A Survey On the Methods and Application of Optimal Control

Zhennan Jiang, *Central South University, Changsha, 410000, China*

***Abstract*—This paper mainly expounds the basic concepts of optimal control problems. To Solve the optimal control problems, variational calculus, minimum principle and dynamic programming are emphatically introduced. First, some basic concepts and methods of optimal control are introduced. It focuses on the connection and difference between the three methods and the applicable processes and systems. Some concrete examples about linear quadratic regulator is shown to make the solution into practice. All demos, including code and simulation shown in this paper, can be dwonload from https://github.com/jzndd/LQR-Application**

***Index Terms*—variational calculus, minimum principle, dynamic programming, optimal control, linear quadratic regulator**

I. Introduction

A. Development of Optimal Control

Optimal control is a field that focuses on finding the best control actions to minimize or maximize a performance criterion while satisfying system dynamics and constraints. It developed in the mid-20th century with concepts like the calculus of variations and the minimum principle introduced by Lev Pontryagin. These principles provide necessary conditions for optimal control solutions. Dynamic programming, developed by Richard Bellman, is another influential approach that breaks down complex control problems into smaller subproblems and solves them recursively. [1]

In recent years, there has been growing interest in optimal control based variational methods, such as the Hamilton-Jacobi-Bellman (HJB) equation and the Pontryagin's maximum principle. These methods combine optimal control theory with concepts from variational calculus and partial differential equations. They provide powerful tools for solving high-dimensional and nonlinear control problems and have contributed to advancements in areas like optimal control of complex systems, optimal trajectory planning, and reinforcement learning. Overall, optimal control still plays an important role in various fields. [2]

B. Article structure

The paper is structured as follows:In Section II,the description of optimal control problem is detailed and relevant formulation is given. Section III mainly introduce the effectiveness and limitation of three solution methods. In Section IV, we mainly discuss the difference of mothods mentioned in III. In Section V, linear quadratic regulator problem is proposed. Section VI shows the concrete examples about LQR to make theory into practice.

II. Problem Formulation and Preliminaries

In order to solve the optimal control problem, this paper establishes a detailed and complete mathematical description of the optimal control problem. The specific description process is divided into the following steps.

1) Model Building:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{1}$$

2) Performance Index: Performance index(also called objective function) is generally expressed as

$$J[\boldsymbol{u}(t)] = \varphi\left[\boldsymbol{x}\left(t_f\right), t_f\right] + \int_{t_0}^{t_f} L[\boldsymbol{x}(t), \boldsymbol{u}(t), t] dt, \varphi, L \in \mathbb{R} \tag{2}$$

t_0 and t_f are the initial and final times. φ is evaluated at the final state and is not necessarily specified. L is the cost function. Different problems have concrete J. Typical applied problems of optimal control are showm as follows.

(i)Minimum time path: $J = t_f - t_0 = \int_{t_0}^{t_f} dt$

(ii) Minimum fuel path: $J = \int_{t_0}^{t_f} |u(t)| dt$

(iii) Minimum control effort: $J = \int_{t_0}^{t_f} u(t)^2 dt$

(iv) Tracking problem:

$J = \frac{1}{2} e\left(t_f\right)^T F e\left(t_f\right) + \frac{1}{2} \int_{t_0}^{t_f} \left[e(t)^T Q e(t) + u(t)^T R u(t)\right] dt$

$e(t) = x(t) - x_r(t)$ or $e(t) = y(t) - y_r(t)$

3) Constraint: Some problems must take the state constraints and the terminal constraints into consideration

$$\boldsymbol{g}\left[\boldsymbol{x}(t), \boldsymbol{u}(t), t\right] = 0,\ \boldsymbol{g} \in \mathbb{R}^p \tag{3}$$

$$\boldsymbol{\xi}\left[\boldsymbol{x}\left(t_f\right), t_f\right] = 0,\ \boldsymbol{\xi} \in \mathbb{R}^r \tag{4}$$

Overall, Given a system, a performance function and constraints, find a control sequence that makes the system stable and the performance function take an extreme value. The goal of optimal control is to solve the control sequence.

III. Solution Methods

This section will make a detailed analysis of the use and limitations of these three methods:variational calculus,minimum principle and dynamic programming.

A. Variational Calculus

In variational calculus, the control problem is represented as an optimization of a functional, which is a mapping from a set of functions to real numbers. The functional measures the system's performance and typically involves control inputs, system states, and their derivatives. The objective is to find the control inputs that minimize or maximize the functional, subject to the system's dynamics and any imposed constraints.

1) Basic concepts: variation of independent variable of functional is defined as

$$\delta \boldsymbol{x}(t) = \boldsymbol{x}(t) - \boldsymbol{x_0}(t),\ \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x_0} \in \mathbb{R}^n$$

the increment of a continuous functional produced from $\delta \boldsymbol{x}(t)$ be expressed by

$$\begin{aligned} \varDelta J[\boldsymbol{x}(t)] &= J[\boldsymbol{x}(t) + \delta \boldsymbol{x}(t)] - J[\boldsymbol{x}(t)] \\ &= L[\boldsymbol{x}(t), \delta \boldsymbol{x}(t)] + R[\boldsymbol{x}(t), \delta \boldsymbol{x}(t)] \end{aligned}$$

where L is a linear functional with respect to $\delta \boldsymbol{x}(t)$, and R is a higher-order infinitesimal quantity with respect to $\delta \boldsymbol{x}(t)$.

