Trajectory Optimization: inverted double-pendulum on a cart via MPPI and DDP

Po-Jen Chen, Yu-Ju Chiu
University of Michigan
cpojen, yujuchiu@umich.edu

***Abstract*— This paper explores the feasibility of two optimal control methods widely used in robotics as trajectory planners for dual pendulum systems, which are nonlinear systems. The first method is DDP, which uses second-order approximations along nominal trajectories and shows quadratic convergence to local solutions. The second method is Model Predictive Path Integral Control (MPPI), a random sampling-based algorithm that can be optimized for general cost criteria, including potentially highly nonlinear formulations. The results show that both methods were able to successfully control a double pendulum to the desired position.**

I. Introduction

Differential dynamic programming (DDP) and Model Predictive Path Integral (MPPI) are both optimization techniques based on Model Predictive Control (MPC), used to solve optimal control problems in dynamic systems, particularly when dealing with nonlinear systems or nonlinear cost functions. These techniques enable robots to optimize their motion trajectories in real-time, based on sensor data and the dynamics of the environment, leading to improved performance, energy efficiency, and safety.

The main difference between DDP and MPPI is their approach to solving optimal control problems. DDP performs dynamics (Taylor Expansion) on the dynamics and cost models and iteratively updates the control input and state trajectory until a local optimal solution is obtained. In contrast, MPPI uses a global optimization approach that connects random paths for current states, samples from the distribution of control inputs, and computes a cost per sample to obtain an approximate global optimum.

The advantage of DDP is that it can converge faster to a locally optimal solution, making it more suitable for problems where real-time control is important. However, DDP can get trapped in local minima and fail to find the global optimum. MPPI, on the other hand, has a higher chance of finding the global optimum but may require more computational resources and time to converge.

In summary, DDP and MPPI are powerful techniques for solving optimal control problems in robotics and other fields. While they have different approaches to solving the problem and different advantages and disadvantages, they both offer valuable tools for improving the performance, safety, and efficiency of autonomous systems.

II. Implementation

A. Dynamics model

An inverted double-pendulum is a mechanical system consisting of two pendulums connected in series, with the second pendulum inverted (i.e., hanging upwards). The two pendulums are connected by a hinge at their common point of suspension, and the system is free to rotate about that point. Our goal of controlling the double-pendulum is to hope that the double-pendulum can stand on the car and make the double-pendulum appear in a straight line.

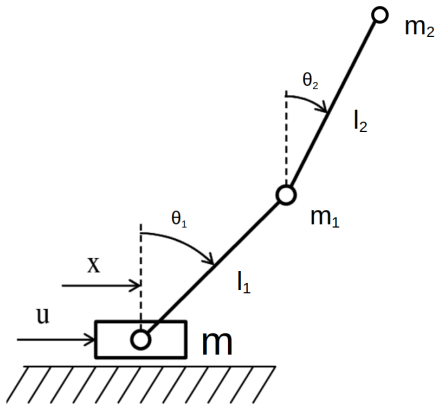

Fig. 1. The dynamics model of inverted double-pendulum system

For this system, the State Vector and the Control Vector of the form.

$$s_t = \left[x, \theta_1, \theta_1, dx/dt, d\theta_1/dt, d\theta_1/dt\right] \tag{1}$$

$$u_t = \left[u_x\right] \tag{2}$$

In the next section we will implement MPPI and DDP to update the $\mathbf{u}_t$ to control and change the state.

Moreover, We modified the urdf file to visualize the double pendulum2, and the basic settings of the double pendulum are as follows:

$$\begin{cases} M = & 1 \\ m_1 = & 0.1 \\ m_2 = & 0.1 \\ l_1 = & 1.0 \\ l_2 = & 1.0 \end{cases} \tag{3}$$

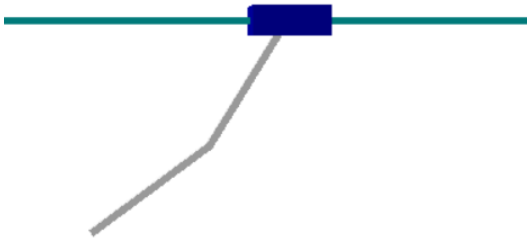

Fig. 2. The visualization of inverted double-pendulum system via urdf (initial state)

