Differential Dynamic Programming

Kamble, Shubham & Duman, Ahmet Hakan*

Abstract– In this report, we describe differential dynamic programming (DDP), an optimal control algorithm of the trajectory optimization class, for determining the optimal control sequence of nonlinear, dynamical systems. First, we shall consider dynamic systems described by nonlinear differential equations. DDP lies on the boundary of open loop optimization and dynamic programming. It finds the optimal control sequence by alternately computing second-order approximate solutions to the Bellman equation around a nominal trajectory (backward pass) and computing new nominal trajectories (forward pass) until convergence. Here, we derive the algorithm of DDP using the Bellman optimality principle. Next, we prove its quadratic convergence. Further, we explain how line search and regularization techniques can help to improve the convergence properties of DDP. Lastly, the results of the DDP algorithm implemented on the ubiquitous cartpole example are presented.

I. Introduction

Optimal control theory is a branch of mathematical optimization that deals with finding a control for a dynamical system over a period of time such that a cost function is optimized. Optimal control problems are generally nonlinear and therefore do not have analytic solutions (e.g., linear-quadratic optimal control problems). As a result, it is necessary to employ numerical methods to solve optimal control problems. As optimal control solutions are now often implemented digitally, contemporary control theory is now primarily concerned with discrete-time systems and solutions.

Most existing numerical methods fall into one of two global methods categories – based on the Hamilton-Jacobi-Bellman (HJB) equations and the idea of dynamic programming [1]. However, due to the *curse of dimensionality*, such methods involve discretizations of the state and control spaces, making them inapplicable to high-dimensional problems. Local methods based on the *Maximum Principle* avoid the curse of dimensionality by solving a set of ODEs via shooting, relaxation, collocation, or gradient descent, to name a few. But, the resulting locally-optimal control laws are open-loop, thus losing the ability to consider modeling and measurement noise.

Differential Dynamic Programming(DDP) [2] provides an ideal blend of the advantages of local and global methods. This method is still local, such that it maintains a representation of a single trajectory and improves it locally. The iterative improvement, however, does not rely on solving ODEs but is based on dynamic programming applied by perturbing locally around the current trajectory. The algorithm was introduced in 1966 by D. Mayne [3]. It is closely related to Pantoja's step-wise Newton's method [4]. DDP is known to have second-order convergence and numerically appears to be more efficient than (efficient implementations of) Newton's method [5]. In exceptional cases where the system's dynamics are linear and the *running costs* are quadratic; it yields the optimal control sequence (i.e., it converges) in a single iteration step [3]. In section II, we derive the DDP algorithm for discrete-time nonlinear problems. Sections III and IV explain the conditions under which convergence of DDP is guaranteed and how DDP can be improved for general nonlinear dynamical systems using line-search and regularization techniques, respectively. Section V discusses the results of the implementation of DDP on the cartpole benchmark. Lastly, section VI concludes with the future scope and directions of research.

II. Differential Dynamic Programming

Considering a nonlinear dynamical system described by the following difference equation

$$\mathrm{x}_{i+1} = f(\mathrm{x}_i, \mathrm{u}_i) \tag{1}$$

with an initial state x_0 and applying a nominal control sequence $\mathbf{U}_0 = \{\mathrm{u}_0, \mathrm{u}_1, ..., \mathrm{u}_{N-1}\}$, yields a sequence of states $\mathbf{x} = \{\mathrm{x}_0, \mathrm{x}_1 ..., \mathrm{x}_N\}$, often referred to as a *trajectory*. The goal of *trajectory optimization* is to find an optimal control sequence $\mathbf{U}^*$ to reach the target state x_N, given any initial state x_0, such that it minimizes

*Email: shubham.kamble@rwth-aachen.de, ahmet.hakan.duman@rwth-aachen.de

a cost function,

$$J_0(\mathbf{x}, \mathbf{U}_0) = \sum_{j=0}^{N-1} \ell(\mathrm{x}_j, \mathrm{u}_j) + \ell_f(\mathrm{x}_N) \tag{2}$$

where $\ell(\mathrm{x}, \mathrm{u})$ and $\ell_f(\mathrm{x}_N)$ denotes the running cost and final cost respectively. To employ dynamic programming (DP), it is necessary to define a partial cost of the trajectory with initial state x_i (at time i),

$$J_i(\mathbf{x}, \mathbf{U}_i) = \sum_{j=i}^{N-1} \ell(\mathrm{x}_j, \mathrm{u}_j) + \ell_f(\mathrm{x}_N) \tag{3}$$

where $\mathbf{U}_i = \{\mathrm{u}_i, \mathrm{u}_{i+1}, ..., \mathrm{u}_{N-1}\}$ is the partial control sequence from time i. The *Value* at time i is the *cost-to-go* (lowest possible cost remaining) given the optimal partial control sequence $\mathbf{U}_i^*$,

$$V(\mathrm{x}, i) = J_i(\mathbf{x}, \mathbf{U}_i^*) = \min_{\mathbf{U}_i} J_i(\mathbf{x}, \mathbf{U}_i) \tag{4}$$

In order to find the optimal control sequence $\mathbf{U}_0^*$ from the start state x_0, we need to find $V(\mathrm{x}, 0)$,

$$V(\mathrm{x}, 0) = \min_{\mathbf{U}_0} J_0(\mathbf{x}, \mathbf{U}_0) \tag{5}$$

At time-step N, it is simply $V(\mathrm{x}, N) = \ell_f(\mathrm{x}_N)$ and going one time-step backwards we get,

$$V(\mathrm{x}, N-1) = \min_{\mathrm{u}_{N-1}} (\ell(\mathrm{x}_{N-1}, \mathrm{u}_{N-1}) + V(\mathrm{x}, N)) \tag{6}$$
$$\text{such that, } \mathrm{x}_N = f(\mathrm{x}_{N-1}, \mathrm{u}_{N-1})$$
$$V(\mathrm{x}, N) = \ell_f(\mathrm{x}_N)$$

Thus using the Bellman's optimality principle, the problem reduces from the minimization over an entire sequence of controls to a sequence of minimizations over a single control recursively backwards in time $i = N-1, ..., 0$ and setting $V(\mathrm{x}_N) = \ell_f(\mathrm{x}_N)$:

$$V(\mathrm{x}, i) = \min_{\mathrm{u}_i} \Big(\ell(\mathrm{x}_i, \mathrm{u}_i) + V(f(\mathrm{x}_i, \mathrm{u}_i), i+1)\Big) \tag{7}$$

Performing the minimization as per (7), yields the optimal control actions $\mathbf{U}_0^*$ and the *optimal value* of starting from state x_0. DP clearly requires the storage of $V(\mathrm{x}, i)$ and u_i for all x_i, $i = \{0, ..., N-1\}$, and this is usually impossible (*curse of dimensionality*). DDP avoids this difficulty by comparing the nominal trajectory with neighboring trajectories and selecting the neighboring trajectory that yields an optimal decrease in the cost function. The new control and state sequence become the nominal sequences for the next iteration of the algorithm. Since, at each iteration, attention is confined to a suitable neighborhood of x_i, it is possible to approximate the improved cost function over this neighborhood by truncated Taylor approximation, and all that is required to be stored are these Taylor approximation coefficients. Thus, defining the argument of the minimum (7) as a function of perturbations around the i^{th} (x, u) pair:

$$Q(\delta\mathrm{x}, \delta\mathrm{u}) = \ell(\mathrm{x}+\delta\mathrm{x}, \mathrm{u}+\delta\mathrm{u}) - \ell(\mathrm{x}, \mathrm{u}) + V(f(\mathrm{x}+\delta\mathrm{x}, \mathrm{u}+\delta\mathrm{u}), i+1) - V(f(\mathrm{x}, \mathrm{u}), i+1) \tag{8}$$

