iLQR Tutorial

Brian Jackson, Taylor Howell

Robotic Exploration Lab, Stanford University

July 2, 2019

1 LQR Derivation

1.1 Discrete Case (Regulator)

$$x_{k+1} = A_k x_t + B_k u_k$$
$$J(x_0, U) = \frac{1}{2} x_N^T Q_f x_N + \frac{1}{2} \sum_{k=1}^{N-1} x_k^T Q_k x_k + u_k^T R_k u_k$$

Using the Principle of Optimality and Pontryagin's Minimum Principle, we can compute the optimal cost-to-go $V_k(x_k) = \frac{1}{2} x_k S_k x_k$ by working backwards from the boundary condition/terminal cost.

$$\begin{aligned}
V_N(x_N) &= \frac{1}{2} x_N^T Q_f x_N = \frac{1}{2} x_N^T S_N x_N \\
V_{N-1}(x_{N-1}) &= \min_{u_{N-1}} \frac{1}{2} x_{N-1}^T Q_{N-1} x_{N-1} + \frac{1}{2} u_{N-1}^T R_{N-1} u_{N-1} + V_N(x_N) \\
&= \min_{u_{N-1}} \frac{1}{2} x_{N-1}^T Q_{N-1} x_{N-1} + \frac{1}{2} u_{N-1}^T R_{N-1} u_{N-1} + V_N(A_{N-1} x_{N-1} + B_{N-1} u_{N-1}) \\
&= \min_{u_{N-1}} \frac{1}{2} x_{N-1}^T Q_{N-1} x_{N-1} + \frac{1}{2} u_{N-1}^T R_{N-1} u_{N-1} + \frac{1}{2} \big(A_{N-1} x_{N-1} + B_{N-1} u_{N-1}\big)^T S_N \big(A_{N-1} x_{N-1} + B_{N-1} u_{N-1}\big) \\
&= \min_{u_{N-1}} \frac{1}{2} x_{N-1}^T Q_{N-1} x_{N-1} + \frac{1}{2} u_{N-1}^T R_{N-1} u_{N-1} \\
&\quad + \frac{1}{2} \big(x_{N-1}^T A_{N-1}^T S_N A_{N-1} x_{N-1} + u_{N-1}^T B_{N-1}^T S_N B_{N-1} u_{N-1} + x_{N-1}^T A_{N-1}^T S_N B_{N-1} u_{N-1} + u_{N-1}^T B_{N-1}^T S_N A_{N-1}
\end{aligned} \tag{1}$$

Using Pontryagin's Minimum Principle we can solve for the optimal control for a single time step u_k, instead of the entire control sequence U.

$$\frac{\partial V}{\partial u} = R_{N-1} u_{N-1} + B_{N-1}^T S_N B_{N-1} u_{N-1} + B_{N-1}^T S_N A_{N-1} x_{N-1} = 0 \tag{2}$$

$$\begin{aligned}
u_{N-1}^* &= -(R_{N-1} + B_{N-1}^T S_N B_{N-1})^{-1} B_{N-1}^T S_N A_{N-1} x_{N-1} \\
&\equiv K_{N-1} x_{N-1}
\end{aligned}$$

The optimal control can be substituted back into the above equation to compute the optimal cost-to-go

$$\begin{aligned}
V_{N-1}(x_{N-1}) &= \frac{1}{2} x_{N-1}^T Q_{N-1} x_{N-1} \frac{1}{2} x_{N-1}^T K_{N-1}^T R_{N-1} K_{N-1} x_{N-1} \\
&\quad + \frac{1}{2} \big(A_{N-1} x_{N-1} + B_{N-1} K_{N-1} x_{N-1}\big)^T P_N \big(A_{N-1} x_{N-1} + B_{N-1} K_{N-1} x_{N-1}\big) \\
&= \frac{1}{2} x_{N-1}^T \Big(Q_{N-1} + K_{N-1}^T R_{N-1} K_{N-1} + (A_{N-1} + B_{N-1} K_{N-1})^{-1} S_N (A_{N-1} + B_{N-1} K_{N-1})\Big) x_{N-1} \\
&\equiv \frac{1}{2} x_{N-1} S_{N-1} x_{N-1}
\end{aligned} \tag{3}$$

2 iLQR Derivation

We first start the derivation by setting up the problem by defining the dynamics, cost function, and cost-to-go variable.

2.1 Discrete Dynamics

The dynamics are typically provided as differential equations. In order to apply iLQR, the dynamics must be discretized with an appropriate quadrature rule (more details to follow in Section ??). Here we assume general, non-linear, discretized dynamics:

$$x_{k+1} = f(x_k, u_k) \tag{4}$$

which we approximate with a first-order Taylor-series expansion about nominal trajectories $X = \{x_0, ...x_N\}, U = \{u_0, ..., u_{N-1}\}$:

$$\begin{aligned} x_{k+1} + \delta x_{k+1} = f(x_k + \delta x_k, u_k + \delta u_k) &\approx f(x_k, u_k) + \frac{\partial f}{\partial x}\Big|_{x_k, u_k}(x - x_k) + \frac{\partial f}{\partial u}\Big|_{x_k, u_k}(u - u_k) \\ \delta x_{k+1} &= A(x_k, u_k)\delta x_k + B(x_k, u_k)\delta u_k \qquad (5) \\ & \qquad (6) \end{aligned}$$

where $A \equiv \frac{\partial f}{\partial x}$ and $B \equiv \frac{\partial f}{\partial u}$.

2.2 Cost Function

Most cost functions used iLQR are linear-quadratic cost functions. However, if the cost function is not linear-quadratic, a second-order Taylor Series Expansion can be used to linearize the dynamics into a form common for optimal control problems (note that constant terms are intentionally dropped as they have no impact on the minimization):

$$\begin{aligned} J(x_0, U) &= \ell_f(x_N) + \sum_{k=1}^{N-1} \ell(x_k, u_k) \\ &\approx \frac{1}{2} x_N^T Q_N x_N + q_N^T x_N + \sum_{k=1}^{N-1} \frac{1}{2} x_k^T Q_k x_k + \frac{1}{2} u_k^T R_k u_k + \frac{1}{2} x_k^T H_k u_k + \frac{1}{2} u_k^T H_k^T x_k + q_k^T x_k + r_k^T u_k \end{aligned} \tag{7}$$

For the we define a few variables of convenience ($\rightarrow$ indicates equivalence for quadratic cost function):

$$\begin{aligned} \ell_x &\equiv \frac{\partial \ell}{\partial x}\Big|_{x_k, u_k} \rightarrow Q_k x_k + q_k \\ \ell_u &\equiv \frac{\partial \ell}{\partial u}\Big|_{x_k, u_k} \rightarrow R_k u_k + r_k \\ \ell_{xx} &\equiv \frac{\partial^2 \ell}{\partial x^2}\Big|_{x_k, u_k} \rightarrow Q_k \\ \ell_{uu} &\equiv \frac{\partial^2 \ell}{\partial u^2}\Big|_{x_k, u_k} \rightarrow R_k \\ \ell_{xu} &\equiv \frac{\partial^2 \ell}{\partial x \partial u}\Big|_{x_k, u_k} \rightarrow H_k \\ \ell_{ux} &\equiv \frac{\partial^2 \ell}{\partial u \partial x}\Big|_{x_k, u_k} \rightarrow H_k^T \end{aligned} \tag{8}$$

With a given cost function, we can apply Bellman's Principle of Optimality to define the optimal cost-to-go $V_k(x)$ by the recurrence relation:

$$\begin{aligned} V_N &= \ell_f(x_N) \\ V_k &= \min_u \{\ell(x_k, u_k) + V_{k+1}(f(x_k, u_k))\} \\ V_k &= \min_u Q_k(x_k, u_k) \end{aligned} \tag{9}$$

We approximate the cost-to-go function as locally quadtratic near the nominal trajectory:

$$V_k + \delta V_k = V_k(x_k + \delta x_k) \approx V(x_k) + \frac{\partial V}{\partial x}\Big|_{x_k}(x - x_k) + \frac{1}{2}(x - x_k)^T \frac{\partial^2 V}{\partial x^2}\Big|_{x_k}(x - x_k)$$

$$\delta V_k(x_k) = s_k^T x_k + \frac{1}{2} x_k^T S_k x_k \tag{10}$$

Similarly:

$$\begin{aligned} Q_k + \delta Q_k &= Q(x_k + \delta x, u_k + \delta u) \\ &\approx Q(x_k, u_k) + \frac{\partial Q}{\partial x}\Big|_{x_k,u_k}(x - x_k) + \frac{\partial Q}{\partial u}\Big|_{x_k,u_k}(u - u_k) \\ &\quad + \frac{1}{2}(x - x_k)^T \frac{\partial^2 Q}{\partial x^2}\Big|_{x_k,u_k}(x - x_k) + \frac{1}{2}(u - u_k)^T \frac{\partial^2 Q}{\partial u^2}\Big|_{x_k,u_k}(u - u_k) \\ &\quad + \frac{1}{2}(u - u_k)^T \frac{\partial^2 Q}{\partial u \partial x}\Big|_{x_k,u_k}(x - x_k) + \frac{1}{2}(x - x_k)^T \frac{\partial^2 Q}{\partial x \partial u}\Big|_{x_k,u_k}(u - u_k) \end{aligned}$$

$$\delta Q_k(x_k, u_k) = \frac{1}{2}\begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{xx}\end{bmatrix}\begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix} + \begin{bmatrix}Q_x \\ Q_u\end{bmatrix}^T \begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix}$$

We define the following variables using matrix calculus :

$$\begin{aligned} Q_x &= \ell_x + s_{k+1} A_k \\ Q_u &= \ell_u + s_{k+1} B_k \\ Q_{xx} &= \ell_{xx} + A_k^T S_{k+1} A_k \\ Q_{uu} &= \ell_{uu} + B_k^T S_{k+1} B_k \\ Q_{ux} &= \ell_{ux} + B_k^T S_{k+1} A_k \end{aligned} \tag{11}$$

and using the fact the $Q_{ux} = Q_{xu}^T$.
which gives us all the values needed to calculate the next step. We can show this by combining Equations 11 and 12:

2.3 Cost-to-go at terminal state

By following a dynamic-programming approach, we can solve the tail problem for $V_N(x)$ for a problem with N time steps. In order to solve the detail we define δV as the deviation from the optimal value: with s_N and S_N defined as follows, given the cost function in Equation 7:

$$\begin{aligned} s_N &\equiv \frac{\partial V}{\partial x}\Big|_{x_N} \\ &= \frac{\partial}{\partial x}\big(\frac{1}{2}(x - x_f)^T Q_f (x - x_f)\big)\Big|_{x_N} \\ &= \frac{\partial}{\partial x}\big(\frac{1}{2}x^T Q_f x - x_f^T Q_f x + \frac{1}{2}x_f^T Q_f x_f\big)\Big|_{x_N} \\ &= Q_f x_N - Q_f x_f \qquad (Q_f = Q_f^T) \\ &= Q_f(x_N - x_f) \\ S_N &\equiv \frac{\partial^2 V}{\partial x^2}\Big|_{x_N} \\ &= \frac{\partial}{\partial x}\big(Q_f(x - x_f)\big)\Big|_{x_N} \\ &= Q_f \end{aligned}$$

2.4 Solving the Dynamic Programming Problem

After solving the tail sub-problem, we can then apply the principle of optimality and define the process for solving for the kth time step given the values at the $k+1$th time step.

$$\begin{aligned} V_k &= \min_{u_k}\{\ell(x_k, u_k) + V_{k+1}(f(x_k, u_k))\} \\ &= \min_{u_k}\{Q_k(x_k, u_k)\} \end{aligned}$$

$$\begin{aligned} \delta V &= \min_{\delta u}\{\delta Q(x, u)\} \\ &= \min_{\delta u}\{Q_x\delta x + Q_u\delta u + \frac{1}{2}\delta x^T Q_{xx}\delta x + \frac{1}{2}\delta u^T Q_{uu}\delta u + \frac{1}{2}\delta x^T Q_{xu}\delta u + \frac{1}{2}\delta u^T Q_{ux}\delta x\} \end{aligned}$$

$$\frac{\partial \delta Q}{\partial \delta u} = Q_u + \frac{1}{2}Q_{ux}\delta x + \frac{1}{2}Q_{xu}^T\delta x + Q_{uu}\delta u = 0$$

$$\begin{aligned} \rightarrow \delta u^* &= -Q_{uu}^{-1}(Q_{ux}\delta x_k + Qu) \\ &= K\delta x + d \end{aligned}$$

So

$$\begin{aligned} d_k &= -Q_{uu}^{-1}Q_u \\ K_k &= -Q_{uu}^{-1}Q_{ux} \end{aligned}$$

After calculating the optimal control as a function of the next time step we can plug it back into Equation **??**.

$$\delta Q_k(x_k, u_k) = \frac{1}{2}\begin{bmatrix} \delta x_k \\ K\delta x + d \end{bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{xx} \end{bmatrix} \begin{bmatrix} \delta x_k \\ K\delta x + d \end{bmatrix} + \begin{bmatrix} Q_x \\ Q_u \end{bmatrix}^T \begin{bmatrix} \delta x_k \\ K\delta x + d \end{bmatrix}$$

By equating the result with Equation **??** we get

$$\begin{aligned} \Delta V &= \frac{1}{2}d^T Q_{uu} d + d^T Q_u \\ s &= Q_x + K^T Q_{uu} d + K^T Q_u + Q_{ux}^T d \\ S &= Q_{xx} + K^T Q_{uu} K + K^T Q_{ux} + Q_{ux}^T K \end{aligned} \tag{12}$$

2.5 Forward Pass

Since the dynamics and the cost function are only approximated at each time step, it is necessary to iteratively solve the previous problem to successively get closer to the local minimum. After each backward pass solving for the optimal correction in control values, δu_k^*, these values are used to calculate a new state trajectory (X) from the nominal trajectories $\bar{X}, \bar{U}$, often referred to as a "rollout". The α term is used for a line search. This is done using the following algorithm:

$$\delta x = x_k - \bar{x}_k$$

$$\begin{aligned} u_k &= \bar{u}_k + \delta u_k^* \\ &= \bar{u}_k + K_k\delta x_k + \alpha d_k \end{aligned}$$

$$x_{k+1} = f(x_k, u_k)$$

where α is the step size, typically used to perform a simple line search (see Section ??).