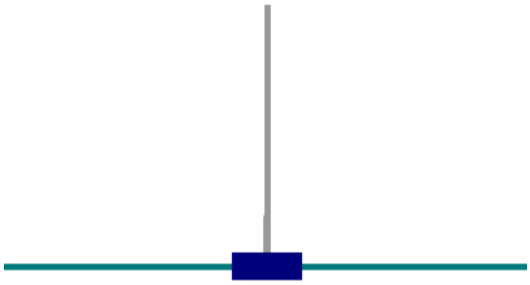

Fig. 3. The visualization of inverted double-pendulum system via urdf (goal state)

B. Optimal Trajectory Problem

The trajectory optimization problem as well as the algorithms that enable planning over nonlinear dynamics. And Our goal is to find optimal control (and states) to achieve a desired end state while minimizing cost function of the form.

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

$$L(\mathbf{x}_t, \mathbf{u}_t) = \sum_{t=1}^{T} (x_t - x_{goal})^T Q (x_t - x_{goal}) + u_t^T R u_t \tag{5}$$

where $L(\mathbf{x}_t, \mathbf{u}_t)$ is running loss and $\phi(\mathbf{x}_T)$ is terminal loss. At the beginning we have to set T to determine the length of the trajectory, which will affect the calculation time and the sum of loss. The algorithm will generate state trajectory and control trajectory. This expresses the planning of the future path in the current state.

C. MPPI

MPPI is a sampling-based approach to solving the stochastic alternative to Eg:5, In the stochastic optimal control problem, the dynamics are stochastic and the goal is to minimize the expected cost. This problem formulation has deep connections with path integral control, Bayesian inference, and statistical mechanics. Since this work focuses on highly nonlinear but deterministic dynamics, stochastic method is introduced in the dynamics through the controls

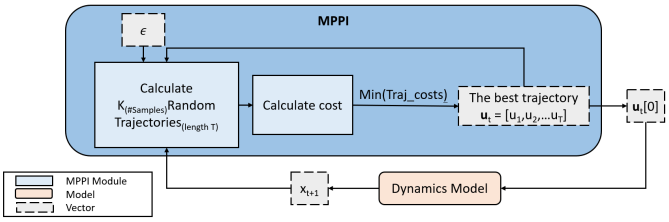

Fig. 4. MPPI framework

According to the paper[1], we implemented MPPI, the method pseudo code is as follows:

Algorithm 2: Model Predictive Path Integral Control - Optimization Loop

Input: Nominal control trajectory $\mathbf{u}_t$
while *not converged* **do**
  `// Parallel block`
  for $t = 1, \ldots, T-1$ **do**
    Sample $\mathbf{v}_t^{(k)} \sim \mathcal{N}(\mathbf{u}_t, \Sigma)$
    $\mathbf{x}_{t+1}^{(k)} \leftarrow \mathbf{f}(\mathbf{x}_t, \mathbf{v}_t^{(k)})$
  Compute costs $\mathcal{J}^{(k)} \leftarrow \mathcal{J}(\mathbf{V}^{(k)})$
  `// End parallel block`
  $\rho \leftarrow \min_k \mathcal{J}^{(k)}$
  $\eta \leftarrow \sum_{k=1}^{K} \exp(-\frac{1}{\lambda}(\mathcal{J}^{(k)} - \rho))$
  Compute weights $w^{(k)} \leftarrow \frac{1}{\eta} \exp(-\frac{1}{\lambda}(\mathcal{J}^{(k)} - \rho))$
  Update controls $\mathbf{u}_t \leftarrow \sum_{k=1}^{K} w^{(k)} \mathbf{v}_t^{(k)}$
return *optimal controls* $\mathbf{u}_t$

Fig. 5. MPPI pseudo code

The process of MPPI involves the following steps:

- Trajectory creating: According to the current state vector, we create trajectory and in this part, we add $\mathbf{v}_t$ to make our u randomly. $vN(u_t, \sum)$, with $\sum \in R^{mxm}$denotes the covariance of the control noise, at this time. At the beginning, we will set the number of samples to determine how many control trajectories to generate and the T to decide the length of the trajectory.
- Cost calculation: According to the control trajectory of the previous step, input $\mathbf{u}_t$ Then we calculate the cost of each state and the target state, and finally add the cost to get the cost of each trajectory.
- Select the optimal trajectory: After obtaining the lowest cost, calculate ρ and η, and calculate the weight for each trajectory, and finally get the final optimal $\mathbf{u}_t$ through weighted average.
- iteration: Repeat the above steps until the loss value approaches zero to form a dynamic balance.

