Model Predictive Control (SC42125)

Sergio Grammatico, Luyao Zhang

7. Introduction to Trajectory Optimization (iLQR/DDP)

References


Rawlings, Mayne, Diehl, "Model Predictive Control: Theory, Computation and Design", Chapter 8.

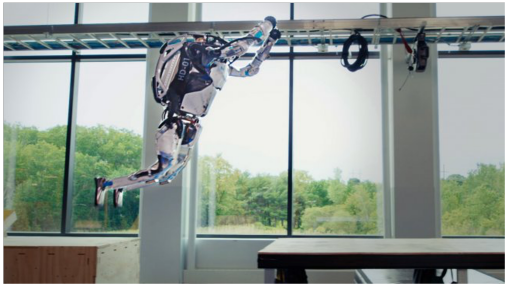

Tassa, Erez and Todorov, "*Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization*", IROS, 2012.


Howell, Jackson and Manchester, "*ALTRO: A Fast Solver for Constrained Trajectory Optimization*", IROS, 2019.

Online course:

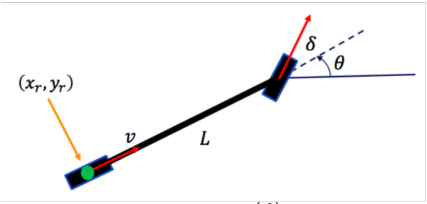
- Optimal Control (CMU 16-745)
- Numerical Optimal Control (University of Freiburg)

Motivation

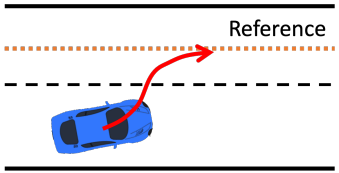


Optimal Control Problem: Example

- Objective: track the reference trajectory + minimize the control efforts



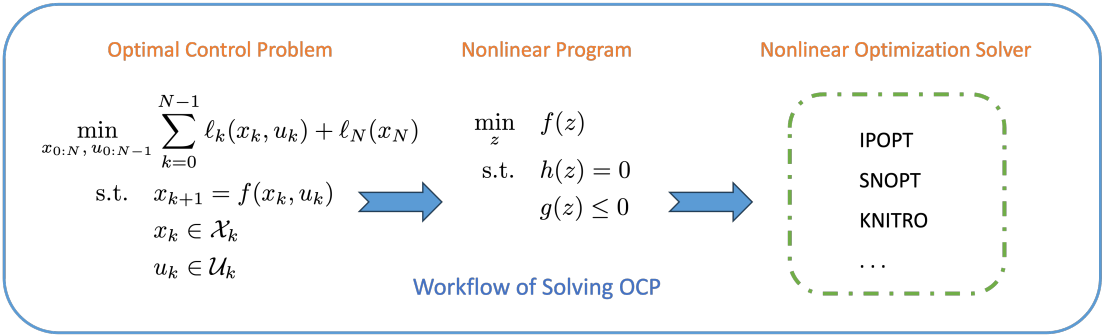
$$\dot{x}_r = v\cos(\theta)$$
$$\dot{y}_r = v\sin(\theta)$$
$$\dot{\theta} = \frac{v\tan(\delta)}{L}$$
$$\dot{v} = a$$



State: (x_r, y_r, θ, v)

Input: (a, δ)

Optimal Control Problem: Structure

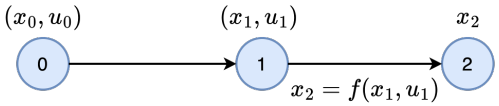


Drawback: no explicit **structure** exploitation

Optimal Control Problem: Structure

$$\min_{x_{0:N},\, u_{0:N-1}} \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_N(x_N)$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k) \longleftarrow \text{nonlinear dynamics}$$

- Dynamic Programming (DP) structure



$$V(x_k) = \min_{u_k} \quad \ell(x_k, u_k) + V(x_{k+1}) \quad \text{Bellman Equation}$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k)$$

iLQR Overview

- Bellman equation

$$V(x_k) = \min_{u_k} \quad \ell(x_k, u_k) + V(x_{k+1}) \longleftarrow \text{need an approximation}$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k)$$

