Controlling a Simple Pendulum via Differential Dynamic Programming

Daniel A. Hagen

November 13, 2018

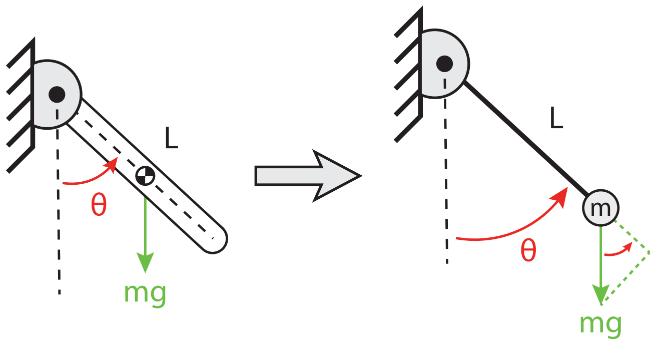

Figure 1: Physical pendulum (left) and the simple pendulum (right) that models it.

1 Simple Pendulum Dynamics

Consider a simple pendulum where a mass m is attached by a weightless rod of length L to a fixed point is controlled by torque input u (Fig. 1) with dynamics given by *Eq. 1*.

$$mL^2\ddot{\theta} = -mgL\sin(\theta) - b\dot{\theta} + u \tag{1}$$

Note that the pendulum is given a damping coefficient, b, but this may be set to zero if desired. If we define $\vec{x} := (x_1, x_2)^T := (\theta, \dot{\theta})^T$, then the state equations can given by *Eq. 2*.

$$\begin{cases} \dot{x_1} = f_1(\vec{x}, u) = x_2 & \text{(2a)} \\ \dot{x_2} = f_2(\vec{x}, u) = -\frac{g}{L}\sin(x_1) - \frac{b}{mL^2}x_2 + u & \text{(2b)} \end{cases}$$

Defining $f(\vec{x}, u) := (f_1(\vec{x}, u), f_2(\vec{x}, u))^T$, then the state dynamics can be rewritten as,

$$\dot{\vec{x}} = f(\vec{x}, u) \tag{3}$$

2 Defining the Cost Function

In order to perform any optimal control over this system, we must first define the cost function over which we wish to optimize. Assume that the controller to seeks to stabilize the pendulum at the unstable vertical position (i.e., $\vec{x}_p = (\pi, 0)$). The cost function can then be defined as,

$$V(x(t),t) = \min_u \left[\phi(\vec{x}(t_f),t_f) + \int_t^{t_f} \mathcal{L}(\vec{x}(\tau),u(\tau))d\tau \right] \tag{4}$$

where ϕ is the *terminal cost* and $\mathcal{L}$ is the *running cost.*

The terminal cost will penalizes large distances between the final state, $\vec{x}(t_f)$, and the desired state, $\vec{x}_p$. If we only are concerned with the pendulum reaching the inverted angle ($x_{p,1} = \pi$) then the terminal cost can be given by *Eq. 5a.* Alternatively, if the controller desires that the pendulum not only reach the inverted state at t_f, but that it be there at zero angular velocity (i.e., reach the unstable *equilibrium point*), then we can use the terminal cost function given by *Eq. 5b.* (Note: k_1 and k_2 are *positive* scaling factors).

$$\phi_1(\vec{x}(t_f),t_f) = \frac{k_1}{2}(x_1(t_f) - x_{p,1})^2 \tag{5a}$$

$$\phi_2(\vec{x}(t_f),t_f) = \frac{k_1}{2}(x_1(t_f) - x_{p,1})^2 + \frac{k_2}{2}(x_2(t_f))^2 \tag{5b}$$

The running cost represents the "cost of getting there" and can penalize things like total input energy (*Eq. 6a*), time not spent at the desired angle (*Eq. 6b*), or time not spent at the desired angular velocity (coincidentally, this also implies a penalty for total angular momentum, *Eq. 6c*). The total running cost can therefore be given any or all of the equations given in *Eq. 6* (Note: k_3, k_4, and k_5 are *positive* scaling factors).