$\delta J[\boldsymbol{x}(t)]|_{\varepsilon=0} = 0$ can prove to be necessary condition for function. Now, we gain the variational problem to an extreme value problem with the necessary condition.

$$\delta J[\boldsymbol{x}(t)] = L[\boldsymbol{x}(t), \delta \boldsymbol{x}(t)] = \frac{\partial}{\partial \varepsilon} J[\boldsymbol{x}(t) + \varepsilon \delta \boldsymbol{x}(t)]|_{\varepsilon=0} \tag{5}$$

Then, the necessary condition for functional extremum can be expressed as:

$$\begin{aligned} \delta J[x] &= \int_{t_0}^{t_f} \left\{ \frac{\partial L}{\partial x} \delta x + \frac{\partial L}{\partial \dot{x}} \delta \dot{x} \right\} dt \\ &= \int_{t_0}^{t_f} \left\{ \frac{\partial L}{\partial x} \delta x \right\} dt + \left[\frac{\partial L}{\partial \dot{x}} \delta x \right]_{t_0}^{t_f} - \int_{t_0}^{t_f} \left\{ \delta x \frac{d}{dt} \frac{\partial L}{\partial \dot{x}} \right\} \\ &= \int_{t_0}^{t_f} \delta x \left\{ \frac{\partial L}{\partial x} - \frac{d}{dt} \frac{\partial L}{\partial \dot{x}} \right\} dt \end{aligned} \tag{6}$$

from (6),we gain :

$$\frac{\partial L}{\partial x} - \frac{d}{dt} \frac{\partial L}{\partial \dot{x}} = 0 \tag{7}$$

(7) is called Euler equation. Futhermore, we can turn The vector form of Euler equation to the expansion of Euler equation(8),which is a significant tool in variational calculus.

$$\frac{\partial L}{\partial x_i} - \frac{\partial^2 L}{\partial t \partial \dot{x}_i} - \frac{\partial^2 L}{\partial x_i \partial \dot{x}_i} \dot{x}_i - \frac{\partial^2 L}{\partial \dot{x}_i \partial \dot{x}_i} \ddot{x}_i = 0 \tag{8}$$

2) Solution of Specific Problems: Many problems have constraints. The way to solve that is to change the functional extremum problem with constraint to a functional extremum problem without constraint by the Lagrange multiplier approach.

Based on this idea, Hamilton function comes up.

$$H(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, t) = L(\boldsymbol{x}, \boldsymbol{\mu}, t) + \boldsymbol{\lambda}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) + \boldsymbol{\mu}^T \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\mu}, t) \tag{9}$$

Based on the analysis of variational calculus, the optimal control must satisfy the following necessary condition:

(I) state equation

$$\dot{\boldsymbol{x}} = f[\boldsymbol{x}(t), \boldsymbol{u}(t), t] \tag{10}$$

(II) costate equation

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial H}{\partial \boldsymbol{x}} = -\frac{\partial \left(L + \boldsymbol{\lambda}^T \boldsymbol{f} + \boldsymbol{\mu}^T \boldsymbol{\varphi} \right)}{\partial \boldsymbol{x}} \tag{11}$$

(III) extremal condition

$$\frac{\partial H}{\partial \boldsymbol{u}} = \frac{\partial \left(L + \boldsymbol{\lambda}^T \boldsymbol{f} + \boldsymbol{\mu}^T \boldsymbol{\varphi} \right)}{\partial \boldsymbol{u}} = 0 \tag{12}$$

(IV) equality constraint

$$\boldsymbol{g}[\boldsymbol{x}(t), \boldsymbol{u}(t), t] = 0 \tag{13}$$

(V) boundary condition

$$\boldsymbol{x}(t_0) = \boldsymbol{x_0},\ \boldsymbol{\xi}[\boldsymbol{x}(t_f), t_f] = 0 \tag{14}$$

(VI) transversality condition

$$\begin{aligned} &\boldsymbol{\lambda}(t_f) = \frac{\partial \varphi}{\partial \boldsymbol{x}(t_f)} + \frac{\partial \boldsymbol{\xi}^T}{\partial \boldsymbol{x}(t_f)} \boldsymbol{\nu}, \\ &\frac{\partial \varphi}{\partial t_f} + \boldsymbol{\nu}^T \frac{\partial \boldsymbol{\xi}^T}{\partial t_f} + H(t_f) = 0 \end{aligned} \tag{15}$$

Variational calculus can solve the following optimal control problem by (10)-(15). However, variational calculus is not applicable to the following case : Hard input-constraint, complex constraint and time discretization constraints.

(Hard constraints mean that some constraints must be strictly satisfied, while variational methods are generally more flexible in dealing with soft constraints, which allow for some degree of violation. [3])

B. The Pontryagin Minimum Principle

As mentioned above, $\boldsymbol{u}(t)$ can't be restricted in variational calculus, which ignite proposal of the minimum principle. It enable optimal control to deal with input-constrained case.