2.6 Square-Root Backward Pass (V3)

2.6.1 $\sqrt{A} + \sqrt{B} \to \sqrt{A+B}$

$$Q, R = \texttt{QR}(\begin{bmatrix}\sqrt{A} \\ \sqrt{B}\end{bmatrix})$$
$$R = \sqrt{A+B}$$

2.6.2 Method

We approximate the state-action cost-to-go $Q(x, u)$ using a second-order Taylor series expansion:

$$\delta Q(x,u) = \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u\end{bmatrix}^T \begin{bmatrix}Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{xx}\end{bmatrix}\begin{bmatrix}\delta x \\ \delta u\end{bmatrix} + \begin{bmatrix}Q_x \\ Q_u\end{bmatrix}^T \begin{bmatrix}\delta x \\ \delta u\end{bmatrix}$$

The terminal cost-to-go Hessian and stage costs are factored:

$$S_N = \sqrt{S_N}^T \sqrt{S_N}$$
$$L_{ii} = \sqrt{L_{ii}}^T \sqrt{L_{ii}}$$

This factorization can be performed using Cholesky (or LDL) decomposition
The backward pass is modified with the diagonal blocks of the state-action cost-to-go $Q(x, u)$ factored as follows:

$$Q_{xx} = \sqrt{L_{xx}}^T \sqrt{L_{xx}} + A^T S'^T S' A = \begin{bmatrix}\sqrt{L_{xx}} & S'A\end{bmatrix}^T \begin{bmatrix}\sqrt{L_{xx}} \\ S'A\end{bmatrix}$$
$$Q_{uu} = \sqrt{L_{uu}}^T \sqrt{L_{uu}} + B^T S'^T S' B = \begin{bmatrix}\sqrt{L_{uu}} & S'B\end{bmatrix}^T \begin{bmatrix}\sqrt{L_{uu}} \\ S'B\end{bmatrix}$$

We can factor $Q_{xx} \to \sqrt{Q_{xx}}^T \sqrt{Q_{xx}}$ using the above technique:

$$-, \sqrt{Q_{xx}} = \texttt{QR}(\begin{bmatrix}\sqrt{L_{xx}} \\ S'A\end{bmatrix})$$

Similarly for Q_{uu}:

$$-, \sqrt{Q_{uu}} = \texttt{QR}(\begin{bmatrix}\sqrt{L_{uu}} \\ S'B\end{bmatrix})$$

The gain $K = -Q_{uu}^{-1} Q_{ux}$ can now be written in its square root form:

$$K = -\sqrt{Q_{uu}}^{-1} \sqrt{Q_{uu}}^{-T} Q_{ux}$$

The gain $d = Q_{uu}^{-1} Q_u$ can be written in its square root form as well:

$$d = -\sqrt{Q_{uu}}^{-1} \sqrt{Q_{uu}}^{-T} Q_u$$

The gradient of the cost-to-go $s = Q_x + K^T Q_{uu} d + K^T Q_u + Q_{ux}^T d$ can be expressed as:

$$s = Q_x + (K^T \sqrt{Q_{uu}}^T)(\sqrt{Q_{uu}} d) + K^T Q_u + Q_{ux}^T d$$

The square root of the Hessian of the cost-to-go $S = Q_{xx} + K^T Q_{uu} K + K^T Q_{ux} + Q_u x^T K$ is derived:

$$S = \begin{bmatrix}I \\ K\end{bmatrix}^T \begin{bmatrix}Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu}\end{bmatrix}\begin{bmatrix}I \\ K\end{bmatrix}$$

$$\begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} = \begin{bmatrix} \alpha^T & 0_{nxm} \\ \beta^T & \gamma^T \end{bmatrix} \begin{bmatrix} \alpha & \beta \\ 0_{mxn} & \gamma \end{bmatrix} = \begin{bmatrix} \alpha^T\alpha & \alpha^T\beta \\ \beta^T\alpha & \beta^T\beta - \gamma^T\gamma \end{bmatrix}$$

$$\begin{aligned} \alpha &= \sqrt{Q_{xx}} \\ \beta &= \sqrt{Q_{xx}}^{-T} Qux \\ \gamma &= \sqrt{\sqrt{Q_{uu}}^T \sqrt{Q_{uu}} - \beta^T\beta} \end{aligned}$$

$$\begin{aligned} \sqrt{S} &= \begin{bmatrix} \alpha & \beta \\ 0_{mxn} & \gamma \end{bmatrix} \begin{bmatrix} I \\ K \end{bmatrix} \\ &= \begin{bmatrix} \alpha + \beta K \\ \gamma K \end{bmatrix} \end{aligned}$$

Note: that $S \in R^{nxn}$, $\sqrt{S} \in R^{(n+m)xn}$ and $\sqrt{S_N}$[1:n,1:n] = chol(S_N)

2.7 Regularization

Due to limited numerical precision, it is common for Q_{uu} to become not positive definite. To address this problem, regularization is added in a way that is equivalent to using trust region methods. Additionally, there are two options for regularization. The first penalizes deviations from a control trajectory and the second penalizes deviations for the state trajectory.

Option 1:

$$\tilde{Q}_{uu} = \ell_{uu} + B^T S' B + \rho I$$

$$\begin{aligned} d &= -\tilde{Q}_{uu}^{-1} Q_u \\ K &= -\tilde{Q}_{uu}^{-1} Q_{ux} \end{aligned}$$

$$\begin{aligned} \Delta V &= \frac{1}{2} d^T Q_{uu} d + d^T Q_u \\ s &= Q_x + K^T Q_{uu} d + K^T Q_u + Q_{ux}^T d \\ S &= Q_{xx} + K^T Q_{uu} K + K^T Q_{ux} + Q_{ux}^T K \end{aligned}$$

Option 2:

$$\begin{aligned} \tilde{Q}_{uu} &= \ell_{uu} + B^T (S' + \rho I) B \\ \tilde{Q}_{ux} &= \ell_{ux} + B^T (S' + \rho I) A \end{aligned}$$

$$\begin{aligned} d &= -\tilde{Q}_{uu}^{-1} Q_u \\ K &= -\tilde{Q}_{uu}^{-1} \tilde{Q}_{ux} \end{aligned}$$

$$\begin{aligned} \Delta V &= \frac{1}{2} d^T Q_{uu} d + d^T Q_u \\ s &= Q_x + K^T Q_{uu} d + K^T Q_u + Q_{ux}^T d \\ S &= Q_{xx} + K^T Q_{uu} K + K^T Q_{ux} + Q_{ux}^T K \end{aligned}$$

Note the subtle but very important distinctive use of Q_{uu} and $\tilde{Q}_{uu}$ (and $\tilde{Q}_{ux}$). The former is used to estimate the cost-to-go and is propagated backward, while the regularized versions are used to compute the optimal gains. Generally, the second option is more robust and is preferred.

3 Constraints

3.1 Augmented Lagrange Method

In order to solve systems with equality and inequality constraints we define a new cost function:

$$\mathcal{L}_A(x_0, U; \mu, \lambda) = \ell_f(x_N) + \sum_{k=1}^{N-1} \{\ell(x_k, u_k)\}$$
$$+\frac{1}{2} c_N(x_N)^T I_{\mu_N} c_N(x_N) + \lambda_N^T c_N(x_N) + \sum_{k=1}^{N-1} \{\frac{1}{2} c_k(x_k, u_k)^T I_{\mu_k} c_k(x_k, u_k) + \lambda_k^T c_k(x_k, u_k)\}$$

We define a single equality constraint as:

$$c_k^i(x, u) = 0, i \in \mathcal{E}$$
$$c_k^i \in C^1, \in \mathbb{R}$$

and a single in equality constraint as:

$$c_k^i(x, u) \leq 0, i \in \mathcal{I}$$
$$c_k^i \in C^1, \in \mathbb{R}$$

using subscripts to denote time index and superscripts to denote a particular constraint. The vector $c_k \in \mathbb{R}^p$ is the ordered (equality, then inequality), vertical concatenation of all p constraints at that time step.
The matrix $I_{\mu_k} \in \mathbb{R}^{pxp}$ is used to turn on and off feasible inequality constraints and is defined as follows:

$$I_{\mu_k}[i, i] = \begin{cases} \mu_k^i & \text{if any: } i \in \mathcal{E},\ c_k^i \in \mathcal{I} > 0,\ \lambda_k^i > 0 \\ 0 & \text{otherwise} \end{cases}$$

The update for λ is defined as follows:

$$\lambda_k^i \leftarrow \lambda_k^i + \mu_k^i c_k^i(x_k, u_k), i \in \mathcal{E}$$
$$\lambda_k^i \leftarrow \max(0, \lambda_k^i + \mu_k^i c_k^i(x_k, u_k)), i \in \mathcal{I}$$

3.2 Modification to Backward Pass

To account for constraints in the optimization, we modify the backward pass as follows:

$$\hat{Q}_x = Q_x + c_x^T I_\mu c + c_x^T \lambda$$
$$\hat{Q}_u = Q_u + c_u^T I_\mu c + c_u^T \lambda$$
$$\hat{Q}_{xx} = Q_{xx} + c_x^T I_\mu c_x$$
$$\hat{Q}_{uu} = Q_{uu} + c_x^T I_\mu c_x$$
$$\hat{Q}_{ux} = Q_{ux} + c_u^T I_\mu c_x$$

where $c_x = \frac{\partial c}{\partial x}|_{x,u}, c_u = \frac{\partial c}{\partial u}|_{x,u}$ Additionally the boundary conditions S_N, s_N are augmented:

$$\hat{S}_N = S_N + c_{x_N}^T I_{\mu_N} c_{x_N}$$
$$\hat{s}_N = s_N + c_{x_N}^T I_{\mu_N} c_N + c_{x_N}^T \lambda_N$$

The backward pass can be performed as before using these augmented versions.

4 Infeasible Initial Trajectory

Given a desired state trajectory $X_d : \{x_{d_1}, ...x_{d_n}\}$ we must find a set of artificial/slack controls $U_i : \{u_{i_1}, ...u_{i_{n-1}}\}$ to achieve this state trajectory despite the system dynamics. At each time step we solve for u_{i_k}:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) + u_{i_k} \\ s.t.\ x_{k+1} &= x_{d_{k+1}} \\ \rightarrow u_{i_k} &= x_{d_{k+1}} - x_{k+1} \end{aligned}$$

Before simulating the dynamics at the next step remember to apply the artificial control we just calculated: $x_{k+1} += u_{i_k}$

5 Minimum Time

6 Taylor Series Approximation

Throughout the following derivations we use Taylor polynomials to approximate nonlinear functions. We are typically interested in first or second order approximations of nonlinear functions of two variables, the state and control. The following derivations may prove helpful.

A second order Taylor Series approximation of $f(x)$, a nonlinear scalar valued function that is dependent one variable ($x \in R$) and linearized about $\bar{x}$ (which is not necessarily zero):

$$f(x) \rightarrow \tag{13}$$

$$f(\bar{x} + \delta x) \approx f(\bar{x}) + \frac{\partial f}{\partial x}\Big|_{\bar{x}}(x - \bar{x}) + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\Big|_{\bar{x}}(x - \bar{x})^2 \tag{14}$$

$$\approx f(\bar{x}) + \frac{\partial f}{\partial x}\Big|_{\bar{x}}\delta x + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\Big|_{\bar{x}}\delta x^2 \tag{15}$$

A second order Taylor Series approximation of $f(x)$, a nonlinear vector valued function that is dependent one variable ($x \in R^n$) and linearized about $\bar{x}$ (which is not necessarily zero):

$$f(x) \rightarrow \tag{16}$$

$$f(\bar{x} + \delta x) \approx f(\bar{x}) + \frac{\partial f}{\partial x}\Big|_{\bar{x}}(x - \bar{x}) + \frac{1}{2}(x - \bar{x})^T\frac{\partial^2 f}{\partial x^2}\Big|_{\bar{x}}(x - \bar{x}) \tag{17}$$

$$\approx f(\bar{x}) + \frac{\partial f}{\partial x}\Big|_{\bar{x}}\delta x + \frac{1}{2}\delta x^T\frac{\partial^2 f}{\partial x^2}\Big|_{\bar{x}}\delta x \tag{18}$$

The most complicated approximation we perform is a second order Taylor Series approximation of a nonlinear function f that is dependent on three variables. The procedure is the same as before:

$$\begin{aligned} & f(x, y, z) \rightarrow \\ & f(\bar{x} + \delta x, \bar{y} + \delta y, \bar{z} + \delta z) \\ & \approx f(\bar{x}, \bar{y}, \bar{z}) + \frac{\partial f}{\partial x}\Big|_{\bar{x},\bar{y},\bar{z}}(x - \bar{x}) + \frac{\partial f}{\partial y}\Big|_{\bar{x},\bar{y},\bar{z}}(y - \bar{y}) + \frac{\partial f}{\partial z}\Big|_{\bar{x},\bar{y},\bar{z}}(z - \bar{z}) \\ & \frac{1}{2}(x - \bar{x})^T\frac{\partial^2 f}{\partial x^2}\Big|_{\bar{x},\bar{y},\bar{z}}(x - \bar{x}) + \frac{1}{2}(y - \bar{y})^T\frac{\partial^2 f}{\partial y^2}\Big|_{\bar{x},\bar{y},\bar{z}}(y - \bar{y}) + \frac{1}{2}(z - \bar{z})^T\frac{\partial^2 f}{\partial z^2}\Big|_{\bar{x},\bar{y},\bar{z}}(z - \bar{z}) \\ & \frac{1}{2}(x - \bar{x})^T\frac{\partial^2 f}{\partial x \partial y}\Big|_{\bar{x},\bar{y},\bar{z}}(y - \bar{y}) + \frac{1}{2}(x - \bar{x})^T\frac{\partial^2 f}{\partial x \partial z}\Big|_{\bar{x},\bar{y},\bar{z}}(z - \bar{z}) \\ & + \frac{1}{2}(y - \bar{y})^T\frac{\partial^2 f}{\partial y \partial x}\Big|_{\bar{x},\bar{y},\bar{z}}(x - \bar{x}) + \frac{1}{2}(y - \bar{y})^T\frac{\partial^2 f}{\partial y \partial z}\Big|_{\bar{x},\bar{y},\bar{z}}(z - \bar{z}) \\ & + \frac{1}{2}(z - \bar{z})^T\frac{\partial^2 f}{\partial z \partial x}\Big|_{\bar{x},\bar{y},\bar{z}}(x - \bar{x}) + \frac{1}{2}(z - \bar{z})^T\frac{\partial^2 f}{\partial z \partial y}\Big|_{\bar{x},\bar{y},\bar{z}}(y - \bar{y}) \end{aligned}$$

7 Appendix

7.1 Square-Root Backward Pass(DONT USE)

It is common for S to become ill-conditioned, which causes numerical instability when performing the updates in (19). This can be quantified using a conditioning number, which is a ratio between the largest and smallest eigenvalues of a square matrix [Kaminkski'71] To alleviate numerical issues, we can derive a backward pass using only the square root of S, defined as $S = LL^T$, where L is the is value to be calculated and propagated backward in time. This is done by using the a factor of S when calculating K, d, s, S (and Q_{uu}) during the backward pass. To reduce visual noise, we drop subscripts and define $S_k \equiv \bar{S}$ and $S_{k+1} \equiv S$. We also define $A^{-T} \equiv (A^{-1})^T$.

$$
\begin{aligned}
K &= Q_{uu}^{-1} Q_{ux} \\
&= \Big(\ell_{uu} + B^T S B\Big)^{-1} B^T S A \\
&= \ell_{uu}^{-1} B^T \Big(S^{-1} + B \ell_{uu}^{-1} B^T\Big)^{-1} A && \text{(Matrix Inversion Lemma}^2\text{)} \\
&= \ell_{uu}^{-1} B^T \Big(L^{-T} L^{-1} + B \ell_{uu}^{-1} B^T\Big)^{-1} A && ((A^{-1})^T = (A^T)^{-1},\ S = LL^T) \\
&= \ell_{uu}^{-1} B^T \Big(L^{-T} L^{-1} + L^{-T} L^T B \ell_{uu}^{-1} B^T L L^{-1}\Big)^{-1} A && (AA^{-1} = I) \\
&= \ell_{uu}^{-1} B^T \Big[L^{-T}(I + L^T B \ell_{uu}^{-1} B^T L) L^{-1}\Big]^{-1} A && \text{(Factor out } L^{-T} \text{ (left), } L^{-1} \text{ (right))} \\
K &= \ell_{uu}^{-1} B^T L \Big(I + L^T B \ell_{uu}^{-1} B^T L\Big)^{-1} L^T A && ((ABC)^{-1} = C^{-1} B^{-1} A^{-1})
\end{aligned}
$$

In order to calculate d we will first calculate Q_{uu} in terms of the factored S:

$$
\begin{aligned}
Q_{uu} &= B^T S B + \ell_{uu} \\
&= B^T L L^T B + L_R L_R^T && (S = LL^T,\ L_R = \texttt{chol}(R)) \\
&= M M^T \\
&\quad \text{where } M \equiv \begin{bmatrix} B^T L & L_R \end{bmatrix} \\
&= Q R R^T Q^T && (Q, R = \texttt{QR}(M)) \\
L_{Q_{uu}} L_{Q_{uu}}^T &= Q D Q^T && (D \text{ is a diagonal matrix}) \\
&\downarrow \\
L_{Q_{uu}} &= Q D^{1/2}
\end{aligned}
$$

With Q_{uu} factorized with respect to L we can compute d in terms of L:

$$
\begin{aligned}
d &= Q_{uu}^{-1} Q_u^T \\
&= \Big(L_{Q_{uu}} L_{Q_{uu}}^T\Big)^{-1} Q_u
\end{aligned}
$$