Taking Taylor approximation up to second order terms for $\ell(\mathrm{x}+\delta\mathrm{x}, \mathrm{u}+\delta\mathrm{u})$ and $V(f(\mathrm{x}+\delta\mathrm{x}, \mathrm{u}+\delta\mathrm{u}))$, and dropping the indices i and $i+1$, and primes denoting the next time-step we get,

$$\ell(\mathrm{x}+\delta\mathrm{x}, \mathrm{u}+\delta\mathrm{u}) \approx \ell(\mathrm{x}, \mathrm{u}) + \ell_\mathrm{x}^T\delta\mathrm{x} + \ell_\mathrm{u}^T\delta\mathrm{u} + \frac{1}{2}\Big(\delta\mathrm{x}^T\ell_{\mathrm{xx}}\delta\mathrm{x} + 2\delta\mathrm{x}^T\ell_{\mathrm{ux}}\delta\mathrm{u} + \delta\mathrm{u}^T\ell_{\mathrm{uu}}\delta\mathrm{u}\Big) \tag{9}$$

$$f(\mathrm{x}+\delta\mathrm{x}, \mathrm{u}+\delta\mathrm{u}) \approx f(\mathrm{x}, \mathrm{u}) + f_\mathrm{x}^T\delta\mathrm{x} + f_\mathrm{u}^T\delta\mathrm{u} + \frac{1}{2}\Big(\delta\mathrm{x}^T f_{\mathrm{xx}}\delta\mathrm{x} + 2\delta\mathrm{x}^T f_{\mathrm{ux}}\delta\mathrm{u} + \delta\mathrm{u}^T f_{\mathrm{uu}}\delta\mathrm{u}\Big) \tag{10}$$

$$V(\mathrm{x}'+\delta\mathrm{x}') \approx V(\mathrm{x}') + \delta\mathrm{x}'^T V_\mathrm{x}' + \frac{1}{2}\delta\mathrm{x}'^T V_{\mathrm{xx}}'\delta\mathrm{x}' \tag{11}$$

$$\delta\mathrm{x}' \approx f(\mathrm{x}+\delta\mathrm{x}, \mathrm{u}+\delta\mathrm{u}) - f(\mathrm{x}, \mathrm{u}) \tag{12}$$

Substituting equations (9), (10), (11) and (12) into (8), and neglecting terms of order higher than two,

$$\begin{aligned} Q(\delta\mathrm{x}, \delta\mathrm{u}) \approx{}& \delta\mathrm{x}^T(\ell_\mathrm{x} + (f_\mathrm{x})^T V_\mathrm{x}') + \delta\mathrm{u}^T(\ell_\mathrm{u} + (f_\mathrm{u})^T V_\mathrm{x}') \\ &+ \frac{1}{2}\delta\mathrm{x}^T n(\ell_{\mathrm{xx}} + (f_\mathrm{x})^T V_{\mathrm{xx}}' f_\mathrm{x} + V_\mathrm{x}' \cdot f_{\mathrm{xx}})\delta\mathrm{x} \\ &+ \frac{1}{2}\delta\mathrm{u}^T(\ell_{\mathrm{uu}} + (f_\mathrm{u})^T V_{\mathrm{xx}}' f_\mathrm{u} + V_\mathrm{x}' \cdot f_{\mathrm{uu}})\delta\mathrm{u} \\ &+ \delta\mathrm{u}^T(\ell_{\mathrm{xu}} + (f_\mathrm{u})^T V_{\mathrm{xx}}' f_\mathrm{x} + V_\mathrm{x}' \cdot f_{\mathrm{xu}})\delta\mathrm{x} \end{aligned}$$

$$\begin{aligned} Q(\delta\mathrm{x}, \delta\mathrm{u}) &\approx \delta\mathrm{x}^T Q_\mathrm{x} + \delta\mathrm{u}^T Q_\mathrm{u} + \frac{1}{2}\delta\mathrm{x}^T Q_{\mathrm{xx}}\delta\mathrm{x} \\ &\quad + \frac{1}{2}\delta\mathrm{u}^T Q_{\mathrm{uu}}\delta\mathrm{u} + \delta\mathrm{u}^T Q_{\mathrm{ux}}\delta\mathrm{x} \qquad (13) \\ &\approx \frac{1}{2}\begin{bmatrix}1\\ \delta\mathrm{x}\\ \delta\mathrm{u}\end{bmatrix}^T \begin{bmatrix}0 & Q_\mathrm{x}^T & Q_\mathrm{u}^T\\ Q_\mathrm{x} & Q_{\mathrm{xx}} & Q_{\mathrm{xu}}\\ Q_\mathrm{u} & Q_{\mathrm{ux}} & Q_{\mathrm{uu}}\end{bmatrix} \begin{bmatrix}1\\ \delta\mathrm{x}\\ \delta\mathrm{u}\end{bmatrix} \end{aligned}$$

where,

$$Q_\mathrm{x} = \ell_\mathrm{x} + (f_\mathrm{x})^T V_\mathrm{x}' \tag{14a}$$
$$Q_\mathrm{u} = \ell_\mathrm{u} + (f_\mathrm{u})^T V_\mathrm{x}' \tag{14b}$$
$$Q_{\mathrm{xx}} = \ell_{\mathrm{xx}} + (f_\mathrm{x})^T V_{\mathrm{xx}}' f_\mathrm{x} + V_\mathrm{x}' \cdot f_{\mathrm{xx}} \tag{14c}$$
$$Q_{\mathrm{uu}} = \ell_{\mathrm{uu}} + (f_\mathrm{u})^T V_{\mathrm{xx}}' f_\mathrm{u} + V_\mathrm{x}' \cdot f_{\mathrm{uu}} \tag{14d}$$
$$Q_{\mathrm{ux}} = \ell_{\mathrm{ux}} + (f_\mathrm{u})^T V_{\mathrm{xx}}' f_\mathrm{x} + V_\mathrm{x}' \cdot f_{\mathrm{ux}} \tag{14e}$$

Minimizing (13) w.r.t $\delta\mathrm{u}$ we get,

$$\delta\mathrm{u}^* = \min_{\delta\mathrm{u}} Q(\delta\mathrm{x}, \delta\mathrm{u}) = -Q_{\mathrm{uu}}^{-1}(Q_\mathrm{u} + Q_{\mathrm{ux}}\delta\mathrm{x}) \tag{15}$$

which consists of open loop and feedback gain terms,

$$\mathbf{k} = -Q_{\mathrm{uu}}^{-1}Q_\mathrm{u} \tag{16a}$$
$$\mathbf{K} = -Q_{\mathrm{uu}}^{-1}Q_{\mathrm{ux}} \tag{16b}$$

Substituting (15) in (13):

$$Q(\delta \mathrm{x}) \approx -\frac{1}{2} Q_{\mathrm{u}} Q_{\mathrm{uu}}^{-1} Q_{\mathrm{u}} + \Big(Q_{\mathrm{x}} - Q_{\mathrm{u}} Q_{\mathrm{uu}}^{-1} Q_{\mathrm{ux}}\Big)\delta \mathrm{x} + \frac{1}{2}\delta \mathrm{x}^T \Big(Q_{\mathrm{xx}} - Q_{\mathrm{xu}} Q_{\mathrm{uu}}^{-1} Q_{\mathrm{ux}}\Big)\delta \mathrm{x} \quad (17)$$

Thus we obtain the Taylor approximation coefficients of the value at time i:

$$\Delta V(i) = -\frac{1}{2} Q_{\mathrm{u}} Q_{\mathrm{uu}}^{-1} Q_{\mathrm{u}} \quad (18a)$$

$$V_{\mathrm{x}}(i) = Q_{\mathrm{x}} - Q_{\mathrm{u}} Q_{\mathrm{uu}}^{-1} Q_{\mathrm{ux}} \quad (18b)$$

$$V_{\mathrm{xx}}(i) = Q_{\mathrm{xx}} - Q_{\mathrm{xu}} Q_{\mathrm{uu}}^{-1} Q_{\mathrm{ux}} \quad (18c)$$

Recursively computing these coefficients of $V(\mathbf{x})$ and the control modifications $\{\mathbf{k}_i, \mathbf{K}_i\}$, constitutes the backward pass. Once it is completed, a forward pass computes a new trajectory:

$$\hat{\mathrm{x}}_0 = \mathrm{x}_0 \quad (19a)$$

$$\hat{\mathrm{u}}_i = \mathrm{u}_i + \mathbf{k}_i + \mathbf{K}_i(\hat{\mathrm{x}}_i - \mathrm{x}_i) \quad (19b)$$

$$\hat{\mathrm{x}}_{i+1} = f(\hat{\mathrm{x}}_i, \hat{\mathrm{u}}_i) \quad (19c)$$

The consequence of the above optimization performed for time-step i justifies the following iteration that is used in DDP and the algorithms derived from it:

1. Given a nominal policy $\bar{\mathbf{U}}_0$, compute the future states $\bar{\mathrm{x}}$.
2. Recursively compute the optimal controls using (14), (16), and (18) for time-steps starting from $N-1$,..,0.
3. Replace the nominal trajectories with the new ones using (19) and repeat until convergence.

How step 2, from above, is performed may vary in different algorithms. The general idea in all DDP-based algorithms is to form the second-order Taylor approximation.

