Georgia Institute of Technology

AE 4803-ROB

Robotic Systems & Autonomy

Homework Assignment 1

Simulated Control of an Inverted Pendulum and Cart-Pole System Using Differential Dynamic Programming (DDP)

Maitreya Venkataswamy

Zachary Chester

Marc Larvie

supervised by

Prof. Evangelos Theodorou

February 12, 2020

I Mathematical Formulation of DDP

I.A Discretization and Linearization of System Dynamics

The continuous dynamics of the systems being controlled are represented in the state space form of Eq. I.1.

$$\dot{\boldsymbol{x}} = F(\boldsymbol{x}, \boldsymbol{u}) \tag{I.1}$$

First, the dynamics are linearized about a nominal state trajectory $\bar{\boldsymbol{x}}$ and the control input $\bar{\boldsymbol{u}}$ in Eq. I.2, where $\delta\boldsymbol{x} \equiv \boldsymbol{x}_k - \bar{\boldsymbol{x}}_k$ and $\delta\boldsymbol{u} \equiv \boldsymbol{u}_k - \bar{\boldsymbol{u}}_k$.

$$\dot{\delta\boldsymbol{x}} = F_x\delta\boldsymbol{x} + F_u\delta\boldsymbol{u} \tag{I.2}$$

The time derivative of the state is discretized using an Euler discretization scheme in Eq. I.3, which is a first order accurate forward difference discretization scheme, where $\Delta t \equiv t_{k+1} - t_k$.

$$\dot{\delta\boldsymbol{x}} \approx \frac{\delta\boldsymbol{x}_{k+1} - \delta\boldsymbol{x}_k}{\Delta t} + \mathcal{O}(\Delta t^2) \tag{I.3}$$

Combining Eqs. I.2 and I.3 and rearranging the result yields the discretized form of the linearized system dynamics in Eq. I.4.

$$\boxed{\delta\boldsymbol{x}_{k+1} = \delta\boldsymbol{x}_k + (F_x\delta\boldsymbol{x} + F_u\delta\boldsymbol{u})\Delta t} \tag{I.4}$$

In Sec. I.B, this equation will be utilized in a mathematical substitution, so it is desired to simplify it to the form of Eq. I.5.

$$\delta\boldsymbol{x}_{k+1} = \Phi_k\delta\boldsymbol{x}_k + \beta_k\delta\boldsymbol{u}_k \tag{I.5}$$

where $\Phi_k \equiv I + F_x\Delta t$ and $\beta_k \equiv F_u\Delta t$. An important note is that Eq. I.5 is used in the math of the derivation of the DDP equations, but to actually propagate the trajectory from the initial state given a control sequence, the scheme in Eq. I.6 is used.

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + F(\boldsymbol{x}_k, \boldsymbol{u}_k)\Delta t \tag{I.6}$$

I.B Second Order Expansion of the State-Action Value Function

Eq. I.7 is the Bellman principle, discretized in time.

$$V(\boldsymbol{x}_k, t_k) = \min_{\boldsymbol{u}_k} Q(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) = \min_{\boldsymbol{u}_k}\Big[\mathcal{L}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) + V(\boldsymbol{x}_{k+1}, t_{k+1})\Big] \tag{I.7}$$

where $\mathcal{L}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) = \ell(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k)\Delta t$ is the running cost, and $V(\boldsymbol{x}, \boldsymbol{u}, t)$ is the value function. The state-action value function Q is expanded using a Taylor series expansion about a nominal state trajectory $\bar{\boldsymbol{x}}$ and the control input $\bar{\boldsymbol{u}}$ in Eq. I.8, up to and including the second order terms, where $\delta\boldsymbol{x} \equiv \boldsymbol{x}_k - \bar{\boldsymbol{x}}_k$ and $\delta\boldsymbol{u} \equiv \boldsymbol{u}_k - \bar{\boldsymbol{u}}_k$.

$$\boxed{Q(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) = Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, t_k) + \begin{bmatrix} Q_x^{\mathrm{T}} & Q_u^{\mathrm{T}} \end{bmatrix}\begin{bmatrix}\delta\boldsymbol{x} \\ \delta\boldsymbol{u}\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\delta\boldsymbol{x} \\ \delta\boldsymbol{u}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{x} \\ \delta\boldsymbol{u}\end{bmatrix}} \tag{I.8}$$

The running cost $\mathcal{L}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k)$ and the value function at the next state $V(\boldsymbol{x}_{k+1}, t_{k+1})$ are expanded about a nominal trajectory $\bar{\boldsymbol{x}}$ and input sequence $\bar{\boldsymbol{u}}$ using Taylor series expansions up to the second order, and written in expanded form.