Next, we can calculate $\bar{S}$:

$$
\begin{aligned}
\bar{S} &= \ell_{xx} + A^T S A_k - A_k^T S B (\ell_{uu} + B^T S B)^{-1} B^T S A \\
&= A^T (S^{-1} + B \ell_{uu}^1 B^T)^1 A + \ell_{xx} && \text{(Matrix Inversion Lemma}^1\text{)} \\
A(\bar{S} - \ell_{xx})^{-1} A^T &= S^{-1} + B \ell_{uu}^{-1} B^T && \text{(Re-arrange and invert)} \\
&= L^{-T} L^{-1} + L^{-T} L^T B L_R^{-T} L_R^{-1} B^T L L^{-1} && (S = LL^T, L_R = \texttt{chol}(\ell_{uu})) \\
&= L^{-T}(I + L^T B L_R^{-T} L_R^{-1} B^T L) L^{-1} \\
&= L^{-T}(I + G G^T) L^{-1} \\
&\quad \text{where } G \equiv L^T B L_R^{-T}, G \in \mathbb{R}^{nxm} \\
\bar{S} &= A^T L (I + G G^T)^{-1} L^T A + \ell_{xx} && \text{(Re-arrange and invert)}
\end{aligned}
$$

$$
\begin{aligned}
&= A^T L(I + W\Lambda W^T)^{-1} L^T A + \ell_{xx} && (W\Lambda W^T = \texttt{LDL}(GG^T)) \\
&= A^T L(WW^T + W\Lambda W^T)^{-1} L^T A + \ell_{xx} && (WW^T = I, W \text{ is orthogonal}) \\
&= A^T L(W(I + \Lambda)W^T)^{-1} L^T A + \ell_{xx} && (\text{Factor out } W, W^T) \\
&= A^T L W^{-T}(I + \Lambda)^{-1} W^{-1} L^T A + \ell_{xx} \\
&= A^T L W (I + \Lambda)^{-1} W^T L^T A + \ell_{xx} && (W^T = W^{-1}, W \text{ is orthogonal}) \\
&= A^T L W (I + \Lambda)^{-1/2} (I + \Lambda)^{-1/2} W^T L^T A + L_Q L_Q^T && (L_Q = \texttt{chol}(\ell_{xx})) \\
&= MM^T \\
&\quad \text{where } M \equiv \begin{bmatrix} A^T L W (I + \Lambda)^{-1/2} & L_Q \end{bmatrix} \\
&= QRR^T Q^T && (Q, R = \texttt{QR}(M)) \\
\bar{L}\bar{L}^T &= QVDV^T Q^T && (V, D = \texttt{LDL}(RR^T)) \\
&\downarrow \\
\bar{L} &= QVD^{1/2}
\end{aligned}
$$

Finally, we can calculate $\bar{s}$:

$$
\begin{aligned}
\bar{s} &= Q_x - K^T Q_u + K^T Q_{uu}^T d - Q_{ux}^T d \\
&= Q_x - (B^T SA)^T (\ell_{uu} + B^T SB)^{-T} Q_u + (B^T SA)^T (\ell_{uu} + B^T SB)^{-1} Q_u - (B^T SA)^T (\ell_{uu} + B^T SB)^{-1} Q_u \\
&= Q_x - (B^T SA)^T (\ell_{uu} + B^T SB)^{-T} Q_u && ((ABC)^T = C^T B^T A^T) \\
&= Q_x - A^T S^T B (\ell_{uu} + B^T SB)^{-T} Q_u \\
&= Q_x - A^T \Big(S^{-T} + B\ell_{uu}^{-T} B^T\Big)^{-1} B\ell_{uu}^{-T} Q_u && (\text{Matrix Inversion Lemma}^2) \\
&= Q_x - A^T \Big(L^{-T} L^{-1} + B\ell_{uu}^{-T} B^T\Big)^{-1} B\ell_{uu}^{-T} Q_u && ((A^{-1})^T = (A^T)^{-1},\ S = LL^T) \\
&= Q_x - A^T \Big(L^{-T} L^{-1} + L^{-T} L^T B\ell_{uu}^{-T} B^T L L^{-1}\Big)^{-1} B\ell_{uu}^{-T} Q_u && (AA^{-1} = I) \\
&= Q_x - A^T \Big[L^{-T}(I + L^T B\ell_{uu}^{-T} B^T L) L^{-1}\Big]^{-1} B\ell_{uu}^{-T} Q_u && (\text{Factor out } L^{-T} \text{ (left)},\ L^{-1} \text{ (right)}) \\
\bar{s} &= Q_x - A^T L(I + L^T B\ell_{uu}^{-T} B^T L)^{-1} L^T B\ell_{uu}^{-T} Q_u && ((ABC)^{-1} = C^{-1} B^{-1} A^{-1})
\end{aligned}
$$

NOTE: $\texttt{chol}(A)$ is the Cholesky decomposition, $A = BB^T$, $\texttt{LDL}(A)$ is an orthogonalization of a symmetric matrix, $A = LDL^T$, where L is an orthogonal matrix and D is a diagonal matrix. This factorization is most easily accomplished using the eigenvector decomposition, where L is the matrix of eigenvectors of A and D is a diagonal matrix comprising the eigenvalues of A. $\texttt{QR}(A)$ is the QR decomposition of A. Matrix Inversion Lemmas:

$$
\begin{aligned}
\Big(A + UCV\Big)^{-1} &= A^{-1} - A^{-1} U\Big(VA^{-1}U + C^{-1})VA^{-1} \\
\Big(A + UCV\Big)^{-1} UC &= A^{-1} U\Big(C^{-1} + VA^{-1}U\Big)^{-1}
\end{aligned}
$$

The algorithm for the square-root backward pass:

$$
\begin{aligned}
G &= L^T B L_R^{-T} \\
W, \Lambda &= \text{LDL}(GG^T) \\
M &= \begin{bmatrix} A^T L W (I + \Lambda)^{-1/2} & L_Q \end{bmatrix} \\
Q, R &= \text{QR}(M) \\
V, D &= \text{LDL}(RR^T) \\
\bar{L} &= QVD^{1/2}
\end{aligned}
$$

7.2 L derivation for FOH

$$
L_{xx} = 4M_1^T L_{xx}^m M_1 + L_{xx}^-
$$

$$
\begin{aligned}
L_{xu} &= 2M_1^T L_{xu}^m + 4M_1 L_{xx}^m M_2 + L_{xu}^- \\
L_{xy} &= M_1^T L_{xx}^m \\
L_{xv} &= 2M_1 L_{xu}^m \\
L_{uu} &= L_{uu}^m + 2(M_2^T L_{xu}^m + L_{ux}^m M_2) + 4M_2 L_{xx}^m M_2 + L_{uu}^- \\
L_{uy} &= \frac{1}{2} L_{ux}^m + M_2^T L_{xx}^m \\
L_{uv} &= L_{uu}^m + 2M_2^T L_{xu}^m \\
L_{yy} &= \frac{1}{4} L_{xx}^m + L_{xx}^+ \\
L_{yv} &= \frac{1}{2} L_{xy}^m + L_{xu}^+ \\
L_{vv} &= L_{uu}^m + L_{uu}^+
\end{aligned}
$$

$$
\begin{aligned}
L_x &= L_x^m M_1 + L_x^+ \\
L_u &= \frac{1}{2} L_u^m + L_x^m M_2 + L_u^- \\
L_y &= \frac{1}{4} L_x^m + L_x^+ \\
L_v &= \frac{1}{2} L_u^m + L_u^+
\end{aligned}
$$

7.3 Verbose iLQR derivation

$$
\begin{aligned}
V_k + \delta V_k &= \min_{u_k, \delta u_k} \ell_k(x_k + \delta x_k, u_k + \delta u_k) + V_{k+1}(f(x_k + \delta x_k, u_k + \delta x_k)) \\
&\approx \min_{u_k, \delta u_k} \ell(x_k, u_k) + \ell_x(x - x_k) + \ell_u(u - u_k) && \text{(Taylor series expansion)} \\
&\quad + \frac{1}{2}(x - x_k)^T \ell_{xx}(x - x_k) + \frac{1}{2}(u - u_k)^T \ell_{uu}(u - u_k) \\
&\quad + \frac{1}{2}(u - u_k)^T \ell_{ux}(x - x_k) + \frac{1}{2}(x - x_k)^T \ell_{xu}(u - u_k) \\
&\quad + V(x_{k+1}, u_{k+1}) + s_{k+1}\delta x_{k+1} + \frac{1}{2}\delta x_{k+1} S_{k+1} \delta x_{k+1} \\
&= \min_{u_k} \ell(x_k, u_k) + V(x_{k+1}, u_{k+1}) && \text{(Separate minimizations)} \\
&\quad + \min_{\delta u_k} \ell_x \delta x_k + \ell_u \delta u_k + \frac{1}{2}\delta x_k^T \ell_{xx} \delta x_k + \frac{1}{2}\delta u_k^T \ell_{uu} \delta u_k \\
&\quad + \frac{1}{2}\delta u_k^T \ell_{ux} \delta x_k + \frac{1}{2}\delta x_k^T \ell_{xu} \delta u_k \\
&\quad + s_{k+1}\delta x_{k+1} + \frac{1}{2}\delta x_{k+1} S_{k+1} \delta x_{k+1} \\
\delta V_k &= \min_{\delta u_k} + \ell_x \delta x_k + \ell_u \delta u_k + \frac{1}{2}\delta x_k^T \ell_{xx} \delta x_k + \frac{1}{2}\delta u_k^T \ell_{uu} \delta u_k && \text{(Cancel terms)} \\
&\quad + \frac{1}{2}\delta u_k^T \ell_{ux} \delta x_k + \frac{1}{2}\delta x_k^T \ell_{xu} \delta u_k \\
&\quad + s_{k+1}\delta x_{k+1} + \frac{1}{2}\delta x_{k+1} S_{k+1} \delta x_{k+1} \\
&= \min_{\delta u_k} \ell_x \delta x_k + \ell_u \delta u_k + \frac{1}{2}\delta x_k^T \ell_{xx} \delta x_k + \frac{1}{2}\delta u_k^T \ell_{uu} \delta u_k && \text{(Plug in dynamics)} \\
&\quad + \frac{1}{2}\delta u_k^T \ell_{ux} \delta x_k + \frac{1}{2}\delta x_k^T \ell_{xu} \delta u_k \\
&\quad + s_{k+1}(A\delta x_k + B\delta u_k) + \frac{1}{2}(\delta u_k^T B^T + \delta x_k^T A^T) S_{k+1}(A\delta x_k + B\delta u_k)
\end{aligned}
$$

$$
\begin{aligned}
&= \min_{\delta u_k} \Big(\ell_x + s_{k+1}A\Big)\delta x_k + \Big(\ell_u + s_{k+1}B\Big)\delta u_k && \text{(Combine Terms)} \\
&\quad + \frac{1}{2}\delta x_k^T \Big(\ell_{xx} + A^T S_{k+1} A\Big)\delta x_k + \frac{1}{2}\delta u_k^T \Big(\ell_{uu} + B^T S_{k+1} B\Big)\delta u_k \\
&\quad + \frac{1}{2}\delta u_k^T \Big(\ell_{ux} + B^T S_{k+1} A\Big)\delta x_k + \frac{1}{2}\delta x_k^T \Big(\ell_{xu} + A^T S_{k+1} B\Big)\delta u_k
\end{aligned}
$$

7.4 (

unnecssary plug in for s,S)

$$
\begin{aligned}
s_k &= q_k + s_{k+1}A_k - (q_k + s_{k+1}B_k)(R_k + B_k^T S_{k+1} B_k)^{-1}(B_k^T S_{k+1} A_k) \\
S_k &= Q_k + A_k^T S_{k+1} A_k - A_k^T S_{k+1} B_k (R_k + B_k^T S_{k+1} B_k)^{-1} B_k^T S_{k+1} A_k
\end{aligned} \tag{19}
$$

which is easy to verify that all values are either functions of the current state and control (subscripts k), or are from the next time step. By working backwards, the $S_k, s_k,$ and u_k^* values can be calculated for each time step.

subsectionincorrect foh derivation

7.4.1 Simpson Integration of Stage Cost

The first step is to express $L(x_k, u_k, x_m, u_m, x_{k+1}, u_{k+1})$ as a linear expression of $(x_k, u_k, x_{k+1}, u_{k+1})$ by finding expressions for (x_m, u_m) in terms of the other variables. Hermite-Simpson integration performs the following approximation:

$$
\begin{aligned}
x(t) &= \int_{t_k}^{t_{k+1}} f(x,u)dt && (20) \\
&\approx at^2 + bt + c && (21) \\
& && (22)
\end{aligned}
$$

Without loss of generality, we assume knowledge of the points $(x_k, u_k, x_{k+1}, u_{k+1})$ and assume $t_k = 0, t_{k+1} = dt$. By specifying 3 known points we can solve for the coefficients (a, b, c). We pick the points $x_k, x_{k+1}, \dot{x}_k = f(x_k, u_k)$:

$$
\begin{aligned}
\begin{bmatrix} x_k \\ \dot{x}_k \\ x_{k+1} \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ dt^2 & dt & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \\
&= T \begin{bmatrix} a \\ b \\ c \end{bmatrix}
\end{aligned} \tag{23}
$$

Solving this linear equation gives

$$
\begin{aligned}
a &= \frac{1}{dt^2}(-x_k - \dot{x}_k dt + x_{k+1}) \\
b &= \dot{x}_k \\
c &= x_k
\end{aligned} \tag{24}
$$

We can now calculate x_m by evaluating the spline (Eq. 22) at $dt/2$:

$$
\begin{aligned}
x_m &= \begin{bmatrix} \frac{1}{4}dt^2 & \frac{1}{2}dt & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{4}dt^2 & \frac{1}{2}dt & 1 \end{bmatrix} T^{-1} \begin{bmatrix} x_k \\ \dot{x}_k \\ x_{k+1} \end{bmatrix} \\
&= \frac{1}{4}\begin{bmatrix} 3 & dt & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ x_{k+1} \end{bmatrix}
\end{aligned} \tag{25}
$$

which can be verified by either plugging in the expressions for a, b, c in Eq 24 or using a symbolic solver (e.g. SymPy).

In order to express $[x_k \ \dot{x}_k \ x_{k+1}]^T$ as a linear expression in the desired variables, we need to perform a 1st Order Taylor-Series expansion of the dynamics:

$$\begin{aligned}\dot{x} + \delta\dot{x} &\approx f(x,u) + \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial u}\delta u \\ &= f(x,u) + \mathcal{A}(x,u)\delta x + \mathcal{B}(x,u)\delta u \\ \delta\dot{x} &= \mathcal{A}(x,u)\delta x + \mathcal{B}(x,u)\delta u\end{aligned} \tag{26}$$

It is important that these Jacobian matrices are distinct from the ones for linearized discrete dynamics in (43). These Jacobians are calculated on the continuous dynamics equation.
Using this result we can now express the following:

$$\begin{bmatrix}\delta x_k \\ \delta\dot{x}_k \\ \delta x_{k+1}\end{bmatrix} = \begin{bmatrix}I_n & 0 & 0 & 0 \\ \mathcal{A} & \mathcal{B} & 0 & 0 \\ 0 & 0 & I_n & 0\end{bmatrix}\begin{bmatrix}\delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1}\end{bmatrix} \tag{27}$$

where I_n is an $n \times n$ identity matrix and I_m is an $m \times m$ identity matrix.
We can now define the following:

$$\begin{aligned}x_m &= \frac{1}{4}\begin{bmatrix}3 & dt & 1\end{bmatrix}\begin{bmatrix}I_n & 0 & 0 & 0 \\ \mathcal{A} & \mathcal{B} & 0 & 0 \\ 0 & 0 & I_n & 0\end{bmatrix}\begin{bmatrix}\delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1}\end{bmatrix} \\ &= \frac{1}{4}\begin{bmatrix}3I_n + dt\mathcal{A} & dt\mathcal{B} & I_n & 0\end{bmatrix}\begin{bmatrix}\delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1}\end{bmatrix} \\ &= M\begin{bmatrix}\delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1}\end{bmatrix}\end{aligned} \tag{28}$$

With an expression for x_m, all we need is an expression for u_m. This expression is obtain using a simple interpolation: $u_m = \frac{1}{2}(u_k + u_{k+1})$. Using both of these relationships we can can convert between point sets:

$$\begin{aligned}\begin{bmatrix}\delta x_k \\ \delta u_k \\ \delta x_m \\ \delta u_m \\ \delta x_{k+1} \\ \delta u_{k+1}\end{bmatrix} &= \begin{bmatrix}I_n & 0 & 0 & 0 \\ 0 & I_m & 0 & 0 \\ & [M] & & \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & I_n & 0 \\ 0 & 0 & 0 & I_m\end{bmatrix}\begin{bmatrix}\delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1}\end{bmatrix} \\ &= E\begin{bmatrix}\delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1}\end{bmatrix}\end{aligned} \tag{29}$$

7.4.2 Approximating Stage Cost

We can find a quadratic expression for $L(x_k, u_k, x_{k+1}, u_{k+1})$ by taking a 2nd Order Taylor Series Expansion of the integral term in (41):

$$\delta L_k \approx \frac{dt}{6} \begin{bmatrix} \delta x_k \\ \delta u_k \\ \delta x_m \\ \delta u_m \\ \delta x_{k+1} \\ \delta u_{k+1} \end{bmatrix}^T \begin{bmatrix} L^-_{xx} & L^-_{xu} & & & & \\ L^-_{ux} & L^-_{uu} & & & & \\ & & 4L^m_{xx} & 4L^m_{xu} & & \\ & & 4L^m_{ux} & 4L^m_{uu} & & \\ & & & & L^+_{xx} & L^+_{xu} \\ & & & & L^+_{ux} & L^+_{uu} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \\ \delta x_m \\ \delta u_m \\ \delta x_{k+1} \\ \delta u_{k+1} \end{bmatrix}^T \tag{30}$$

$$+ \frac{dt}{6} \begin{bmatrix} L^-_x & L^-_u & 4L^m_x & 4L^m_u & L^+_x & L^+_u \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \\ \delta x_m \\ \delta u_m \\ \delta x_{k+1} \\ \delta u_{k+1} \end{bmatrix}^T \tag{31}$$

$$= \frac{dt}{6} \begin{bmatrix} \delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1} \end{bmatrix}^T E^T \begin{bmatrix} L^-_{xx} & L^-_{xu} & & & & \\ L^-_{ux} & L^-_{uu} & & & & \\ & & 4L^m_{xx} & 4L^m_{xu} & & \\ & & 4L^m_{ux} & 4L^m_{uu} & & \\ & & & & L^+_{xx} & L^+_{xu} \\ & & & & L^+_{ux} & L^+_{uu} \end{bmatrix} E \begin{bmatrix} \delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1} \end{bmatrix} \tag{32}$$

$$+ \frac{dt}{6} \begin{bmatrix} L^-_x & L^-_u & 4L^m_x & 4L^m_u & L^+_x & L^+_u \end{bmatrix} E \begin{bmatrix} \delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1} \end{bmatrix} \tag{33}$$

where $L_x := \frac{\partial L}{\partial x}, L_{xx} := \frac{\partial^2 L}{\partial x^2}, L_{xu} := \frac{\partial^2 L}{\partial x u}$, etc. and L^-, L^m, L^+ are the partials evaluated at $(x_k, u_k), (x_m, u_m), (x_{k+1}, u_{k+1})$, respectively.