III. Convergence Proof of DDP

The purpose of this section is to carefully examine the convergence properties of the DDP algorithm. There was no rigorous proof of convergence until the early 1980s. Proofs of quadratic convergence of the DDP were given independently by Pantoja [6, 4], and Murray and Yakowitz [7]. Both proofs were based on the comparison between DDP and stagewise Newton's method but neither used dynamic programming and Bellman's Optimality Principle. Here, we provide the proof of quadratic convergence which is alongside [8]. The results of the quadratic convergence proof is summarized in the following Theorem 1.

Theorem 1 *: There exists a constant $c > 0$ such that $\|\boldsymbol{U}^{j+1} - \boldsymbol{U}^*\| \leq c \cdot \|\boldsymbol{U}^j - \boldsymbol{U}^*\|^2$ for all $j \geq 0$ provided $\|\boldsymbol{U}^0 - \boldsymbol{U}^*\|$ is sufficiently small and the following set of assumptions are satisfied.*

Assumption 1 *: $\ell(x_i, u_i)$, $f(x_i, u_i)$, and $\ell(x_N)$ have continuous third partial derivatives with respect to x_i and u_i over a closed bounded convex set $D \subset R^{n+m}$, where n and m are the dimensions of x_i and u_i respectively.*

Assumption 2 *: $\boldsymbol{U}^j$ is the control trajectory obtained by the j^{th} iteration of the DDP algorithm.*

Assumption 3 *: The matrices $Q^j_{uu_i}$ for $i = 0, ..., N-1$ computed at the j^{th} iteration of the DDP algorithm are all positive definite in D.*

Assumption 4 *: (x_i^j, u_i^j) and $x_N^j \in D$ for $i = 0, ..., N-1$ and for all $j > 0$, where x_i^j, $i = 0, ..., N$ and u_i^j, $i = 0, ..., N-1$ are the components of the $\boldsymbol{x}^j$ and $\boldsymbol{U}^j$ computed in the j^{th} iteration of the DDP algorithm.*

Assumption 5 *: $\boldsymbol{U}^j$ converges to $\boldsymbol{U}^*$ which is a solution to problem (5), where x^* is the trajectory associated with $\boldsymbol{U}^*$, and (x_i^*, u_i^*) and $x_N^* \in int(D)$ for $i = 0, ..., N-1$.*

Assumption 6 *: $\ell(x_i, u_i)$ and $f(x_i, u_i)$, i.e., the running cost functions and transition functions are identical for $i = 0, ..., N-1$*

Proof : For $j \geq 0$, let the vectors $\mathbf{U}^j(k)$ and $\mathbf{U}^{j+1}(k)$ be the solutions resulting from the j^{th} and $(j+1)^{th}$ iterations of the DDP algorithm, respectively, for solving (5) with $N = k$. Also, let $\mathbf{U}^*(k)$ denote the optimal solution to problem (5) with $N = k$. We will use the induction on the number of time steps N to prove this theorem.

a) First, for $N = 2$, consider

$$\min_{\mathbf{U}} J_0(\mathbf{x}, \mathbf{U}) = \min_{\mathrm{u}_0, \mathrm{u}_1} \Big(\ell(\mathrm{x}_0, \mathrm{u}_0) + \ell(\mathrm{x}_1, \mathrm{u}_1) + \ell_f(\mathrm{x}_2)\Big) \quad (20)$$

where $\mathrm{x}_0 = \bar{\mathrm{x}}_0$,
$\mathrm{x}_1 = f(\mathrm{x}_0, \mathrm{u}_0)$,
$\mathrm{x}_2 = f(\mathrm{x}_1, \mathrm{u}_1)$,
$(\bar{\mathrm{u}}_0, \bar{\mathrm{u}}_1)$ is the initial policy
$(\bar{\mathrm{x}}_0, \bar{\mathrm{x}}_1, \bar{\mathrm{x}}_2)$ is the initial trajectory

The application of DDP algorithm to problem (20) starts with computing (14) (16), and (18) for decreas-

ing $i = 1, 0$ and setting $V(\mathrm{x}_2) = \ell_f(\mathrm{x}_2)$.

$$Q_{\mathrm{x}_i} = \ell_{\mathrm{x}_i} + (f^1_{\mathrm{x}_i})^T V_{\mathrm{x}_{i+1}} \tag{21a}$$
$$Q_{\mathrm{u}_i} = \ell_{\mathrm{u}_i} + (f^1_{\mathrm{u}_i})^T V_{\mathrm{x}_{i+1}} \tag{21b}$$
$$Q_{\mathrm{xx}_i} = \ell_{\mathrm{xx}_i} + (f^1_{\mathrm{x}_i})^T V_{\mathrm{xx}_{i+1}} f^1_{\mathrm{x}_i} + V_{\mathrm{x}_{i+1}} \cdot f^1_{\mathrm{xx}_i} \tag{21c}$$
$$Q_{\mathrm{uu}_i} = \ell_{\mathrm{uu}_i} + (f^1_{\mathrm{u}_i})^T V_{\mathrm{xx}_{i+1}} f^1_{\mathrm{u}_i} + V_{\mathrm{x}_{i+1}} \cdot f^1_{\mathrm{uu}_i} \tag{21d}$$
$$Q_{\mathrm{ux}_i} = \ell_{\mathrm{ux}_i} + (f^1_{\mathrm{u}_i})^T V_{\mathrm{xx}_{i+1}} f^1_{\mathrm{x}_i} + V_{\mathrm{x}_{i+1}} \cdot f^1_{\mathrm{ux}_i} \tag{21e}$$

$$\mathbf{k}_i = -Q^{-1}_{\mathrm{uu}_i} Q_{\mathrm{u}_i} \tag{22a}$$
$$\mathbf{K}_i = -Q^{-1}_{\mathrm{uu}_i} Q_{\mathrm{ux}_i} \tag{22b}$$

$$\Delta V(i) = -\frac{1}{2} Q_{\mathrm{u}_i} Q^{-1}_{\mathrm{uu}_i} Q_{\mathrm{u}_i} \tag{23a}$$
$$V_{\mathrm{x}_i}(i) = Q_{\mathrm{x}_i} - Q_{\mathrm{u}_i} Q^{-1}_{\mathrm{uu}_i} Q_{\mathrm{ux}_i} \tag{23b}$$
$$V_{\mathrm{xx}_i}(i) = Q_{\mathrm{xx}_i} - Q_{\mathrm{xu}_i} Q^{-1}_{\mathrm{uu}_i} Q_{\mathrm{ux}_i} \tag{23c}$$

Next, performing the forward pass to update the trajectory to the new nominal one using the dynamics of (20), for increasing $i = 0, 1$,

$$\mathrm{x}_0 = \bar{\mathrm{x}}_0 \tag{24a}$$
$$\mathrm{u}_0 = \bar{\mathrm{u}}_0 + \mathbf{k}_0 + \mathbf{K}_0(\mathrm{x}_0 - \bar{\mathrm{x}}_0) = \bar{\mathrm{u}}_0 + \mathbf{k}_0 \tag{24b}$$
$$\mathrm{x}_1 = f(\mathrm{x}_0, \mathrm{u}_0) \tag{24c}$$
$$\mathrm{u}_1 = \bar{\mathrm{u}}_1 + \mathbf{k}_1 + \mathbf{K}_1(\mathrm{x}_1 - \bar{\mathrm{x}}_1) \tag{24d}$$
$$\mathrm{x}_2 = f(\mathrm{x}_1, \mathrm{u}_1) \tag{24e}$$

Since $(\bar{\mathrm{x}}_0, \mathrm{u}^*_0, \mathrm{x}^*_1, \mathrm{u}^*_1, \mathrm{x}^*_2)$ is the solution to problem (20),

$$J_0(\bar{\mathrm{x}}_0, \mathrm{u}^*_0, \mathrm{x}^*_1, \mathrm{u}^*_1, \mathrm{x}^*_2) = \min_{\mathrm{u}_0, \mathrm{u}_1} J_0(\mathrm{x}, \mathbf{U}) \tag{25}$$
$$(J_0)_{\mathrm{u}_1} = Q_{\mathrm{u}_1}|_{(\mathrm{x}^*_1, \mathrm{u}^*_1, \mathrm{x}^*_2)} = 0 \tag{26}$$
$$(J_0)_{\mathrm{u}_0} = Q_{\mathrm{u}_0}|_{(\bar{\mathrm{x}}_0, \mathrm{u}^*_0, \mathrm{x}^*_1, \mathrm{u}^*_1, \mathrm{x}^*_2))} = 0 \tag{27}$$

