LQR→iLQR→DDP(Differential Dynamic Programming)

— CHH3213

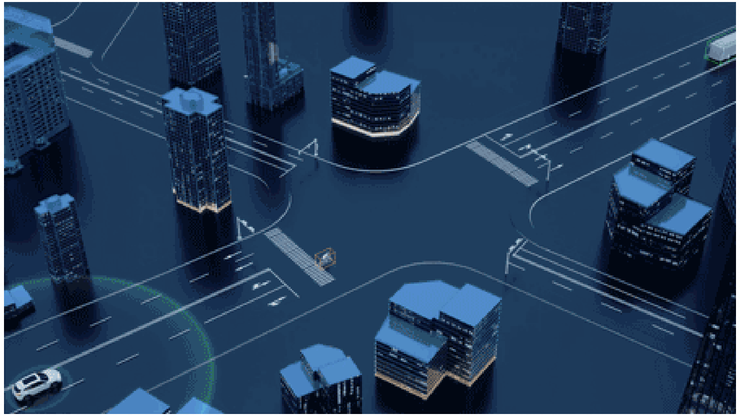
1

Prerequisite

Joint Spatio-temporal Planning vs. Decoupled Planning

Joint Spatio-temporal Planning

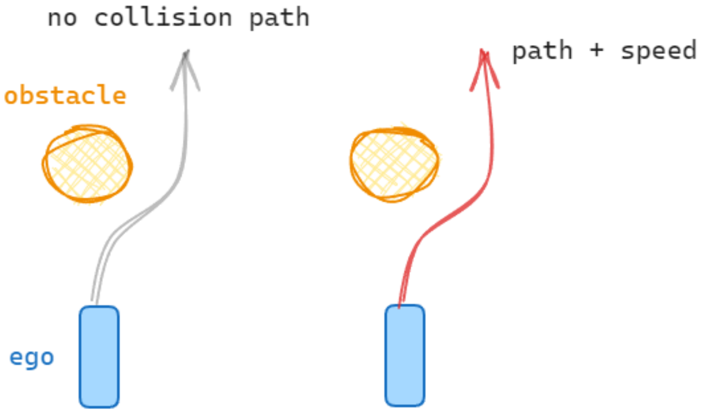
1. Plans path and speed simultaneously
2. Output is spatio-temporal trajectory
3. Globally optimal



Source：

Decoupled Planning

1. First path planning
2. Then speed planning
3. Combine path with speed to generate a trajectory
4. Local optimal



Overall for LQR , iLQR and DDP

LQR (Linear Quadratic Regulator):

1. Optimal control algorithm for linear systems

2. Finds a feedback controller to take the system from initial state to goal state while minimizing a quadratic cost function

3. Requires a linear dynamics model and quadratic cost function

4. Can compute control inputs online efficiently

iLQR (Iterative LQR):

1. Iterative version of LQR that handles nonlinear systems

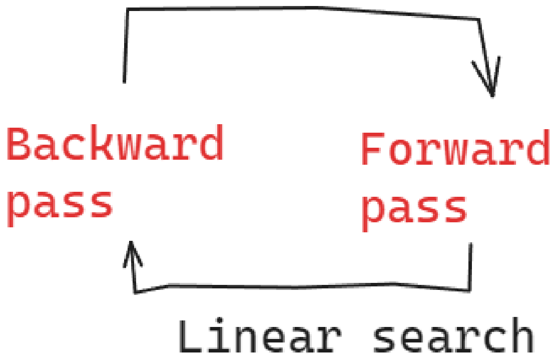
2. Linearizes the nonlinear dynamics through Gaussian Newton iterations and applies LQR

3. Requires gradients of dynamics model and cost function

4. More costly than LQR but can still compute online

DDP (Differential Dynamic Programming):

1. Online nonlinear optimal control algorithm
2. Similar to iLQR but uses second order information to better locally approximate the system as quadratic
3. Reduces linearization errors compared to iLQR