- **Iterative LQR** / Differential Dynamic Programming
  - Approximate cost function and value function using second-order Taylor expansion
  - Approximate nonlinear dynamics using first-order Taylor expansion

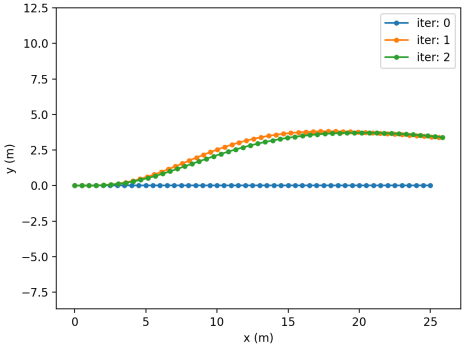
iLQR Overview (+)

$$\min_{x_{0:N},\, u_{0:N-1}} \sum_{k=0}^{N-1} \ell(x_k, u_k) + \ell_N(x_N)$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k)$$

Approximate around $\bar{x}_{0:N}, \bar{u}_{0:N-1}$

$$\min_{x_{0:N},\, u_{0:N-1}} \sum_{k=0}^{N-1} \ell^{\text{QP}}(x_k, u_k) + \ell_N^{\text{QP}}(x_N)$$
$$\text{s.t.} \quad x_{k+1} - \bar{x}_{k+1} = A_k(x_k - \bar{x}_k) + B_k(u_k - \bar{u}_k)$$

Lane-changing example



Improve trajectory iteratively!

LQR: Recap

$$\min_{x_{0:N},\, u_{0:N-1}} \sum_{k=0}^{N-1} \left(\frac{1}{2} x_k^\top Q x_k + \frac{1}{2} u_k^\top R u_k \right) + \frac{1}{2} x_N^T Q_N x_N$$
$$\text{s.t.} \quad x_{k+1} = A x_k + B u_k$$

- Value function (cost-to-go): $V(x) = \frac{1}{2} x^\top P x$
- Bellman equation

$$\begin{aligned} V(x_k) &= \min_{u_k} \quad \ell(x_k, u_k) + V(x_{k+1}) \\ &= \min_{u_k} \quad \frac{1}{2} x_k^\top Q x_k + \frac{1}{2} u_k^\top R u_k + \frac{1}{2} (A x_k + B u_k)^\top P_{k+1} (A x_k + B u_k) \\ &= \min_{u_k} \quad \frac{1}{2} u_k^\top (R + B^\top P_{k+1} B) u_k + x_k^\top (A^\top P_{k+1} B) u_k + \text{Constant} \end{aligned}$$

LQR: Recap

- Optimal control input u_k^*

$$\begin{aligned} u_k^* &= -(R + B^\top P_{k+1} B)^{-1} B^\top P_{k+1} A x_k \\ &= K_k x_k \end{aligned}$$

- Cost-to-go: $V(x_k) = \frac{1}{2} x_k^\top P_k x_k$

$$V(x_k) = \frac{1}{2} x_k^\top Q x_k + \frac{1}{2} u_k^{*\top} R u_k^* + \frac{1}{2} (A x_k + B u_k^*)^\top P_{k+1} (A x_k + B u_k^*)$$

$$P_k = Q + K_k^\top R K_k + (A + B K_k)^\top P_{k+1} (A + B K_k)$$

General Case: Use Iterative Approximations

- Approximate dynamics with first-order Taylor expansion

$$\begin{aligned}\bar{x}_{k+1} + \delta x_{k+1} &= f(\bar{x}_k + \delta x_k, \bar{u}_k + \delta u_k) \\ &\approx f(\bar{x}_k, \bar{u}_k) + \frac{\partial f}{\partial x}\Big|_{\bar{x}_k, \bar{u}_k} (x - \bar{x}_k) + \frac{\partial f}{\partial u}\Big|_{\bar{x}_k, \bar{u}_k} (u - \bar{u}_k) \\ \delta x_{k+1} &= A_k \delta x_k + B_k \delta u_k, \quad \{\bar{x}_k\} \text{ and } \{\bar{u}_k\} \text{ are nominal trajectories.}\end{aligned}$$