From Taylor expansion[1], we have

$$\begin{aligned} &Q_{\mathrm{u}_1}|_{(\bar{\mathrm{x}}_1, \bar{\mathrm{u}}_1, \bar{\mathrm{x}}_2)} - Q_{\mathrm{u}_1}|_{(\mathrm{x}^*_1, \mathrm{u}^*_1, \mathrm{x}^*_2)} \\ &= Q_{\mathrm{xu}_1}(\bar{\mathrm{x}}_1 - \mathrm{x}^*_1) + Q_{\mathrm{uu}_1}(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (Q_{\mathrm{u}_1})_{\mathrm{x}_2}(\bar{\mathrm{x}}_2 - \mathrm{x}^*_2) \\ &\quad + O([(\bar{\mathrm{x}}_1 - \mathrm{x}^*_1) + (\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (\bar{\mathrm{x}}_2 - \mathrm{x}^*_2)]^2) \end{aligned}$$

Since $\bar{\mathrm{x}}_1 - \mathrm{x}^*_1 = f_{\mathrm{u}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) + O((\bar{\mathrm{u}}_0 - \mathrm{u}^*_0)^2)$ and, $\bar{\mathrm{x}}_2 - \mathrm{x}^*_2 = f_{\mathrm{u}_1}(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + O((\bar{\mathrm{u}}_1 - \mathrm{u}^*_1)^2)$,

$$\begin{aligned} &Q_{\mathrm{u}_1}|_{(\bar{\mathrm{x}}_1, \bar{\mathrm{u}}_1, \bar{\mathrm{x}}_2)} - Q_{\mathrm{u}_1}|_{(\mathrm{x}^*_1, \mathrm{u}^*_1, \mathrm{x}^*_2)} \\ &= (Q_{\mathrm{uu}_1} + (Q_{\mathrm{u}_1})_{\mathrm{x}_2} f_{\mathrm{u}_1})(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + Q_{\mathrm{xu}_1} f_{\mathrm{u}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) \\ &\quad + O([(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (\bar{\mathrm{u}}_0 - \mathrm{u}^*_0)]^2) \end{aligned}$$

From (26),

$$\begin{aligned} &Q_{\mathrm{u}_1}|_{(\bar{\mathrm{x}}_1, \bar{\mathrm{u}}_1, \bar{\mathrm{x}}_2)} = (Q_{\mathrm{uu}_1} + (Q_{\mathrm{u}_1})_{\mathrm{x}_2} f_{\mathrm{u}_1})(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) \\ &+ Q_{\mathrm{xu}_1} f_{\mathrm{u}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) + O([(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (\bar{\mathrm{u}}_0 - \mathrm{u}^*_0)]^2) \end{aligned} \tag{28}$$

From (22) we get,

$$\begin{aligned} \mathbf{k}_1 &= -(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) - Q^{-1}_{\mathrm{uu}_1}(Q_{\mathrm{u}_1})_{\mathrm{x}_2} f_{\mathrm{u}_1}(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) \\ &\quad - Q^{-1}_{\mathrm{uu}_1} Q_{\mathrm{xu}_1} f_{\mathrm{u}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) \\ &\quad + O([(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (\bar{\mathrm{u}}_0 - \mathrm{u}^*_0)]^2) \end{aligned} \tag{29a}$$
$$\begin{aligned} \mathbf{K}_1(\mathrm{x}_1 - \bar{\mathrm{x}}_1) &= -Q^{-1}_{\mathrm{uu}_1} Q_{\mathrm{ux}_1}[f(\bar{\mathrm{x}}_0, \mathrm{u}_0) - f(\bar{\mathrm{x}}_0, \bar{\mathrm{u}}_0)] \\ &= -Q^{-1}_{\mathrm{uu}_1} Q_{\mathrm{ux}_1} f_{\mathrm{u}_0}(\mathrm{u}_0 - \bar{\mathrm{u}}_0) + O((\mathrm{u}_0 - \bar{\mathrm{u}}_0)^2) \end{aligned} \tag{29b}$$

Since $\mathrm{u}_1 = \bar{\mathrm{u}}_1 + \mathbf{k}_1 + \mathbf{K}_1(\mathrm{x}_1 - \bar{\mathrm{x}}_1)$,

$$\begin{aligned} \mathrm{u}_1 &= \bar{\mathrm{u}}_1 - (\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) - Q^{-1}_{\mathrm{uu}_1}(Q_{\mathrm{u}_1})_{\mathrm{x}_2} f_{\mathrm{u}_1}(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) \\ &\quad - Q^{-1}_{\mathrm{uu}_1} Q_{\mathrm{xu}_1} f_{\mathrm{u}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) - Q^{-1}_{\mathrm{uu}_1} Q_{\mathrm{ux}_1} f_{\mathrm{u}_0}(\mathrm{u}_0 - \bar{\mathrm{u}}_0) \\ &\quad + O([(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (\bar{\mathrm{u}}_0 - \mathrm{u}^*_0)]^2 + (\mathrm{u}_0 - \bar{\mathrm{u}}_0)^2) \end{aligned}$$

$$\begin{aligned} \mathrm{u}_1 - \mathrm{u}^*_1 &= -Q^{-1}_{\mathrm{uu}_1}(Q_{\mathrm{u}_1})_{\mathrm{x}_2} f_{\mathrm{u}_1}(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) \\ &\quad - Q^{-1}_{\mathrm{uu}_1} Q_{\mathrm{xu}_1} f_{\mathrm{u}_0}(\mathrm{u}_0 - \mathrm{u}^*_0) \\ &\quad + O([(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (\bar{\mathrm{u}}_0 - \mathrm{u}^*_0)]^2 + (\mathrm{u}_0 - \bar{\mathrm{u}}_0)^2) \end{aligned} \tag{30}$$

Similarly from Taylor expansion[1] of

$$\begin{aligned} &Q_{\mathrm{u}_0}|_{(\bar{\mathrm{x}}_0, \bar{\mathrm{u}}_0, \bar{\mathrm{x}}_1, \bar{\mathrm{u}}_1, \bar{\mathrm{x}}_2)} - Q_{\mathrm{u}_0}|_{(\mathrm{x}^*_0, \mathrm{u}^*_0, \mathrm{x}^*_1, \mathrm{u}^*_1, \mathrm{x}^*_2)} \\ &= Q_{\mathrm{xu}_0}(\bar{\mathrm{x}}_0 - \mathrm{x}^*_0) + Q_{\mathrm{uu}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) + (Q_{\mathrm{u}_0})_{\mathrm{x}_1}(\bar{\mathrm{x}}_1 - \mathrm{x}^*_1) \\ &+ (Q_{\mathrm{u}_0})_{\mathrm{u}_1}(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (Q_{\mathrm{u}_0})_{\mathrm{x}_2}(\bar{\mathrm{x}}_2 - \mathrm{x}^*_2) + O([(\bar{\mathrm{x}}_0 - \mathrm{x}^*_0) \\ &+ (\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) + (\bar{\mathrm{x}}_1 - \mathrm{x}^*_1) + (\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (\bar{\mathrm{x}}_2 - \mathrm{x}^*_2)]^2) \end{aligned}$$

Since $\bar{\mathrm{x}}_0 - \mathrm{x}^*_0 = 0$, $\bar{\mathrm{x}}_1 - \mathrm{x}^*_1 = f_{\mathrm{u}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) + O((\bar{\mathrm{u}}_0 - \mathrm{u}^*_0)^2)$ and, $\bar{\mathrm{x}}_2 - \mathrm{x}^*_2 = f_{\mathrm{u}_1}(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + O((\bar{\mathrm{u}}_1 - \mathrm{u}^*_1)^2)$,

$$\begin{aligned} &Q_{\mathrm{u}_0}|_{(\bar{\mathrm{x}}_0, \bar{\mathrm{u}}_0, \bar{\mathrm{x}}_1, \bar{\mathrm{u}}_1, \bar{\mathrm{x}}_2)} - Q_{\mathrm{u}_0}|_{(\mathrm{x}^*_0, \mathrm{u}^*_0, \mathrm{x}^*_1, \mathrm{u}^*_1, \mathrm{x}^*_2)} \\ &\quad = (Q_{\mathrm{uu}_0} + (Q_{\mathrm{u}_0})_{\mathrm{x}_1} f_{\mathrm{u}_0})(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) \\ &+ ((Q_{\mathrm{u}_0})_{\mathrm{u}_1} + (Q_{\mathrm{u}_0})_{\mathrm{x}_2} f_{\mathrm{u}_1})(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + O([(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) + (\bar{\mathrm{u}}_1 - \mathrm{u}^*_1)]^2) \end{aligned}$$