$$\mathcal{L}(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) = \mathcal{L}(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, t_k) + \mathcal{L}_x^{\mathrm{T}}\delta\boldsymbol{x} + \mathcal{L}_u^{\mathrm{T}}\delta\boldsymbol{u} + \frac{1}{2}\Big(\delta\boldsymbol{x}^{\mathrm{T}}\mathcal{L}_{xx}\delta\boldsymbol{x} + \delta\boldsymbol{x}^{\mathrm{T}}\mathcal{L}_{xu}\delta\boldsymbol{u} + \delta\boldsymbol{u}^{\mathrm{T}}\mathcal{L}_{ux}\delta\boldsymbol{x} + \delta\boldsymbol{u}^{\mathrm{T}}\mathcal{L}_{uu}\delta\boldsymbol{u}\Big)$$

$$V(\boldsymbol{x}_{k+1}, t_{k+1}) = V(\bar{\boldsymbol{x}}_{k+1}, t_{k+1}) + V_x(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})^{\mathrm{T}}\delta\boldsymbol{x}_{k+1} + \frac{1}{2}\delta\boldsymbol{x}_{k+1}^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\delta\boldsymbol{x}_{k+1}$$

Substituting Eq. I.5 into the expansion of the value function yields

$$V(\boldsymbol{x}_{k+1}, t_{k+1}) = V(\bar{\boldsymbol{x}}_{k+1}, t_{k+1}) + V_x(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})^{\mathrm{T}}\Big(\Phi_k\delta\boldsymbol{x}_k + \beta_k\delta\boldsymbol{u}_k\Big) + \frac{1}{2}\Big(\Phi_k\delta\boldsymbol{x}_k + \beta_k\delta\boldsymbol{u}_k\Big)^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\Big(\Phi_k\delta\boldsymbol{x}_k + \beta_k\delta\boldsymbol{u}_k\Big)$$

which is expanded and simplified to obtain

$$\begin{aligned} V(\boldsymbol{x}_{k+1}, t_{k+1}) = {} & V(\bar{\boldsymbol{x}}_{k+1}, t_{k+1}) + V_x(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})^{\mathrm{T}}\Phi_k\delta\boldsymbol{x}_k + V_x(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})^{\mathrm{T}}\beta_k\delta\boldsymbol{u}_k \\ & + \frac{1}{2}\delta\boldsymbol{x}_k^{\mathrm{T}}\Phi_k^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\Phi_k\delta\boldsymbol{x}_k + \frac{1}{2}\delta\boldsymbol{u}_k^{\mathrm{T}}\beta_k^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\beta_k\delta\boldsymbol{u}_k \\ & + \frac{1}{2}\delta\boldsymbol{x}_k^{\mathrm{T}}\Phi_k^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\beta_k\delta\boldsymbol{u}_k + \frac{1}{2}\delta\boldsymbol{u}_k^{\mathrm{T}}\beta_k^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\Phi_k\delta\boldsymbol{x}_k \end{aligned}$$

We expand Eq. I.8 and group the $\delta\boldsymbol{x}$ and $\delta\boldsymbol{u}$ terms to get Eq. I.9.

$$\begin{aligned} Q(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) = {} & Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, t_k) + Q_x^{\mathrm{T}}\delta\boldsymbol{x} + Q_u^{\mathrm{T}}\delta\boldsymbol{u} \\ & + \frac{1}{2}\Big(\delta\boldsymbol{x}^{\mathrm{T}}Q_{xx}\delta\boldsymbol{x} + \delta\boldsymbol{x}^{\mathrm{T}}Q_{xu}\delta\boldsymbol{u} + \delta\boldsymbol{u}^{\mathrm{T}}Q_{ux}\delta\boldsymbol{x} + \delta\boldsymbol{u}^{\mathrm{T}}Q_{uu}\delta\boldsymbol{u}\Big) \end{aligned} \tag{I.9}$$

Knowing that the state-action value function is defined as the sum of the running cost and the value function of the next state, the gradient and Jacobian terms in the expansion of the state action value function can be written in the terms of the expansions of the running cost and value function at the next state using Eq. I.10 through Eq. I.15.

$$\boxed{\begin{aligned} Q_x &= \mathcal{L}_x + \Phi_k^{\mathrm{T}}V_x(\bar{\boldsymbol{x}}_{k+1}, t_{k+1}) && \text{(I.10)} \\ Q_u &= \mathcal{L}_u + \beta_k^{\mathrm{T}}V_x(\bar{\boldsymbol{x}}_{k+1}, t_{k+1}) && \text{(I.11)} \\ Q_{xx} &= \mathcal{L}_{xx} + \Phi_k^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\Phi_k && \text{(I.12)} \\ Q_{uu} &= \mathcal{L}_{uu} + \beta_k^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\beta_k && \text{(I.13)} \\ Q_{xu} &= \mathcal{L}_{xu} + \Phi_k^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\beta_k && \text{(I.14)} \\ Q_{ux} &= \mathcal{L}_{ux} + \beta_k^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_{k+1}, t_{k+1})\Phi_k && \text{(I.15)} \end{aligned}}$$

I.C Derivation of Optimal Control Update

The objective of the trajectory optimization in DDP is to find a $\delta\boldsymbol{u}^*$ such that

$$\delta\boldsymbol{u}^* = \arg\min_{\delta\boldsymbol{u}} Q(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k)$$

which, since the state-action value function is in a quadratic form, can be found by equating the gradient of the state-action value function with respect to $\delta\boldsymbol{u}$, evaluated at the optimal control correction $\delta\boldsymbol{u}^*$, to the zero vector in Eq. I.16.