The stochastic alternative method can make the best solution jump out of the local solution to the global solution, but it may also take too much time to calculate too many samples.

D. DDP

DDP utilizes linear or quadratic approximations of the dynamics and quadratic approximations of the cost function along a nominal trajectory and produces an update to the controls that results in a reduction of the cost. Repeating this process iteratively results in quadratic convergence to a locally-optimal nominal trajectory. DDP can effectively perform trajectory planning for nonlinear systems.

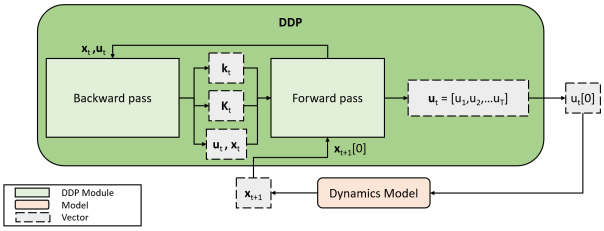

Fig. 6. DDP framework

According to the paper[1], we implemented DDP, the method pseudo code is as follows:

Algorithm 1: Differential Dynamic Programming

Input: Nominal state and control trajectory $\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t$
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 $t = T-1, \ldots, 1$ **do**
    Calculate $Q_t^x, Q_t^u, Q_t^{xx}, Q_t^{xu}, Q_t^{uu}$
    if Q_t^{uu} *not invertible* **then**
      Increase regularization parameters μ_1, μ_2
      Restart backward pass
    Calculate gains $\mathbf{k}_t, \mathbf{K}_t$
    Calculate V_t^0, V_t^x, V_t^{xx}
  // End backward pass
  Decrease regularization parameters μ_1, μ_2
  // Line search
  $\epsilon \leftarrow 1$
  while *cost decrease not sufficient* **do**
    // Forward pass
    $\mathbf{x}_1 \leftarrow \bar{\mathbf{x}}_1$
    for $t = 1, \ldots, T-1$ **do**
      $\delta\mathbf{x}_t \leftarrow \mathbf{x}_t - \bar{\mathbf{x}}_t$
      $\mathbf{u}_t \leftarrow \bar{\mathbf{u}}_t + \epsilon\mathbf{k}_t + \mathbf{K}_t\delta\mathbf{x}_t$
      $\mathbf{x}_{t+1} \leftarrow \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)$
    // End forward pass
    $\epsilon \leftarrow \rho\epsilon$ // Reduce ϵ
  // End line search
  $\bar{\mathbf{x}}_t \leftarrow \mathbf{x}_t$
  $\bar{\mathbf{u}}_t \leftarrow \mathbf{u}_t$
return *optimal controls* $\mathbf{u}_t$

Fig. 7. DDP pseudo code

The quadratic value function expansion at t:

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

The gain value function at t:

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

The value function is defined as:

$$V(x_i) = minJ((\mathbf{x}_i, \mathbf{u}_i)) \tag{6}$$

Over the entire control sequence to be rewritten as sequence of minimization over each u_t proceeding backwards in time:

$$V(x_i) = min\left[L(\mathbf{x}_i, \mathbf{u}_i) + V(x_{i+1})\right] \tag{7}$$

The process of DDP involves the following steps:

- Backward Pass: Calculate V and Q according to the x_t at the previous moment. If Q_t^{uu} is invertible, you need to add regularization parameters μ_1, μ_2, and you can get K_t and $k_t$through Q.
- Forward Pass: Utilize the $x_t[0]$ from dynamic model and the Kt and kt of the previous step to generate control trajectory $\mathbf{u}_t$. And use line search every time to ensure that the cost decreases every time.
- Iteration: Repeat the above steps until the loss value approaches zero to form a dynamic balance.