$$\mathcal{L}_1(\vec{x}(t),u(t)) = \frac{k_3}{2}(u(t))^2 \tag{6a}$$

$$\mathcal{L}_2(\vec{x}(t),u(t)) = \frac{k_4}{2}(x_1(t) - x_{p,1})^2 \tag{6b}$$

$$\mathcal{L}_3(\vec{x}(t),u(t)) = \frac{k_5}{2}(x_2(t))^2 \tag{6c}$$

For the sake of the derivation, we will assume that the terminal and running costs are given by,

$$\begin{cases} \phi(\vec{x}(t_k),t_k) = \frac{k_1}{2}(x_1(t_f) - x_{p,1})^2 + \frac{k_2}{2}(x_2(t_f))^2 & \text{(7a)} \\ \mathcal{L}(\vec{x}(t),u(t)) = \frac{k_3}{2}(u(t))^2 + \frac{k_4}{2}(x_1(t) - x_{p,1})^2 + \frac{k_5}{2}(x_2(t))^2 & \text{(7b)} \end{cases}$$

3 Differential Dynamic Programming

In order to utilize differential dynamic programming, we assume a an initial action (input) trajectory. We then perform dynamic programming *around* the trajectory (*backward-pass*). We use this to find an update for our input, and then we create a new state trajectory from this new input (*forward-pass*). This process is repeated until convergence is achieved.

3.1 Linearized dynamics

For the dynamics given by *Eq. 3* and some prescribed reference state and action trajectories $(\bar{x}, \bar{u})$, the dynamics are then linearized as follows.

$$\begin{aligned} \frac{\mathrm{d}\vec{x}}{\mathrm{d}t} &= f(\vec{x},u) \\ &= f(\vec{x} + \bar{x} - \bar{x}, u + \bar{u} - \bar{u}) \\ &= f(\bar{x} + \delta\vec{x}, \bar{u} + \delta u) && \text{(where } \delta\vec{x} = \vec{x} - \bar{x} \text{ and } \delta u = u - \bar{u}) \\ &\approx f(\bar{x},\bar{u}) + f_{\vec{x}}\,\delta\vec{x} + f_u\,\delta u && \text{(where } f_{\vec{x}} \text{ and } f_u \text{ are the Jacobians of } f \text{ w.r.t. } \vec{x} \text{ and } u\text{, respectively.)} \\ &= \frac{\mathrm{d}\bar{x}}{\mathrm{d}t} + f_{\vec{x}}\,\delta\vec{x} + f_u\,\delta u \end{aligned}$$

Note that $f_{\vec{x}} \in \mathbb{R}^{2\times 2}$ and $f_u \in \mathbb{R}^{2\times 1}$. Therefore, moving $\mathrm{d}\bar{x}/\mathrm{d}t$ to the L.H.S. yields,

$$\begin{aligned}\frac{\mathrm{d}\vec{x}}{\mathrm{d}t} - \frac{\mathrm{d}\bar{x}}{\mathrm{d}t} &= f_{\vec{x}}\,\delta\vec{x} + f_u\,\delta u \\ \frac{\mathrm{d}}{\mathrm{d}t}(\delta\vec{x}) &= f_{\vec{x}}\,\delta\vec{x} + f_u\,\delta u \\ &= \begin{bmatrix} 0 & 1 \\ -\frac{g}{L}\cos(x_1) & -\frac{b}{mL^2}\end{bmatrix}\delta\vec{x} + \begin{bmatrix} 0 \\ \frac{1}{mL^2}\end{bmatrix}\delta u\end{aligned} \tag{8}$$

3.2 Discretized Linearized Dynamics

We can now discretize the linearized dynamics by utilizing a forward integration technique, like *forward Euler*.

$$\delta\vec{x}_{k+1} = \delta\vec{x}_k + \frac{\mathrm{d}}{\mathrm{d}t}(\delta\vec{x}_k)\cdot\mathrm{d}t$$

where $\mathrm{d}t = t_{k+1} - t_k$. Plugging in the dynamics yields,

$$\begin{aligned}\delta\vec{x}_{k+1} &= (I + f_{\vec{x}}\,\mathrm{d}t)\,\delta\vec{x}_k + f_u\,\delta u_k\,\mathrm{d}t \\ &= \Phi\,\delta\vec{x}_k + B\,\delta u_k\end{aligned} \tag{9}$$

Therefore we can define the discretized, linearized dynamics matrices as,

$$\begin{cases}\Phi(\bar{x}(t_k), \bar{u}(t_k)) := \begin{bmatrix} 1 & dt \\ -\frac{g}{L}\cos(\bar{x}_1(t_k))dt & 1 - \frac{b}{mL^2}dt\end{bmatrix} & \text{(10a)} \\ B(\bar{x}(t_k), \bar{u}(t_k)) := \begin{bmatrix} 0 \\ \frac{1}{mL^2}dt\end{bmatrix} & \text{(10b)}\end{cases}$$