$$\nabla_{\delta\boldsymbol{u}}Q\Big|_{\delta\boldsymbol{u}=\delta\boldsymbol{u}^*} = Q_u^{\mathrm{T}} + \delta\boldsymbol{x}^{\mathrm{T}}Q_{xu}^{\mathrm{T}} + (\delta\boldsymbol{u}^*)^{\mathrm{T}}Q_{uu} = 0 \tag{I.16}$$

Rearranging Eq. I.16 while recognizing that Q_{uu} is a symmetric matrix and $Q_{ux}^{\mathrm{T}} = Q_{xu}$ yields the optimal control correction in Eq. I.17.

$$\boxed{\delta\boldsymbol{u}^* = -Q_{uu}^{-1}(Q_u + Q_{ux}\delta\boldsymbol{x})} \tag{I.17}$$

The control correction, or update, is composed of a feed-forward term $-Q_{uu}^{-1}Q_u$ and a feed-back term $-Q_{uu}^{-1}Q_{ux}\delta\boldsymbol{x}$.

I.D Derivation of the Backward-Pass Equations

Substituting Eq. I.17 into Eq. I.9 yields

$$\begin{aligned}Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) = Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, t_k) &+ Q_x^{\mathrm{T}}\delta\boldsymbol{x} + Q_u^{\mathrm{T}}\Big[-Q_{uu}^{-1}(Q_u + Q_{ux}\delta\boldsymbol{x})\Big] \\ &+ \frac{1}{2}\Big(\delta\boldsymbol{x}^{\mathrm{T}}Q_{xx}\delta\boldsymbol{x} + \delta\boldsymbol{x}^{\mathrm{T}}Q_{xu}\Big[-Q_{uu}^{-1}(Q_u + Q_{ux}\delta\boldsymbol{x})\Big] + \Big[-Q_{uu}^{-1}(Q_u + Q_{ux}\delta\boldsymbol{x})\Big]^{\mathrm{T}}Q_{ux}\delta\boldsymbol{x} \\ &+ \Big[-Q_{uu}^{-1}(Q_u + Q_{ux}\delta\boldsymbol{x})\Big]^{\mathrm{T}}Q_{uu}\Big[-Q_{uu}^{-1}(Q_u + Q_{ux}\delta\boldsymbol{x})\Big]\Big)\end{aligned}$$

which is expanded to

$$\begin{aligned}Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) = Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, t_k) &+ Q_x^{\mathrm{T}}\delta\boldsymbol{x} - Q_u^{\mathrm{T}}Q_{uu}^{-1}Q_u - Q_u^{\mathrm{T}}Q_{uu}^{-1}Q_{ux}\delta\boldsymbol{x} \\ &+ \frac{1}{2}\Big(\delta\boldsymbol{x}^{\mathrm{T}}Q_{xx}\delta\boldsymbol{x} - \delta\boldsymbol{x}^{\mathrm{T}}Q_{xu}Q_{uu}^{-1}Q_u - \delta\boldsymbol{x}^{\mathrm{T}}Q_{xu}Q_{uu}^{-1}Q_{ux}\delta\boldsymbol{x} \\ &- Q_u^{\mathrm{T}}Q_{uu}^{-1}Q_{ux}\delta\boldsymbol{x} - \delta\boldsymbol{x}^{\mathrm{T}}Q_{ux}^{\mathrm{T}}Q_{uu}^{-1}Q_{ux}\delta\boldsymbol{x} \\ &+ Q_u^{\mathrm{T}}Q_{uu}^{-1}Q_{uu}Q_{uu}^{-1}Q_u + \delta\boldsymbol{x}^{\mathrm{T}}Q_{ux}^{\mathrm{T}}Q_{uu}^{-1}Q_{uu}Q_{uu}^{-1}Q_{ux}\delta\boldsymbol{x} \\ &+ Q_u^{\mathrm{T}}Q_{uu}^{-1}Q_{uu}Q_{uu}^{-1}Q_{ux}\delta\boldsymbol{x} + \delta\boldsymbol{x}^{\mathrm{T}}Q_{ux}^{\mathrm{T}}Q_{uu}^{-1}Q_{uu}Q_{uu}^{-1}Q_u\Big)\end{aligned}$$

and then simplified to the minimum value of the state-action value function in Eq. I.18.

$$Q^*(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) = Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, t_k) + \Big[Q_x^{\mathrm{T}} - Q_u^{\mathrm{T}}Q_{uu}^{-1}Q_{ux}\Big]\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{x}^{\mathrm{T}}\Big[Q_{xx} - Q_{xu}Q_{uu}^{-1}Q_{ux}\Big]\delta\boldsymbol{x} \tag{I.18}$$

The discretized Bellman principle in Eq. I.7 states that the value function is equivalent to Eq. I.18, which means that expanding the value function using a Taylor series up to second order about a nominal state trajectory yields Eq. I.19

$$\begin{aligned}&V(\bar{\boldsymbol{x}}_k, t_k) + V_x(\bar{\boldsymbol{x}}_k, t_k)^{\mathrm{T}}\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{x}^{\mathrm{T}}V_{xx}(\bar{\boldsymbol{x}}_k, t_k)\delta\boldsymbol{x} \\ &\qquad = Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, t_k) + \Big[Q_x^{\mathrm{T}} - Q_u^{\mathrm{T}}Q_{uu}^{-1}Q_{ux}\Big]\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{x}^{\mathrm{T}}\Big[Q_{xx} - Q_{xu}Q_{uu}^{-1}Q_{ux}\Big]\delta\boldsymbol{x}\end{aligned} \tag{I.19}$$