- Approximate stage cost with second-order Taylor expansion

$$\ell(x_k, u_k) \approx \underbrace{\ell_k}_{\ell(\bar{x}_k, \bar{u}_k)} + \begin{bmatrix}\ell_{x,k} \\ \ell_{u,k}\end{bmatrix}^\top \begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix}^\top \begin{bmatrix}\ell_{xx,k} & \ell_{ux,k}^\top \\ \ell_{ux,k} & \ell_{uu,k}\end{bmatrix} \begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix},$$

where $\ell_{x,k} = \nabla_x \ell(\bar{x}_k, \bar{u}_k), \ell_{u,k} = \nabla_u \ell(\bar{x}_k, \bar{u}_k)$ are the gradients, and $\ell_{xx,k} = \nabla^2_{xx} \ell(\bar{x}_k, \bar{u}_k), \ell_{ux,k} = \nabla^2_{ux} \ell(\bar{x}_k, \bar{u}_k), \ell_{uu,k} = \nabla^2_{uu} \ell(\bar{x}_k, \bar{u}_k)$ are the Hessians.

Dynamic Programming

- Bellman equation

$$\begin{aligned} V(x_k) &= \min_{u_k} \quad \ell(x_k, u_k) + V(x_{k+1}) \\ &= \min_{u_k} \quad Q(x_k, u_k) \end{aligned}$$

- Approximate the value function (cost-to-go)

$$V(x_{k+1}) \approx \underbrace{V_{k+1}}_{V(\bar{x}_{k+1})} + \underbrace{V_{x,k+1}{}^{\top}}_{\nabla_x V(\bar{x}_{k+1})} \delta x_{k+1} + \frac{1}{2} \delta x_{k+1}^{\top} \underbrace{V_{xx,k+1}}_{\nabla^2_{xx} V(\bar{x}_{k+1})} \delta x_{k+1}$$

$$\begin{aligned} V_{x,N} &= \ell_{x,N} \\ V_{xx,N} &= \ell_{xx,N} \end{aligned}$$

Approximated Q-function

- Substitute δx_{k+1} using the linearized dynamics

$$
\begin{aligned}
V(x_{k+1}) &\approx V_{k+1} + V_{x,k+1}^\top \left(A_k \delta x_k + B_k \delta u_k\right) + \frac{1}{2}\left(A_k \delta x_k + B_k \delta u_k\right)^\top V_{xx,k+1}\left(A_k \delta x_k + B_k \delta u_k\right) \\
&= V_{k+1} + \begin{bmatrix} A_k^\top V_{x,k+1} \\ B_k^\top V_{x,k+1} \end{bmatrix}^\top \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}^\top \begin{bmatrix} A_k^\top V_{xx,k+1} A_k & (B_k^\top V_{xx,k+1} A_k)^\top \\ B_k^\top V_{xx,k+1} A_k & B_k^\top V_{xx,k+1} B_k \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}
\end{aligned}
$$

- Approximate the Q function

$$
Q(x_k, u_k) \approx Q_k + \begin{bmatrix} Q_{x,k} \\ Q_{u,k} \end{bmatrix}^\top \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}^\top \begin{bmatrix} Q_{xx,k} & Q_{ux,k}^\top \\ Q_{ux,k} & Q_{uu,k} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}
$$

Approximated Q-function (+)