3.3 Discretized Cost Function

Now that the dynamics have been discretized in time and linearized we can employ a similar approach to dynamic programming *along* the trajectory. But first we need to discretize the cost function in time as well. Consider the value function for a given state when t_f is some small timestep ($\mathrm{d}t$) away.

$$\begin{aligned}V(\vec{x}(t), t) &= \min_{u(t)}\left[\int_t^{t+\mathrm{d}t}\mathcal{L}(\vec{x}(s), u(s))\,\mathrm{d}s + V(\vec{x}(t+\mathrm{d}t), t+\mathrm{d}t)\right] \\ &\approx \min_{u(t)}\Big[\mathcal{L}(\vec{x}(t), u(t))\,\mathrm{d}t + V(\vec{x}(t+\mathrm{d}t), t+\mathrm{d}t)\Big]\end{aligned} \tag{11}$$

where $\ell(\vec{x}(t), u(t)) := \mathcal{L}(\vec{x}(t), u(t))\,\mathrm{d}t$ is the left Riemann rectangular approximation of the integral. This can be rewritten in discrete time as,

$$V(\vec{x}(t_k), t_k) \approx \min_{u(t_k)}\Big[\ell(\vec{x}(t_k), u(t_k)) + V(\vec{x}(t_{k+1}), t_{k+1})\Big] \tag{12}$$

3.4 Taylor Quadratic Expansion of Value Function

Let $V_{\vec{x}} \in \mathbb{R}^{2\times 1}$ be the gradient of V with respect to $\vec{x}$ and $V_u \in \mathbb{R}$ be the derivative of V w.r.t. u. Let $V_{\vec{x}\vec{x}} \in \mathbb{R}^{2\times 2}$ be the Hessian of V w.r.t. $\vec{x}$, $V_{uu} \in \mathbb{R}$ be the second derivative of V w.r.t. u, and $V_{\vec{x}u} = V_{u\vec{x}}^T \in \mathbb{R}^{2\times 1}$ be the gradient of V_u w.r.t $\vec{x}$.

Additionally, let $\ell_{\vec{x}} \in \mathbb{R}^{2\times 1}$ be the gradient of ℓ w.r.t. $\vec{x}$ and $\ell_u \in \mathbb{R}$ be derivative of ℓ with respect to u. Let $\ell_{\vec{x}\vec{x}} \in \mathbb{R}^{2\times 2}$ be the Hessian of ℓ w.r.t. $\vec{x}$, $\ell_{uu} \in \mathbb{R}$ be the second derivative of ℓ w.r.t. u, and $\ell_{\vec{x}u} = \ell_{u\vec{x}}^T \in \mathbb{R}^{2\times 1}$ be the gradient of ℓ_u w.r.t. $\vec{x}$. Therefore, we can expand the discrete value function as,