Relating the terms on both sides of Eq. I.19 yields the backward pass equations.

$$\boxed{\begin{aligned}V(\bar{\boldsymbol{x}}_k, t_k) &= Q(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, t_k) && \text{(I.20)}\\ V_x(\bar{\boldsymbol{x}}_k, t_k) &= Q_x - Q_{xu}Q_{uu}^{-1}Q_u && \text{(I.21)}\\ V_{xx}(\bar{\boldsymbol{x}}_k, t_k) &= Q_{xx} - Q_{xu}Q_{uu}^{-1}Q_{ux} && \text{(I.22)}\end{aligned}}$$

II Implementation of DDP in Simulation

II.A General Implementation Details

II.A.1 Cost Functions

For both the inverted pendulum and cart-pole systems, we utilize quadratic cost functions for the terminal cost and the transitions cost of the form in Eq. II.1, where Q_f and R are tuned for the specific control problem to yield an acceptable control sequence and trajectory from the DDP algorithm.

$$\begin{aligned}\phi(\boldsymbol{x}_f, t_f) &= \frac{1}{2}(\boldsymbol{x}_f - \boldsymbol{x}_*)^{\mathrm{T}} Q_f \frac{1}{2}(\boldsymbol{x}_f - \boldsymbol{x}_*) \\ \ell(\boldsymbol{x}_k, \boldsymbol{u}_k, t_k) &= \frac{1}{2}\boldsymbol{u}_k^{\mathrm{T}} R \boldsymbol{u}_k\end{aligned} \tag{II.1}$$

II.A.2 Discretization of Time

Since the discretization of the dynamics in time was done using an Euler discretization scheme, a sufficiently small time step Δt needs to be selected in order to minimize the artificial introduction/removal of energy to/from the simulated systems due to truncation errors. For both systems, the time horizon was no more than $t_f = 5\,\mathrm{s}$, so a time step of $\Delta t = 10\,\mathrm{ms}$ was considered sufficient. This was determined by simulating the free response of the system with a non-equilibrium initial condition with no control input while monitoring the energy of the system during the evolution of the free response. The time step was decreased until the accumulation/reduction of energy from the systems was considered minimal.

II.A.3 Control Update Learning Rate

A line searching parameter was used to reduce the learning rate in the DDP algorithm by decreasing the magnitude of the control update at each DDP iteration by a constant factor α. The control update equation took to the form of Eq. II.2.

$$\boldsymbol{u}_k \leftarrow \boldsymbol{u}_k + \alpha \cdot \delta \boldsymbol{u}_k, \;\; \alpha \in (0, 1] \tag{II.2}$$

This allowed the algorithm to steadily converge to the locally optimal solution instead of diverging due to a sudden increase in the control update. The learning rate α was generally kept around 0.5 for the inverted pendulum simulations, and 0.1 for the cart-pole simulations.

II.A.4 Control Energy

For each system, we define the "control energy" as a means to examine the amount of control being used during the trajectory, using Eq. II.3

$$E = \frac{1}{2}\sum_{k=1}^{f-1} \boldsymbol{u}^T M \boldsymbol{u} \tag{II.3}$$

Matrix M is a diagonal matrix to specify the relative value of control between the different inputs, and to resolve differences in units of the controls. For a multiple input system, this could be the cost matrix R, but for a single input system, this is set to unity. Although none are shown in this study, this function would be used in comparisons between two locally optimal control sequences that may be been determined using different cost functions, for the purpose of determining which policy used the lesser amount of control. In this study, this is used to monitor the total amount of control being used by a control policy during the refinement of this policy in the DDP algorithm.

II.B Implementation for an Inverted Pendulum

An inverted pendulum was simulated with mass $m = 1\,\text{kg}$, length $l = 1\,\text{m}$, and damping coefficient $b = 1\,\text{N}\,\text{s}\,\text{rad}^{-1}$. The time horizon was set as $t_f = 2\,\text{s}$, and the cost matrices were set as

$$Q_f = \begin{bmatrix} 100\,\text{rad}^{-2} & 0 \\ 0 & 100\,\text{s}^2\,\text{rad}^{-2} \end{bmatrix}, \quad R = 1 \times 10^{-4}\,\text{N}^{-2}\,\text{m}^{-2}\,\text{s}^{-1}$$

The initial state is the stationary pendulum-down configuration, and the target state was the stationary pendulum-up configuration. Fig. 1 shows the solution found by the DDP algorithm for the righting of the pendulum. We can see that the pendulum was successfully brought from the initial state to the target state, where the pendulum was brought to the upwards configuration by swinging back and forth twice with increasing amplitude until it was brought completely upwards. This back and forth movement of the pendulum is represented in the oscillating shape of the pendulum angle, as well as the spiral shape in the state-space representation of the trajectory.