Using a symbolic solver or chugging through the matrix multiplication by hand we arrive at the final expression for $L(x_k, u_k, x_{k+1}, u_{k+1})$:

7.5 Calculating $\bar{P}$

Once we have $\delta\bar{L}$, we also need $\delta\bar{P}$ from Equation **??** in order to calculate the elements of Q in Equation 51. We simply plug in the linearized discrete dynamics (Eq. 43) into our cost-go-to for the next time step:

$$\delta P(\delta y, \delta v) = \frac{1}{2} \begin{bmatrix} \delta y \\ \delta v \end{bmatrix}^T \begin{bmatrix} S_{yy} & S_{yv} \\ S_{vy} & S_{vv} \end{bmatrix} \begin{bmatrix} \delta y \\ \delta v \end{bmatrix} + \begin{bmatrix} S_y & S_v \end{bmatrix} \begin{bmatrix} \delta y \\ \delta v \end{bmatrix} \tag{34}$$

$$\delta\bar{P}(\delta x, \delta u, \delta v) = \frac{1}{2} \begin{bmatrix} A\delta x + B\delta u + C\delta v \\ \delta v \end{bmatrix}^T \begin{bmatrix} S_{yy} & S_{yv} \\ S_{vy} & S_{vv} \end{bmatrix} \begin{bmatrix} A\delta x + B\delta u + C\delta v \\ \delta v \end{bmatrix} + \begin{bmatrix} S_y & S_v \end{bmatrix} \begin{bmatrix} A\delta x + B\delta u + C\delta v \\ \delta v \end{bmatrix} \tag{35}$$

$$= \frac{1}{2} \begin{bmatrix} \delta x \\ \delta u \\ \delta v \end{bmatrix}^T \begin{bmatrix} G_{xx} & G_{xu} & G_{xv} \\ G_{ux} & G_{uu} & G_{uv} \\ G_{vx} & G_{vu} & G_{vv} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta v \end{bmatrix} G + \begin{bmatrix} G_x & G_y & G_v \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta v \end{bmatrix} \tag{36}$$

$$\begin{aligned} G_{xx} &= A^T S_{yy} A \\ G_{uu} &= B^T S_{yy} B \\ G_{vv} &= C^T S_{yy} C + C^T S_{yv} + S_{vy} C + S_{vv} \\ G_{xu} &= A^T S_{yy} B \end{aligned} \tag{37}$$

$$
\begin{aligned}
G_{xv} &= A^T S_{yy} C + A^T S_{yv} \\
G_{uv} &= B^T S_{yy} C + B^T S_{yv}
\end{aligned}
$$

$$
\begin{aligned}
G_x &= S_y A \\
G_y &= S_y B \\
G_v &= S_y C + S_v
\end{aligned} \tag{38}
$$

7.6 Calculating the Gains

With $\bar{L}$ and $\bar{P}$ we can now find the partials of Q in Eq. 51.

$$
\begin{aligned}
\delta Q(x,u,v) &= \delta\bar{L}(\delta x, \delta u, \delta v) + \delta\bar{P}(\delta x, \delta u, \delta v) \\
&= \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix}^T H \begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix}^T G \begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} + h\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} + g\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix}^T (H+G)\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} + (h+g)\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix}^T \begin{bmatrix}Q_{xx} & Q_{xu} & Q_{xv} \\ Q_{ux} & Q_{uu} & Q_{uv} \\ Q_{vx} & Q_{vu} & Q_{vv}\end{bmatrix}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} + \begin{bmatrix}Q_x & Q_u & Q_v\end{bmatrix}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix}
\end{aligned} \tag{39}
$$

$$
\begin{aligned}
Q_{xx} &= H_{xx} + G_{xx} \\
Q_{xy} &= H_{xy} + G_{xy} \\
&\text{etc } \ldots
\end{aligned}
$$

However, we need S in its square root form U. To accomplish this, we perform recursive rank-1 downdates on W_{xx} using the rows of $W_{uu}^{-1}Q_{ux}$. An example of this process in Julia:

```
    U = LinAlg.Cholesky(Wxx, 'U')
tmp = Wuu'\Qux
for i = 1:size(luu,1)
    U = LinAlg.lowrankdowndate(U,tmp[i,:])
end
U = U[:U]
```

$$
\begin{aligned}
\delta V(x_k) &= Q_x\delta x_k + Q_u\delta x_u + \frac{1}{2}\delta x_k^T Q_{xx}\delta x_k + \frac{1}{2}\delta u_k^T Q_{uu}\delta u_k \\
&\quad + \frac{1}{2}\delta u_k^T Q_{ux}\delta x_k + \frac{1}{2}\delta x_k^T Q_{ux}^T\delta u_k \\
\delta V_k &= Q_x\delta x_k + Q_u(-Q_{uu}^{-1}(Q_u^T + Q_{ux}\delta x_k)) + \frac{1}{2}\delta x_k^T Q_{xx}\delta x_k + \frac{1}{2}(-Q_{uu}^{-1}(Q_u^T + Q_{ux}\delta x_k))^T Q_{uu}(-Q_{uu}^{-1}(Q_u^T + Q_{ux}\delta x_k)) \\
&\quad + \frac{1}{2}(-Q_{uu}^{-1}(Q_u^T + Q_{ux}\delta x_k))^T Q_{ux}\delta x_k + \frac{1}{2}\delta x_k^T Q_{ux}^T(-Q_{uu}^{-1}(Q_u^T + Q_{ux}\delta x_k)) \\
&= Q_x\delta x_k - Q_u Q_{uu}^{-1}Q_u^T - Q_u Q_{uu}^{-1}Q_{ux}\delta x_k \\
&\quad + \frac{1}{2}\delta x_k^T Q_{xx}\delta x_k + \frac{1}{2}(\delta x_k^T Q_{ux}^T + Q_u)Q_{uu}^{-T}(Q_u^T + Q_{ux}\delta x_k) \\
&\quad + \frac{1}{2}(-Q_u Q_{uu}^{-T}Q_{ux}\delta x_k - \delta x_k^T Q_{ux}^T Q_{uu}^{-T}Q_{ux}\delta x_k) + \frac{1}{2}(-\delta x_k^T Q_{ux}^T Q_{uu}^{-1}Q_u^T - \delta x_k^T Q_{ux}^T Q_{uu}^{-1}Q_{ux}\delta x_k) \\
&= Q_x\delta x_k - Q_u Q_{uu}^{-1}Q_u^T - Q_u Q_{uu}^{-1}Q_{ux}\delta x_k \\
&\quad + \frac{1}{2}\delta x_k^T Q_{xx}\delta x_k + \frac{1}{2}(\delta x_k^T Q_{ux}^T Q_{uu}^{-T}Q_u^T + Q_u Q_{uu}^{-T}Q_u^T + \delta x_k^T Q_{ux}^T Q_{uu}^{-T}Q_{ux}\delta x_k + Q_u Q_{uu}^{-T}Q_{ux}\delta x_k)
\end{aligned}
$$

$$
\begin{aligned}
&+ \frac{1}{2}(-Q_u Q_{uu}^{-T} Q_{ux} \delta x_k - \delta x_k^T Q_{ux}^T Q_{uu}^{-T} Q_{ux} \delta x_k) + \frac{1}{2}(-\delta x_k^T Q_{ux}^T Q_{uu}^{-1} Q_u^T - \delta x_k^T Q_{ux}^T Q_{uu}^{-1} Q_{ux} \delta x_k) \\
&= -Q_u Q_{uu}^{-1} Q_u^T + \frac{1}{2}(Q_u Q_{uu}^{-T} Q_u^T) + (Q_x - Q_u Q_{uu}^{-1} Q_{ux} + Q_u Q_{uu}^{-1} Q_{ux} - 1 Q_u Q_{uu}^{-1} Q_{ux}) \delta x_k \\
&\quad + \frac{1}{2} \delta x_k^T (Q_{xx} + Q_{ux}^T Q_{uu}^{-T} Q_{ux} - 2 Q_{ux}^T Q_{uu}^{-T} Q_{ux}) \delta x_k \\
&= -\frac{1}{2}(Q_u Q_{uu}^{-T} Q_u^T) + (Q_x - Q_u Q_{uu}^{-1} Q_{ux}) \delta x_k + \frac{1}{2} \delta x_k^T (Q_{xx} - Q_{ux}^T Q_{uu}^{-T} Q_{ux}) \delta x_k
\end{aligned}
$$

7.7 Active Set - Second-order Multiplier Update

[See: *Constrained Optimization and Lagrange Multiplier Methods* (Bertsekas 1996, Chapter 2.3)]

At each iteration of the backward pass we solve the following minimization:

$$
\min_{x_k, u_k} \quad \mathcal{L}_A(x_k, u_k, \lambda_k) = L(x_k, u_k) + V_{k+1}(x_{k+1}) + \frac{1}{2} c_k(x_k, u_k)^T I_{\mu_k} c_k(x_k, u_k) + \lambda_k^T c_k(x_k, u_k)
$$

Where L is the stage cost and may incorporate infeasible or minimum time costs.

For notational convenience we define $z_k = [x_k, u_k]^T$ and $Q(z_k) = L(z_k) + V_{k+1}(z_{k+1})$. Active constraints are denoted with a bar, for example: $\bar{c}$. All equality and any violating inequality constraints are considered active.

Solving for the KKT conditions:

$$
\begin{aligned}
\frac{\partial \mathcal{L}_A}{\partial z} &= \frac{\partial Q}{\partial z} + \frac{\partial \bar{c}^T}{\partial z} \lambda_k + \frac{\partial \bar{c}^T}{\partial z} \bar{I}_{\mu_k} \bar{c}_k = 0 \\
\frac{\partial \mathcal{L}_A}{\partial \lambda} &= \bar{c} = 0
\end{aligned}
$$

Taking the first order expansion of the KKT conditions:

$$
\begin{aligned}
\frac{\partial \mathcal{L}_A}{\partial z} &\approx \frac{\partial \mathcal{L}_A}{\partial z}|_{z_k, \lambda_k} + \frac{\partial}{\partial z}(\frac{\partial \mathcal{L}_A}{\partial z})|_{z_k, \lambda_k} \delta z + \frac{\partial}{\partial \lambda}(\frac{\partial \mathcal{L}_A}{\partial z})|_{z_k, \lambda_k} \delta \lambda \\
\frac{\partial \mathcal{L}_A}{\partial \lambda} &\approx \frac{\partial \mathcal{L}_A}{\partial \lambda}|_{z_k, \lambda_k} + \frac{\partial}{\partial z}(\frac{\partial \mathcal{L}_A}{\partial \lambda})|_{z_k, \lambda_k} \delta z + \frac{\partial}{\partial \lambda}(\frac{\partial \mathcal{L}_A}{\partial \lambda})|_{z_k, \lambda_k} \delta \lambda
\end{aligned}
$$

The partial derivatives:

$$
\begin{aligned}
\frac{\partial \mathcal{L}_A}{\partial z} &= \frac{\partial Q}{\partial z}|_{z_k, \lambda_k} + (\frac{\partial \bar{c}}{\partial z}^T \lambda_k)|_{z_k, \lambda_k} + (\frac{\partial \bar{c}}{\partial z}^T \bar{I}_{\mu_k} \bar{c})|_{z_k, \lambda_k} \\
\frac{\partial \mathcal{L}_A}{\partial \lambda} &= \bar{c} \\
\frac{\partial}{\partial z}(\frac{\partial \mathcal{L}_A}{\partial \lambda}) &= \frac{\partial}{\partial \lambda}(\frac{\partial \mathcal{L}_A}{\partial z})^T = \frac{\partial \bar{c}}{\partial z}|_{z_k, \lambda_k} \\
\frac{\partial}{\partial z}(\frac{\partial \mathcal{L}_A}{\partial z}) &= \frac{\partial^2 Q}{\partial z^2}|_{z_k, \lambda_k} + (\frac{\partial \bar{c}}{\partial z}^T \bar{I}_{u_k} \frac{\partial \bar{c}}{\partial z})|_{z_k, \lambda_k} \\
\frac{\partial}{\partial \lambda}(\frac{\partial \mathcal{L}_A}{\partial \lambda}) &= 0_{\bar{p} x \bar{p}}
\end{aligned}
$$

Forming the ... system:

$$
\begin{bmatrix} \frac{\partial \mathcal{L}_A^2}{\partial z^2} & \frac{\partial \bar{c}}{\partial z}^T \\ \frac{\partial \bar{c}}{\partial z} & 0_{\bar{p} x \bar{p}} \end{bmatrix} \begin{bmatrix} \delta z \\ \delta \lambda \end{bmatrix} = \begin{bmatrix} -\frac{\partial \mathcal{L}_A}{\partial z} \\ -\bar{c} \end{bmatrix}
$$

$$\frac{\partial \mathcal{L}_A^2}{\partial z^2}\delta z + \frac{\partial \bar{c}}{\partial z}^T \delta\lambda = -\frac{\partial \mathcal{L}_A}{\partial z}$$
$$\frac{\partial \bar{c}}{\partial z}^T \delta z = -\bar{c}$$

If $\frac{\partial \mathcal{L}_A^2}{\partial z^2}$ is invertible and $\frac{\partial \bar{c}}{\partial z}$ is full rank then the system can be solved. Premultiply the first equation by $\frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}$:

$$\frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \mathcal{L}_A^2}{\partial z^2}\delta z + \frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \bar{c}}{\partial z}^T \delta\lambda = -\frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \mathcal{L}_A}{\partial z}$$
$$-\bar{c} + \frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \bar{c}}{\partial z}^T \delta\lambda = -\frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \mathcal{L}_A}{\partial z}$$

$$\delta\lambda = (\frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \bar{c}}{\partial z}^T)^{-1}(\bar{c} - \frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \mathcal{L}_A}{\partial z})$$

This result can be used to update z:

$$\delta z = -\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \mathcal{L}_A}{\partial z}$$