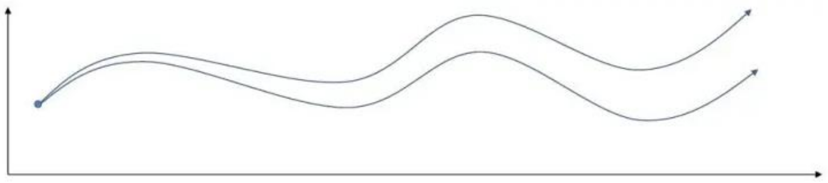
2

Details of iLQR/DDP

Shooting method vs. collocation method

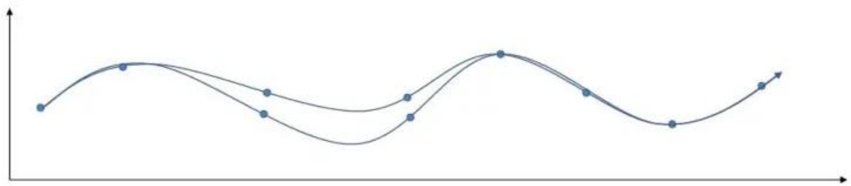
LQR: https://blog.csdn.net/weixin_42301220/article/details/124542242

Shooting method



- focus on optimizing the action/control at each time step, treating the state as a result of the actions.

collocation method

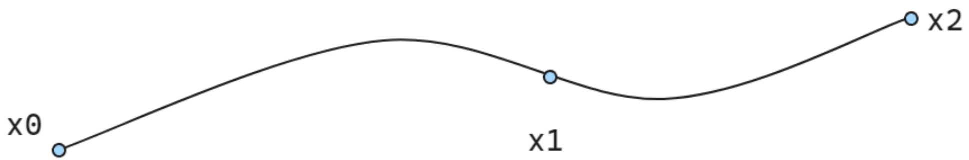


- Optimize state and action at each time step together
- Use constraints to relate state and action
- Sometimes only optimize state, treating action as way to change states

iLQR and DDP belong to shooting method

The Bellman Optimality Principle

- If a policy π has the property that it achieves the optimal value function Vπ(s) for some initial state s, then π must be an optimal policy for the process for all states.
- Each component of an optimal policy must also be optimal.
- Bellman’s principle suggests starting from the end of the trajectory and walking backwards.



Key idea of dynamic programming!

Discrete Dynamics and Cost objectives

Given a discrete-time nonlinear system dynamics:

$$x_k + \delta x_k = x$$

$$u_k + \delta u_k = u$$

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

Use first-order Taylor-series expansion:

$$\boxed{x_{k+1} + \delta x_{k+1} = f(x_k + \delta x_k, u_k + \delta u_k)} \approx f(x_k, u_k) + \boxed{\left.\frac{\partial f}{\partial x}\right|_{x_k, u_k}}(x - x_k) + \boxed{\left.\frac{\partial f}{\partial u}\right|_{x_k, u_k}}(u - u_k)$$

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

If we use second-order Taylor-series expansion:

$$f(x_k + \delta x_k, u_k + \delta u_k) \approx f(x_k, u_k) + \left.\frac{\partial f}{\partial x}\right|_{(x_k, u_k)} \delta x_k + \left.\frac{\partial f}{\partial u}\right|_{(x_k, u_k)} \delta u_k + \frac{1}{2}\left.\frac{\partial^2 f}{\partial x^2}\right|_{(x_k, u_k)} (\delta x_k)^2 + \frac{1}{2}\left.\frac{\partial^2 f}{\partial u^2}\right|_{(x_k, u_k)} (\delta u_k)^2 + \left.\frac{\partial^2 f}{\partial x \partial u}\right|_{(x_k, u_k)} \delta x_k \delta u_k$$

The iLQR/DDP algorithm's goal is to find an input sequence, $U = \{u_0, u_1, \cdots, u_{m-1}\}$, that minimizes a cost function,

$$J(x_0, U) = \mathcal{L}(x_N) + \sum_{k=0}^{N-1} \mathcal{L}(x_k, u_k)$$

Backward pass

Using Bellman Optimality Principle, we can define the optimal cost-to-go

$$V_N(x_N) = \mathcal{L}(x_N)$$
$$V_k(x_k) = \min_u \{\mathcal{L}(x_k, u_k) + V_{k+1}(f(x_k, u_k))\}$$

We define $Q(x_k, u_k) = \boxed{\mathcal{L}(x_k, u_k) + V_{k+1}(f(x_k, u_k)),}$

$$V_k(x_k) = \min_u \{Q(x_k, u_k)\} \xrightarrow{\text{A small perturbation}} \delta V = \min_{\delta u}\{\delta Q(x, u)\}$$