In summary, DDP is a second-order algorithm like Newton's method. It therefore takes large steps toward the minimum and often requires regularization and line-search to achieve convergence.

$$V(x_i) = min\left[L(\mathbf{x}_i, \mathbf{u}_i) + V(x_{i+1})\right] \tag{8}$$

III. Results

We conducted several tests on our DDP controller to evaluate its stability, convergence, and performance. We then compared it to the MPPI controller to identify the differences between them.

A. DDP convergence

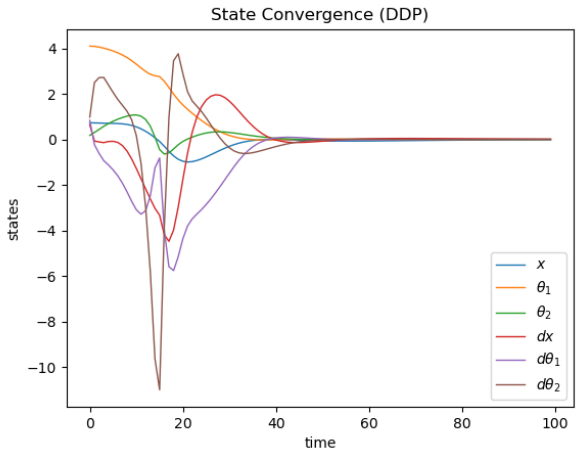

Fig. 8. State Convergence of DDP

We found that our controller converges pretty well. Even with a random initial position, our controller can guide the pendulum to the proper positions with a low pendulum angular speed so it can stay stable on the top (our goal position).

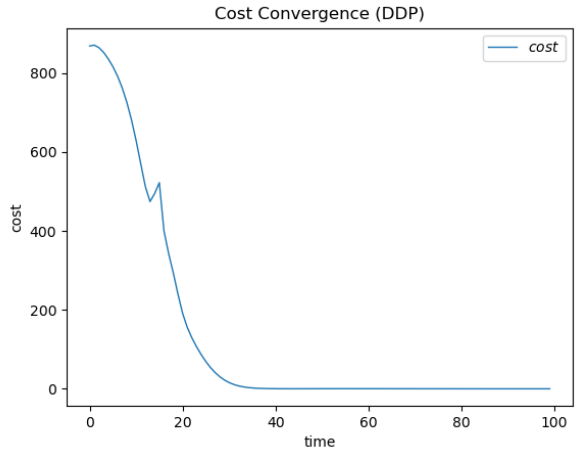

Fig. 9. Cost convergence of DDP

6Our cost function is a quadratic approximation that converges rapidly. Additionally, we defined a very good cost function that allows our system to continuously move to the desired pose.

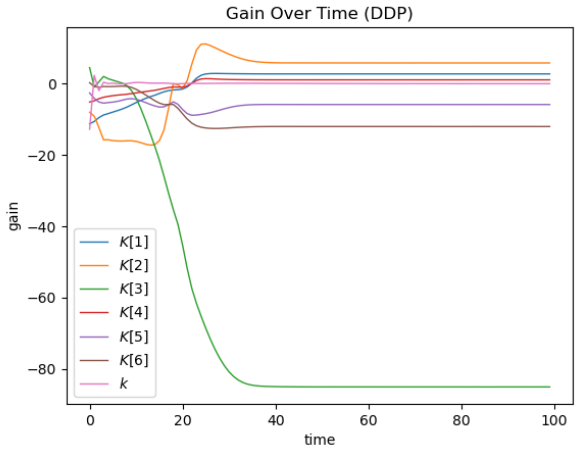

Fig. 10. Gain vs Time (DDP)

These gains are used to update the actions and will be multiplied by $(x_t - x_{goal})$ to update the actions. Since it is very close to our goal position at the end, these gains will not converge.