Finally, the active-set second-order multiplier update is:

$$\lambda_{k+1} = \lambda_k + (\frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \bar{c}}{\partial z}^T)^{-1}(\bar{c} - \frac{\partial \bar{c}}{\partial z}\frac{\partial \mathcal{L}_A^2}{\partial z^2}^{-1}\frac{\partial \mathcal{L}_A}{\partial z})|_{z_k,u_k}$$

8 First Order Hold iLQR

This Differential Dynamic Programming formulation of the discrete optimal control problem includes Simpson quadrature stage costs, first-order hold controls, infeasible controls, minimum time, and general nonlinear equality and inequality constraints to solve the following problem:

$$\begin{aligned}
\min_{u,t_f} \quad & \ell_f(x(t_f)) + \int_0^{t_f} \ell(x,u)dt \\
\text{s.t.} \quad & \dot{x}(t) = f(x(t),u(t)) + u_i(t) \\
& x_{min} \le x(t) \le x_{max},\ 0 \le t \le tf \\
& u_{min} \le u(t) \le u_{max} \\
& h(x(t),u(t)) = 0 \\
& g(x(t),u(t)) \le 0 \\
& u_i(t) = 0 \\
& t_{f_{min}} \le t_f \le t_{f_{max}} \\
& x(t_f) = x_f \\
& x(0) = x_0
\end{aligned}$$

Converting to discrete time

$$\begin{aligned}
\min_{\bar{u}_{1:N}} \quad & \ell_N(x_N) + \sum_{k=1}^{N-1} h^2(\frac{1}{6}\ell(x_k,u_k) + \frac{4}{6}\ell(x_m,u_m) + \frac{1}{6}\ell(x_{k+1},u_{k+1})) + h^2 R_m \\
\text{s.t.} \quad & x_{k+1} = f_d(x_k,u_k,u_{k+1}) + u_{i_k} \\
& x_{min} \le x_k \le x_{max},\ \forall k = 1,...,N \\
& u_{min} \le u_k \le u_{max}
\end{aligned}$$

$$
\begin{aligned}
&h_k(x_k, u_k) = 0 \\
&g_k(x_k, u_k) \leq 0 \\
&u_{i_k} = 0, \forall k = 1, ..., N-1 \\
&\sqrt{dt_{min}} \leq h_k \leq \sqrt{dt_{max}} \\
&h_k = h_{k+1}, \forall k = 1, ..., N-2 \\
&x_N = x_f \\
&x_1 = x_0
\end{aligned}
$$

We form and solve the following Augmented Lagrangian:

$$
\begin{aligned}
\mathcal{L}(x_{1:N}, \bar{u}_{1:N}, \lambda) = \frac{1}{2}(x_N - x_f)^T Q_f (x_N - x_f) &+ \sum_{k=1}^{N-1} L_k(x_k, u_k, x_{k+1}, u_{k+1}) \\
&+ \sum_{k=1}^{N-1} R_m h^2 \\
&+ \sum_{k=1}^{N-1} \frac{1}{2} u_{i_k}^T R_i u_{i_k} \\
+\frac{1}{2} c(x_N, u_N)^T I_{\mu_N} c(x_N, u_N) + \lambda_N^T c(x_N, u_N) &+ \sum_{k=1}^{N} \frac{1}{2} c(x_k, u_k, x_{k+1}, u_{k+1})^T I_{\mu_k} c(x_k, u_k, x_{k+1}, u_{k+1}) + \lambda_k^T c(x_k, u_k, x_{k+1}, u_{k+1})
\end{aligned}
$$

We define:

$$
\begin{aligned}
L_k(x_k, u_k, x_{k+1}, u_{k+1}) &= \frac{h^2}{6}(\ell(x_k, u_k) + 4\ell(x_m, u_m) + \ell(x_{k+1}, u_{k+1})) \\
x_m &= \frac{1}{2}(x_k + x_{k+1}) + \frac{h^2}{8}\dot{x}_k - \frac{h^2}{8}\dot{x}_{k+1} \\
u_m &= \frac{u_k + u_{k+1}}{2} \\
h_k &= \sqrt{dt_k} \\
\bar{u}_k &= \begin{bmatrix} u_k \\ h_k \\ u_{i_k} \end{bmatrix} \\
\hat{u}_k &= \begin{bmatrix} u_k \\ h_k \end{bmatrix}
\end{aligned}
$$

With infeasible controls the dynamics become:

$$
x_{k+1} = f_d(x_k, u_k, u_{k+1}, h_k) + u_{i_k}
$$

Constraints are defined as:

$$
c(x_k, \bar{u}_k, x_{k+1}, \bar{u}_{k+1}) = \left[\begin{array}{c} u_k - u_{\max} \\ (h_k - \sqrt{dt_{\max}})[k \neq N] \\ u_{\min} - u_k \\ (\sqrt{dt_{\min}} - h_k)[k \neq N] \\ x_k - x_{\max} \\ x_{\min} - x_k \\ c_I(x_k, u_k) \\ \hline c_E(x_k, u_k) \\ u_{i_k}[k \neq N] \\ (h_k - h_{k+1})[k < N-1] \end{array}\right]
$$

where the horizontal line indicates the separation between inequality (above) and equality (below) constraints. Gradients

$$\nabla_{x_k} c(x_k, \bar{u}_k, x_{k+1}, \bar{u}_{k+1}) = \begin{bmatrix} \mathbf{0_{\bar{m}\times n}} \\ \mathbf{0_{\bar{m}\times n}} \\ I_n \\ -I_n \\ \nabla_x c_I(x_k, u_k) \\ \nabla_x c_E(x_k, u_k) \\ \mathbf{0_{1\times n}} \end{bmatrix}$$

$$\nabla_{\bar{u}_k} c(x_k, \bar{u}_k, x_{k+1}, \bar{u}_{k+1}) = \begin{bmatrix} I_{m_{(\bar{m},\bar{m})[k\neq N]}} & \mathbf{0_{\bar{m}\times n}} \\ -I_{m_{(\bar{m},\bar{m})[k\neq N]}} & \mathbf{0_{\bar{m}\times n}} \\ \mathbf{0_{n\times mm}} & \\ \mathbf{0_{n\times mm}} & \\ [\nabla_{\bar{u}} c_I(x_k, u_k)\ \mathbf{0}] & \\ [\nabla_{\bar{u}} c_E(x_k, u_k)\ \mathbf{0}] & \\ \mathbf{0_{1\times m}} & 1[k < N-1] \quad \mathbf{0_{1\times n}} \end{bmatrix}$$

$$\nabla_{x_{k+1}} c(x_k, \bar{u}_k, x_{k+1}, \bar{u}_{k+1}) = \begin{bmatrix} \mathbf{0_{p\times n}} \end{bmatrix}$$

$$\nabla_{\bar{u}_{k+1}} c(x_k, \bar{u}_k, x_{k+1}, \bar{u}_{k+1}) = \begin{bmatrix} \mathbf{0_{p-1\times mm}} & \\ \mathbf{0_{1\times m}} & -1[k < N-1] \quad \mathbf{0_{1\times n}} \end{bmatrix}$$

8.1 Notation

For convenience we define the following:

$$\begin{aligned} x &= x_k \\ u &= u_k \\ y &= x_{k+1} \\ v &= u_{k+1} \end{aligned}$$

8.2 Backward pass

Normal

$$\begin{aligned} L_x &= \frac{h^2}{6}\ell_{1x} + \frac{4h^2}{6}\ell_{2x} \\ L_u &= \frac{h^2}{6}\ell_{1u} + \frac{4h^2}{6}\ell_{2u} \\ L_y &= \frac{4h^2}{6}\ell_{2y} + \frac{h^2}{6}\ell_{3y} \\ L_v &= \frac{4h^2}{6}\ell_{2v} + \frac{h^2}{6}\ell_{3v} \\ L_{xx} &= \frac{h^2}{6}\ell_{1xx} + \frac{4h^2}{6}\ell_{2xx} \\ L_{uu} &= \frac{h^2}{6}\ell_{1uu} + \frac{4h^2}{6}\ell_{2uu} \\ L_{yy} &= \frac{4h^2}{6}\ell_{2yy} + \frac{h^2}{6}\ell_{3yy} \\ L_{vv} &= \frac{4h^2}{6}\ell_{2vv} + \frac{h^2}{6}\ell_{3vv} \\ L_{xu} &= \frac{4h^2}{6}\ell_{2xu} \\ L_{xy} &= \frac{4h^2}{6}\ell_{2xy} \end{aligned}$$

$$L_{xv} = \frac{4h^2}{6}\ell_{2xv}$$
$$L_{uy} = \frac{4h^2}{6}\ell_{2uy}$$
$$L_{uv} = \frac{4h^2}{6}\ell_{2uv}$$
$$L_{yv} = \frac{4h^2}{6}\ell_{2yv}$$

Minimum time

$$L_x = L_x$$
$$L_{\hat{u}} = \begin{bmatrix} L_u \\ \frac{2h}{6}\ell_1 + L_{2h} + \frac{2h}{6}\ell_3 + 2R_m h \end{bmatrix}$$
$$L_y = L_y$$
$$L_{\hat{v}} = \begin{bmatrix} L_v \\ 0 \end{bmatrix}$$
$$L_{xx} = L_{xx}$$
$$L_{\hat{u}\hat{u}} = \begin{bmatrix} L_{uu} & (\frac{2h}{6}\ell_{1u} + L_{2uh}) \\ (\frac{2h}{6}\ell_{1u} + L_{2hu})^T & (\frac{2}{6}\ell_1 + L_{2hh} + \frac{2}{6}\ell_3 + 2R_m) \end{bmatrix}$$
$$L_{yy} = L_{yy}$$
$$L_{\hat{v}\hat{v}} = \begin{bmatrix} L_{vv} & 0 \\ 0 & 0 \end{bmatrix}$$
$$L_{x\hat{u}} = \begin{bmatrix} L_{xu} & (\frac{2h}{6}\ell_{1x} + L_{2xh}) \end{bmatrix}$$
$$L_{xy} = L_{xy}$$
$$L_{x\hat{v}} = \begin{bmatrix} L_{xv} & 0 \end{bmatrix}$$
$$L_{\hat{u}y} = \begin{bmatrix} L_{yu} & (\frac{2h}{6}\ell_{3y} + L_{2yh}) \end{bmatrix}^T$$
$$L_{\hat{u}\hat{v}} = \begin{bmatrix} L_{vu} & (\frac{2h}{6}\ell_{3v} + L_{2vh}) \\ 0 & 0 \end{bmatrix}^T$$
$$L_{y\hat{v}} = \begin{bmatrix} L_{yv} & 0 \end{bmatrix}^T$$

$$x_{m_h}7 = \frac{2}{8}h(f_c(x,u) - f_c(y,v))$$
$$x_{m_u} = \frac{h^2}{8}\mathcal{B}_u$$
$$x_{m_y} = 0.5I - \frac{h^2}{8}\mathcal{A}_y$$
$$x_{m_v} = \frac{h^2}{8}\mathcal{B}_b$$
$$\ell_{2h} = x_{m_h}^T Q(x_m - x_f)$$
$$\ell_{2hh} = \frac{2}{8}((f_c(x,u) - f_c(y,v))^T Q(x_m - x_f) + h(f_c(x,u) - f_c(y,v))^T Q x_{m_h}$$
$$L_{2h} = \frac{4}{6}(h^2\ell_{2h} + 2h\ell_2)$$
$$L_{2hh} = \frac{4}{6}(2h\ell_{2h} + 2\ell_2 + h^2\ell_{2hh} + 2h\ell_{2h})$$
$$L_{2hu} = \frac{4}{6}(2h\ell_{2u} + \frac{2}{8}h^3(\mathcal{B}_u^T Q(x_m - x_f) + x_{m_u}^T Q(f_c(x,u) - f_c(y,v))))$$
$$L_{2xh} = \frac{4}{6}(2h\ell_{2x} + h^2(0.5I + \frac{h^2}{8} * (A)_x^T Q x_{m_h} + \frac{2}{8}hA_x^T Q(x_m - x_f))$$
$$L_{2uh} = \frac{4}{6}(2h\ell_{2u} + h^2(\frac{h^2}{8}\mathcal{B}_u^T Q x_{m_h} + \frac{2}{8}h\mathcal{B}_u Q(x_m - x_f))$$

$$L_{2yh} = \frac{4}{6}(2h\ell_{2y} + h^2(-\frac{h^2}{8}\mathcal{A}_y^T Q x_{m_h} + \frac{2}{8}h\mathcal{A}_y Q(x_m - x_f))$$
$$L_{2vh} = \frac{4}{6}(2h\ell_{2v} - h^2(\frac{h^2}{8}\mathcal{B}_v^T Q x_{m_h} + \frac{2}{8}h\mathcal{B}_v Q(x_m - x_f))$$

Infeasible
-Note: the bold variable is used to indicate that either u or $\hat{u}$ can be used if the problem is infeasible or minimum time and infeasible, respectively

$$\begin{aligned}
L_x &= L_x \\
L_{\tilde{u}} &= \begin{bmatrix} L_{\mathbf{u}} \\ R_i u_i \end{bmatrix} \\
L_y &= L_y \\
L_{\tilde{v}} &= \begin{bmatrix} L_{\mathbf{v}} \\ 0 \end{bmatrix} \\
L_{xx} &= L_{xx} \\
L_{\tilde{u}\tilde{u}} &= \begin{bmatrix} L_{\mathbf{uu}} & 0 \\ 0 & R_i \end{bmatrix} \\
L_{yy} &= L_{yy} \\
L_{\tilde{v}\tilde{v}} &= \begin{bmatrix} L_{\mathbf{vv}} & 0 \\ 0 & 0 \end{bmatrix} \\
L_{x\tilde{u}} &= \begin{bmatrix} L_{x\mathbf{u}} & 0 \end{bmatrix} \\
L_{xy} &= L_{xy} \\
L_{x\mathbf{v}} &= \begin{bmatrix} L_{x\mathbf{v}} & 0 \end{bmatrix} \\
L_{\tilde{u}y} &= \begin{bmatrix} L_{\mathbf{u}}y \\ 0 \end{bmatrix}^T \\
L_{\tilde{u}\tilde{v}} &= \begin{bmatrix} L_{\mathbf{vu}} & 0 \\ 0 & 0 \end{bmatrix}^T \\
L_{y\tilde{v}} &= \begin{bmatrix} L_{y\mathbf{v}} & 0 \end{bmatrix}^T
\end{aligned}$$

8.3 Preliminaries

Similar to the setup for the regular iLQR problem (Section **??**), we define a generic nonlinear cost function

$$J = \ell_N(x_N) + \int_0^{t_f} \ell(x, u)dt \tag{40}$$

which we can approximate over N discrete time steps using Simpson integration:

$$\begin{aligned}
J &\approx \ell_N(x_N) + \sum_{k=1}^{N-1} \frac{dt}{6}(\ell(x_k, u_k) + 4\ell(x_m, u_m) + \ell(x_{k+1}, u_{k+1})) \\
&= \ell_N(x_N) + \sum_{k=1}^{N-1} L(x_k, u_k, x_{k+1}, u_{k+1})
\end{aligned} \tag{41}$$

where x_m, u_m are the controls at the midpoint of the interval $[k, k+1]$.

We choose to linearly interpolate the controls such that

$$u_m = \frac{u_k + u_{k+1}}{2} \tag{42}$$

and define a function g that such $x_m = g(x_k, u_k, x_{k+1}, u_{k+1})$, which will be derived later.

We discretize the dynamics as before, but now include a dependence on the control at $k+1$ to implement a first order hold on the controls so that our dynamics are now $x_{k+1} = f_d(x_k, u_k, u_{k+1})$, with a first-order Taylor-series approximation:

$$\delta x_{k+1} = A(x_k, u_k, u_{k+1})\delta x_k + B(x_k, u_k, u_{k+1})\delta u_k + C(x_k, u_k, u_{k+1})\delta u_{k+1} \tag{43}$$

8.4 Cost-To-Go

With a given cost function, we can apply Bellman's Principle of Optimality to define the optimal cost-to-go $V_k(x)$ by the recurrence relation:

$$\begin{aligned} V_N &= \ell_f(x_N) \\ V_k &= \min_{u_k, u_{k+1}} \{L(x_k, u_k, x_{k+1}, u_{k+1}) + V_{k+1}(f_d(x_k, u_k, u_{k+1}))\} \end{aligned} \tag{44}$$

However, the minimization is now over controls at time steps k and $k+1$. Since the cost at the previous time step $k-1$ is dependent on the control u_k, we cannot optimize over u_k without taking into account the effect at $k-1$. To avoid this, we define a new control-dependent cost-to-go function:

$$\begin{aligned} P(x_k, u_k) &\equiv \min_{u_{k+1}} \left\{L(x_k, u_k, x_{k+1}, u_{k+1}) + P_{k+1}(x_{k+1}, u_{k+1})\right\} \\ &= \min_{u_{k+1}} \left\{Q(x_k, u_k, x_{k+1}, u_{k+1})\right\} \end{aligned} \tag{45}$$

such that

$$V_k = \min_{u_k} P(x_k, u_k) \tag{46}$$

We now perform a second-order Taylor-series expansion on P to get:

$$\delta P = \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u\end{bmatrix}^T S \begin{bmatrix}\delta x \\ \delta u\end{bmatrix} + s \begin{bmatrix}\delta x \\ \delta u\end{bmatrix} \tag{47}$$

where $s \in \mathbb{R}^{n+m}$ and $S \in \mathbb{R}^{(n+m)\times(n+m)}$ are the gradient and Hessian of δP, similar to their definitions for the normal iLQR case, except that now they are functions of both the state and control and the current time step.