In fact compared with classical variational calculus, only the extremal conditions are different. In classical variational calculus,we have $\frac{\partial H}{\partial u} = 0$. Howver,in the minimum principle, we have :

$$H(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{\lambda}, \boldsymbol{\mu}, t) = \min_{\boldsymbol{u}(t) \in U} H(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}, t) \tag{16}$$

The latter means that in all admissible controls, the one making take minimum is optimal control, or and reach minimum simultaneously.

Combined with (10)(11)(13)(14)(15) , we can apply minimum principle to solve problems.

What's more, discrete minimum principle can solve time discretization constraints. The necessary conditions for functional extremum with constraints are:

(I) state equation

$$x(k+1)^* = f[x(k)^*, u(k)^*, k] \tag{17}$$

(II) costate equation

$$\lambda(k) = \frac{\partial H[x(k)^*, u(k)^*, \lambda(k+1), k]}{\partial x(k)^*} \tag{18}$$

(III) extremal condition

$$H[x(k)^*, u(k)^*, \lambda(k+1), k] = \min_{u(k) \in U} H[x(k)^*, u(k), \lambda(k+1), k] \tag{19}$$

(IV) initial constraint

$$x(0)^* = x_0 \tag{20}$$

(VI) transversality condition

$$\lambda(k_f) = \frac{\partial \varphi[x(N)^*, N]}{\partial x(N)^*} \tag{21}$$

Hence, the pontryagin minimum principle is an extension of variational calculus, which is useful to solve the problems with input-constraint and time discretization constraints.

C. Dynamic Programming

Dynamic programming(DP) is very different from the two methods mentioned above. DP relies on the principle of optimality, which states that an optimal control policy for a given time step is independent of the past and depends only on the current state. This property allows DP to solve complex problems by breaking them down into smaller subproblems. Figure 1 shows the classic DP problem. [4] [5]

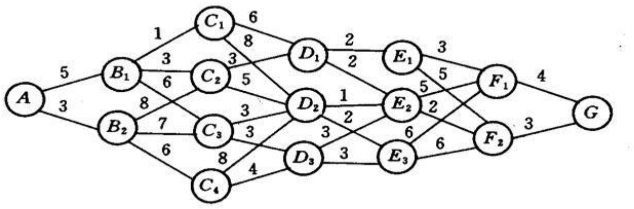

Fig. 1: Classic DP problem

As mentioned above, DP solves problems by breaking them down into smaller subproblems and utilizing the principle of optimality. Let's consider a discrete-time optimal control problem with a time horizon from $t = 0$ to $t = T$. The goal is to find the optimal control policy, making J minimal.

$$J_N = \sum_{k=0}^{N-1} L[x(k), u(k), k], \quad J, L \in R \tag{22}$$

The value function $V_{N-j}[x(j)]$ is defined as the (N-j)-stage optimal decision starting from state $x(j)$

$$\begin{aligned} V_{N-j}[x(j)] = J^*_{N-j} &= \min_{\{u(k)\}} \left\{ \sum_{k=j}^{N-1} L[x(k), u(k), k] \right\} \\ &= \sum_{k=j}^{N-1} L\left[x(k)^*, u(k)^*, k\right] \end{aligned} \tag{23}$$

Then,

$$\begin{aligned} J^*_N = V_N[x(0)] &= \min_{\{u(k)\}} \left\{ \sum_{k=0}^{N-1} L[x(k), u(k), k] \right\} \\ &= \min_{u(0),u(1),\cdots,u(N-1)} \\ &\left\{ L[x(0), u(0), 0] + \sum_{k=1}^{N-1} L[x(k), u(k), k] \right\} \end{aligned} \tag{24}$$

The principle of optimality states that an optimal control policy for a given time step is independent of the future and depends only on the current state. Based on this principle, if the N-stage decision $V_N[x(0)]$ is optimal, then the (N-1)-stage decision $V_{N-1}[x(1)]$, regarding the x(1) resulting from x(0) and u(0) as the initial state,should be also optimal:

$$\begin{aligned} V_N[x(0)] &= \min_{u(0)} \left\{ L[x(0), u(0), 0] + \min_{u(1),\cdots,u(N-1)} \sum_{k=1}^{N-1} L \right\} \\ &= \min_{u(0)} \left\{ L[x(0), u(0), 0] + V_{N-1}[x(1)] \right\} \end{aligned} \tag{25}$$

In general, it holds that:

$$V_{N-j}[x(j)] = \min_{u(j)} \left\{ L[x(j), u(j), j] + V_{N-(j+1)}[x(j+1)] \right\} \tag{26}$$

DP can also work in continous system. To solve 2, define $V[x(t), t]$ as the minimum of starting from state $x(t)$ and time t :

$$\begin{aligned} V\left[\boldsymbol{x}^*(t), t\right] &= J^*\left[\boldsymbol{x}^*(t), t\right] \\ &= \min_{\boldsymbol{u}(t)\in U} \left\{ \varphi\left[\boldsymbol{x}^*(t_f), t_f\right] + \int_{t_0}^{t_f} L\left(\boldsymbol{x}^*, \boldsymbol{u}, t\right) dt \right\} \end{aligned} \tag{27}$$

Then, we gain the Hamilton-Jacobi-Bellman equation:

$$\frac{\partial V\left(\boldsymbol{x}^*, t\right)}{\partial t} = -\min_{\boldsymbol{u}(t)\in U} \left\{ L\left(\boldsymbol{x}^*, \boldsymbol{u}, t\right) + \frac{\partial V\left(\boldsymbol{x}, t\right)}{\partial \boldsymbol{x}^T} \boldsymbol{f}\left(\boldsymbol{x}^*, \boldsymbol{u}, t\right) \right\} \tag{28}$$

In (28), it needs a condition that $V[x(t), t]$ is continuously differentiable in x(t) and t.