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

Backward pass

Similarly,

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

$$\boxed{\delta Q_k(x_k, u_k) = \ Q_x \delta x + Q_u \delta u + \frac{1}{2}\delta x^T Q_{xx} \delta x + \frac{1}{2}\delta u^T Q_{uu} \delta u + \frac{1}{2}\delta x^T Q_{xu} \delta u + \frac{1}{2}\delta u^T Q_{ux} \delta x}$$

$$\Updownarrow$$

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

$$\Updownarrow$$

$$\delta Q(x_k, u_k) \approx \frac{1}{2}\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k\end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{xu}^T & Q_{uu}\end{bmatrix}\begin{bmatrix} 1 \\ \delta x_k \\ \delta u_k\end{bmatrix}$$

Note that, $Q_{ux} = Q_{xu}^T$

$$\begin{aligned}
Q_x &= \frac{\partial Q}{\partial x} = \frac{\partial \mathcal{L}}{\partial x} + \frac{\partial f}{\partial x}^T \frac{\partial V_{k+1}}{\partial_x} \\
Q_u &= \frac{\partial Q}{\partial u} = \frac{\partial \mathcal{L}}{\partial u} + \frac{\partial f}{\partial u}^T \frac{\partial V_{k+1}}{\partial_x} \\
Q_{xx} &= \frac{\partial^2 Q}{\partial x^2} = \frac{\partial^2 \mathcal{L}}{\partial x^2} + \frac{\partial f}{\partial x}^T \frac{\partial^2 V_{k+1}}{\partial x^2}\frac{\partial f}{\partial x} + \frac{\partial V_{k+1}}{\partial x}\frac{\partial^2 f}{\partial x^2} \\
Q_{uu} &= \frac{\partial^2 Q}{\partial u^2} = \frac{\partial^2 \mathcal{L}}{\partial u^2} + \frac{\partial f}{\partial u}^T \frac{\partial^2 V_{k+1}}{\partial x^2}\frac{\partial f}{\partial u} + \frac{\partial V_{k+1}}{\partial x}\frac{\partial^2 f}{\partial u^2} \\
Q_{xu} &= \frac{\partial^2 Q}{\partial x \partial u} = \frac{\partial^2 \mathcal{L}}{\partial x \partial u} + \frac{\partial f}{\partial x}^T \frac{\partial^2 V_{k+1}}{\partial x^2}\frac{\partial f}{\partial u} + \frac{\partial V_{k+1}}{\partial x}\frac{\partial^2 f}{\partial x \partial u}
\end{aligned}$$

Backward pass

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

$$\Rightarrow$$

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

$$\delta Q_k(x_k,u_k) = Q_x\delta x + Q_u\delta u + \frac{1}{2}\delta x^T Q_{xx}\delta x + \frac{1}{2}\delta u^T Q_{uu}\delta u + \frac{1}{2}\delta x^T Q_{xu}\delta u + \frac{1}{2}\delta u^T Q_{ux}\delta x$$

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

$$\Rightarrow$$

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

After calculating the optimal control as a function of the next time step

$$\delta V = \delta Q(\delta x, \delta u^*) = \frac{1}{2}\begin{bmatrix} 1 \\ \delta x_k \\ (K\delta x_k + d) \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{xu}^T & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ (K\delta x_k + d) \end{bmatrix}$$
$$= (Q_x + K^T Q_{uu} d + K^T Q_u + Q_{ux}^T d)^T \delta x_k + \frac{1}{2}\delta x_k^T (Q_{xx} + K^T Q_{uu} K + K^T Q_{ux} + Q_{ux}^T K)\delta x_k + \frac{1}{2} d^T Q_{uu} d + d^T Q_u$$

Backward pass

After calculating the optimal control as a function of the next time step

$$\delta V = \delta Q(\delta x, \delta u^*) = \frac{1}{2}\begin{bmatrix} 1 \\ \delta x_k \\ (K\delta x_k + d) \end{bmatrix}^T \begin{bmatrix} 0 & Q_x^T & Q_u^T \\ Q_x & Q_{xx} & Q_{xu} \\ Q_u & Q_{xu}^T & Q_{uu} \end{bmatrix} \begin{bmatrix} 1 \\ \delta x_k \\ (K\delta x_k + d) \end{bmatrix}$$

$$= \boxed{(Q_x + K^T Q_{uu} d + K^T Q_u + Q_{ux}^T d\ )^T} \delta x_k + \frac{1}{2}\delta x_k^T \boxed{(Q_{xx} + K^T Q_{uu} K + K^T Q_{ux} + Q_{ux}^T K)} \delta x_k + \boxed{\frac{1}{2} d^T Q_{uu} d + d^T Q_u}$$