8.5 Terminal Cost-to-go

The terminal cost-to-go is found by taking the the second order Taylor-series approximation of the terminal cost $\ell_N(x_N)$:

$$\begin{aligned} \delta P_N &= \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u\end{bmatrix}^T \begin{bmatrix}\frac{\partial^2 \ell_N}{\partial x^2} & 0 \\ 0 & 0\end{bmatrix} + \begin{bmatrix}\frac{\partial \ell_N}{\partial x} & 0\end{bmatrix} \begin{bmatrix}\delta x \\ \delta u\end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u\end{bmatrix}^T S_N \begin{bmatrix}\delta x \\ \delta u\end{bmatrix} + s_N \begin{bmatrix}\delta x \\ \delta u\end{bmatrix} \end{aligned} \tag{48}$$

8.6 Solving the Dynamic Programming Problem

As before, we find the deviations of the cost-to-go by taking a Taylor-series expansion and plugging in our linearized dynamics (and foregoing subscripts for the more succinct notation defined in Section 8.1) to get

$$\begin{aligned} \delta P_k &= \min_{\delta v} \left\{\delta L(\delta x, \delta u, \delta y, \delta v) + \delta P(\delta y, \delta v)\right\} \\ &= \min_{\delta v} \delta \bar{Q}(\delta x, \delta u, \delta y, \delta v) && (49) \\ &= \min_{\delta v} \delta \bar{Q}(\delta x, \delta u, f(\delta x, \delta u, \delta v), \delta v) && (50) \end{aligned}$$

Substituting in the dynamics produces:

$$\delta Q(\delta x, \delta u, \delta v) = \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix}^T \begin{bmatrix}Q_{xx} & Q_{xu} & Q_{xv} \\ Q_{ux} & Q_{uu} & Q_{uv} \\ Q_{vx} & Q_{vu} & Q_{vv}\end{bmatrix} \begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} + \begin{bmatrix}Q_x & Q_u & Q_v\end{bmatrix} \begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} \tag{51}$$

The terms of the Q are composed of the elements of $L(x, u, y, v)$ and $P(y, v)$. These calculations and the substitution of the dynamics are detailed in Sections to avoid clutter when deriving the dynamic programming step.

To calculate the optimal modification to the nominal control at the next time step δv^*, we take the derivative of δQ with respect to δv and solve for the minimizing δv^*:

$$\begin{aligned} \frac{\partial \delta Q}{\partial v} &= Q_v + Q_{vv}\delta v + Q_{vx}\delta x + Q_{vu}\delta u \\ \delta v^* &= -Q_{vv}^{-1}(Q_{vx}\delta x + Q_{vu}\delta u + Q_v) \\ &\equiv K\delta x + b\delta u + d \end{aligned} \tag{52}$$

It is important to note that the subscripts on the gains match the time index of the control deviation being calculated, not the deviations by which they are multiplied (for more detail see Section 7.6)

By plugging in δv^* into Q we find our approximated cost-to-go which we can propagate backward one time step.

$$\begin{aligned} \delta P = \delta Q^* &= \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u\end{bmatrix}^T \begin{bmatrix} Q_{xx}^* & Q_{xu}^* \\ Q_{ux}^* & Q_{uu}^*\end{bmatrix}\begin{bmatrix}\delta x \\ \delta u\end{bmatrix} + \begin{bmatrix} Q_x^* & Q_u^*\end{bmatrix}\begin{bmatrix}\delta x \\ \delta u\end{bmatrix} \\ = \delta Q^* &= \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u\end{bmatrix}^T S \begin{bmatrix}\delta x \\ \delta u\end{bmatrix} + s\begin{bmatrix}\delta x \\ \delta u\end{bmatrix} \end{aligned} \tag{53}$$

This completes the dynamic programming step. The equations to calculate S and s at the current time step are derived in a following section.

8.6.1 Final time step of backwards recurssion

The recursive steps derived above are completed backwards, starting from the terminal state, as done for regular iLQR. At the final step, however, we have a final optimization. Once we have calculated δP_1 we have

$$\delta V_1 = \min_{\delta u_1} \delta P(x_1, u_1) \tag{54}$$

from Equation 46. This minimization can now be carried out since u_1 does not affect any previous time steps. This problem is identical to that of iLQR:

$$\begin{aligned} \delta u_1^* &= -Q_{uu}^{-1}(Q_{ux}\delta x_1 + Q_x) \\ &\equiv K_1\delta x_1 + d_1 \end{aligned}$$

8.7 Calculating L

We now find a closed-form expression for $\delta L(\delta x, \delta u, \delta y, \delta v)$ in Equation **??**, which is the 2nd order approximation of $L(x, u, y, u)$

8.7.1 Interpolation of states and controls

The state midpoint x_m will be interpolated using a cubic polynomial. This choice is made because we are using RK3 integration (3rd order accuracy) which requires a cubic representation for the interpolation to be the same order of accuracy [reference]. The control midpoint u_m will be linearly interpolated.

The state $x(t)$ over each interval $t_k < t < t_{k+1}$ can be approximated using a cubic polynomial:

$$x(t) \approx at^3 + bt^2 + ct + d$$

with the following boundary conditions:

$$x(0) = x_k$$

$$\begin{aligned}\dot{x}(0) &= f(x_k, u_k)\\ x(dt) &= x_{k+1}\\ \dot{x}(dt) &= f(x_{k+1}, u_{k+1})\end{aligned}$$

The coefficients enter into the system linearly and can be solved for as follows:

$$\begin{bmatrix} x_k \\ \dot{x}_k \\ x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ dt^3 & dt^2 & dt & 1 \\ 3dt^2 & 2dt & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \frac{2}{dt^3} & \frac{1}{dt^2} & \frac{-2}{dt^3} & \frac{1}{dt^2} \\ \frac{-3}{dt^2} & \frac{-2}{dt} & \frac{3}{dt^2} & \frac{-1}{dt} \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix}$$

To evaluate the midpoint x_m:

$$\begin{aligned} x_m = x(dt/2) &= a(dt/2)^3 + b(dt/2)^2 + c(dt/2) + d \\ &= \begin{bmatrix} \frac{dt^3}{8} & \frac{dt^2}{4} & \frac{dt}{2} & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \\ &= \begin{bmatrix} \frac{dt^3}{8} & \frac{dt^2}{4} & \frac{dt}{2} & 1 \end{bmatrix} \begin{bmatrix} \frac{2}{dt^3} & \frac{1}{dt^2} & \frac{-2}{dt^3} & \frac{1}{dt^2} \\ \frac{-3}{dt^2} & \frac{-2}{dt} & \frac{3}{dt^2} & \frac{-1}{dt} \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ x_{k+1} \\ \dot{x}_{k+1} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} x_m &= \frac{1}{2}(x_k + x_{k+1}) + \frac{dt}{8}\dot{x}_k - \frac{dt}{8}\dot{x}_{k+1} \\ &= \frac{1}{2}(x_k + x_{k+1}) + \frac{dt}{8}f(x_k, u_k) - \frac{dt}{8}f(x_{k+1}, u_{k+1}) \\ &= g(x_k, u_k, x_{k+1}, u_{k+1}) \end{aligned}$$

The control midpoint is defined to be:

$$u_m = \frac{u_k + u_{k+1}}{2}$$

8.7.2 Second order expansion of $L(x, u, y, v)$

(see section on Taylor Series expansions)

$$\begin{aligned} \delta L_k = \frac{1}{2} \begin{bmatrix} \delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1} \end{bmatrix}^T \begin{bmatrix} L_{xx} & L_{xu} & L_{xy} & L_{xv} \\ L_{ux} & L_{uu} & L_{uy} & L_{uv} \\ L_{yx} & L_{yu} & L_{yy} & L_{yv} \\ L_{vx} & L_{vu} & L_{vy} & L_{vv} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1} \end{bmatrix} \\ + \begin{bmatrix} L_x & L_u & L_y & L_v \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \\ \delta x_{k+1} \\ \delta u_{k+1} \end{bmatrix} \end{aligned} \tag{55}$$

We can consider the pieces that sum to form $L(x, u, y, v)$ separately (i.e., $\ell(x, u), \ell(x_m, u_m), \ell(y, v)$). First, we consider the second order expansion of $\ell(x, u)$:

$$\ell(x + \delta x, u + \delta u) \approx \ell(x, u)$$

$$
\begin{aligned}
&+ \frac{\partial \ell}{\partial x}|_{x,u}\delta x + \frac{\partial \ell}{\partial u}|_{x,u}\delta u \\
&+ \frac{1}{2}\delta x \frac{\partial^2 \ell}{\partial x^2}|_{x,u}\delta x + \frac{1}{2}\delta u \frac{\partial^2 \ell}{\partial u^2}|_{x,u}\delta u
\end{aligned}
$$

Note that for our quadratic stage cost there are no cross terms so $\frac{\partial \ell^2}{\partial x \partial u}$ and $\frac{\partial \ell^2}{\partial u \partial x}$ are both zero. The terms of interest for our quadratic stage cost are:

$$
\begin{aligned}
\frac{\partial \ell}{\partial x}|_{x,u} &= Q(x - x_f) \\
\frac{\partial \ell}{\partial u}|_{x,u} &= Ru \\
\frac{\partial^2 \ell}{\partial x^2}|_{x,u} &= Q \\
\frac{\partial^2 \ell}{\partial u^2}|_{x,u} &= R
\end{aligned}
$$

The expansion for $\ell(y, v)$ is defined identically, using y and v in the appropriate places.

$$
\begin{aligned}
\ell_{1x} &= Q(x - x_f) \\
\ell_{1u} &= Ru \\
\ell_{1xx} &= Q \\
\ell_{1uu} &= R
\end{aligned}
$$

$$
\begin{aligned}
\ell_{3y} &= Q(y - x_f) \\
\ell_{3v} &= Rv \\
\ell_{3yy} &= Q \\
\ell_{3vv} &= R
\end{aligned}
$$

The expansion of $\ell(x_m, u_m)$ requires application of the matrix chain rule, our cubic interpolation for x_m, linear interpolation for u_m, and linear approximation of the continuous dynamics:

$$
\begin{aligned}
\ell_{2x} &= \frac{\partial x_m}{\partial x}^T \frac{\partial \ell}{\partial x_m} \\
&= (\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(x_m - xf) \\
\ell_{2u} &= \frac{\partial x_m}{\partial u}^T \frac{\partial \ell}{\partial x_m} + \frac{\partial u_m}{\partial u}^T \frac{\partial \ell}{\partial u_m} \\
&= (\frac{dt}{8}\mathcal{B}_u)^T Q(x_m - x_f) + \frac{1}{2}Ru_m \\
\ell_{2y} &= \frac{\partial x_m}{\partial y}^T \frac{\partial \ell}{\partial x_m} \\
&= (\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y)^T Q(x_m - xf) \\
\ell_{2v} &= \frac{\partial x_m}{\partial v}^T \frac{\partial \ell}{\partial x_m} + \frac{\partial u_m}{\partial v}^T \frac{\partial \ell}{\partial u_m} \\
&= (-\frac{dt}{8}\mathcal{B}_v)^T Q(x_m - x_f) + \frac{1}{2}Ru_m
\end{aligned}
$$

$$
\ell_{2xx} = \frac{\partial x_m}{\partial x}^T \frac{\partial^2 \ell}{\partial x_m^2} \frac{\partial x_m}{\partial x}
$$

$$
\begin{aligned}
&= (\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x) \\
\ell_{2uu} &= \frac{\partial x_m}{\partial u}^T \frac{\partial^2 \ell}{\partial x_m^2}\frac{\partial x_m}{\partial u} + \frac{\partial u_m}{\partial u}^T \frac{\partial^2 \ell}{\partial u_m^2}\frac{\partial u_m}{\partial u} \\
&= (\frac{dt}{8}\mathcal{B}_u)^T Q(\frac{dt}{8}\mathcal{B}_u) + \frac{1}{2}R\frac{1}{2} \\
\ell_{2yy} &= \frac{\partial x_m}{\partial y}^T \frac{\partial^2 \ell}{\partial x_m^2}\frac{\partial x_m}{\partial y} \\
&= (\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y)^T Q(\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y) \\
\ell_{2vv} &= \frac{\partial x_m}{\partial v}^T \frac{\partial^2 \ell}{\partial x_m^2}\frac{\partial x_m}{\partial v} + \frac{\partial u_m}{\partial v}^T \frac{\partial^2 \ell}{\partial u_m^2}\frac{\partial u_m}{\partial v} \\
&= (-\frac{dt}{8}\mathcal{B}_v)^T Q(-\frac{dt}{8}\mathcal{B}_v) + \frac{1}{2}R\frac{1}{2}
\end{aligned}
$$

$$
\begin{aligned}
\ell_{2xu} &= \frac{\partial x_m}{\partial x}^T \frac{\partial^2 \ell}{\partial x_m^2}\frac{\partial x_m}{\partial u} \\
&= (\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(\frac{dt}{8}\mathcal{B}_u) \\
\ell_{2xy} &= \frac{\partial x_m}{\partial x}^T \frac{\partial^2 \ell}{\partial x_m^2}\frac{\partial x_m}{\partial y} \\
&= (\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y) \\
\ell_{2xv} &= \frac{\partial x_m}{\partial x}^T \frac{\partial^2 \ell}{\partial x_m^2}\frac{\partial x_m}{\partial v} \\
&= (\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(-\frac{dt}{8}\mathcal{B}_v) \\
\ell_{2uy} &= \frac{\partial x_m}{\partial u}^T \frac{\partial^2 \ell}{\partial x_m^2}\frac{\partial x_m}{\partial y} \\
&= (\frac{dt}{8}\mathcal{B}_u)^T Q(\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y) \\
\ell_{2uv} &= \frac{\partial x_m}{\partial u}^T \frac{\partial^2 \ell}{\partial x_m^2}\frac{\partial x_m}{\partial v} + \frac{\partial u_m}{\partial u}^T \frac{\partial^2 \ell}{\partial u_m^2}\frac{\partial u_m}{\partial v} \\
&= (\frac{dt}{8}\mathcal{B}_u)^T Q(-\frac{dt}{8}\mathcal{B}_v) + \frac{1}{2}R\frac{1}{2} \\
\ell_{2yv} &= \frac{\partial x_m}{\partial y}^T \frac{\partial^2 \ell}{\partial x_m^2}\frac{\partial x_m}{\partial v} \\
&= (\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y)^T Q(-\frac{dt}{8}\mathcal{B}_v)
\end{aligned}
$$