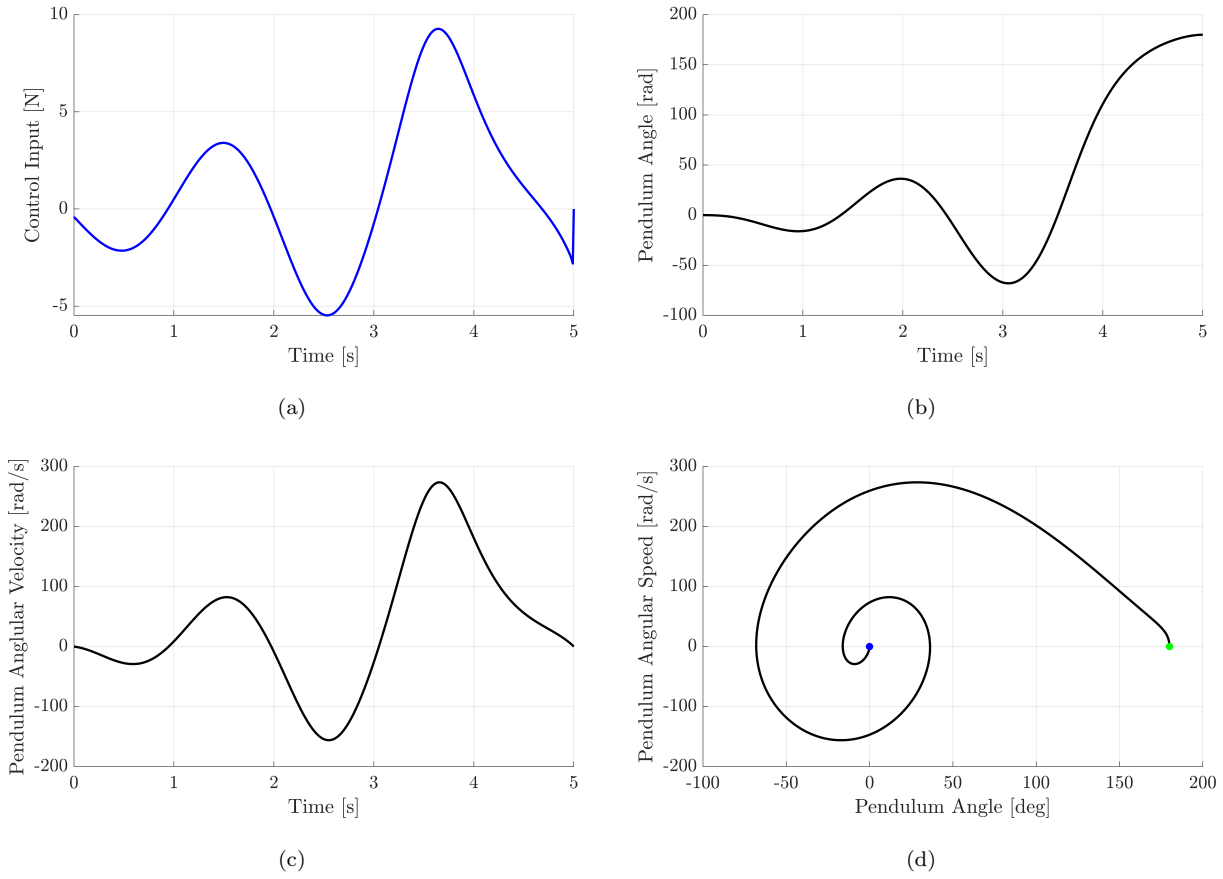
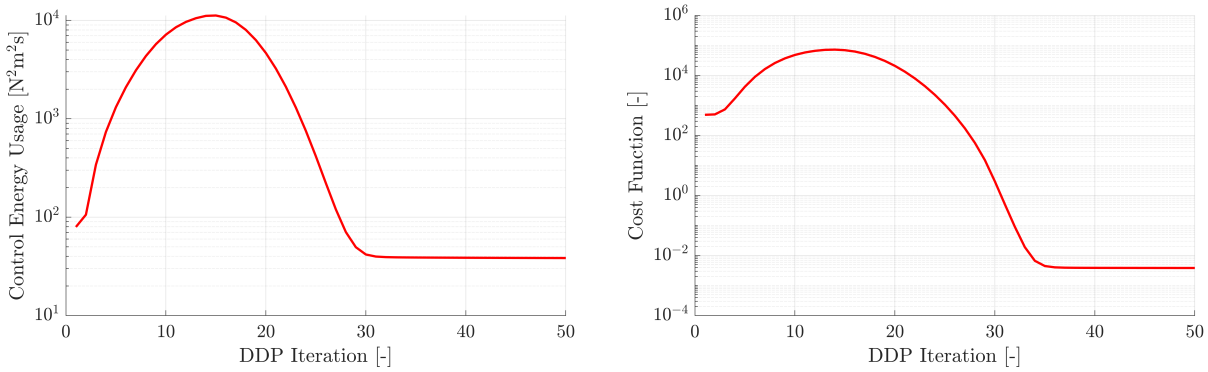


Figure 1: Locally optimal control sequence (a), pendulum angle trajectory (b), pendulum angular velocity trajectory (c), and trajectory in state-space (d) found using the DDP algorithm for the inverted pendulum system.