B. MPPI convergence

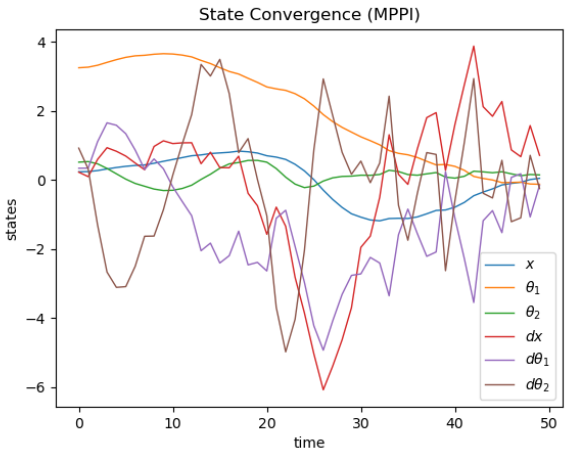

Fig. 11. State Convergence of MPPI

The positions converge very well, but the velocities do not. Therefore, our MPPI controller cannot maintain the pendulum system at its goal positions, and it is easy for it to fall down after reaching the goal.

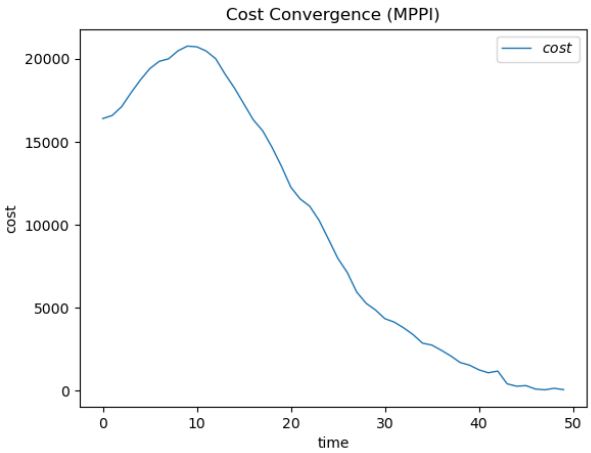

Fig. 12. Cost convergence of MPPI

IV. Discussion

First of all, we neglected the second derivative term of the dynamics function in our DDP controller. As a result, the performance of our controller is very similar to that of the iLQR controller, and it shares the same feature with iLQR.

To compute the second derivative of the dynamics function for an inverted double-pendulum on a cart system, which is often referred to as a chaotic system, it is easy to encounter numerical issues on certain ”special” points. When this happens, the entire system can break down. This is why we chose to neglect the second derivative term in order to avoid such issues

To ensure optimal performance of our DDP controller, we need to carefully tune the weight of the quadratic term (Q) in our cost function. If this weight is too small, the system may not have enough momentum to move the pendulum to the top. Conversely, if it is too large, the system is at risk of becoming unstable and breaking down.

DDP is a deterministic method, which means if you give the DDP controller the same initial conditions, it will result in the same final position. In comparison, MPPI uses the shooting method, which contains stochastic processes, so it will produce different results each time we run the controller. There is no guarantee that the system will always reach its goal pose with MPPI.

The regularization terms μ_1 and μ_2 are crucial. Adding μ_1 helps to prevent rapid changes in the trajectory, while μ_2 ensures that the Q_{uu} matrix is always invertible. Without these regularization terms, numerical issues can arise when running the controller.

After all, our fine-tuned DDP controller works very well and is very robust, even against random initial conditions. In fact, almost all reasonable initial conditions can lead to our goal position.

V. Conclusion

In conclusion, we have successfully designed and implemented a DDP controller for an inverted double-pendulum on a cart system. With neglecting the second derivative term in the dynamics function, our controller performed similarly to the iLQR controller. The weight of the quadratic term in the cost function and the regularization terms were carefully tuned to ensure optimal performance and stability. Our DDP controller demonstrated its robustness by being able to reach the goal position from almost all reasonable initial conditions. Overall, our DDP controller is a viable option for controlling chaotic systems such as the inverted double-pendulum on a cart.

References

[1] et al. Houghton, Matthew D. Path planning: Differential dynamic programming and model predictive path integral control on vtol aircraft. *AIAA SCITECH 2022 Forum*, 2022.