$$\begin{aligned}
V\left(\vec{x}(t_k), t_k\right) &\approx \min_{u(t_k)} \Big[\ell\left(\vec{x}(t_k), u(t_k)\right) + V\left(\vec{x}(t_{k+1}), t_{k+1}\right) \Big] \\
&= \min_{u(t_k)} \begin{bmatrix} \ell\left(\vec{x}(t_k) - \bar{x}(t_k) + \bar{x}(t_k), u(t_k) - \bar{u}(t_k) + \bar{u}(t_k)\right) \\ + V\left(\vec{x}(t_{k+1}) - \bar{x}(t_{k+1}) + \bar{x}(t_{k+1}), t_{k+1}\right) \end{bmatrix} \\
&= \min_{u(t_k)} \begin{bmatrix} \ell\left(\bar{x}(t_k) + \delta\vec{x}(t_k), \bar{u}(t_k) + \delta u(t_k)\right) \\ + V\left(\bar{x}(t_{k+1}) + \delta\vec{x}(t_{k+1}), t_{k+1}\right) \end{bmatrix} \\
&= \min_{u(t_k)} \begin{bmatrix} \ell\left(\bar{x}(t_k), \bar{u}(t_k)\right) + \begin{bmatrix} \ell_{\vec{x}}^T & \ell_u^T \end{bmatrix} \begin{bmatrix} \delta\vec{x}(t_k) \\ \delta u(t_k) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta\vec{x}(t_k)^T & \delta u(t_k)^T \end{bmatrix} \begin{bmatrix} \ell\vec{x}\vec{x} & \ell\vec{x}u \\ \ell u\vec{x} & \ell uu \end{bmatrix} \begin{bmatrix} \delta\vec{x}(t_k) \\ \delta u(t_k) \end{bmatrix} \\ + V\left(\bar{x}(t_{k+1}), t_{k+1}\right) + V_{\vec{x}}\left(\bar{x}(t_{k+1}), t_{k+1}\right)^T \delta\vec{x}(t_{k+1}) \\ + \frac{1}{2}\delta\vec{x}(t_{k+1})^T V_{\vec{x}\vec{x}}\left(\bar{x}(t_{k+1}), t_{k+1}\right) \delta\vec{x}(t_{k+1}) \end{bmatrix} \\
&= \min_{\delta u(t_k)} \begin{bmatrix} \ell\left(\bar{x}(t_k), \bar{u}(t_k)\right) + \begin{bmatrix} \ell_{\vec{x}}^T & \ell_u^T \end{bmatrix} \begin{bmatrix} \delta\vec{x}(t_k) \\ \delta u(t_k) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta\vec{x}(t_k)^T & \delta u(t_k)^T \end{bmatrix} \begin{bmatrix} \ell\vec{x}\vec{x} & \ell\vec{x}u \\ \ell u\vec{x} & \ell uu \end{bmatrix} \begin{bmatrix} \delta\vec{x}(t_k) \\ \delta u(t_k) \end{bmatrix} \\ + V\left(\bar{x}(t_{k+1}), t_{k+1}\right) + V_{\vec{x}}\left(\bar{x}(t_{k+1}), t_{k+1}\right)^T \left(\Phi\, \delta\vec{x}(t_k) + B\, \delta u(t_k)\right) \\ + \frac{1}{2}\left(\left(\Phi\, \delta\vec{x}(t_k) + B\, \delta u(t_k)\right)\right)^T V_{\vec{x}\vec{x}}\left(\bar{x}(t_{k+1}), t_{k+1}\right) \left(\Phi\, \delta\vec{x}(t_k) + B\, \delta u(t_k)\right) \end{bmatrix}
\end{aligned}$$

We change the variable that we minimize over from $u(t_k)$ to $\delta u(t_k)$ because, from the definition, $\delta u(t_k) = u(t_k) - \bar{u}(t_k)$ where $\bar{u}(t_k)$ is constant for a given instant. Therefore minimizing over $u(t_k)$ is equivalent to minimizing over $\delta u(t_k)$. We could solve this minimization by then taking the gradient of the cost function with respect to $\delta u(t_k)$, but this is rather tedious. Instead let's define a new function Q as,

$$Q(\vec{x}(t_k), u(t_k)) := \ell(\vec{x}(t_k), u(t_k)) + V(\vec{x}(t_{k+1}), u(t_{k+1})) \tag{13}$$

Such that,

$$\begin{aligned}
Q(\vec{x}(t_k), u(t_k)) &= Q\left(\vec{x}(t_k) - \bar{x}(t_k) + \bar{x}(t_k), u(t_k) - \bar{u}(t_k) + \bar{u}(t_k)\right) \\
&= Q\left(\bar{x}(t_k) + \delta\vec{x}(t_k), \bar{u}(t_k) + \delta u(t_k)\right) \\
&= Q(\bar{x}(t_k), \bar{u}(t_k)) + Q_{\vec{x}}^T \delta\vec{x}(t_k) + Q_u^T \delta u(t_k) \\
&\quad + \frac{1}{2} \begin{bmatrix} \delta\vec{x}(t_k)^T & \delta u(t_k)^T \end{bmatrix} \begin{bmatrix} Q_{\vec{x}\vec{x}} & Q_{\vec{x}u} \\ Q_{u\vec{x}} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta\vec{x}(t_k) \\ \delta u(t_k) \end{bmatrix}
\end{aligned} \tag{14}$$

where $Q_{\vec{x}} \in \mathbb{R}^{2\times 1}$ is the gradient of Q with respect to $\vec{x}$, $Q_u \in \mathbb{R}$ is the derivative of Q w.r.t. u, $Q_{\vec{x}\vec{x}} \in \mathbb{R}^{2\times 2}$ is the Hessian of Q w.r.t. $\vec{x}$, $Q_{uu} \in \mathbb{R}$ is the second derivative of Q w.r.t. u, and $Q_{\vec{x}u} = Q_{u\vec{x}}^T \in \mathbb{R}^{2\times 1}$ is the gradient of Q_u w.r.t $\vec{x}$. Therefore it is possible to define these values of Q in terms of our previous cost function in terms of ℓ and V.