In summary, DP uses the Bellman equation and the principle of optimality to compute the optimal value function and control policy by breaking down the original optimization problem into a sequence of smaller subproblems. By solving these subproblems iteratively, DP finds the optimal solution for both continuos and discrete problems.

IV. Comparison of Methods

This section will make a comparison of three methods proposed above in terms of core ideas, application scenarios and so on.

A. Difference of Core Ideas

Optimal control refers to the problem of determining the control inputs that minimize or maximize a certain objective while satisfying system constraints. To solve this problem, different methods have different solution due to their different core ideas.

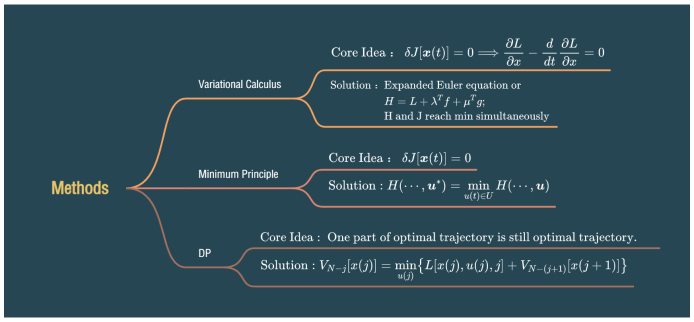
Methods
Variational Calculus
Core Idea : $\delta J[\boldsymbol{x}(t)] = 0 \Longrightarrow \frac{\partial L}{\partial x} - \frac{d}{dt}\frac{\partial L}{\partial x} = 0$
Solution : Expanded Euler equation or $H = L + \lambda^T f + \mu^T g$; H and J reach min simultaneously
Minimum Principle
Core Idea : $\delta J[\boldsymbol{x}(t)] = 0$
Solution : $H(\cdots, \boldsymbol{u}^*) = \min_{u(t)\in U} H(\cdots, \boldsymbol{u})$
DP
Core Idea : One part of optimal trajectory is still optimal trajectory.
Solution : $V_{N-j}[x(j)] = \min_{u(j)}\{L[x(j), u(j), j] + V_{N-(j+1)}[x(j+1)]\}$

Fig. 2: Idea of different methods

B. Variational Calculus and The Minimum Principle

The minimum principle can be derived using variational methods. By applying the calculus of variations to the Hamiltonian function associated with the optimal control problem, the necessary conditions for optimality can be obtained, which are consistent with those derived from the minimum principle. In this sense, variational methods provide a mathematical foundation for understanding and solving problems governed by the minimum principle.

In fact, these two methods have the same solution in cases that there is no restriction on control variable u, where formula (19) is equal to (12)

However, they have some differences in their formulations and applications shown as follows:

1) Problem formulation: Variational calculus formulate optimal control problems as variational principles, where the goal is to find the extremum of a functional over a set of admissible control functions or trajectories. On the other hand, the minimum principle provides a set of necessary conditions for an optimal control problem, stating that an optimal control policy must satisfy these conditions.

2) Optimization approach: Variational calculus involve solving the variational problem using techniques such as the Euler-Lagrange equations. The minimum principle, on the other hand, provides a set of necessary conditions in the form of a set of differential equations known as the Hamiltonian equations.

3) Applicability: Variational calculus is well-suited for problems with continuous control and state variables, as they rely on the calculus of variations. The minimum principle, on the other hand, is applicable to both continuous-time and discrete-time optimal control problems ,and particularly useful for problems involving bang-bang control.

4) Requirements: Variational calculus requires the existence of $\frac{\partial H}{\partial \boldsymbol{u}}$ and $\frac{\partial H}{\partial \boldsymbol{x}}$.And only the existence of $\frac{\partial H}{\partial \boldsymbol{x}}$ is needed in The minimum principle.

C. The Minimum Principle and Dynamic Programming

Under certain conditions, dynamic programming can be derived as a discrete-time approximation of the continuous-time minimum principle. For the problem (1)-(4) and $H = L + \lambda^T f$,we have:

$$\begin{aligned}\tilde{J} &= \varphi\left[x\left(t_f\right), t_f\right] + v^T \xi\left[x\left(t_f\right), t_f\right] + \int_t^{t_f} \left\{H - \lambda^T \dot{x}\right\} d\tau \\ &= \varphi + v^T \xi + \int_t^{t_f} \left\{H + \dot{\lambda}^T x\right\} d\tau - \lambda\left(t_f\right)^T x\left(t_f\right) + \lambda^T x\end{aligned} \tag{29}$$