- Find $Q_{x,k}$, $Q_{u,k}$, $Q_{xx,k}$, $Q_{ux,k}$, $Q_{ux,k}$, $Q_{uu,k}$

$$
\begin{aligned}
Q(x_k, u_k) &\approx Q_k + \begin{bmatrix} Q_{x,k} \\ Q_{u,k} \end{bmatrix}^\top \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}^\top \begin{bmatrix} Q_{xx,k} & Q_{ux,k}^\top \\ Q_{ux,k} & Q_{uu,k} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} \\
&= \ell_k + \begin{bmatrix} \ell_{x,k} \\ \ell_{u,k} \end{bmatrix}^\top \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}^\top \begin{bmatrix} \ell_{xx,k} & \ell_{ux,k}^\top \\ \ell_{ux,k} & \ell_{uu,k} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} \\
&+ V_{k+1} + V_{x,k+1}^\top \delta x_{k+1} + \frac{1}{2} \delta x_{k+1}^\top V_{xx,k+1} \delta x_{k+1}
\end{aligned}
$$

$$
\begin{aligned}
Q_{x,k} &= \ell_{x,k} + A_k^\top V_{x,k+1} \\
Q_{u,k} &= \ell_{u,k} + B_k^\top V_{x,k+1} \\
Q_{xx,k} &= \ell_{xx,k} + A_k^\top V_{xx,k+1} A_k \\
Q_{uu,k} &= \ell_{uu,k} + B_k^\top V_{xx,k+1} B_k, \\
Q_{ux,k} &= \ell_{ux,k} + B_k^\top V_{xx,k+1} A_k = Q_{xu,k}^\top
\end{aligned}
$$

Affine Control Law

- Bellman equation

$$
\begin{aligned}
V(x_k) &= \min_{u_k} \quad Q(x_k, u_k) \\
&= \min_{\delta u_k} \quad Q_k + \begin{bmatrix} Q_{x,k} \\ Q_{u,k} \end{bmatrix}^\top \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}^\top \begin{bmatrix} Q_{xx,k} & Q_{ux,k}^\top \\ Q_{ux,k} & Q_{uu,k} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k \end{bmatrix}
\end{aligned}
$$

- Find the optimal control input δu_k^*

$$
\begin{aligned}
\frac{\partial Q}{\partial \delta u_k} &= Q_{u,k} + \frac{1}{2} Q_{ux,k} \delta x_k + \frac{1}{2} Q_{xu,k}^\top \delta x_k + Q_{uu,k} \delta u_k = 0 \\
\delta u_k^* &= -Q_{uu,k}^{-1} (Q_{u,k} + Q_{ux,k} \delta x_k) \\
&= \underbrace{-Q_{uu,k}^{-1} Q_{ux,k}}_{K_k} \delta x_k \underbrace{-Q_{uu,k}^{-1} Q_{u,k}}_{d_k}
\end{aligned}
$$

Approximated Value Function Update

- Plug δu_k^* into approximated $Q(x_k, u_k)$ to get $V(x_k)$

$$
\begin{aligned}
V(x_k) &\approx V_k + V_{x,k}{}^\top \delta x_k + \frac{1}{2}\delta x_k^\top V_{xx,k}\delta x_k \\
&= Q_k + \begin{bmatrix} Q_{x,k} \\ Q_{u,k} \end{bmatrix}^\top \begin{bmatrix} \delta x_k \\ \delta u_k^* \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \delta x_k \\ \delta u_k^* \end{bmatrix}^\top \begin{bmatrix} Q_{xx,k} & Q_{ux,k}^\top \\ Q_{ux,k} & Q_{uu,k} \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta u_k^* \end{bmatrix} \\
&\approx Q_k + \begin{bmatrix} Q_{x,k} \\ Q_{u,k} \end{bmatrix}^\top \begin{bmatrix} \delta x_k \\ K_k\delta x_k + d_k \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \delta x_k \\ K_k\delta x_k + d_k \end{bmatrix}^\top \begin{bmatrix} Q_{xx,k} & Q_{ux,k}^\top \\ Q_{ux,k} & Q_{uu,k} \end{bmatrix} \begin{bmatrix} \delta x_k \\ K_k\delta x_k + d_k \end{bmatrix}
\end{aligned}
$$

- Find $V_{x,k}$, $V_{xx,k}$

$$
\begin{aligned}
V_{x,k} &= Q_{x,k} + K_k^\top Q_{u,k} + Q_{ux,k}^\top d_k + K_k^\top Q_{uu,k} d_k \\
V_{xx,k} &= Q_{xx,k} + K_k^\top Q_{uu,k} K_k + K_k^\top Q_{ux,k} + Q_{ux,k}^\top K_k
\end{aligned}
$$