From (27),

$$\begin{aligned} Q_{\mathrm{u}_0}|_{(\bar{\mathrm{x}}_0, \bar{\mathrm{u}}_0, \bar{\mathrm{x}}_1, \bar{\mathrm{u}}_1, \bar{\mathrm{x}}_2)} &= (Q_{\mathrm{uu}_0} + (Q_{\mathrm{u}_0})_{\mathrm{x}_1} f_{\mathrm{u}_0})(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) \\ &\quad + ((Q_{\mathrm{u}_0})_{\mathrm{u}_1} + (Q_{\mathrm{u}_0})_{\mathrm{x}_2} f_{\mathrm{u}_1})(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) \\ &\quad + O([(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) + (\bar{\mathrm{u}}_1 - \mathrm{u}^*_1)]^2) \end{aligned} \tag{31}$$

From (22) we get,

$$\begin{aligned} \mathbf{k}_0 &= -(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) - Q^{-1}_{\mathrm{uu}_0}(Q_{\mathrm{u}_0})_{\mathrm{x}_1} f_{\mathrm{u}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) \\ &\quad - Q^{-1}_{\mathrm{uu}_0}((Q_{\mathrm{u}_0})_{\mathrm{u}_1} + (Q_{\mathrm{u}_0})_{\mathrm{x}_2} f_{\mathrm{u}_1})(\bar{\mathrm{u}}_0 = 1 - \mathrm{u}^*_1) \\ &\quad + O([(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (\bar{\mathrm{u}}_0 - \mathrm{u}^*_0)]^2) \end{aligned} \tag{32a}$$
$$\mathbf{K}_0(\mathrm{x}_0 - \bar{\mathrm{x}}_0) = 0 \tag{32b}$$

Since $\mathrm{u}_0 = \bar{\mathrm{u}}_0 + \mathbf{k}_0$,

$$\begin{aligned} \mathrm{u}_0 &= \bar{\mathrm{u}}_0 - (\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) - Q^{-1}_{\mathrm{uu}_0}(Q_{\mathrm{u}_0})_{\mathrm{x}_1} f_{\mathrm{u}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}^*_0) \\ &\quad - Q^{-1}_{\mathrm{uu}_0}((Q_{\mathrm{u}_0})_{\mathrm{u}_1} + (Q_{\mathrm{u}_0})_{\mathrm{x}_2} f_{\mathrm{u}_1})(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) \\ &\quad + O([(\bar{\mathrm{u}}_1 - \mathrm{u}^*_1) + (\bar{\mathrm{u}}_0 - \mathrm{u}^*_0)]^2) \end{aligned}$$

[1] All partial derivatives on the right hand side are taken at the nominal points

$$\begin{aligned}\mathrm{u}_0 - \mathrm{u}_0^* = &-Q_{\mathrm{uu}_0}^{-1}(Q_{\mathrm{u}_0})_{\mathrm{x}_1} f_{\mathrm{u}_0}(\bar{\mathrm{u}}_0 - \mathrm{u}_0^*) \\ &- Q_{\mathrm{uu}_0}^{-1}((Q_{\mathrm{u}_0})_{\mathrm{u}_1} + (Q_{\mathrm{u}_0})_{\mathrm{x}_2} f_{\mathrm{u}_1})(\bar{\mathrm{u}}_1 - \mathrm{u}_1^*) \\ &+ O([(\bar{\mathrm{u}}_1 - \mathrm{u}_1^*) + (\bar{\mathrm{u}}_0 - \mathrm{u}_0^*)]^2) \end{aligned} \tag{33}$$

Plugging equation (33) into equation (30), and further combining them we get,

$$\|\mathbf{U}(2) - \mathbf{U}^*(2)\| \leq c(\bar{\mathbf{x}}(2), \bar{\mathbf{U}}(2)) \cdot \|\bar{\mathbf{U}}(2) - \mathbf{U}^*(2)\|^2 \tag{34}$$

where $c(\bar{\mathbf{x}}(2), \bar{\mathbf{U}}(2))$ is a positive function of ℓ, and f and its derivatives through equations (21). From assumptions 1), 4) and 5), and from the continuity of $c(\bar{\mathbf{x}}(2), \bar{\mathbf{U}}(2))$, we know that $c(\bar{\mathbf{x}}(2), \bar{\mathbf{U}}(2))$ is bounded in D; hence there exists a constant $c > 0$ such that $c(\bar{\mathbf{x}}(2), \bar{\mathbf{U}}(2)) \leq c$ for all $(\bar{\mathrm{x}}_i, \bar{\mathrm{u}}_i) \in D, i = 0, 1$. From (34) we have,

$$\|\mathbf{U}(2) - \mathbf{U}^*(2)\| \leq c \cdot \|\bar{\mathbf{U}}(2) - \mathbf{U}^*(2)\|^2 \tag{35}$$

Thus the induction hypothesis is true for $N = 2$.

b) Suppose the induction hypothesis is true for $N = k - 1$, then there exists a constant $c_1 > 0$ such that,

$$\|\mathbf{U}(k-1) - \mathbf{U}^*(k-1)\| \leq c_1 \cdot \|\bar{\mathbf{U}}(k-1) - \mathbf{U}^*(k-1)\|^2 \tag{36}$$

where $\mathbf{U}^*(k-1)$ is the optimal solution for the following $(k-1)$ time-step problem,

$$\min_{\mathbf{U}} J_0(\mathbf{x}, \mathbf{U}) = \min_{\mathrm{u}_0, \ldots, \mathrm{u}_{k-1}} \sum_{i=0}^{k-2} \ell(\mathrm{x}_i, \mathrm{u}_i) + \ell(\mathrm{x}_{k-1}) \tag{37}$$

$$\begin{aligned}&\text{where } \mathrm{x}_0 = \bar{\mathrm{x}}_0, \\ &\qquad \mathrm{x}_{i+1} = f(\mathrm{x}_i, \mathrm{u}_i), \text{ for } i = 0, \ldots, k-2\end{aligned}$$

Below we will show that for $N = k$, our induction hypothesis is still true. In the case of $N = k$, our problem is as folllows,

$$\min_{\mathbf{U}} J_0(\mathbf{x}, \mathbf{U}) = \min_{\mathrm{u}_0, \ldots, \mathrm{u}_k} \sum_{i=0}^{k-1} \ell(\mathrm{x}_i, \mathrm{u}_i) + \ell(\mathrm{x}_k) \tag{38}$$

$$\begin{aligned}&\text{where } \mathrm{x}_0 = \bar{\mathrm{x}}_0, \\ &\qquad \mathrm{x}_{i+1} = f(\mathrm{x}_i, \mathrm{u}_i), \text{ for } i = 0, \ldots, k-1\end{aligned}$$

The application to DDP algorithm to problem (38) results in $V(\mathrm{x}_k), V_{\mathrm{x}_k}, V_{\mathrm{xx}_k}, Q_{\mathrm{x}_{k-1}}, Q_{\mathrm{u}_{k-1}}, Q_{\mathrm{xx}_{k-1}}, Q_{\mathrm{uu}_{k-1}}, Q_{\mathrm{ux}_{k-1}}, V_{\mathrm{x}_{k-1}}, V_{\mathrm{xx}_{k-1}}, \mathbf{k}_{k-1}, \mathbf{K}_{k-1}$. The approach here is to combine the last two stages (k and $k-1$) in the problem (38) and convert it into a $(k-1)$ step problem like (37). We have defined the following in the previous section,

$$\begin{aligned}V(\mathrm{x}_{k-1}) = &\ell(\mathrm{x}_{k-1}, \bar{\mathrm{u}}_{k-1} + \mathbf{k}_{k-1} + \mathbf{K}_{k-1}(\mathrm{x}_{k-1} - \bar{\mathrm{x}}_{k-1})) \\ &+ \ell(\mathrm{x}_k)\end{aligned} \tag{39}$$

$$\begin{aligned}&\text{where } \mathrm{u}_{k-1} = \bar{\mathrm{u}}_{k-1} + \mathbf{k}_{k-1} + \mathbf{K}_{k-1}(\mathrm{x}_{k-1} - \bar{\mathrm{x}}_{k-1}) \\ &\qquad \mathrm{x}_k = f(\mathrm{x}_{k-1}, \mathrm{u}_{k-1}) \\ &\qquad \mathbf{k}_{k-1} = -Q_{\mathrm{uu}_{k-1}}^{-1} Q_{\mathrm{u}_{k-1}} \\ &\qquad \mathbf{K}_{k-1} = -Q_{\mathrm{uu}_{k-1}}^{-1} Q_{\mathrm{ux}_{k-1}}\end{aligned}$$