$$\left\{ \begin{aligned}
Q(\bar{x}(t_k), \bar{u}(t_k)) &= \ell(\bar{x}(t_k), \bar{u}(t_k)) + V(\bar{x}(t_{k+1}), t_{k+1}) && \text{(15a)} \\
Q_{\vec{x}}(\bar{x}(t_k), \bar{u}(t_k)) &= \ell_{\vec{x}}(\bar{x}(t_k), \bar{u}(t_k)) + \Phi(\bar{x}(t_k), \bar{u}(t_k))^T V_{\vec{x}}(\bar{x}(t_{k+1}), t_{k+1}) && \text{(15b)} \\
Q_u(\bar{x}(t_k), \bar{u}(t_k)) &= \ell_u(\bar{x}(t_k), \bar{u}(t_k)) + B(\bar{x}(t_k), \bar{u}(t_k))^T V_{\vec{x}}(\bar{x}(t_{k+1}), t_{k+1}) && \text{(15c)} \\
Q_{\vec{x}u}(\bar{x}(t_k), \bar{u}(t_k)) &= \ell_{\vec{x}u}(\bar{x}(t_k), \bar{u}(t_k)) + \Phi(\bar{x}(t_k), \bar{u}(t_k))^T V_{\vec{x}\vec{x}}(\bar{x}(t_{k+1}), t_{k+1}) B(\bar{x}(t_k), \bar{u}(t_k)) && \text{(15d)} \\
Q_{u\vec{x}}(\bar{x}(t_k), \bar{u}(t_k)) &= \ell_{u\vec{x}}(\bar{x}(t_k), \bar{u}(t_k)) + B(\bar{x}(t_k), \bar{u}(t_k))^T V_{\vec{x}\vec{x}}(\bar{x}(t_{k+1}), t_{k+1}) \Phi(\bar{x}(t_k), \bar{u}(t_k)) && \text{(15e)} \\
Q_{\vec{x}\vec{x}}(\bar{x}(t_k), \bar{u}(t_k)) &= \ell_{\vec{x}\vec{x}}(\bar{x}(t_k), \bar{u}(t_k)) + \Phi(\bar{x}(t_k), \bar{u}(t_k))^T V_{\vec{x}\vec{x}}(\bar{x}(t_{k+1}), t_{k+1}) \Phi(\bar{x}(t_k), \bar{u}(t_k)) && \text{(15f)} \\
Q_{uu}(\bar{x}(t_k), \bar{u}(t_k)) &= \ell_{uu}(\bar{x}(t_k), \bar{u}(t_k)) + B(\bar{x}(t_k), \bar{u}(t_k))^T V_{\vec{x}\vec{x}}(\bar{x}(t_{k+1}), t_{k+1}) B(\bar{x}(t_k), \bar{u}(t_k)) && \text{(15g)}
\end{aligned} \right.$$

We can now solve the equivalent problem,

$$\min_{\delta u(t_k)} Q\left(\vec{x}(t_k), u(t_k)\right)$$

3.5 Minimization of the Cost Function

To find the minimum of the cost function we take the gradient with respect to $\delta u(t_k)$ and find where it is equal to zero.

$$\begin{aligned}
\nabla_{\delta u(t_k)} & \begin{bmatrix} Q(\bar{x}(t_k), \bar{u}(t_k)) + Q_{\vec{x}}^T \delta\vec{x}(t_k) + Q_u^T \delta u(t_k) \\ \qquad + \frac{1}{2} \begin{bmatrix} \delta\vec{x}(t_k)^T & \delta u(t_k)^T \end{bmatrix} \begin{bmatrix} Q_{\vec{x}\vec{x}} & Q_{\vec{x}u} \\ Q_{u\vec{x}} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta\vec{x}(t_k) \\ \delta u(t_k) \end{bmatrix} \end{bmatrix} \\
&= Q_u + Q_{u\vec{x}}\,\delta\vec{x}(t_k) + Q_{uu}\,\delta u(t_k) = 0
\end{aligned}$$