Collect the expanded terms from $\ell(x,u), \ell(x_m,u_m), \ell(y,v)$. Importantly, multiply by the correct scaling factor $\frac{dt}{6}$ or $\frac{4dt}{6}$:

$$
\begin{aligned}
L_x &= \frac{dt}{6}Q(x - x_f) + \frac{4dt}{6}(\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(x_m - xf) \\
L_u &= \frac{dt}{6}Ru + \frac{4dt}{6}\Big((\frac{dt}{8}\mathcal{B}_u)^T Q(x_m - x_f) + \frac{1}{2}Ru_m\Big) \\
L_y &= \frac{dt}{6}Q(y - x_f) + \frac{4dt}{6}(\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y)^T Q(x_m - xf) \\
L_v &= \frac{dt}{6}Rv + \frac{4dt}{6}\Big((-\frac{dt}{8}\mathcal{B}_v)^T Q(x_m - x_f) + \frac{1}{2}Ru_m\Big)
\end{aligned}
$$

$$
\begin{aligned}
L_{xx} &= \frac{dt}{6}Q + \frac{4dt}{6}(\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x) \\
L_{uu} &= \frac{dt}{6}R + \frac{4dt}{6}\Big((\frac{dt}{8}\mathcal{B}_u)^T Q(\frac{dt}{8}\mathcal{B}_u) + \frac{1}{2}R\frac{1}{2}\Big) \\
L_{yy} &= \frac{dt}{6}Q + \frac{4dt}{6}(\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y)^T Q(\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y) \\
L_{vv} &= \frac{dt}{6}R + \frac{4dt}{6}\Big((-\frac{dt}{8}\mathcal{B}_v)^T Q(-\frac{dt}{8}\mathcal{B}_v) + \frac{1}{2}R\frac{1}{2}\Big)
\end{aligned}
$$

$$
\begin{aligned}
L_{xu} &= \frac{4dt}{6}(\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(\frac{dt}{8}\mathcal{B}_u) \\
L_{xy} &= \frac{4dt}{6}(\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y) \\
L_{xv} &= \frac{4dt}{6}(\frac{1}{2}I + \frac{dt}{8}\mathcal{A}_x)^T Q(-\frac{dt}{8}\mathcal{B}_v) \\
L_{uy} &= \frac{4dt}{6}(\frac{dt}{8}\mathcal{B}_u)^T Q(\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y) \\
L_{uv} &= \frac{4dt}{6}\Big(\frac{dt}{8}\mathcal{B}_u)^T Q(-\frac{dt}{8}\mathcal{B}_v) + \frac{1}{2}R\frac{1}{2}\Big) \\
L_{yv} &= \frac{4dt}{6}(\frac{1}{2}I - \frac{dt}{8}\mathcal{A}_y)^T Q(-\frac{dt}{8}\mathcal{B}_v)
\end{aligned}
$$

8.7.3 Substituting in the Dynamics

Prior to taking the summation of the stage cost δL and the cost-to-go δP to form $\delta Q(\delta x, \delta u, \delta v) = \delta L + \delta P$ we will substitute in dynamics

First, we substitute the linearized discrete dynamics into $\delta P(\delta y, \delta v) \rightarrow \delta P(\delta x, \delta u, \delta v)$:

$$
\delta P(\delta y, \delta v) = \frac{1}{2}\begin{bmatrix}\delta y \\ \delta v\end{bmatrix}^T \begin{bmatrix}S_{yy} & S_{yv} \\ S_{vy} & S_{vv}\end{bmatrix}\begin{bmatrix}\delta y \\ \delta v\end{bmatrix} + \begin{bmatrix}S_y & S_v\end{bmatrix}\begin{bmatrix}\delta y \\ \delta v\end{bmatrix} \tag{56}
$$

$$
\begin{aligned}
\delta P(\delta x, \delta u, \delta v) = \frac{1}{2}&\begin{bmatrix}A\delta x + B\delta u + C\delta v \\ \delta v\end{bmatrix}^T \begin{bmatrix}S_{yy} & S_{yv} \\ S_{vy} & S_{vv}\end{bmatrix}\begin{bmatrix}A\delta x + B\delta u + C\delta v \\ \delta v\end{bmatrix} \\
&+ \begin{bmatrix}S_y & S_v\end{bmatrix}\begin{bmatrix}A\delta x + B\delta u + C\delta v \\ \delta v\end{bmatrix}
\end{aligned} \tag{57}
$$

$$
= \frac{1}{2}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix}^T \begin{bmatrix}G_{xx} & G_{xu} & G_{xv} \\ G_{ux} & G_{uu} & G_{uv} \\ G_{vx} & G_{vu} & G_{vv}\end{bmatrix}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} G + \begin{bmatrix}G_x & G_y & G_v\end{bmatrix}\begin{bmatrix}\delta x \\ \delta u \\ \delta v\end{bmatrix} \tag{58}
$$

$$
\begin{aligned}
G_{xx} &= A^T S_{yy} A \\
G_{uu} &= B^T S_{yy} B \\
G_{vv} &= C^T S_{yy} C + C^T S_{yv} + S_{vy} C + S_{vv} \\
G_{xu} &= A^T S_{yy} B \\
G_{xv} &= A^T S_{yy} C + A^T S_{yv} \\
G_{uv} &= B^T S_{yy} C + B^T S_{yv}
\end{aligned} \tag{59}
$$

$$
\begin{aligned}
G_x &= A^T S_y \\
G_y &= B^T S_y \\
G_v &= C^T S_y + S_v
\end{aligned} \tag{60}
$$

Similarly, we substitute the linearized discrete dynamics into $\delta L(\delta x, \delta u, \delta y, \delta v) \rightarrow \delta L(\delta x, \delta u, \delta v)$:

$$\delta L_k = \frac{1}{2}\begin{bmatrix} \delta x \\ \delta u \\ A\delta x + B\delta u + C\delta v \\ \delta v \end{bmatrix}^T \begin{bmatrix} L_{xx} & L_{xu} & L_{xy} & L_{Lxv} \\ L_{ux} & L_{uu} & L_{uy} & L_{Luv} \\ L_{yx} & L_{yu} & L_{yy} & L_{Lyv} \\ L_{vx} & L_{vu} & L_{vy} & L_{Lvv} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ A\delta x + B\delta u + C\delta v \\ \delta v \end{bmatrix} + \begin{bmatrix} L_x & L_u & L_y & L_v \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ A\delta x + B\delta u + C\delta v \\ \delta v \end{bmatrix} \tag{61}$$

This simplifies to: (62)

$$\delta L_k = \begin{bmatrix} \delta x \\ \delta u \\ \delta v \end{bmatrix}^T \frac{1}{2}\begin{bmatrix} H_{xx} & H_{xu} & H_{xv} \\ H_{ux} & H_{uu} & H_{uv} \\ H_{vx} & H_{vu} & H_{vv} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta v \end{bmatrix} + \begin{bmatrix} H_x & H_y & H_v \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta v \end{bmatrix}$$

where: (63)

$$\begin{aligned} H_{xx} &= L_{xx} + L_{xy}A + A^T L_{yx} + A^T L_{yy} A \\ H_{uu} &= L_{uu} + L_{uy}B + B^T L_{yu} + B^T L_{yy} B \\ H_{vv} &= L_{vv} + L_{vy}C + C^T L_{yv} + C^T L_{yy} C \\ H_{xu} &= L_{xu} + L_{xy}B + A^T L_{yu} + A^T L_{yy} B \\ H_{xv} &= L_{xv} + L_{xy}C + A^T L_{yv} + A^T L_{yy} C \\ H_{uv} &= L_{uv} + L_{uy}C + B^T L_{yv} + B^T L_{yy} C \end{aligned} \tag{64}$$

$$\begin{aligned} H_x &= L_x + A^T L_y \\ H_u &= L_u + B^T L_y \\ H_v &= L_v + C^T L_y \end{aligned} \tag{65}$$

This yields

$$\delta Q(\delta x, \delta u, \delta v) = \frac{1}{2}\begin{bmatrix} \delta x \\ \delta u \\ \delta v \end{bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} & Q_{xv} \\ Q_{ux} & Q_{uu} & Q_{uv} \\ Q_{vx} & Q_{vu} & Q_{vv} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta v \end{bmatrix} + \begin{bmatrix} Q_x & Q_u & Q_v \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta v \end{bmatrix} \tag{66}$$

where:

$$\begin{aligned} Q_{xx} &= G_{xx} + H_{xx} \\ Q_{uu} &= G_{uu} + H_{uu} \\ Q_{vv} &= G_{vv} + H_{vv} \\ Q_{xu} &= G_{xu} + H_{xu} \\ Q_{xv} &= G_{xv} + H_{xv} \\ Q_{uv} &= G_{uv} + H_{uv} \end{aligned} \tag{67}$$

$$\begin{aligned} Q_x &= G_x + H_x \\ Q_u &= G_u + H_u \\ Q_v &= G_v + H_v \end{aligned} \tag{68}$$

8.8 Calculating S and s

With the gains obtained by optimizing over the control at the next timestep, we are ready to calculate the cost-to-go Hessian and gradient at the current time step:

$$\begin{aligned}
\delta P(x,u) &= \frac{1}{2}\begin{bmatrix} \delta x \\ \delta u \\ K\delta x + b\delta u + d \end{bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} & Q_{xv} \\ Q_{ux} & Q_{uu} & Q_{uv} \\ Q_{vx} & Q_{vu} & Q_{vv} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ K\delta x + b\delta u + d \end{bmatrix} + \begin{bmatrix} Q_x & Q_u & Q_v \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ K\delta x + b\delta u + d \end{bmatrix} \\
&= Q_x\delta x + Q_u\delta u + (Q_vK\delta x + Q_vb\delta u + Q_vd) \\
&\quad + \frac{1}{2}(\delta x^T Q_{xx}\delta x + \delta u^T Q_{uu}\delta u) \\
&\quad + \frac{1}{2}(\delta x^T Q_{xu}\delta u + \delta u^T Q_{ux}\delta x) \\
&\quad + \frac{1}{2}(\delta x^T Q_{xv}K\delta x + \delta x^T Q_{xv}b\delta u + \delta x^T Q_{xv}d) \\
&\quad + \frac{1}{2}(\delta u^T Q_{uv}K\delta x + \delta u^T Q_{uv}b\delta u + \delta u^T Q_{uv}d) \\
&\quad + \frac{1}{2}(\delta x^T K^T Q_{vx}\delta x + \delta u^T b^T Q_{vx}\delta x + d^T Q_{vx}\delta x) \\
&\quad + \frac{1}{2}(\delta x^T K^T Q_{vu}\delta u + \delta u^T b^T Q_{vu}\delta u + d^T Q_{vx}\delta u) \\
&\quad + \frac{1}{2}(\delta x^T K^T Q_{vv}K\delta x + \delta x^T K^T Q_{vv}b\delta u + \delta x^T K^T Q_{vv}d) \\
&\quad + \frac{1}{2}(\delta u^T b^T Q_{vv}K\delta x + \delta u^T b^T Q_{vv}b\delta u + \delta u^T b^T Q_{vv}d) \\
&\quad + \frac{1}{2}(d^T Q_{vv}K\delta x + d^T Q_{vv}b\delta u + d^T Q_{vv}d) \\
&= \frac{1}{2}\begin{bmatrix} \delta x \\ \delta u \end{bmatrix} \begin{bmatrix} Q^*_{xx} & Q^*_{xu} \\ Q^*_{ux} & Q^*_{uu} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix} + \begin{bmatrix} Q^*_x & Q^*_u \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix} \\
&= \frac{1}{2}\begin{bmatrix} \delta x \\ \delta u \end{bmatrix}^T S_k \begin{bmatrix} \delta x \\ \delta u \end{bmatrix} + s_k \begin{bmatrix} \delta x \\ \delta u \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
Q_{xx}* &= Q_{xx} + Q_{xv}K + K^TQ_{vx} + K^TQ_{vv}K \\
Q_{uu}* &= Q_{uu} + Q_{vu}b + b^TQ_{vu} + b^TQ_{vv}b \\
Q_{ux}* &= Q_{ux} + Q_{uv}K + b^TQ_{vx} + b^TQ_{vv}K
\end{aligned}$$

$$\begin{aligned}
Q^*_x &= Q_x + Q_vK + d^TQ_{vx} + d^TQ_{vv}K \\
Q^*_u &= Q_u + Q_vb + D^TQ_{vu} + d^TQ_{vv}b
\end{aligned}$$

$$\Delta Q = Q_vd + \frac{1}{2}d^TQ_{vv}d$$

8.9 Forward Pass

8.9.1 Initial Rollout

We begin the first rollout with a set of controls, $U \in \mathbb{R}^{m\times N}$ and an initial state, x_0. To calculate the state at the next time step we use the discrete dynamics with a first-order hold on the control:

$$x_{k+1} = f_d(x_k, u_k, u_{k+1}), \text{ for } k = 1, \ldots, N-1 \tag{69}$$

8.9.2 Rollout with Gains

We assume we have a control U and a trajectory X from the previous iteration, along with the gains $K \in \mathbb{R}^{m\times n\times N}, b \in \mathbb{R}^{m\times m\times N}$ and $d \in \mathbb{R}^{m\times N}$. (Note: $\alpha \in [0,1]$ is the line search paramter)

Prior to iterating over the time steps we calculate the deviation of the first control:

$$\begin{aligned} \delta u_1 &= K_1 \delta x_1 + \alpha d_1 \\ &= \alpha d_1 \\ \bar{u}_1 &= u_1 + \delta u_1 \end{aligned} \tag{70}$$

since $\delta x_1 = 0$.

We can then perform the rollout as follows for $k = 1, \ldots, N-1$:

$$\begin{aligned} \delta x_k &= \bar{x}_k - x_k \\ \delta u_{k+1} &= K_{k+1}\delta x_k + b_{k+1}\delta u_k + \alpha d_{k+1} \\ \bar{u}_{k+1} &= u_{k+1} + \delta u_{k+1} \\ \bar{x}_{k+1} &= f_d(\bar{x}_k, \bar{u}_k, \bar{u}_{k+1}) \end{aligned}$$

8.10 Regularization

As before, there are two types of regularization that can be employed: control and state:

Option 1:

$$\tilde{Q}_{vv} = Q_{vv} + \rho I$$

$$\begin{aligned} K &= -\tilde{Q}_{vv}^{-1} Q_{vx} \\ b &= -\tilde{Q}_{vv}^{-1} Q_{vu} \\ d &= -\tilde{Q}_{vv}^{-1} Q_v \end{aligned}$$

$$\begin{aligned} Q^*_{xx} &= Q_{xx} + Q_{xv}K + K^T Q_{vx} + K^T Q_{vv} K \\ Q^*_{uu} &= Q_{uu} + Q_{vu}b + b^T Q_{vu} + b^T Q_{vv} b \\ Q^*_{ux} &= Q_{ux} + Q_{uv}K + b^T Q_{vx} + b^T Q_{vv} K \end{aligned}$$

$$\begin{aligned} Q^*_x &= Q_x + K^T Q_v + Q_{xv} d + K^T Q_{vv} d \\ Q^*_u &= Q_u + b^T Q_v + Q_{uv} d + b^T Q_{vv} d \end{aligned}$$

$$\Delta Q = Q_v d + \frac{1}{2} d^T Q_{vv} d$$

Option 2:

$$\begin{aligned} \tilde{Q}_{vv} &= L_{vv} + L_{yv}^T C + C^T L_{yv} + C^T L_{yy} C + C^T (S_{yy} + \rho I) C + C^T S_{yv} + S_{yv}^T C + S_{vv} \\ \tilde{Q}_{xv} &= L_{xv} + L_{xy} C + A^T L_{yv} + A^T L_{yy} C + A^T (S_{yy} + \rho I) C + A^T S_{yv} \\ \tilde{Q}_{uv} &= L_{uv} + L_{uy} C + B^T L_{yv} + B^T L_{yy} C + B^T (S_{yy} + \rho I) C + B^T S_{yv} \end{aligned}$$

$$\begin{aligned} K &= -\tilde{Q}_{vv}^{-1} \tilde{Q}_{vx} \\ b &= -\tilde{Q}_{vv}^{-1} \tilde{Q}_{vu} \\ d &= -\tilde{Q}_{vv}^{-1} Q_v \end{aligned}$$

$$
\begin{aligned}
Q^*_{xx} &= Q_{xx} + Q_{xv}K + K^T Q_{vx} + K^T Q_{vv} K \\
Q^*_{uu} &= Q_{uu} + Q_{vu}b + b^T Q_{vu} + b^T Q_{vv} b \\
Q^*_{ux} &= Q_{ux} + Q_{uv}K + b^T Q_{vx} + b^T Q_{vv} K
\end{aligned}
$$

$$
\begin{aligned}
Q^*_{x} &= Q_{x} + K^T Q_{v} + Q_{xv}d + K^T Q_{vv} d \\
Q^*_{u} &= Q_{u} + b^T Q_{v} + Q_{uv}d + b^T Q_{vv} d
\end{aligned}
$$

$$
\Delta Q = Q_v d + \frac{1}{2} d^T Q_{vv} d
$$

For both regularization types, when timestep $k = 1$:

$$
\tilde{Q}^*_{uu} = Q^*_{uu} + \rho I
$$

$$
\begin{aligned}
K_1 &= -(\tilde{Q}^*_{uu})^{-1} Q^*_{xu} \\
b_1 &= 0 \\
d_1 &= -(\tilde{Q}^*_{uu})^{-1} Q^*_{u}
\end{aligned}
$$