Forward Pass

- Nominal trajectories: $\{\bar{x}_k\}$ and $\{\bar{u}_k\}$
- Update control input

$$\begin{aligned} u_k \leftarrow \bar{u}_k + \delta u_k^* &= \bar{u}_k + K_k \delta x_k + d_k \\ &= \bar{u}_k + K_k(x_k - \bar{x}_k) + d_k \end{aligned}$$

- Rollout (forward simulation)

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

Backward Pass

1. Start from $V(x_N) = \ell_N(x_N),\ \ k \leftarrow N-1$
2. Compute approximated $Q(x_k, u_k)$
3. Compute K_k, d_k
4. Update cost-to-go function $V(x_k),\ \ k \leftarrow k-1$
5. Go to Step 3 if $k > 0$
6. Return $\{K_k\}_{k=0}^{N-1}$, $\{d_k\}_{k=0}^{N-1}$

Optimal Control Problem with Constraints

$$\min_{x_{0:N},\, u_{0:N-1}} \quad \ell_N(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k)$$
$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k)$$
$$c_{k,i}(x_k, u_k) = 0 \quad i \in \mathcal{E}_k$$
$$c_{k,j}(x_k, u_k) \leq 0 \quad j \in \mathcal{I}_k$$

- Methods for handling constraints
  - Interior point method
  - **Augmented Lagrangian method**

Chen, Zhan, and Tomizuka, "*Autonomous Driving Motion Planning With Constrained Iterative LQR*", TITS.

Vanroye, Sathya, Schutter, and Decré, "*A Fast Constrained Optimal Control Problem Solver for Robot Trajectory Optimization and Control*", IROS, 2023.

Recap: Penalty Method

- Constrained optimization problem:

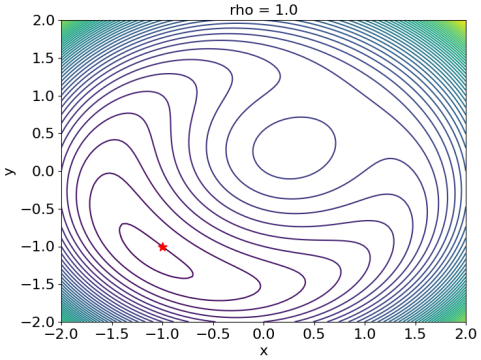
$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & c_i(x) = 0 \quad i \in \mathcal{E} \\ & c_i(x) \leq 0 \quad i \in \mathcal{I} \end{aligned}$$

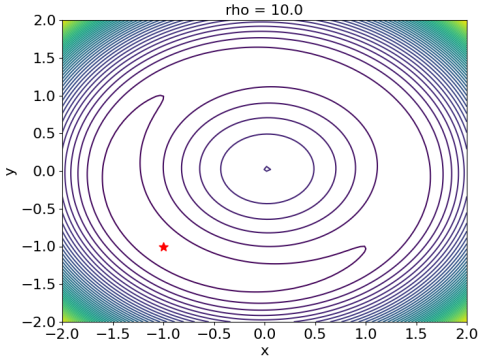
- Replace constraints with penalty terms:

$$P(x, \rho) = f(x) + \underbrace{\frac{1}{2}\rho \sum_{i \in \mathcal{E}} c_i^2(x) + \frac{1}{2}\rho \sum_{i \in \mathcal{I}} \max(c_i(x), 0)^2}_{\text{quadratic penalty}}$$

- P is not twice continuously differentiable due to $\max(\cdot, \cdot)^2$
- $\nabla_{xx} P(x, \rho)$ becomes ill conditioned near the minimizer $\longrightarrow$ Solving $\min_x P(x, \rho)$ becomes more difficult

Recap: Penalty Method





$$\min_{x,y} \; x + y \quad \text{s.t. } x^2 + y^2 - 2 = 0$$

$$P(x, y, \rho) = x + y + \frac{1}{2}\rho(x^2 + y^2 - 2)^2$$

- Hessian is **ill conditioned**
- **Banana shape** rather than elliptical shape near minimizer