Therefore the optimal control for time t_k is given by,

$$\begin{aligned}
\delta u(t_k) &= -Q_{uu}^{-1}\Big(Q_u + Q_{u\vec{x}}\,\delta\vec{x}(t_k)\Big) \\
&= -Q_{uu}^{-1}Q_{u\vec{x}}\,\delta\vec{x}(t_k) - Q_{uu}^{-1}Q_u
\end{aligned} \tag{16}$$

3.6 Solving for the values of V, $V_{\vec{x}}$, and $V_{\vec{x}\vec{x}}$

Substituting the values from the optimal input (δu^*) given by DDP back into the value function we find,

$$\begin{aligned}
V(\vec{x}(t_k), t_k) &\approx \begin{bmatrix} Q(\bar{x}(t_k), \bar{u}(t_k)) + Q_{\vec{x}}^T \delta\vec{x}(t_k) + Q_u^T \delta u^*(t_k) \\ \qquad + \frac{1}{2} \begin{bmatrix} \delta\vec{x}(t_k)^T & \delta u^*(t_k)^T \end{bmatrix} \begin{bmatrix} Q_{\vec{x}\vec{x}} & Q_{\vec{x}u} \\ Q_{u\vec{x}} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta\vec{x}(t_k) \\ \delta u^*(t_k) \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} Q(\bar{x}(t_k), \bar{u}(t_k)) + Q_{\vec{x}}^T \delta\vec{x}(t_k) + Q_u^T \Big(-Q_{uu}^{-1}Q_u - Q_{uu}^{-1}Q_{u\vec{x}}\,\delta\vec{x}(t_k)\Big) \\ + \frac{1}{2} \begin{bmatrix} \delta\vec{x}(t_k)^T \\ \Big(-Q_{uu}^{-1}Q_u - Q_{uu}^{-1}Q_{u\vec{x}}\,\delta\vec{x}(t_k)\Big)^T \end{bmatrix}^T \begin{bmatrix} Q_{\vec{x}\vec{x}} & Q_{\vec{x}u} \\ Q_{u\vec{x}} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta\vec{x}(t_k) \\ \Big(-Q_{uu}^{-1}Q_u - Q_{uu}^{-1}Q_{u\vec{x}}\,\delta\vec{x}(t_k)\Big) \end{bmatrix} \end{bmatrix} \\
&= Q(\bar{x}(t_k), \bar{u}(t_k)) + Q_{\vec{x}}^T \delta\vec{x}(t_k) - Q_u^T Q_{uu}^{-1} Q_u - Q_u^T Q_{uu}^{-1} Q_{u\vec{x}}\,\delta\vec{x}(t_k) \\
&\quad + \frac{1}{2}\Bigg(\delta\vec{x}(t_k)^T Q_{\vec{x}\vec{x}}\,\delta\vec{x}(t_k) \\
&\qquad + \Big(-Q_u^T Q_{uu}^{-1} - \delta\vec{x}(t_k)^T Q_{\vec{x}u} Q_{uu}^{-1}\Big) Q_{u\vec{x}}\,\delta\vec{x}(t_k) \\
&\qquad + \delta\vec{x}(t_k)^T Q_{\vec{x}u}\Big(-Q_{uu}^{-1}Q_u - Q_{uu}^{-1}Q_{u\vec{x}}\,\delta\vec{x}(t_k)\Big) \\
&\qquad + \Big(-Q_u^T Q_{uu}^{-1} - \delta\vec{x}(t_k)^T Q_{\vec{x}u} Q_{uu}^{-1}\Big) Q_{uu} \Big(-Q_{uu}^{-1}Q_u - Q_{uu}^{-1}Q_{u\vec{x}}\,\delta\vec{x}(t_k)\Big)\Bigg) \\
&= \Big[Q(\bar{x}(t_k), \bar{u}(t_k)) - \frac{1}{2} Q_u^T Q_{uu}^{-1} Q_u \Big] \\
&\qquad + \Big[Q_{\vec{x}}^T - Q_u^T Q_{uu}^{-1} Q_{u\vec{x}} \Big]\,\delta\vec{x}(t_k) \\
&\qquad + \frac{1}{2}\delta\vec{x}(t_k)^T \Big[Q_{\vec{x}\vec{x}} - Q_{\vec{x}u} Q_{uu}^{-1} Q_{u\vec{x}} \Big]\,\delta\vec{x}(t_k) \\
&\approx V(\bar{x}(t_k), t_k) + V_{\vec{x}}(\bar{x}(t_k), t_k)\,\delta\vec{x}(t_k) + \frac{1}{2}\delta\vec{x}(t_k)^T V_{\vec{x}\vec{x}}(\bar{x}(t_k), t_k)\,\delta\vec{x}(t_k)
\end{aligned}$$