Fig. 2 shows the history of the control energy and the cost function of the control sequence during the DDP iterations. We see both in the control energy history and the cost function, that 50 iterations was sufficient to converge to the locally optimal solution. An interesting phenomenon in both plots is the initial increase in the cost function and control energy at the beginning of the algorithm's iterations. This is due to the way we have tuned the cost functions; the magnitude of R is much smaller than elements in Q_f, which means the DDP algorithm will "prioritize" decreasing the terminal state cost over decreasing the control cost. So

initially in the first 10 or so iterations, the algorithm is increasing the amount of control being used in order to drive the final state towards the terminal state, resulting in the increase in the control energy. Around 15 iterations, the algorithm has most likely produced a control sequence that is not optimal, but does results in the target state being reached. Then, during the remaining iterations, the algorithm reduces the amount of control in the control sequence in order to reduce the control energy and the cost function, until it cannot reduce them any farther.



(a) (b)

Figure 2: Control energy (a) and cost function (b) histories during the DDP algorithm iterations for the inverted pendulum system.

To study the robustness of the control policy to disturbances, we simulate the systems with Gaussian noise present in the control input, to mimic the effects of an additional disturbance torque acting on the system alongside the control torque. The discretized dynamics used to propagate the trajectory after the locally optimal control sequence $\boldsymbol{u}$ has been determined are given in Eq. II.4, which was modified from Eq. I.6.

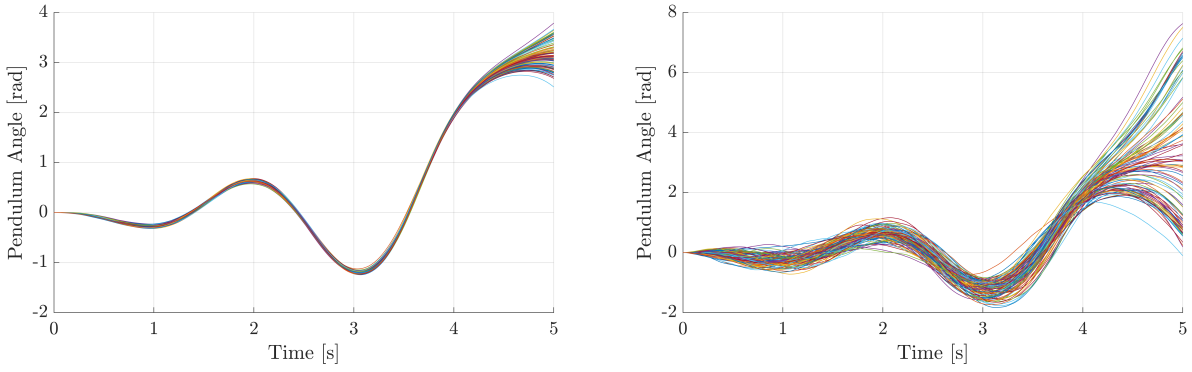
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + F(\boldsymbol{x}_k, \boldsymbol{u}_k + f)\Delta t, \quad f(x) \sim \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \quad [\mathrm{N\,m}] \tag{II.4}$$

The noise f is sampled from a Gaussian distribution with variance σ^2. Fig. 3 shows 100 simulated trajectories of the pendulum angle with the disturbance force, at two different values of σ. We see that for low variances in the noise, the control policy is able to maintain proximity to the designed trajectory, but as the variance is increases, the trajectory begins to deviate, particularly at the final state. By only using the determined locally optimal control policy as is, the system is not able to reach to target state in almost all of the trials.

In order to increase the robustness of the control policy to disturbances, the feedback policy that was used in the last DDP iteration is used to adjust the control sequence at each step during the control process. This happens by updating the control through Eq. II.5, where $\widehat{\boldsymbol{x}}_k$ and $\widehat{\boldsymbol{u}}_k$ are the kth state and control input in the locally optimal solution that was determined through by the DDP algorithm.

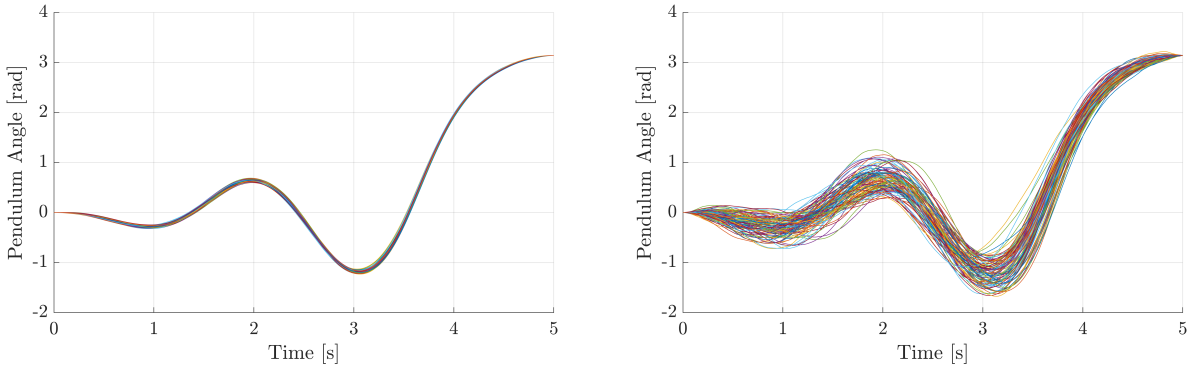
$$\boldsymbol{u}_k = \widehat{\boldsymbol{u}}_k - Q_{uu,k}^{-1} Q_{ux,k}(\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k) \tag{II.5}$$

Again, we simulate 100 trajectories of the system, but in which we implement the feedback control update in Eq. II.5 at each time-step in the simulation. Fig. 4 shows the history of the pendulum angles for these simulations for two levels of noise. We see that the feedback policy is able to effectively account for the disturbance, and it successfully brings the system to the target state in all the simulations.