Along the optimal trajectory $V\left[x^*, t\right] = \tilde{J}^* = J^*$, yields :

$$\begin{aligned}\frac{\partial V\left[x^*, t\right]}{\partial t} &= -\left\{H\left(x^*, u^*, \lambda, t\right) + \dot{\lambda}^T x^*\right\} + \dot{\lambda}^T x^* \\ &= -H\left(x^*, u^*, \lambda, t\right)\end{aligned} \tag{30}$$

Insert $\frac{\partial V[x^*, t]}{\partial x^*} = \lambda$ into Hamilton-Jacobi-Bellman equation (28) , yields :

$$\begin{aligned}H\left(x^*, u^*, \lambda, t\right) &= \min_{u(t)} \left\{L\left(x^*, u, t\right) + \frac{\partial V\left[x^*, t\right]}{\partial x^T} f\left(x^*, u, t\right)\right\} \\ &= \min_{u(t)} \left\{L\left(x^*, u, t\right) + \lambda^T f\left(x^*, u, t\right)\right\}\end{aligned} \tag{31}$$

So we gain $H\left(x^*, u^*, \lambda, t\right) = \min_{u(t)} H\left(x^*, u, \lambda, t\right)$, which is as same as(16).It can be proved that the dynamic programming equation is equal to the the minimum principle in given conditions.

However, they have some differences shown as follows:

1) Problem Formulation: The minimum principle addresses continuous-time optimal control problems, providing necessary conditions for optimality based on the Hamiltonian function. Dynamic Programming covers both continuous-time and discrete-time problems, breaking them into subproblems.

2) State and Control Space Dimensions: The minimum principle focuses on continuous state and control spaces, while Dynamic Programming accommodates both continuous and discrete spaces.

3) Optimization Approach: The minimum principle establishes necessary conditions for optimality but lacks a direct computational method. Dynamic Programming breaks down the problem into subproblems, solving them iteratively.

4) Requirements: The minimum principle requires the existence of $\frac{\partial H}{\partial \boldsymbol{x}}$. Dynamic programming requires the existence of $\frac{\partial V}{\partial \boldsymbol{u}}$ and $\frac{\partial V}{\partial \boldsymbol{x}}$

V. Linear Quadratic Optimal Control

In this section,we will illustrate the basic facts of LQR [8]-[11], and then introduced an widly algorithm LQR-RRT*. This is a process from theory to practice in optimal control.

A. LQR Problem

Consider a continuous time dynamical control system.

$$\dot{x} = f(x) + g(x)u, \tag{32}$$

where $x \in \mathcal{X} \subset \mathbb{R}^n$, $u \in \mathcal{U} \subset \mathbb{R}^m$, and $f(x) : \mathbb{R}^n \mapsto \mathbb{R}^n$, $g(x) : \mathbb{R}^n \mapsto \mathbb{R}^{n \times m}$ are locally Lipschitz continuous.

We use LQR to compute optimal control policies. The cost function with the infinite time horizon is defined as

$$J = \int_0^\infty x^T Q x + u^T R u, \tag{33}$$

where $Q = Q^T \succeq 0$ and $R = R^T \succeq 0$ are weight matrices for state x and control u, respectively. For linear system dynamics, we can compute the closed form solution for the optimal control for a given linear-time invariant systems. For general nonlinear system (32), we can obtain its linearized form using Taylor-expansion near a local state and control as

$$\dot{x} = Ax + Bu. \tag{34}$$

Given the system (34) and the cost function (33), we can compute the LQR gain matrix K_{LQR} by solving the algebraic Riccati equation with cost matrix P as

$$A^T P + PA - PBR^{-1}B^T P + Q = 0, \tag{35}$$

and

$$K_{\text{LQR}} = R^{-1}B^TP. \tag{36}$$

Finally, the optimal control policy is denoted as

$$\pi^*(x) = -K_{\text{LQR}}x \tag{37}$$

For nonlinear systems, we a linearization can be made w.r.t. an equilibrium point (x_{eq}, u_{eq}) using first-order taylor expansion. Given a nonlinear system,

$$\begin{aligned}\dot{x} &= F(x,u)\\ &\approx F(x_{\text{eq}},u_{\text{eq}}) + \frac{\partial F(x_{\text{eq}},u_{\text{eq}})}{\partial x}(x-x_{\text{eq}})\\ &+ \frac{\partial F(x_{\text{eq}},u_{\text{eq}})}{\partial u}(u-u_{\text{eq}})\end{aligned}$$

Based on the equilibrium point , the linearized system can be re-written as

$$\dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}\hat{u}, \tag{38}$$

with $\hat{A} = \frac{\partial F(x_{\text{eq}},u_{\text{eq}})}{\partial x}$ and $\hat{B} = \frac{\partial F(x_{\text{eq}},u_{\text{eq}})}{\partial u}$. Finally, we can compute the optimal gain K_{LQR} with the Riccati equation (35).

B. LQR-RRT Algorithm*

This algorithm is just for path planning. The detailed algorithm is introduced as the following [14]. First, we define a tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set of edges.