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

$$\frac{\partial V}{\partial x} = Q_x + K^T Q_{uu} d + K^T Q_u + Q_{ux}^T d$$

$$\frac{\partial^2 V}{\partial x^2} = Q_{xx} + K^T Q_{uu} K + K^T Q_{ux} + Q_{ux}^T K$$

$$\Delta V = \frac{1}{2} d^T Q_{uu} d + d^T Q_u$$

Algorithm 1 Backward pass

Initialize: Initialize cost-to-go function V_N at the terminal step N.

for $k \in [N-1, \cdots, 1]$ **do**

- Compute $\delta Q(x_k, u_k)$;
- minmize $\delta Q(x_k, u_k)$ and get optimal policy: K, d;
- Compute δV_k;
- Update function V_k according to K, d;

Forward pass

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"

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

where 0<=α<=1 is the step size, typically used to perform a simple line search.

$$z = \frac{J(X,U) - J(\bar{X},\bar{U})}{-\Delta V(\alpha)}$$

$$\Delta V = \frac{1}{2} d^T Q_{uu} d - d^T Q_u \xrightarrow{d = \alpha d_k} \Delta V(\alpha) = \sum_{k=0}^{N-1} \alpha d_k^T Q_u + \alpha^2 \frac{1}{2} d_k^T Q_{uu} d_k$$

where J is the total cost.

$$\boxed{J(x_0, U) = \mathcal{L}(x_N) + \sum_{k=0}^{N-1} \mathcal{L}(x_k, u_k)}$$

https://bjack205.github.io/papers/AL_iLQR_Tutorial.pdf

Line search

where α is the step size, typically used to perform a simple line search.

$$z = \frac{J(X,U) - J(\bar{X},\bar{U})}{-\Delta V(\alpha)}$$

$$\boxed{J(x_0,U) = \mathcal{L}(x_N) + \sum_{k=0}^{N-1} \mathcal{L}(x_k,u_k)}$$

$$\Delta V = \frac{1}{2} d^T Q_{uu} d - d^T Q_u \xrightarrow{d = \alpha d_k} \Delta V(\alpha) = \sum_{k=0}^{N-1} \alpha d_k^T Q_u + \alpha^2 \frac{1}{2} d_k^T Q_{uu} d_k$$

where J is the total cost.

- If z lies within a the interval [β1, β2], usually [1e-4, 10], we accept the candidate trajectories.
- If it does not, we update the scaling parameter α = γα, where 0 < γ < 1 is the backtracking scaling parameter. γ = 0.5 is typical.
- Increasing the lower bound on the acceptance interval will require that more significant progress is made during each backward-forward pass iteration. Decreasing the upper bound will keep the progress closer to the expected decrease. These values aren't changed much in practice.

https://bjack205.github.io/papers/AL_iLQR_Tutorial.pdf

Algorithm 2 Forward pass

Initialize: Initialize $\bar{x}_0 = x_0, \alpha = 1.0.$

for $k \in [0, \cdots, N-1]$ **do**

  Compute $\bar{u}_k$;

  Update $\bar{x}_k$;

Compute the total cost J and $\Delta V(\alpha)$;

Compute z;

if *z satisfies line search conditions* **then**

  Update $X \leftarrow \bar{X}$;

  Update $U \leftarrow \bar{U}$;

else

  Reduce α and go to the loop again;

return X, U, J;

Side note: If we use RL?

- We will use nueral network to approximate V and Q!
- V is state value function;
- Q is state-action value function.

Appendix

- https://openreview.net/pdf?id=BNDO0dxvjD
- http://roboticexplorationlab.org/papers/iLQR_Tutorial.pdf
- https://bjack205.github.io/papers/AL_iLQR_Tutorial.pdf
- https://arxiv.org/pdf/2103.03293.pdf
- https://zhuanlan.zhihu.com/p/600500268

Thanks!