8.11 Constraints (Augmented Lagrange Method)

Previously, we defined the constraint stage cost as:

$$
g(x_k, u_k) = \frac{1}{2} c_k(x_k, u_k)^T I_{\mu_k} c_k(x_k, u_k) + \lambda_k^T c_k(x_k, u_k)
$$

While we are using Simpson integration to approximate the state and control stage cost, here we use trapezoidal integration (which is not dependent on the midpoint state).

$$
\begin{aligned}
\mathcal{L}_A(X, U; \mu, \lambda) = \ell_f(x_N) & \\
&+ \frac{1}{2} c_N(x_N)^T I_{\mu_N} c_N(x_N) + \lambda_N^T c_N(x_N) \\
&+ \sum_{k=1}^{N-1} \frac{dt}{6} \Big(\ell(x_k, u_k) + 4\ell(x_m, u_m) + \ell(x_{k+1}, u_{k+1}) \Big) \\
&+ \sum_{k=1}^{N} g(x_k, u_k)
\end{aligned}
$$

Importantly, we must modify the backward pass. We defined $\delta \bar{Q}(x, u, y, v)$ for the first order holdand will modify it in a similar manner to the Section 3.2:

$$
\begin{aligned}
\hat{\bar{Q}}_x &= \bar{Q}_x + c_x^T I_\mu c + c_x^T \lambda \\
\hat{\bar{Q}}_u &= \bar{Q}_u + c_u^T I_\mu c + c_u^T \lambda
\end{aligned}
$$

$$
\begin{aligned}
\hat{\bar{Q}}_{xx} &= \bar{Q}_{xx} + c_x^T I_\mu c_x \\
\hat{\bar{Q}}_{uu} &= \bar{Q}_{uu} + c_u^T I_\mu c_u
\end{aligned}
$$

$$
\hat{\bar{Q}}_{xu} = \bar{Q}_{xu} + c_x^T I_\mu c_u
$$

The boundary conditions are augmented as well, similar to Section 3.2, they fill the P Hessian and gradient.

$$
\hat{s}_x = s_x + (c_{x_N}^T I_{\mu_N} c_N + c_{x_N}^T \lambda_N)
$$

$$\hat{s}_u = s_u + (c_{u_N}^T I_{\mu_N} c_N + c_{u_N}^T \lambda_N)$$

$$\hat{S}_{xx} = S_{xx} + (c_{x_N}^T I_{\mu_N} c_{x_N}) + (c_x^T I_\mu c_x)$$
$$\hat{S}_{uu} = (c_u^T I_\mu c_u)$$

$$\hat{S}_{xu} = (c_x^T I_\mu c_u)$$
$$\hat{S}_{ux} = (c_u^T I_\mu c_x)$$

9 Minimum Time: NOTE: THIS FORMULATION IS NOT UP TO DATE

It is often desirable to minimize the time required to reach the goal state. Minimum-time policies very often results in "bang-bang" control policies, where the control is saturated at the limits. It is therefore imperative to specify control bounds when solving minimum time problems. Here we set forth the necessary modifications to the iLQR algorithm to compute minimum-time policies.

9.1 Notation

As will be shown, it is natural to treat the time step dt as an additional control. We then define the following:

1. $u \in \mathbb{R}^m$ = vector of controls for the original problem
2. $\hat{u} \in \mathbb{R}^{\hat{m}}$ = controls augmented with time

In general, any value related to the controls with a hat will denote values augmented with the time step.

9.2 General Methodology

In order to minimize over time, we must make time a decision variable. It is very important that all decision variables are only locally coupled between time-steps in order to leverage the sparsity structure of the Hessian and solve the problem using dynamic programming. Therefore, rather than have a single decision variable represent the total time or even the time step, we assign a separate time step variable to each time step: dt_k = the time between time step k and $k+1$, i.e. $dt_k = t_{k+1} - t_k$. We then constrain each of these values to be positive and less than some maximum threshold: $0 < dt_k < dt_{\max}$.

We also want the time steps to be equal, since we want to avoid having the algorithm exploit the time step to artificially lower the cost. We then impose $N-2$ constraints: $dt_k = dt_{k+1}, \quad k = 1, \ldots, N-2$ to set the time steps equal.

It is also imperative that we never allow the time steps to become negative when plugged into the dynamics, since this effectives "reverses" time and will result in strange behavior. To ensure dt is strictly positive we store the square root of the time step, which is then squared whenever plugged into dynamics, ensuring strict positivity. This these high-level insights and design decision, we formulate the problem.

9.3 Problem Formulation

9.3.1 Cost Function

We first set up the cost function, or Lagrangian since we are solving a constrained problem:

$$J = \ell(x_N) + \sum_{k=1}^{N-1} (\ell(x_k, u_k) + cdt)dt$$

we now define $h := \sqrt{dt}$ so that

$$J = \ell(x_N) + \sum_{k=1}^{N-1} (\ell(x_k, u_k) + ch^2)h^2$$

9.3.2 Optimization Problem

Using the previous cost function we now formulate the optimization problem:

$$\begin{aligned}
\min_{u_k, h_k \forall k} \quad & \ell(x_N) + \sum_{k=1}^{N-1} (\ell(x_k, u_k) + ch^2)h^2 \\
\text{subject to} \quad & x_{lb} \leq x_k \leq x_{ub}, \ \forall k \\
& u_{lb} \leq u_k \leq u_{ub}, \ \forall k \\
& x_N = x_f \\
& 0 \leq h_k \leq \sqrt{dt_{\max}}, \ \forall k \\
& h_k = h_{k+1}, \ \forall k
\end{aligned}$$

Note that enforcing equality between h_k, h_{k+1} and dt_k, dt_{k+1} are equivalent, so we avoid unnecessary squares in the constraints by simply enforcing equality on h.

9.3.3 Augmented Lagrangian

We use the Augmented Lagrangian method to convert the problem to an unconstrained optimization problem. As mentioned previously, we can conveniently treat h_k as an additional control and form $\hat{u}_k = [u_k^T \ h_k]^T$. The Lagrangian is formed as usual, except this time we special case the equality constraint on the time steps:

$$\mathcal{L}_A = \ell(x_N) + c(x_N)^T I_{\mu,N} c(x_N) + \lambda_N^T c(x_N) + \sum_{k=1}^{N-1} \left[(\ell(x_k, u_k) + \frac{1}{2}ch^2)h^2 + \frac{1}{2}c(x_k, \hat{u}_k)^T I_{\mu,k} c(x_k, \hat{u}_k) + \lambda_k^T c(x_k, \hat{u}_k) \right] + \sum_{k=1}^{N-2} \frac{1}{2}\mu_k^t (h_k - h_{k+1})^2 + \lambda_k^t (h_k - h_{k+1})$$

which turns our problem into:

$$\min_{u_k, h_k, \lambda_k, \lambda_k^t \forall k} \quad \mathcal{L}_A$$

We define $c(x_k, \hat{u}_k)$ in the next section.

9.3.4 Discrete Dynamics

It is also necessary to account for the impact of the changing dt on the dynamics. In general, the discrete dynamics are a function of the state, control, and time step:

$$x_{k+1} = f(x_k, u_k, dt_k) = f(x_k, u_k, h_k^2)$$

So when taking the first-order Taylor Series expansion we also take the expansion about h_k:

$$\begin{aligned}
\delta x_{k+1} &= \left.\frac{\partial f}{\partial x}\right|_{x_k, u_k, h_k} \delta x_k + \left.\frac{\partial f}{\partial u}\right|_{x_k, u_k, u_k} \delta u_k + \left.\frac{\partial f}{\partial h}\right|_{x_k, u_k, h_k} \delta h_k \\
&= A(x_k, u_k, h_k)\delta x_k + B(x_k, u_k, h_k)\delta u_k + H(x_k, u_k, h_k)\delta h_k \\
&= A(x_k, \hat{u}_k)\delta x_k + \hat{B}(x_k, \hat{u}_k)
\end{aligned}$$

where $\hat{B}(x_k, \hat{u}_k) = \begin{bmatrix} B(x_k, u_k, h_k) & H(x_k, u_k, h_k) \end{bmatrix}$.

9.4 Backwards Pass

With these changes, we just treat the time step as an additional control, so nothing changes about the derivation of the Dynamic Programming steps to calculate the closed-loop gains during the backwards pass. However, since we now have a different Lagrangian, we need to be careful to correctly the specify the expansion for the augmented controls.

9.4.1 Terminal Cost-to-Go

Nothing changes here, since the Lagrangian at the terminal state is not a function of dt. So,

$$s_N = \frac{\partial \ell_N}{\partial x} + \nabla_x c(x_N)^T (I_{\mu,N}) c(x_N) + \nabla_x^T \lambda_N$$
$$S_N = \frac{\partial^2 \ell_N}{\partial x^2} + \nabla_x c(x_N)^T (I_{\mu,N}) \nabla_x c(x_N)$$

9.4.2 Dynamic Programming Step

Similar to before, our goal is to find the cost-to-go at step k:

$$\delta V_k = \min_{u_k, h_k} \{\delta Q_k(x_k, u_k, h_k)\}$$

where

$$\begin{aligned}\delta Q = Q_x \delta x + Q_u \delta u + Q_h \delta h + \\ \delta x^T Q_{xx} \delta x + \delta u^T Q_{uu} \delta u + \delta h^T Q_{hh} \delta h + \\ \delta x^T Q_{xu} \delta u + \delta x^T Q_{xh} \delta h + \delta u^T Q_{uh} \delta h\end{aligned}$$

and $Q_{xx} = \mathcal{L}_{xx} + \nabla_{xx} V_{k+1}(f(x_k, u_k, h_k))$ and so forth for the other terms of the expansion.

We re-state the Lagrangian here for reference, prior to defining the partials:

$$\mathcal{L}_A = \ell(x_N) + c(x_N)^T I_{\mu,N} c(x_N) + \lambda_N^T c(x_N) + \sum_{k=1}^{N-1} \left[(\ell(x_k, u_k) + \frac{1}{2} c h^2) h^2 + \frac{1}{2} c(x_k, \hat{u}_k)^T I_{\mu,k} c(x_k, \hat{u}_k) + \lambda_k^T c(x_k, \hat{u}_k)\right] +$$
$$\sum_{k=1}^{N-2} \frac{1}{2} \mu_k^t (h_k - h_{k+1})^2 + \lambda_k^t (h_k - h_{k+1})$$

Taking the expansion of the Lagrangian and the cost-to-go we get the following terms. To simplify the notation, any variable without a subscript is assumed to at the current time step k, and terms with a superscript (+) or (-) are the values at the next $(k+1)$ or previous $(k-1)$ time step.

$$\begin{aligned}\mathcal{L}_x &= \ell_x h^2 + c_x^T I \mu c + c_x^T \lambda \\ \mathcal{L}_u &= \ell_u h^2 + c_u^T I \mu c + c_u^T \lambda \\ \mathcal{L}_h &= 2\ell h + 2ch^3 + c_h^T I \mu c + c_h^T \lambda + \mu_t (h - h^+) + \lambda_t - \mu^t (h^- - h) - \lambda_t^-\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{xx} &= \ell_{xx} h^2 + c_x^T I \mu c_x \\ \mathcal{L}_{uu} &= \ell_{uu} h^2 + c_u^T I \mu c_u \\ \mathcal{L}_{hh} &= 2\ell + 6ch^2 + c_h^T I \mu c_h + \mu_t + \mu_t^-\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{xu} &= \ell_{xu} h^2 + c_x^T I \mu c_u \\ \mathcal{L}_{xh} &= 2\ell_x h + c_x^T I \mu c_h \\ \mathcal{L}_{uh} &= 2\ell_u h + c_u^T I \mu c_h\end{aligned}$$

by augmenting the control with dt we get the following

$$\begin{aligned}\mathcal{L}_x &= \ell_x h^2 + c_x^T I \mu c + c_x^T \lambda \\ \mathcal{L}_{\hat{u}} &= \ell_u h^2 + c_{\hat{u}}^T I \mu c + c_{\hat{u}}^T \lambda + \begin{bmatrix}\mathbf{0} & 2\ell h + 2ch^3 + \mu_t (h - h^+) + \lambda_t - \mu^t (h^- - h) - \lambda_t^-\end{bmatrix}\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{xx} &= \ell_{xx} h^2 + c_x^T I \mu c_x \\ \mathcal{L}_{\hat{u}\hat{u}} &= \ell_{\hat{u}\hat{u}} h^2 + c_u^T I \mu c_u + \begin{bmatrix}\mathbf{0} & 2h\ell_u \\ 2h\ell_u^T & 2\ell + 6ch^2 + \mu_t + \mu_t^-\end{bmatrix}\end{aligned}$$

$$\mathcal{L}_{x\hat{u}} = \ell_{x\hat{u}} h^2 + c_x^T I \mu c_{\hat{u}} + \begin{bmatrix}\mathbf{0} & 2h\ell_x\end{bmatrix}$$

and

$$
\begin{aligned}
Q_x &= \mathcal{L}_x + s^+ A \\
Q_{\hat{u}} &= \mathcal{L}_{\hat{u}} + s^+ \hat{B} \\
Q_{xx} &= \mathcal{L}_{xx} + A^T S^+ A \\
Q_{\hat{u}\hat{u}} &= \mathcal{L}_{\hat{u}\hat{u}} + \hat{B}^T S^+ \hat{B} \\
Q_{x\hat{u}} &= \mathcal{L}_{x\hat{u}} + A^T s^+ \hat{B}
\end{aligned}
$$

The gains can then be readily calculated used an identical procedure as in the non-minimum time case by using these augmented partials of Q.

9.5 Forward Pass

The forward pass is identical, just make sure to extract out the updated h and plug into the dynamics, i.e.

$$
\begin{aligned}
\delta x &= \bar{x}_k - x_k \\
\bar{u}_k &= u_k + K_k \delta x_k + \alpha d_k \\
\bar{h}_k &= \bar{u}_k[\bar{m}] \\
\bar{x}_{k+1} &= f(\bar{x}_k, \bar{u}_k[1:m], \bar{h})
\end{aligned}
$$

where the square brackets are vector indexing operations (similar to syntax used in Julia or Python). Constraints: In the previous section we used $c(x_k, \hat{u}_k) \in \mathbb{R}^p$, which is equivalent to $c(x_k, u_k, h_k)$, which we define as:

$$
c(x,u,h) = \left[\begin{array}{c} u - u_{\max} \\ h - \sqrt{dt_{\max}} \\ u_{\min} - u \\ -h \\ x - x_{\max} \\ x_{\min} - x \\ c_I(x,u) \\ \hline c_E(x,u) \\ (h_k - h_{k+1})[k < N-1] \end{array}\right] = c(x,\hat{u}) = \left[\begin{array}{c} \hat{u} - \hat{u}_{\max} \\ \hat{u}_{\min} - \hat{u} \\ x - x_{\max} \\ x_{\min} - x \\ c_I(x,u) \\ \hline c_E(x,u) \\ (h_k - h_{k+1})[k < N-1] \end{array}\right]
$$

where the horizontal line indicates the separation between inequality (above) and equality (below) constraints.
We now write down the gradients of this equation, since they will be used when re-deriving the backwards pass later:

$$
\nabla_x c(x,\hat{u}) = \begin{bmatrix} \mathbf{0}_{\hat{\mathbf{m}}\times\mathbf{n}} \\ \mathbf{0}_{\hat{\mathbf{m}}\times\mathbf{n}} \\ I_n \\ -I_n \\ \nabla_x c_I(x,u) \\ \nabla_x c_E(x,u) \\ \mathbf{0}_{\mathbf{1}\times\mathbf{n}} \end{bmatrix}
$$

$$
\nabla_{\hat{u}} c(x,\hat{u}) = \begin{bmatrix} I_m \\ -I_m \\ \mathbf{0}_{\hat{\mathbf{n}}\times\hat{\mathbf{m}}} \\ \mathbf{0}_{\hat{\mathbf{n}}\times\hat{\mathbf{m}}} \\ [\nabla_u c_I(x,u)\ \mathbf{0}] \\ [\nabla_u c_E(x,u)\ \mathbf{0}] \\ 1[k < N-1] \end{bmatrix}
$$