Therefore, we can set the coefficients of $\delta\vec{x}$ equal to each other obtain a close, discrete approximation of the value function.

$$\begin{cases} V(\bar{x}(t_k), t_k) \approx Q(\bar{x}(t_k), \bar{u}(t_k)) - \frac{1}{2} Q_u^T Q_{uu}^{-1} Q_u & \text{(17a)} \\ \qquad = \ell(\bar{x}(t_k), \bar{u}(t_k)) + V(\bar{x}(t_{k+1}), t_{k+1}) - \frac{1}{2} Q_u^T Q_{uu}^{-1} Q_u & \text{(17b)} \\ V_{\vec{x}}(\bar{x}(t_k), t_k) \approx Q_{\vec{x}} - Q_{\vec{x}u} Q_{uu}^{-1} Q_u & \text{(17c)} \\ V_{\vec{x}\vec{x}}(\bar{x}(t_k), t_k) \approx Q_{\vec{x}\vec{x}} - Q_{\vec{x}u} Q_{uu}^{-1} Q_{u\vec{x}} & \text{(17d)} \end{cases}$$

4 Computational Work

In order to perform the *backward pass*, we first start with a discrete input array $\bar{u}$. We then calculate the discrete state array, $\bar{x}$, using a forward integration technique like *Forward Euler*. Now that the values of both $\bar{x}$ and $\bar{u}$ are known for a given trial, we can solve for all values of Φ and B (linearized dynamics) as well as all of the values and derivatives of ℓ (discretized integral approximation of the running cost). The equations for Φ and B are given by *Eq. 10*, but we define ℓ and its derivatives as,

$$\begin{cases} \ell(\bar{x}(t_k), \bar{u}(t_k)) := \left(\frac{k_3}{2}(\bar{u}(t_k))^2 + \frac{k_4}{2}(\bar{x}_1(t_k) - x_{p,1})^2 + \frac{k_5}{2}(\bar{x}_2(t_k))^2\right) dt \in \mathbb{R} & \text{(18a)} \\ \ell_{\vec{x}}(\bar{x}(t_k), \bar{u}(t_k)) := \begin{pmatrix} k_4(\bar{x}_1(t_k) - x_{p,1})dt \\ k_5 \bar{x}_2(t_k) dt \end{pmatrix} \in \mathbb{R}^{2\times 1} & \text{(18b)} \\ \ell_u(\bar{x}(t_k), \bar{u}(t_k)) := k_3 \bar{u}(t_k) dt \in \mathbb{R} & \text{(18c)} \\ \ell_{\vec{x}u}(\bar{x}(t_k), \bar{u}(t_k)) := \begin{pmatrix} 0 \\ 0 \end{pmatrix} \in \mathbb{R}^{2\times 1} & \text{(18d)} \\ \ell_{u\vec{x}}(\bar{x}(t_k), \bar{u}(t_k)) := \begin{pmatrix} 0 & 0 \end{pmatrix} \in \mathbb{R}^{1\times 2} & \text{(18e)} \\ \ell_{\vec{x}\vec{x}}(\bar{x}(t_k), \bar{u}(t_k)) := \begin{pmatrix} k_4 dt & 0 \\ 0 & k_5 dt \end{pmatrix} \in \mathbb{R}^{2\times 2} & \text{(18f)} \\ \ell_{uu}(\bar{x}(t_k), \bar{u}(t_k)) := k_3 dt \in \mathbb{R} & \text{(18g)} \end{cases}$$

