, it is worth noting that DDP's performance is significantly improved after the first few iterations when convergence is reached. In conclusion, our results demonstrate that MPPI is a better choice for real-time control scenarios where fast convergence is required, while DDP can still be an effective choice for control scenarios where computational efficiency is not the primary concern or convergence can be easily reached.

TABLE I
COMPARISON OF AVERAGE ITERATION TIME BETWEEN DDP AND MPPI CONTROLLERS

Controller	First 5 Steps Avg. (s/iter)	Total Avg. (s/iter)
DDP	7.5	0.58
MPPI	0.8	0.8

2) Convergence Rate: In addition to the average iteration time, we also analyzed the convergence rate of the controllers by measuring the number of iterations required to reach the goal state, defined as a goal error less than 0.2. Because this is a more appropriate metric for comparing the performance of the controllers considering computational limitations can be overcome through parallel computing and other methods [8]. We ran both DDP and MPPI controllers for the same set of initial states and horizon length as in the previous section. Figure. [2 3 4] shows the number of iterations required by each controller to reach the goal state from the three different initial states.

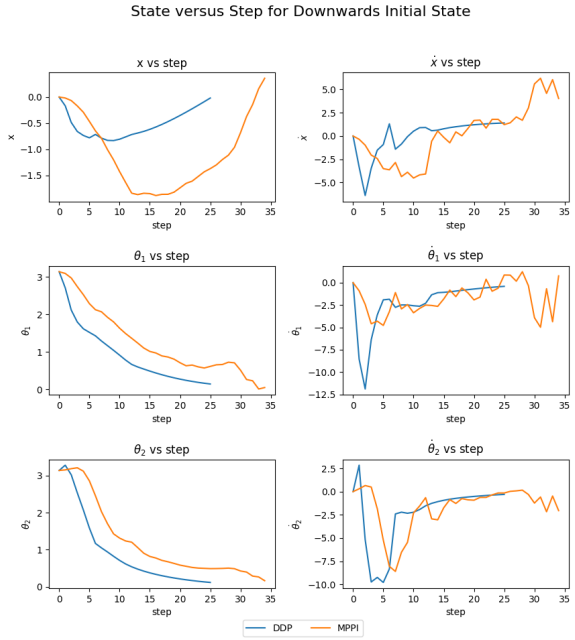

Fig. 2. States vs Step Comparison for DDP and MPPI from Downward State

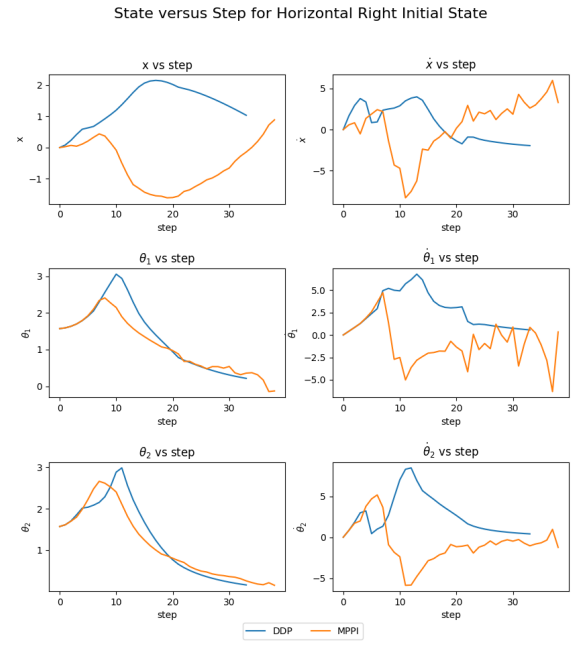

Fig. 3. States vs Step Comparison for DDP and MPPI from Horizontal State

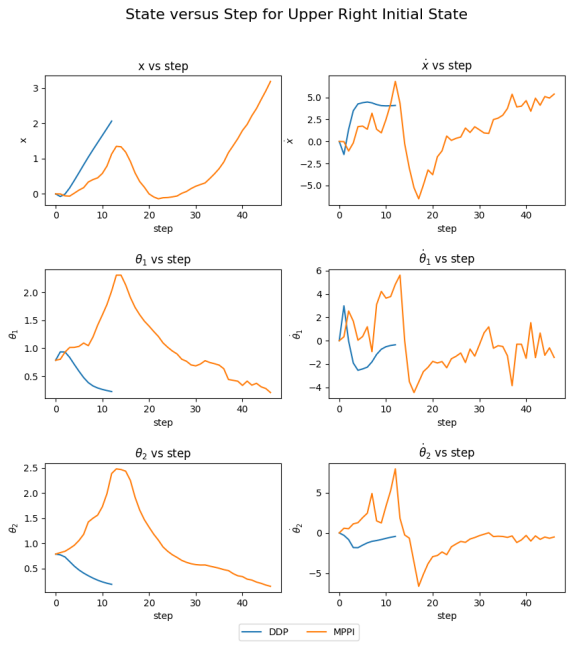

Fig. 4. States vs Step Comparison for DDP and MPPI from Upper-right State

As can be seen from the figure, DDP consistently outperforms MPPI in terms of convergence rate, requiring significantly fewer iterations to reach the goal state from any initial state ($\sim$ 40% less). The average converge step for DDP is aorund 24.2 while 40.3 for MPPI. This result suggests that DDP is more effective in controlling the double pendulum system and achieving the desired state and DDP should be prioritized when implementing control in systems with sufficient computing power.