Recap: Augmented Lagrangian Method

- Introduce **Lagrangian multiplier estimate** into penalty method:

$$\mathcal{L}_{\mathsf{a}}(x, \tilde{\lambda}, \mu) = \underbrace{f(x) + \tilde{\lambda}^\top c(x)}_{\mathcal{L}} + \underbrace{\frac{1}{2} c(x)^\top I_\mu c(x)}_{\text{penalty}},$$

  where $\tilde{\lambda}$ are the Lagrange multipliers, μ are the penalty multipliers, and

$$I_{\mu,ii} = \begin{cases} 0 & \text{if } c_i(x) < 0 \wedge \lambda_i = 0, i \in \mathcal{I} \\ \mu_i & \text{otherwise} \end{cases}$$

- Relieve ill-conditioning of the penalty method
- Converge with finite μ

Recap: Augmented Lagrangian Method

$$\min_{x,y} \; x + y \quad \text{s.t. } x^2 + y^2 - 2 = 0$$

$$\mathcal{L}_a = x + y + \tilde{\lambda}(x^2 + y^2 - 2) + \frac{\rho}{2}(x^2 + y^2 - 2)^2$$

$$(\hat{x}^*_{\mathsf{AL}}, \hat{y}^*_{\mathsf{AL}}) = \arg\min_{x,y} \; \mathcal{L}_a(x, y, \tilde{\lambda} = 0.3, \mu = 1.0)$$

$$= (-1.02, -1.02)$$

$$(\hat{x}^*_{\mathsf{P}}, \hat{y}^*_{\mathsf{P}}) = \arg\min_{x,y} \; P(x, y, \rho = 1.0)$$

$$= (-1.10, -1.10)$$

- Good Lagrangian multiplier estimate + small penalty weight $\longrightarrow$ better approximated solution

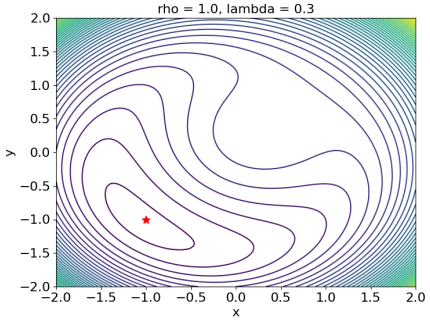


Figure: $\lambda^* = 0.5$, $\tilde{\lambda} = 0.3$, $\mu = 1.0$

Recap: Augmented Lagrangian Method

1. Solve $\min_x \mathcal{L}_{\mathsf{a}}(x, \lambda, \mu)$, holding λ and μ constant
2. Update the Lagrangian multipliers

$$\lambda_i \leftarrow \begin{cases} \lambda_i + \mu_i c_i(x^*) & i \in \mathcal{E} \\ \max(0, \lambda_i + \mu_i c_i(x^*)) & i \in \mathcal{I} \end{cases}$$

3. Update penalty multipliers: $\mu \leftarrow \beta\mu$, $\beta > 1$
4. Check constraint feasibility
5. Go to Step 1 if tolerance not satisfied

Augmented Lagrangian iLQR

- Optimal control problem

$$\begin{aligned} \min_{x_{0:N},\, u_{0:N-1}} \quad & \ell_N(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) \\ \text{s.t.} \quad & x_{k+1} = f(x_k, u_k) \\ & c_{k,i}(x_k, u_k) = 0 \quad i \in \mathcal{E}_k \\ & c_{k,i}(x_k, u_k) \leq 0 \quad i \in \mathcal{I}_k \end{aligned}$$

- Augmented Lagrangian

$$\mathcal{L}_{\mathsf{a}} = \ell_N(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) + \lambda_N^\top c_N(x_N) + \frac{1}{2} c_N(x_N)^\top I_{\mu,N} c_N(x_N) + \sum_{k=0}^{N-1} \left\{ \lambda_k^\top c_k(x_k, u_k) + \frac{1}{2} c_k(x_k, u_k)^\top I_{\mu,k} c_k(x_k, u_k) \right\}$$