- **Nearby Node** The function utilizes a pre-defined euclidean distance d to find a set of nearby nodes $\mathcal{X}_{\text{near}}$ in $\mathcal{T}$ that is closest to x_{sample}.
- **Nearest** The function returns the nearest node from $\mathcal{X}_{\text{near}}$ w.r.t. x_{sample}.
- **ChooseParent** The procedure (Algorithm 1 Line 10-14) tries to find a collision-free paths between x_{new} w.r.t. all its neighboring nodes. If there exists a a collision-free path between x_{new} and $x_{\text{nn}} \in \mathcal{X}_{\text{near}}$, the corresponding cost is calculated. The `ChooseParent` procedure then selects the x_{nn} with the lowest cost (33) as the parent of x_{new}.
- **Rewire** The rewiring procedure [12] (Algorithm 1 Line 16-20) evaluates and optimizes the LQR cost (33). The rewiring function checks a selected node's neighborhood and calculates the costs w.r.t. all the neighboring nodes. Given that a a collision-free path exists between the current node and the nearby node, by taking the path from the current node to the nearby node, the nearby node's current path is removed if the new cost is lower than the existing cost. The new path from the current node to the nearby node is added to the tree $\mathcal{T}$.

Instead of using Euclidean distance between two nodes as a cost metric, we use LQR cost (33) as the metric in both `ChooseParent` and `Rewire` procedures. Therefore, the control efforts can be captured as part of the optimization process, and we can adjust the weight matrices to tune the performance of our motion planner.

Algorithm 1: LQR-RRT*

```
1  Initialization: $\mathcal{T} = (\mathcal{V},\mathcal{E})$, $\mathcal{V} \leftarrow \{x_{\text{init}}\}$; $\mathcal{E} \leftarrow \emptyset$; $i = 0$;
     adapFlag = True, optDensityFlag = False, and $r = \eta$
2  while $i < N$ do
3      $x_{samp} \leftarrow$ Sample($\mathcal{G}, \mathcal{V}$, adapFlag)
4      $x_{nearest} \leftarrow$ Nearest($\mathcal{V}, x_{samp}$)
5      $x_{\text{new}}, \sigma \leftarrow$ LQR-Steer($x_{\text{nearest}}, x_{\text{samp}}$)
6      $\mathcal{X}_{near} \leftarrow$ NearbyNode($\mathcal{V}, x_{nearest}$)
7      $r = \min\{\lambda(\log(|\mathcal{V}|)/|\mathcal{V}|)^{1/(d+1)}, \eta\}$
8      $\mathcal{X}_{near} \leftarrow$ Near($\mathcal{V}, r, x_{new}$)
9      minCost $\leftarrow \infty$; $x_{\min}, \sigma_{\min} \leftarrow$ None, None
10     foreach $x_{\text{near}} \in \mathcal{X}_{\text{near}}$ do
11         $\sigma \leftarrow$ LQR-Steer ($x_{\text{new}}, x_{\text{nn}}$)
12         if $x_{\text{nn}}$.cost+Cost($\sigma$) < minCost then
13             minCost $\leftarrow x_{\text{nn}}$.cost+Cost($\sigma$)
14             $x_{\min} \leftarrow x_{\text{nn}}$; $\sigma_{\min} \leftarrow \sigma$
15     $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{new}\}$; $\mathcal{E} \leftarrow \mathcal{E} \cup \{x_{\min}, x_{\text{new}}\}$
16     foreach $x_{\text{nn}} \in \mathcal{X}_{near}$ do
17         $\sigma \leftarrow$ LQR-Steer($x_{\text{new}}, x_{\text{nn}}$)
18         if $x_{\text{new}}$.cost+Cost($\sigma$) < $x_{\text{nn}}$.cost then
19             $x_{\text{nn}}$.parent $\leftarrow x_{\text{new}}$
20             $x_{\text{new}}$.cost =Cost($x_{\text{new}}$)
21     $\mathcal{T}$, $\mathcal{G} \leftarrow$ extToGoal($\mathcal{V}, x_{\text{new}}$, adapFlag); $i \leftarrow i+1$
22 return $\mathcal{T}$
```

Algorithm 2: LQR-Steer($x_{\text{current}}, x_{\text{next}}$)

```
1  $\mathbf{x} \leftarrow$ None, $\mathbf{u} \leftarrow$ None
2  $\mathbf{x}$.add($x_{\text{current}}$)
3  $\text{K}_{\text{LQR}} \leftarrow$ LQRsolver($x_{\text{current}}, x_{\text{next}}$)
4  foreach $t <$ T do
5      $x^{'}, u \leftarrow$ Integrator($x_{\text{current}}, \text{K}_{\text{LQR}}$)
6      if Satisfied $\leftarrow$ CBFconstraints($x^{'}, u$) then
7          $\mathbf{x}$.add($x^{'}$); $\mathbf{u}$.add($u$)
8          $x_{\text{current}} \leftarrow x^{'}$
9      else
10         return $\sigma = (\mathbf{x}, \mathbf{u})$
```

- **LQRSolver** The method (Algorithm 1 Line 4) computes optimal gain matrix K_{LQR} using (36) from Riccati equation (35). For a nonlinear system, the system can be linearized locally [13] and then computes K_{LQR} based on linearized system dynamics.
- **LQR-Steer** Given two states $(x_{\text{current}}, x_{\text{next}})$, the LQR controller generates a sequence of optimal controls u_i^*, that steers state trajectory based on (34). At each time steps, the constraints are checked to ensure generated path is collision free.Then the `LQR-steer` is used for steering x_{current} to x_{next}, and node x_{next} is added to the tree.