4.1 Backward Pass

To perform the backwards pass, we start at the final timestep (t_f) and work backwards in time. At t_f we only have the terminal cost as there are no more steps left. Therefore we can initialize $V(\bar{x}(t_f), t_f)$, $V_{\vec{x}}(\bar{x}(t_f), t_f)$, and $V_{\vec{x}\vec{x}}(\bar{x}(t_f), t_f)$ as,

$$\begin{cases} V(\bar{x}(t_f), t_f) = \frac{k_1}{2}(\bar{x}_1(t_f) - x_{p,1})^2 + \frac{k_2}{2}(\bar{x}_2(t_f))^2 \in \mathbb{R} & \text{(19a)} \\ V_{\vec{x}}(\bar{x}(t_f), t_f) = \begin{pmatrix} k_1(\bar{x}_1(t_f) - x_{p,1}) \\ k_2 \bar{x}_2 \end{pmatrix} \in \mathbb{R}^{2\times 1} & \text{(19b)} \\ V_{\vec{x}\vec{x}}(\bar{x}(t_f), t_f) = \begin{bmatrix} k1 & 0 \\ 0 & k2 \end{bmatrix} \in \mathbb{R}^{2\times 2} & \text{(19c)} \end{cases}$$

Utilizing *Eqs. 15* and *18*, we can the solve for the previous timestep's values for V, $V_{\vec{x}}$ and $V_{\vec{x}\vec{x}}$ using *Eq. 17*. This backward iterative process is referred to as the *backward pass*.

4.2 Forward Pass

To then perform the forward pass, we must solve for the optimal values of $\delta u^*(t_k)$. In order to do this, we utilize the discrete linear dynamics given by *Eq. 10* as well as the equation for optimal $\delta u^*(t_k)$ given by *Eq. 16*. If $\delta x(t_o) = (0,0)^T$, then $\delta u^*(t_o) = -Q_{uu}^{-1} Q_u$ and the remaining values for $\delta x(t_k)$ and $\delta u^*(t_k)$ can be found from,

$$\begin{cases} \delta u^*(t_k) = -Q_{uu}^{-1}(\bar{x}(t_k), \bar{u}(t_k)) \Big(Q_{u\vec{x}}(\bar{x}(t_k), \bar{u}(t_k))\, \delta \vec{x}(t_k) + Q_u(\bar{x}(t_k), \bar{u}(t_k)) \Big) & (20a) \\ \delta x(t_{k+1}) = \Phi(\bar{x}(t_k), \bar{u}(t_k))\, \delta x(t_k) + B(\bar{x}(t_k), \bar{u}(t_k))\, \delta u^*(t_k) & (20b) \end{cases}$$

Performing this over all of the timesteps is referred to as the *forward pass*. Once the values of δu^* are known for every timestep, a new $\bar{u}$ can be defined as $\bar{u}_{\text{new}} = \bar{u} + \delta u^*$. The DDP process is then repeated for this new input until the cost function converges.

5 Results

Running DDP for this system and the cost function defined in *Eq. 7* produced the following results.

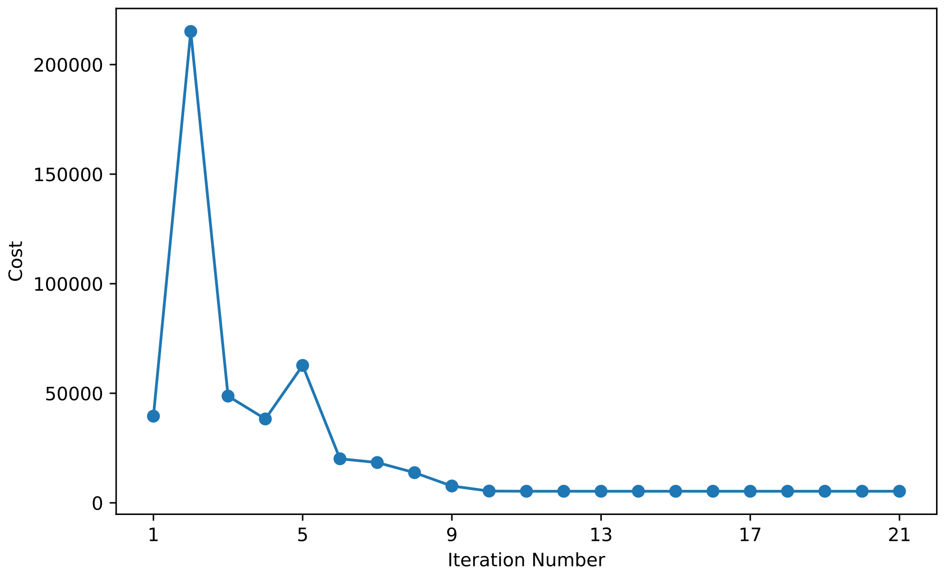

Figure 2: Cost versus iteration number.

To see an animation of the last iteration, Click Here!