(a) (b)

Figure 3: 100 simulated trajectories of the inverted pendulum system with Gaussian noise in the control with $\sigma = 1\,\mathrm{N\,m}$ (a) and $\sigma = 10\,\mathrm{N\,m}$.



(a) (b)

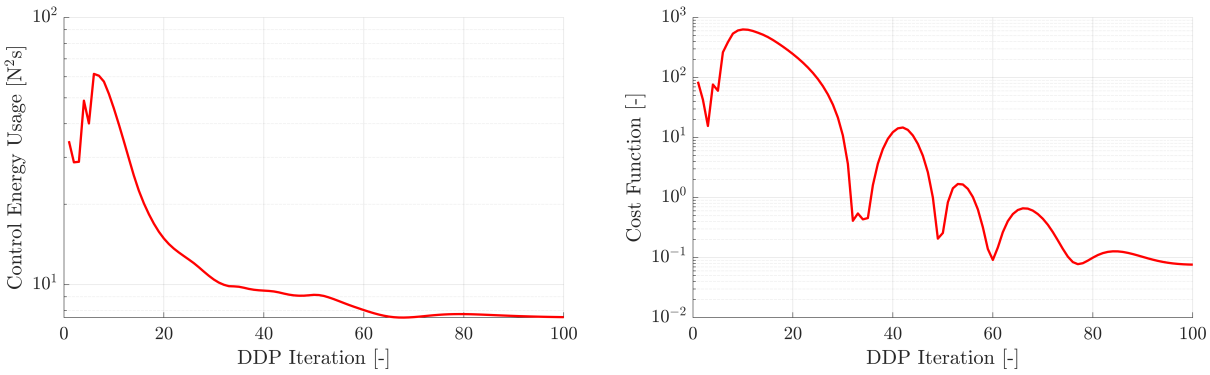
Figure 4: 100 simulated trajectories of the inverted pendulum system with Gaussian noise in the control with $\sigma = 1\,\mathrm{N\,m}$ (a) and $\sigma = 10\,\mathrm{N\,m}$ with the feedback policy to correct for the disturbances.

II.C Implementation for a Cart-Pole

A cart-pole system was simulated with a cart mass $m_c = 1\,\mathrm{kg}$, pole mass $m_p = 0.01\,\mathrm{kg}$, and pole length $l = 0.25\,\mathrm{m}$. The time horizon was set as $t_f = 5\,\mathrm{s}$, and the cost matrices were set as

$$Q_f = \begin{bmatrix} 10\,\mathrm{m}^{-2} & 0 & 0 & 0 \\ 0 & 10\,\mathrm{s}^2\,\mathrm{m}^{-2} & 0 & 0 \\ 0 & 0 & 10\,\mathrm{rad}^{-2} & 0 \\ 0 & 0 & 0 & 10\,\mathrm{s}^2\,\mathrm{rad}^{-2} \end{bmatrix}, \quad R = 0.01\,\mathrm{N}^{-2}\,\mathrm{s}^{-1}$$

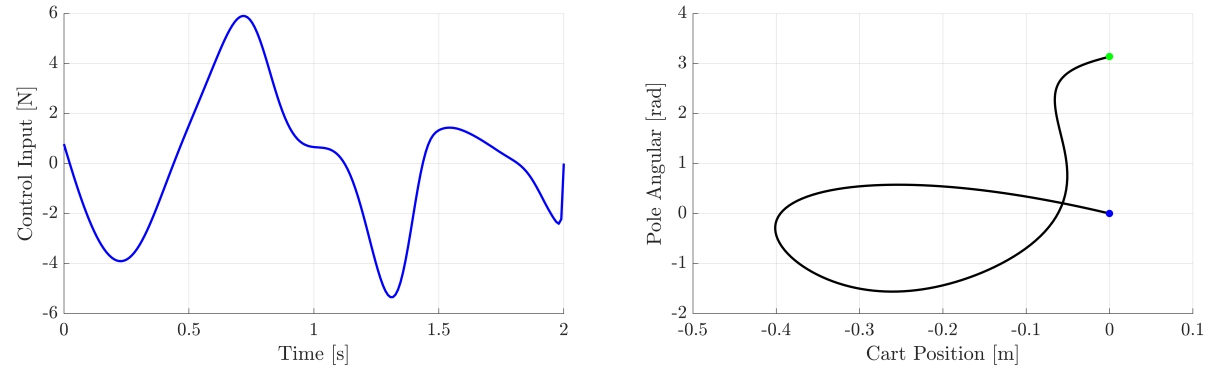
The initial state is the stationary pole-down configuration with the cart stationary, and the target state was the stationary pole-up configuration with the cart at the same position. Fig. 5 shows the history of the control energy and the cost function of the control sequence during the DDP algorithm. Similar to the inverted pendulum system, the cost and the control energy exhibit an initial increase followed by a decrease down to the solution. However, for this system, the history of the cost function during the DDP iterations includes multiple sudden increases in the cost function that would threaten the convergence if the learning parameter was not properly decreased to around 0.1. It was found that if the learning parameter was reduced an order of magnitude below this, the history of the cost function was smoother, but the convergence process took much longer than the 100 iterations that were used in the simulations shown in the figures.