By introducing this explicitly, we can convert the problem (38) to a similar form of problem (37) as follows,

$$\min_{\mathbf{U}} J_0(\mathbf{x}, \mathbf{U}) = \min_{\mathrm{u}_0, \ldots, \mathrm{u}_{k-1}} \sum_{i=0}^{k-2} \ell(\mathrm{x}_i, \mathrm{u}_i) + V(\mathrm{x}_{k-1}) \tag{40}$$

$$\begin{aligned}&\text{where } \mathrm{x}_0 = \bar{\mathrm{x}}_0, \\ &\qquad \mathrm{x}_{i+1} = f(\mathrm{x}_i, \mathrm{u}_i), \text{ for } i = 0, \ldots, k-2 \\ &\qquad V(\mathrm{x}_{k-1}) = \ell(\mathrm{x}_{k-1})\end{aligned}$$

$V(\mathrm{x}_{k-1})$ is the output of minimization over u_{k-1} which is what we will use to minimize over u_{k-2} (Bellman's optimality principle), and for that we will need $V_{\mathrm{x}_{k-1}}$ and $V_{\mathrm{xx}_{k-1}}$, which are dependent on $\hat{Q}_{\mathrm{x}_{k-1}}, \hat{Q}_{\mathrm{u}_{k-1}}, \hat{Q}_{\mathrm{xx}_{k-1}}, \hat{Q}_{\mathrm{uu}_{k-1}}$, and $\hat{Q}_{\mathrm{ux}_{k-1}}$ evaluated from problem (40). The value of x_{k-1} remains the same till the end of the backward pass and hence,

$$\begin{aligned}&\hat{Q}_{\mathrm{x}_{k-1}} = Q_{\mathrm{x}_{k-1}}, \ \hat{Q}_{\mathrm{u}_{k-1}} = Q_{\mathrm{u}_{k-1}}, \ \hat{Q}_{\mathrm{xx}_{k-1}} = Q_{\mathrm{xx}_{k-1}}, \\ &\hat{Q}_{\mathrm{ux}_{k-1}} = Q_{\mathrm{ux}_{k-1}}, \ \hat{Q}_{\mathrm{uu}_{k-1}} = Q_{\mathrm{uu}_{k-1}}\end{aligned} \tag{41}$$

Assumption 3) indicates that $Q_{\mathrm{uu}_{k-1}}$ is positive definite and above (41) implies that $\hat{Q}_{\mathrm{uu}_{k-1}}$ is also positive definite. Since the DDP algorithm is backward, recursive method, from (41) and assumption 6) (i.e., running cost functions and transition functions are identical for $i = 0, \ldots, k-2$), in problems (37) and (38), it follows that,

$$\begin{aligned}&\hat{Q}_{\mathrm{x}_i} = Q_{\mathrm{x}_i}, \ \hat{Q}_{\mathrm{u}_i} = Q_{\mathrm{u}_i}, \ \hat{Q}_{\mathrm{xx}_i} = Q_{\mathrm{xx}_i}, \\ &\hat{Q}_{\mathrm{ux}_i} = Q_{\mathrm{ux}_i}, \ \hat{Q}_{\mathrm{uu}_i} = Q_{\mathrm{uu}_i} \text{ for } i = 0, \ldots, k-1..\end{aligned} \tag{42}$$

Therefore, (42) and the function V generates the same sequence of $\mathbf{U}(k)$ for problems (38) and (40), where $\mathrm{u}_{k-1} = \bar{\mathrm{u}}_{k-1} + \mathbf{k}_{k-1} + \mathbf{K}_{k-1}(\mathrm{x}_{k-1} - \bar{\mathrm{x}}_{k-1})$ for problem (40). The application of DDP algorithm to problem (40) at the last step is just a Newton's method,

$$|\mathrm{u}_k - \mathrm{u}_k^*| \leq c_2 \cdot \ |\bar{\mathrm{u}}_k - \mathrm{u}_k^*|^2 \text{ for some } c_2 > 0. \tag{43}$$

Combining (36) and (43), we get

$$\|\mathbf{U}(k) - \mathbf{U}^*(k)\| \leq c(\bar{\mathbf{x}}(k), \bar{\mathbf{U}}(k)) \cdot \|\bar{\mathbf{U}}(k) - \mathbf{U}^*(k)\|^2 \tag{44}$$

Following the same reason as (35), there exists a constant $c > 0$ such that $c(\bar{\mathbf{x}}(k), \bar{\mathbf{U}}(k)) \leq c$ for all $(\bar{\mathrm{x}}_i, \bar{\mathrm{u}}_i) \in D, i = 0, \ldots, k-1$ and therefore,

$$\|\mathbf{U}(k) - \mathbf{U}^*(k)\| \leq c \cdot \|\bar{\mathbf{U}}(k) - \mathbf{U}^*(k)\|^2 \tag{45}$$

■

The characteristics of matrices $Q_{\mathrm{uu}i}$ plays an important role in determining the convergence properties for different classes of problems. In the above proof we set the assumption that the matrices $Q_{\mathrm{uu}i}$ are positive definite, which is generally not the case for both systems with linear and nonlinear transition functions $f(\mathrm{x}_i, \mathrm{u}_i)$. For the former case, sufficient conditions

on the running cost functions $\ell(\mathrm{x}_i, \mathrm{u}_i)$ in accordance with [9, Theroem (2)] guarantees that the matrices Q_{uu_i} are positive definite for DDP applied to them. Further, [9, Theroem (3)] indicates that for the latter, even for positive definite convex $\ell(\mathrm{x}_i, \mathrm{u}_i)$ and positive semidefinite convex transition functions $f(\mathrm{x}_i, \mathrm{u}_i)$, there will always be some nonpositive definite matrices Q_{uu_i} and hence the algorithm may not converge. In the following section we will discuss three regularization techniques which helps us better the convergence rate of DDP to the general class of systems with nonlinear transition functions.

IV. Regularization and Line-search

Luenberger [10] suggested a shift when the Hessian matrix for Newton's method was not positive definite. His demonstration that this shift for Newton's method converges quadratically gave Yakowitz et al. [11] the idea to replace Q_{uu} with

$$\tilde{Q}_{\mathrm{uu}} = Q_{\mathrm{uu}} + \mu^c I \tag{46}$$

where μ^c is a positive number large enough to assure that $\tilde{Q}_{\mathrm{uu}}$ is positive definite. This is referred to as *constant shift*. Instead of the above, [11] also suggested an *active shift* as follows:

$$\tilde{Q}_{\mathrm{uu}} = Q_{\mathrm{uu}} + \mu^a(\delta) I \tag{47}$$

$$\text{where } \mu^a(\delta) = \begin{cases} \delta - \lambda(Q_{\mathrm{uu}}) & \text{if } \lambda(Q_{\mathrm{uu}}) < \delta, \\ 0 & \text{if } \lambda(Q_{\mathrm{uu}}) \geq \delta \end{cases}$$