VI. Simulation Examples

In this section, simulation examples are given to demonstrate the effectiveness of LQR. The three examples are path planning, fixed-value output tracking, and trajectory tracking.

A. Example One: Path Planning

In this example, I use the LQR-RRT* to implement path planning.

The experiment is conducted with initial state $x_{init} = [0.0, 0.0]$ and the goal state $x_{goal} = [6.0, 7.0]$. We consider obstacles to be circular in a 2-dimensional workspace. The environment contains seven obstacles with positions:

$[5, 5], [4, 6], [4, 7.5], [4, 9], [6, 5], [7, 5]$,and the size of obstacles is 1.

The parameters of LQR are shown in Table I.

TABLE I: LQR Parameters

Parameter	Value
A	$\begin{bmatrix} dt & 1 \\ 0 & dt \end{bmatrix}$
B	$\begin{bmatrix} 0 & 1 \end{bmatrix}^T$
Q	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
R	$\begin{bmatrix} 1 \end{bmatrix}$
dt	0.1
L	0.5

LQR-RRT* enables agent to search for shorter paths in relatively less time. The result is shown in Fig 3

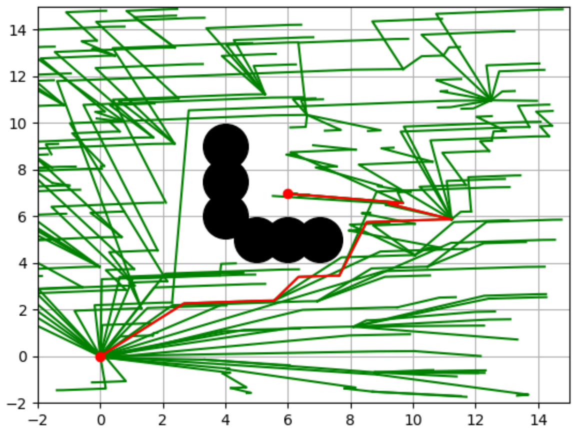

Fig. 3: LQR applied in path planning

B. *Example second:Fixed-Value Output Tracking*

In this example, Simulink is uesd to create an env for simualtion. The controller is LQR.

The experiment is conducted in the following model:

$$\begin{cases} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} u \\ y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{cases} \quad (39)$$

initial state :$x_{t0} = [0, 5]$
terminal state :$x_{tf} = [0, 0]$
from (39),a sim can be built.

We adjust the different values of Q and R to observe the controller's transformation. Different cost leads to different Q and R and parameters are shown in Table II. Result is shown in Fig 5.

From table II, we get the following information larger Q parameter will make the state variables x converge to the set value more quickly, and larger R parameter will result in smaller control variables u.

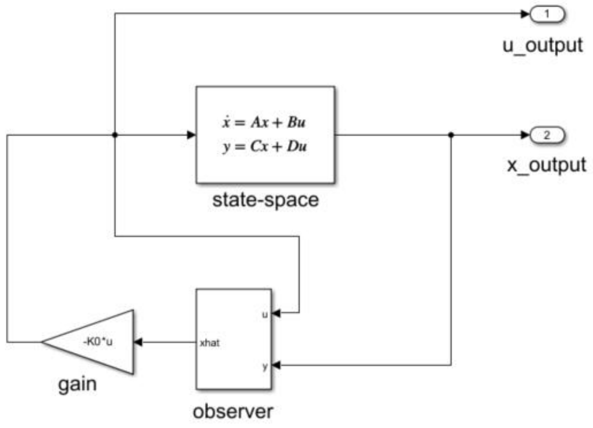

Fig. 4: LQR applied in fixed-value output tracking

Make a brief conclusion:

- Q parameter: The Q parameter determines the importance of the state variables. Increasing the value of a state variable in the Q parameter increases its weight in the controller's optimization objective, making the controller prioritize the optimization of that state variable's tracking. Decreasing the value of a state variable in the Q parameter reduces its weight in the controller's optimization objective, allowing the controller to be more lenient in tracking that state variable.
- R parameter: The R parameter determines the penalty on control inputs. Increasing the value of the R parameter increases the penalty on control inputs, encouraging the controller to generate smaller control inputs. This can be used to limit the magnitude of system actions. Decreasing the value of the R parameter reduces the penalty on control inputs, making the controller's influence on control inputs more lenient.

TABLE II: Different Cost(Differnet Q-R)

	Large Q	Normal	Large R
Q	$\begin{bmatrix} 1000 & 0 \\ 0 & 10 \end{bmatrix}$	$\begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$
R	1	1	100
u peak	133	111	106
x peak	5.48	5.82	5.97
x reach 0(s)	0.6	0.83	0.97

C. *Example third:Path Tracking*

In this section, I use the LQR to implement path tracking.

The experiment is conducted with initial state $x_{init} = [0.0, 0.0]$ and the goal state $x_{goal} = [-1.0, -2.0]$. The state of the intermediate process are $[6.0, -3.0], [12.5, -5.0], [10.0, 6.5], [7.5, 3.0], [3.0, 5.0]$. First of all, we use cubic spline for trajectory planning. It is not important here, so the details are not listed. LQR, as a controller, is then used for path tracking.