3) Goal Error: The phase plot for initial state as downward is shown in Figure. 5. In the phase plot, the end point represents the final error of the controller after convergence. The fact that the end point for MPPI is farther from the zero point compared to DDP indicates that MPPI has a higher final error compared to DDP. This suggests that DDP may be a better controller in terms of minimizing the final goal error.

4) Stability and Robustness Analysis: In the phase plot, the smoothness of the trajectory indicates the stability and robustness of the controller. The smoother the trajectory, the more stable and robust the controller is. In this case, the phase plot for DDP is relatively smooth compared to MPPI, indicating that DDP is a more stable and robust controller compared to MPPI. This suggests that DDP may be a better controller for applications where stability and robustness are critical factors. However, it is important to note that the stability and robustness of a controller depend on various factors such as the system dynamics, controller parameters, and operating conditions, and further analysis may be needed to confirm this conclusion.

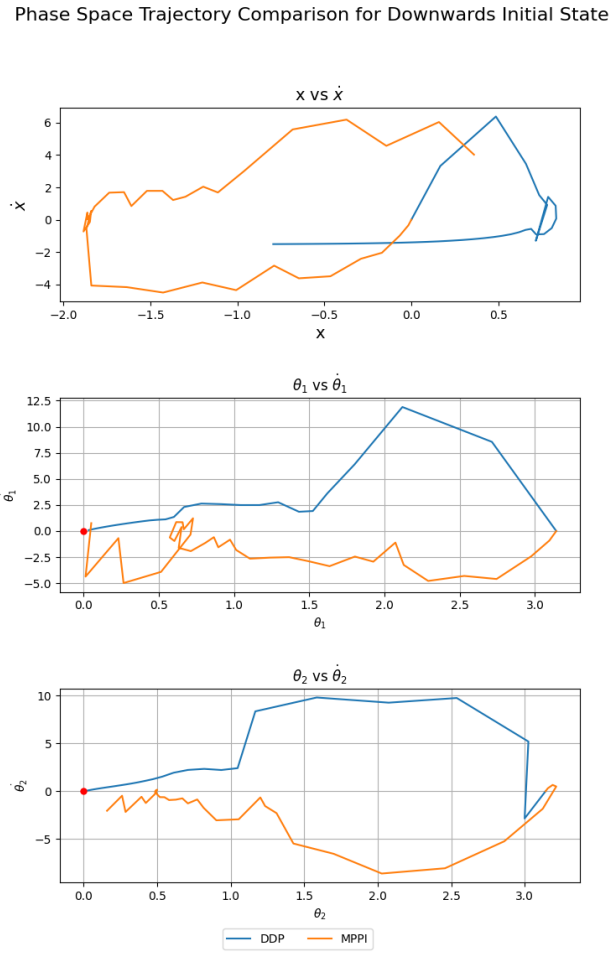

Fig. 5. Phase Plot Comparison for DDP and MPPI from Downward State

IV. Discussion

Despite achieving good results in angle control of the two rods, stabilizing the cart on the ground using both MPPI and DDP remains challenging. MPPI, being a stochastic controller, can lead to suboptimal control due to its randomness, which might not provide the best control for the system's stability. Moreover, the highly sensitive nature of the dynamic system can cause even subtle control errors to result in the collapse of the stable state. Both MPPI and DDP suffered from imperfect weight assignment for each state element.

During parameter tuning, we found that a large horizon for both DDP and MPPI can lead to SVD errors and numerical instability issues during the backward pass of the algorithm due to the ill-conditioned or singular Hessian matrix.

To sum up, in this project, we have investigated the application of two optimization methods, DDP and MPPI, to control an inverted double pendulum system on a cart. Our results show that DDP outperforms MPPI in terms of control accuracy but at the cost of longer computation time.

Future work includes exploring other optimization methods

like reinforcement learning, parallel computing control algorithm, and exploring the applicability of the developed control methods to other nonlinear systems.

REFERENCES

[1] N. J. Medrano, "Optimal control of the inverted double pendulum using differential dynamic programming and model predictive path integral," Master's Thesis, University of New Mexico, 2018.

[2] G. Williams, H. Chen, B. Houska, A. Mitsos, and J. B. Rawlings, "Model predictive path integral control: From theory to parallel computation," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1392–1399, 2017.

[3] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *http://dx.doi.org/10.1080/00207176608921369*, vol. 3, pp. 85–95, 2007. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00207176608921369

[4] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4906–4913, 2012.

[5] M. D. Houghton, A. Oshin, M. J. Acheson, E. A. Theodorou, and I. M. Gregory, "Path planning: Differential dynamic programming and model predictive path integral control on vtol aircraft," *AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022*, 2022. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2022-0624

[6] E. Todorov and W. Li, "A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems," *Proceedings of the American Control Conference*, vol. 1, pp. 300–306, 2005.

[7] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, pp. 335–348, 5 1989.

[8] B. Plancher and S. Kuindersma, "A performance analysis of parallel differential dynamic programming on a gpu."