$\lambda(Q_{\mathrm{uu}})$ denotes the minimum eigenvalue of Q_{uu}

The third kind of regularizer called the *adaptive shift* was also introduced by [9], which combined both constant and active shifts. Irrespective of the regularization techniques adopted, the first drawback is that the selection of μ is not obvious. Second, if μ (for all three shifts) is too large then the convergence is slow and if it is small then it may run into numerical difficulties. Third, any of these modifications amounts to adding a quadratic cost around the current control sequence, making the steps more conservative, and also the same control perturbation can have different effects at different time steps i in the backward pass, depending on the control-transition matrix f_{u}. Therefore introducing a scheme that penalizes deviations from the future states rather than controls is more suitable to tackle the third drawback:

$$\tilde{V}'_{\mathrm{xx}} = V'_{\mathrm{xx}} + \mu I \tag{48}$$

This regularization places a quadratic state cost around the previous sequence. Unlike the standard control-based regularization, the feedback gains $\mathbf{K}_i$ do not vanish as $\mu \to \infty$ but rather force the new trajectory closer to the old one, significantly improving robustness. Finally last two equations from (14), the open loop and feedback gain terms (16), and the value updates (18) are improved as follows [12]:

$$\tilde{Q}_{\mathrm{uu}} = \ell_{\mathrm{uu}} + (f_{\mathrm{u}})^T \tilde{V}'_{\mathrm{xx}} f_{\mathrm{u}} + V'_{\mathrm{x}} \cdot f_{\mathrm{uu}} \tag{49a}$$

$$\tilde{Q}_{\mathrm{ux}} = \ell_{\mathrm{ux}} + (f_{\mathrm{u}})^T \tilde{V}'_{\mathrm{xx}} f_{\mathrm{x}} + V'_{\mathrm{x}} \cdot f_{\mathrm{ux}} \tag{49b}$$

$$\mathbf{k} = -\tilde{Q}_{\mathrm{uu}}^{-1} Q_{\mathrm{u}} \tag{49c}$$

$$\mathbf{K} = -\tilde{Q}_{\mathrm{uu}}^{-1} \tilde{Q}_{\mathrm{ux}} \tag{49d}$$

$$\Delta V(i) = +\frac{1}{2} \mathbf{k}^T Q_{\mathrm{uu}} \mathbf{k} + \mathbf{k}^T Q_{\mathrm{u}} \tag{49e}$$

$$V_{\mathrm{x}}(i) = Q_{\mathrm{x}} + \mathbf{K}^T Q_{\mathrm{uu}} \mathbf{k} + \mathbf{K}^T Q_{\mathrm{u}} + Q_{\mathrm{ux}}^T \mathbf{k} \tag{49f}$$

$$V_{\mathrm{xx}}(i) = Q_{\mathrm{xx}} + \mathbf{K}^T Q_{\mathrm{uu}} \mathbf{K} + \mathbf{K}^T Q_{\mathrm{ux}} + Q_{\mathrm{ux}}^T \mathbf{K} \tag{49g}$$

Resolving the first two drawbacks gives rise to three conflicting requirements. If we are near the minimum we would like μ to quickly go to zero for fast convergence. If the backward pass fails due to encountering a non-PD Q_{uu}, we would like it to increase very rapidly since the minimum value of μ which prevents divergence is often very large. Finally, if we are in a regime where some $\mu > 0$ is required, we would like to accurately tweak it to be as close as possible to the minimum value, but not smaller. A suitable solution is to adopt a quadratic modification scheme as follows:

Choose μ_{min} (typically 10^{-6})
Choose Δ_0 (typically 2)
Increase μ :
$\quad \Delta \leftarrow \max(\Delta_0, \Delta \cdot \Delta_0)$
$\quad \mu \leftarrow \max(\mu_{min}, \mu \cdot \Delta)$
Decrease μ :
$\quad \Delta \leftarrow \min(\frac{1}{\Delta_0}, \frac{\Delta}{\Delta_0})$
$\quad \mu \leftarrow \begin{cases} \mu \cdot \Delta & \text{if } \mu \cdot \Delta > \mu_{min} \\ 0 & \text{if } \mu \cdot \Delta < \mu_{min} \end{cases}$

DDP is a second-order algorithm similar to Newton's method [4, 7] and hence sometimes suffers from taking large steps towards the minimum despite small changes in the nominal trajectory. Backtracking line search helps to scale the open loop term of equation (16) using a parameter α such that $0 < \alpha \leq 1$.

$$\hat{\mathrm{u}}_i = \mathrm{u}_i + \alpha \mathbf{k}_i + \mathbf{K}_i(\hat{\mathrm{x}}_i - \mathrm{x}_i) \tag{50}$$

Backtracking line search is a scheme based on the *Armijo* condition. It involves starting with a relatively large estimate of the step size $\alpha = 1$ for movement along the search direction, and iteratively shrinking the step size (i.e., "backtracking") using the control parameters $\tau \in (0, 1)$ and $c \in (0, 1)$ until a decrease of the objective function is observed that adequately corresponds to the decrease that is expected, based on the local gradient of the objective function [2]. It is

roughly as follows:

Choose $\alpha_0 > 0, \tau \in (0,1), c \in (0,1)$
Set $\alpha \leftarrow \alpha_0$
While $f(\mathrm{x}_k + \alpha p_k) > f(\mathrm{x}_k) + c\alpha \nabla f_k^T p_k$
  $\alpha_k \leftarrow \tau\alpha$
End while
Return $\alpha_k = \alpha$

The advantage of this approach is that the step length found at each line search iterate is short enough to satisfy sufficient decrease but large enough to still allow the algorithm to make reasonable progress towards convergence. Therefore as per the new improved equations (50) and (49e), we can better estimate the reduction in the total cost (or value) as,

$$\Delta V(\alpha) = \alpha \sum_{i=0}^{N-1} \mathbf{k}_i^T Q_{\mathrm{u}_i} + \frac{\alpha^2}{2} \sum_{i=0}^{N-1} \mathbf{k}_i^T Q_{\mathrm{uu}_i} \mathbf{k}_i \qquad (51)$$

The step size α must be accepted by comparing the above estimate with the actual estimate using (52) such that $0 < c < z$ (this condition goes into the while loop of the above simple backtracking line search algorithm).

$$z = (V(\mathrm{x}, 0) - V(\hat{\mathrm{x}}, 0))/\Delta V(\alpha) \qquad (52)$$

V. Case Study and Results

Here, we demonstrate the implementation of the modified DDP algorithm. The convergence of the algorithm is stabilized using backtracking line-search and regularization which penalizes the cost with deviation from the old sequence (see (48)). The Modified DDP Algorithm 1 is implemented in Python and solved using JAX framework [13]: a python library that has the capability of doing automatic differentiation and just-in-time compilation. Additionally, we used a library called Diffrax [14]: JAX-based numerical differential equation solver library. In the implementation, a continuous system model is used and the transition function is obtained by integrating the continuous model using the *Dormand-Prince 5(4)* numerical differential equation solver ($diffeqsolve$) provided by Diffrax.

$$\dot{\mathbf{x}}_i = f(\mathbf{x}_i, \mathbf{u}_i)$$
$$\mathbf{x}_{i+1} = diffeqsolve(\dot{\mathbf{x}}_i, T)$$

where T is the sampling time between i^{th} and $(i+1)^{th}$ states. This summarizes the first forward loop with a given nominal control sequence.

Algorithm 1 Modified DDP

Inputs:
$\bar{\mathbf{U}}_0, \mathbf{x}_0$
Initialize:
$\mathbf{u}_{0\ldots N-1} \leftarrow \bar{\mathbf{U}}_0$
$\mathbf{x}_{i+1} \leftarrow f(\mathbf{x}_i, \mathbf{u}_i) \quad i = 0, ..., N-1$
$V(\mathrm{N}) \leftarrow \ell_f(\mathbf{x}_N)$
$V_{\mathbf{x}}(N) \leftarrow \frac{\partial}{\partial x}\ell_f(\mathbf{x}_N)$
$V_{\mathbf{xx}}(N) \leftarrow \frac{\partial^2}{\partial x^2}\ell_f(\mathbf{x}_N)$
repeat
  backward pass:
  for $i \leftarrow N-1$ **to** 0 **do**
    $Q_{\mathrm{x}_i}, Q_{\mathrm{u}_i}, Q_{\mathrm{xx}_i}, Q_{\mathrm{uu}_i}, Q_{\mathrm{ux}_i} \leftarrow$ eq. (14)
    $\mathbf{k}_i, \mathbf{K}_i, \tilde{Q}_{\mathrm{uu}_i}, \tilde{Q}_{\mathrm{ux}_i} \leftarrow$ eqs. (18), (48) and (49)
    if $\tilde{Q}_{\mathrm{uu}_i}$ **is not** positive-definite **then**
      Increase μ
      restart *backward pass.*
    end if
  end for
  Decrease μ
  $\alpha \leftarrow 1$
  $\hat{\mathbf{x}}_0 \leftarrow \mathbf{x}_0$
  forward pass:
  for $i \leftarrow 0$ **to** $N-1$ **do**
    $\hat{\mathrm{u}}_i \leftarrow \mathrm{u}_i + \alpha\mathbf{k}_i + \mathbf{K}_i(\hat{\mathrm{x}}_i - \mathrm{x}_i)$
    $\hat{\mathbf{x}}_{i+1} = f(\hat{\mathbf{x}}_i, \hat{\mathbf{u}}_i)$
  end for
  if $z > c > 0$ **then** accept changes
    $\mathbf{u}_{0\ldots N-1} \leftarrow \hat{\mathbf{u}}_{0\ldots N-1}$
    $\mathbf{x}_{0\ldots N-1} \leftarrow \hat{\mathbf{x}}_{0\ldots N-1}$
  else
    Decrease α: $\alpha \leftarrow \tau\alpha$
    restart *forward pass.*
  end if
until convergence

1. Inverted pendulum on a cart (Cart-Pole)

An inverted pendulum on a cart (cart-pole) is widely studied in control system research chiefly because it is a simple nonlinear dynamical system that is unstable without control at its inverted position. The Modified DDP algorithm is implemented to find an optimal trajectory that steers the cart to the target position (i.e., the pole in its inverted position). We assume that a point mass is placed at the end of the rod and the rod is massless. We also consider the system is con-

strained such that the cart can move horizontally as can be seen in Figure 1. Lastly, we assume no friction in the system.