The parameters of LQR are same as example 1 except A and B, which are shown in Table III. v is set by cubic spline.

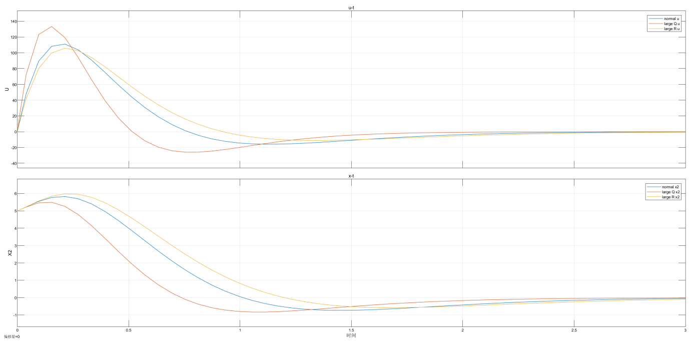

Fig. 5: Different Q-R leads to different u and x

Fig 6 demonstrates the effectiveness of LQR applying in path tracking. The example here is just tracing the path. In fact, it can also track speed by altering A and B (give two u,one is for path tracking,other is for speed tracking).

TABLE III: Matrices A and B

Parameter	Value
A	$\begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 0 & v & 0 \\ 0 & 0 & 1.0 & dt \\ 0 & 0 & 0 & 0 \end{bmatrix}$
B	$\begin{bmatrix} 0 & 0 & 0 & v/L \end{bmatrix}^T$

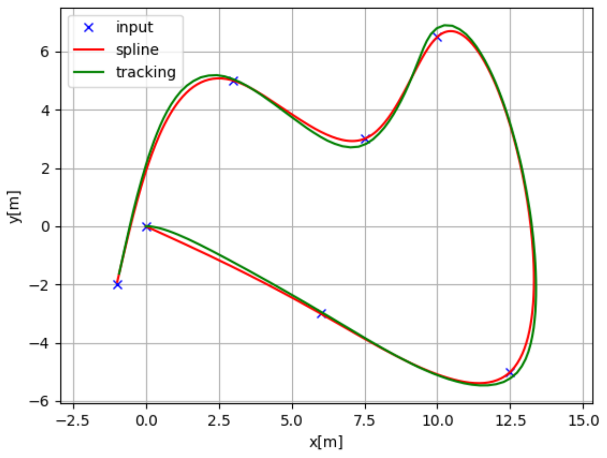

Fig. 6: LQR applied in path tracking

Acknowledgments

I would like to acknowledge Prof.Hui Peng for his valuable guidance throughout my studies about optimal control.

References

[1] Shousong Hu, Optimal control theory and system[M], Beijing Science Press, 2005.
[2] Lian Zhang, Optimal control theory, Beijing: Tsinghua University Press, 2008.1
[3] Mu. Trajectory optimization: Hard constraints and Soft constraints
[4] Bellman R., Dynamic Programming, Princeton University Press,1957
[5] Bertsekas, D. P., Dynamic Programming and Optimal Control, Athena Scientific,2005
[6] Lu J, Chen G. A time-varying complex dynamical network model and its controlled synchronization criteria[J]. IEEE Transactions on Automatic Control, 2005, 50(6): 841-846.
[7] Olfati-Saber, R., and R. M. Murray, Consensus problems in networks of agents with switching topology and time-delays, IEEE Trans. Automatic Control , 49 (9), 15201533 (2004).
[8] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar, Pid, lqr and lqr-pid on a quadcopter platform, in 2013 International Conference on Informatics, Electronics and Vision (ICIEV). IEEE, 2013, pp. 16.
[9] E. Okyere, A. Bousbaine, G. T. Poyi, A. K. Joseph, and J. M. Andrade, Lqr controller design for quad-rotor helicopters, The Journal of Engineering, vol. 2019, no. 17, pp. 40034007, 2019.
[10] X. Xiao, T. Zhang, K. Choromanski, E. Lee, A. Francis, J. Varley, S. Tu, S. Singh, P. Xu, F. Xia et al., Learning model predictive controllers with real-time attention for real-world navigation, arXiv preprint arXiv:2209.10780, 2022
[11] J. Chen, W. Zhan, and M. Tomizuka, Autonomous driving motion planning with constrained iterative lqr, IEEE Transactions on Intelligent Vehicles, vol. 4, no. 2, pp. 244254, 2019
[12] K. Solovey, L. Janson, E. Schmerling, E. Frazzoli, and M. Pavone, Revisiting the asymptotic optimality of rrt, in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 21892195.
[13] Perez. LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. IEEE International Conference on Robotics and Automation. IEEE, 2012, pp. 25372542.
[14] B. Doerr and R. Linares, Motion planning and control for onorbit assembly using lqr-rrt and nonlinear mpc, arXiv preprint arXiv:2008.02846, 2020.

Zhennan Jiang is pursuing his B.E. degree at the School of Automation, Central South University. He rank 1% in major and his current research interests include AI for robotics, Autonomous navigation and dynamic programming.
email:stjzn0410@gmail.com
github:https://github.com/jzndd