Augmented Lagrangian iLQR

- Augmented Lagrangian

$$\mathcal{L}_{\mathsf{a}} = \underbrace{\ell_N(x_N) + \lambda_N^\top c_N(x_N) + \frac{1}{2} c_N(x_N)^\top I_{\mu,N} c_N(x_N)}_{\mathcal{L}_{\mathsf{a},N}(x_N, \lambda_N, \mu_N)} + \sum_{k=0}^{N-1} \underbrace{\left\{ \ell(x_k, u_k) + \lambda_k^\top c_k(x_k, u_k) + \frac{1}{2} c_k(x_k, u_k)^\top I_{\mu,k} c_k(x_k, u_k) \right\}}_{\mathcal{L}_{\mathsf{a},k}(x_k, u_k, \lambda_k, \mu_k)}$$

- Cost-to-go function

$$\begin{aligned} V_N(x_N)|_{\lambda_N, \mu_N} &= \mathcal{L}_{\mathsf{a},N}(x_N, \lambda_N, \mu_N) \\ V_k(x_k)|_{\lambda_k, \mu_k} &= \min_{u_k} \left\{ \mathcal{L}_{\mathsf{a},k}(x_k, u_k, \lambda_k, \mu_k) + V_{k+1}(x_{k+1})|_{\lambda_{k+1}, \mu_{k+1}} \right\} \\ &= \min_{u_k} Q(x_k, u_k)|_{\lambda_k, \mu_k} \end{aligned}$$

Augmented Lagrangian iLQR

- Take the gradient and the Hessian of $\mathcal{L}_{\mathsf{a},k}(x_k, u_k, \lambda_k, \mu_k)$

$$
\begin{aligned}
Q_{x,k} &= \ell_{x,k} + A_k^\top V_{x,k+1} + c_{x,k}^\top \lambda_k + c_{x,k}^\top I_{\mu,k} c_k \\
Q_{u,k} &= \ell_{u,k} + B_k^\top V_{x,k+1} + c_{u,k}^\top \lambda_k + c_{u,k}^\top I_{\mu,k} c_k \\
Q_{xx,k} &= \ell_{xx,k} + A_k^\top V_{xx,k+1} A_k + c_{x,k}^\top I_{\mu,k} c_{x,k} \\
Q_{uu,k} &= \ell_{uu,k} + B_k^\top V_{xx,k+1} B_k + c_{u,k}^\top I_{\mu,k} c_{u,k} \\
Q_{ux,k} &= Q_{xu,k}^\top = \ell_{ux,k} + B_k^\top V_{xx,k+1} A_k + c_{u,k}^\top I_{\mu,k} c_{x,k}
\end{aligned}
$$

- Note: Only the first-order terms are left (**Gauss-Newton approximation**)

Gauss-Newton Approximation

- Hessian approximation of a specific class of functions

$$g(x) = \frac{1}{2}\|r(x)\|^2 = \frac{1}{2}\sum_{i=1}^{m} r_i(x)^2$$

- Gradient of $g(x)$ is

$$\nabla g(x) = \sum_{i=1}^{m} \nabla r_i(x) r_i(x) = Dr(x)^\top r(x)$$

- Hessian of $g(x)$ is

$$\begin{aligned}\nabla^2 g(x) &= \sum_{i=1}^{m} \nabla r_i(x) \nabla r_i(x)^\top + \sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x) \\ &= Dr(x)^\top Dr(x) + \sum_{i=1}^{m} r_i(x) \nabla^2 r_i(x) \\ &\approx Dr(x)^\top Dr(x) \quad \text{ignore second-order terms}\end{aligned}$$

Software Packages

- iLQR/DDP
  -  Altro.jl, https://github.com/RoboticExplorationLab/Altro.jl
  -  aligator, https://github.com/Simple-Robotics/aligator
- Modeling tools
  -  CasADi, https://web.casadi.org/
  -  do-mpc, https://www.do-mpc.com/en/latest/
- Automatic differentiation
  -  autodiff, https://autodiff.github.io/
  -  jax, https://github.com/google/jax