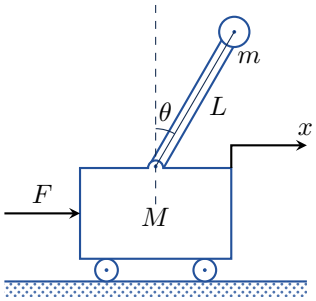

Figure 1: Schematic representation of the Inverted Pendulum on cart model

The following state-space ODE summarizes the modeling of our nonlinear cart-pole system which uses the system parameters from Table 1.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{F + m\sin(\theta)(L\dot{\theta}^2 - g\cos(\theta))}{M + m\sin(\theta)^2} \\ \dot{\theta} \\ \frac{-F\cos(\theta) - mL\dot{\theta}^2\sin(\theta)\cos(\theta) + (M+m)g\sin(\theta)}{L(M + m\sin(\theta)^2)} \end{bmatrix} \tag{53}$$

where the input is the horizontal force F.

$$\mathbf{u} = F \tag{54}$$

Table 1: Model Simulation Parameters

Parameter	Unit	Value
Cart mass (M)	kg	1
Point mass (m)	kg	0.1
Rod length (L)	m	0.5
Gravitiy constant (g)	m/s^2	9.81

2. Implementation

For initialization, we use randomized nominal control input sequence $\mathbf{U}_0$ that is sampled from a uniform distribution between [-1, 1], and an initial state $\mathbf{x}_0 = [0\ 0\ \pi\ 0]^T$(in the downward position). We chose quadratic running and final costs along with parameters from Table 2 for our final implementation. [1]

$$\ell(\mathrm{x}, \mathrm{u}) = \tfrac{1}{2}\mathrm{u}^T\mathrm{u}$$

$$\ell_f(\mathrm{x}_N) = \tfrac{1}{2}(\mathrm{x}_N - \mathrm{x}_{target})^T \begin{bmatrix} 10000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 10000 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} (\mathrm{x}_N - \mathrm{x}_{target})$$

where $\mathrm{x}_{target} = [2\ 0\ 0\ 0]^T$

[1] https://github.com/dumanah/ddp-jax.

Table 2: DDP Simulation Parameters

Parameter	Value
μ_{min}	10^{-6}
Δ_0	2
τ	0.8
N	100
T	0.02s

3. Results

The following results using the model (53) and paramaters given in Table 1 and 2 justify that x_{target} is reached and therefore we found the desired optimal trajectory $\mathbf{U}_0^*$.

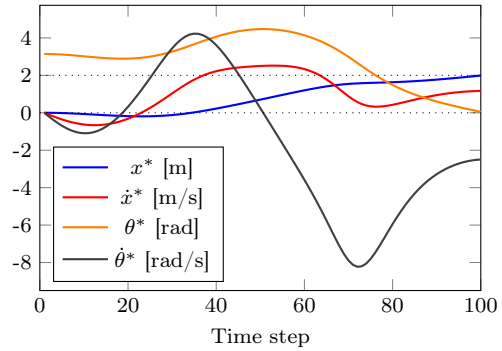

Figure 2: Optimal state trajectory

We observed that the final cost function needs to be penalized greatly compared to the running costs because it converges to the nearest local optimum and does not decrease the total cost any further. We tuned these penalization constants by trial and error. The results of optimized trajectory and the sequence of optimal force inputs applied to cart are shown in Figures 2 and 3 respectively. Thus, the total cost convergences to a near global optimum (see Figure 4)

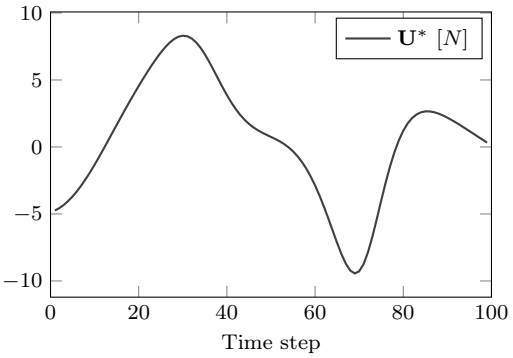

Figure 3: Optimal control sequence

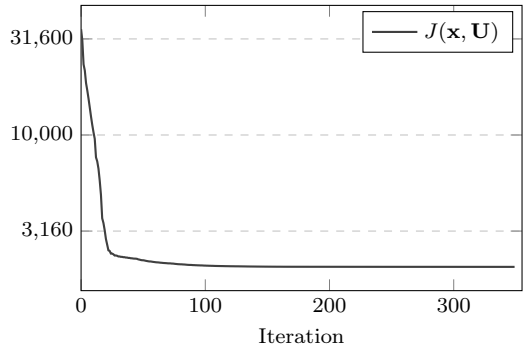

Figure 4: Cost over iteration

VI. Conclusion

In this report, we have presented the classical DDP method. In comparison, the steps taken by DDP correspond to Newton steps on the unconstrained optimal-control problem. DDP searches in the space of control trajectories $\mathbf{U} \in \mathbf{R}^{m \times N}$ and solves the m-dimensional problem N times and not a single problem of size mN. The difference is made stark when considering N Hessians of size $m \times m$ rather than a large $Nm \times Nm$ matrix. As with all second-order methods, in order to guarantee a decent direction, regularization must be used when the Hessian loses positive definiteness. Backtracking line-search addressed the problem of diverging costs. The regularization parameter μ and the step length α were adapted as per Levenberg-Marquardt heuristic. The obtained modified DDP algorithm was applied the problem of cart-pole. The applicability of these improvements and convergence has been demonstrated in a jupyter notebook. In the general framework of trajectory optimization, there are possibilities for improvements through improved trajectory optimization algorithms, the physics engine, and cost function design. With some additional refinements, we believe our methodology may apply to complex robots. How well it will work on physical robots performing different tasks still remains. Nevertheless, having the tools to apply trajectory optimization to complex robots will enable many robotic control tasks that are beyond the reach of existing methods for real-time feedback control.

References

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

[2] D. Jacobson and D. Mayne, *Differential Dynamic Programming*, ser. Modern analytic and computational methods in science and mathematics. American Elsevier Publishing Company, 1970.

[3] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, pp. 85–95, 1966.

[4] J. F. A. D. O. PANTOJA, "Differential dynamic programming and newton's method," *International Journal of Control*, vol. 47, no. 5, pp. 1539–1553, 1988.

[5] L.-Z. Liao and C. Shoemaker, "Advantages of Differential Dynamic Programming Over Newton's Method for Discrete-time Optimal Control Problems," Cornell University, Technical Report, 1993.

[6] J. F. A. d. O. Pantoja, "Algorithms for constrained optimization," 1984.

[7] D. M. Murray and S. J. Yakowitz, "Differential dynamic programming and Newton's method for discrete optimal control problems," *Journal of Optimization Theory and Applications*, vol. 43, no. 3, pp. 395–414, Jul. 1984.

[8] L.-Z. Liao and C. Shoemaker, "The proof of quadratic convergence of differential dynamic programming," Cornell University, Technical Report, 1990.

[9] ——, "Convergence in unconstrained discrete-time differential dynamic programming," *IEEE Transactions on Automatic Control*, vol. 36, no. 6, pp. 692–706, 1991.

[10] D. G. Luenberger and D. G. Luenberger, *Linear and nonlinear programming*, 2nd ed. Reading, Mass: Addison-Wesley, 1984.

[11] S. Yakowitz and B. Rutherford, "Computational aspects of discrete-time optimal control," *Applied Mathematics and Computation*, vol. 15, no. 1, pp. 29–45, 1984.

[12] E. Todorov and Weiwei Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proceedings of the 2005, American Control Conference, 2005.* Portland, OR, USA: IEEE, 2005, pp. 300–306.

[13] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[14] P. Kidger, "On Neural Differential Equations," Ph.D. dissertation, University of Oxford, 2021.