(a) (b)

Figure 5: Control energy (a) and cost function (b) histories during the DDP algorithm iterations for the cart-pole system.

Fig. 6 and Fig. 7 show the solution found by the DDP algorithm to for the righting of the pole. We can see that the system was successfully brought from the initial state to the target state, where the pole was brought to the upwards configuration by the cart's back and forth motion. The cart was able to swing the pole up by moving the cart away from the initial position, and then quickly changing direction to bring it back. This caused the pole to enter a large swing, which the cart used to bring the pole all the way up by suddenly braking as the cart returned to the initial position. Both the cart velocity and the pole angular velocity were brought to zero at the target state, which means that the system was indeed brought to rest at the target state.



(a) (b)

Figure 6: Locally optimal control sequence (a) and trajectory in state-space (b) found using the DDP algorithm for the cart-pole system.

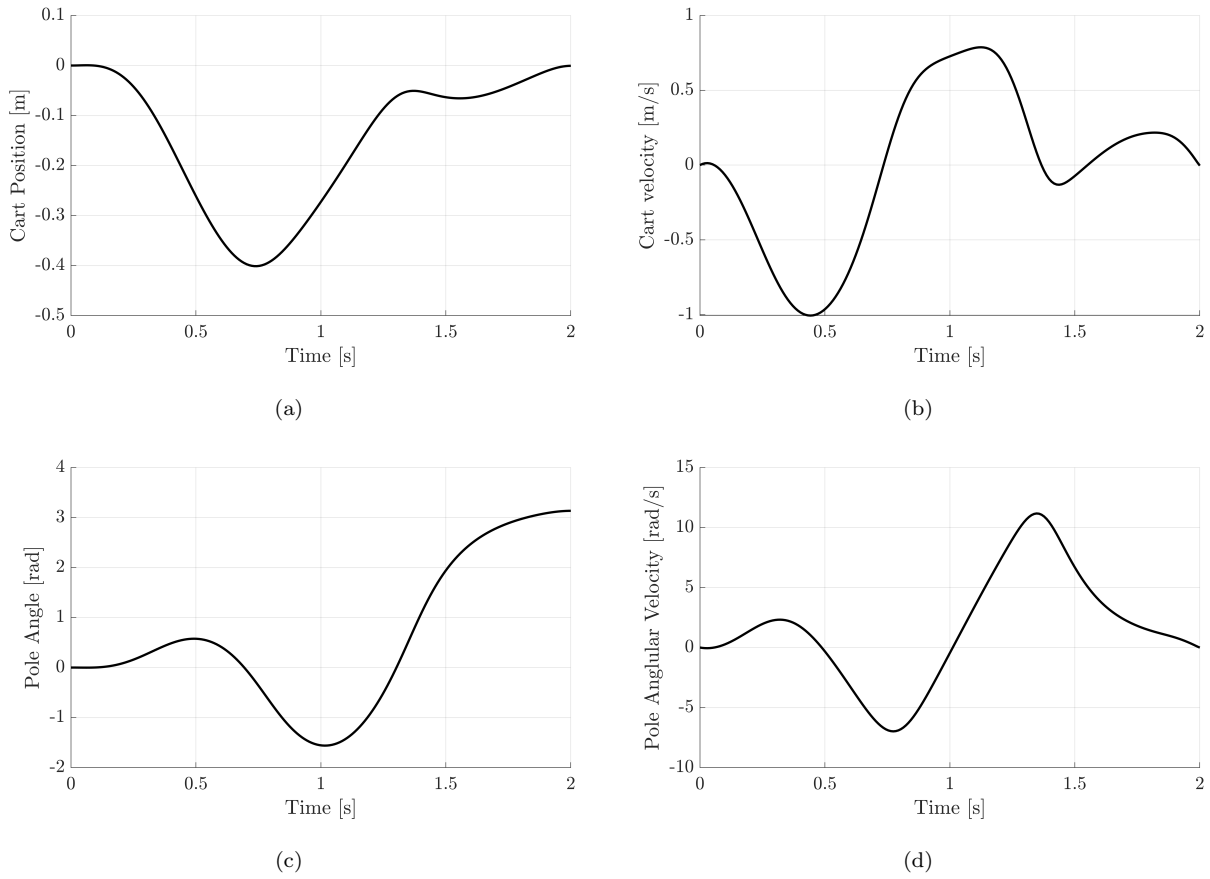


Figure 7: Locally optimal cart position (a), cart velocity (b), pole angle (c), and pole angular velocity (d) trajectories found using the DDP algorithm for the cart-pole system.