# AA274A: Principles of Robot Autonomy I
# Course Notes

Oct 1, 2019

## 3 Open-loop motion control and differential flatness

This lecture introduces optimal control problems and two different forms of control law, open-loop and closed-loop. After a brief overview on solving optimal control problems, we will focus on differential flatness of a robot system that can grealy simplify the solution to optimal control problems.

### 3.1 Kinematic and Dynamic Models [Kir04]

In Chapter 1, we looked at how to derive a kinematic and dynamic model of a robot in the form of ordinary differential equations (ODE) in state variable form

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

where

$$x_1(t), x_2(t), \ldots, x_n(t) \tag{2}$$

are the *state variables* of the process at time $t$, and

$$u_1(t), u_2(t), \ldots, u_m(t) \tag{3}$$

are *control inputs* to the process at time $t$. Eq.1 is therefore a compact notation for the system that can be described by a $n$ first-order differential equations

$$\begin{aligned}
\dot{x}_1(t) &= a_1(x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_m(t), t) \\
\dot{x}_2(t) &= a_2(x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_m(t), t) \\
&\vdots \\
\dot{x}_n(t) &= a_1(x_1(t), x_2(t), \ldots, x_n(t), u_1(t), u_2(t), \ldots, u_m(t), t).
\end{aligned}$$

Note that in Chapter 1, we used $\xi$ to denote the configuration of a robot. In this chapter, we will use $\boldsymbol{x}$ to indicate the combined state of configuration and respective velocities.

## 3.2 Optimal Control Problem

Much of this section is a direct excerpt from Chapter 1 of [Kir04].
Based on the model derived in Chapter 1, we can control our robot based on a certain performance measure. The performance measure can be anything from "getting from point A to point B as quickly as possible" to "keeping the robot velocity close to zero zero when it is close to the goal". Optimal control derives a control history $\boldsymbol{u}(t)$ that minimizes (or maximizes) the performance measure. The mathematical definition of the optimal control problem can be written as below:

**Definition 3.1** (Optimal Control Problem)**.** *Optimal control problem aims at finding an admissible control $\boldsymbol{u}^*$ which causes the system (1) to follow an admissible trajectory $\boldsymbol{x}^*$ that minimizes the performance measure $J$, or*

$$\begin{aligned} \min_{u} \; & h(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\boldsymbol{x}(t), \boldsymbol{u}(t), t) dt \\ & \boldsymbol{x}(t) \in \mathcal{X}, \quad \boldsymbol{u}(t) \in \mathcal{U} \end{aligned} \tag{4}$$

*where $t_0$ and $t_f$ are the initial and the final time; $h$ and $g$ are scalar functions. $t_f$ may be specified or 'free', depending on the problem statement.*

In the objective function, $h(\boldsymbol{x}(t_f), t_f)$ represents the *terminal cost*, where the integral can be viewed as a total sum of *stage-wise costs* induced along the path from $t_0$ to $t_f$.

In general, $\boldsymbol{x}(t) \in \mathbb{R}^n$, $\boldsymbol{u}(t) \in \mathbb{R}^m$, and the initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$ is given. In the context of this course, the constraint $\boldsymbol{x}(t) \in \mathcal{X}$ is typically relaxed to just be $\boldsymbol{x}(t) \in \mathbb{R}^n$, i.e., there are no state constraints. Recall that system dynamics (Eq. 1) is still a constraint, and it is often nonlinear.

## 3.3 Form of the Optimal Control

### 3.3.1 Optimal Control Law

**Definition 3.2** (Optimal control law)**.** *If a functional relationship of the form*

$$\boldsymbol{u}^*(t) = \boldsymbol{f}(\boldsymbol{x}(t), t) \tag{5}$$

*can be found for the optimal control for (Eq.4 at time $t$, then the function $\boldsymbol{f}$ is called the optimal control law, or the optimal policy.*

### 3.3.2 Closed-loop Control

**Definition 3.3** (Closed-loop control)**.** *If the optimal control law is a function of the state and time, i.e.,*

$$\boldsymbol{u}^*(t) = \pi(\boldsymbol{x}(t), t) \tag{6}$$

*then the optimal control is said to be in closed-loop form.*

The closed-loop stems from the fact that your control is a function that maps the state to a given control input. In general, closed-loop control is the most accurate form of motion control since the pairing of state and time within the feedback loop allows the vehicle to determine any error in its configuration and correct it along the way. In other words, any deviation from a determined path (i.e. disturbances from wind, slipping, etc) will be detected and corrected by the vehicle. We will look at closed-loop control in next lecture, so let's focus on open-loop control for now.

### 3.3.3 Open-loop Control

**Definition 3.4** (Open-loop control)**.** *If the optimal control law is determined as a function of time for a specified initial state value, i.e.,*

$$\boldsymbol{u}^*(t) = f(\boldsymbol{x}(t_0), t), \tag{7}$$

*then the optimal control is said to be in open-loop form.*

Open-loop control law depends only on time and initial conditions, and does not update the state in the feedback loop. Thus, open-loop control law cannot correct any deviations from nominal trajectory. The optimal open-loop control is optimal only for a particular initial state value, whereas, if the optimal control law is known, the optimal control history starting from any state value can be generated. [Kir04] Certain engineering applications prefer open-loop control over closed-loop control, nevertheless; optimizing over open-loop control laws (i.e. only over time, not time *and* state) is much easier and computationally inexpensive.

### 3.3.4 Two-Step Design

A good compromise between closed-loop and open-loop control is a two-step design where your control function (A) is an additive combination between a reference open-loop control function (B) plus some reference tracking closed-loop control policy (C, a closed-loop function that keeps the system as close as possible to the nominal behavior that is a function of the current state and the tracking error (D)).

$$\underbrace{\mathbf{u}^*(t)}_{A} = \underbrace{\mathbf{u}_d(t)}_{B} + \underbrace{\pi}_{C}(\mathbf{x}(t), \underbrace{\mathbf{x}(t) - \mathbf{x}_d(t)}_{D}) \tag{8}$$

Here, the closed-loop control law is only responsible for tracking the reference trajectory, and thus can be easier to compute than a closed-loop optimal control law that solves the optimal control problem for any state. This is a similar concept to adding a feed-forward control term to a feedback control law.

## 3.4 Solving the Open-loop Control Problem

There are largely two approaches to solve pen-loop control problems: *direct* and *indirect* methods.

### 3.4.1 Direct Methods (Nonlinear Programming transcription)

Direct methods transcribe the optimal control problem into a discretized finite-dimensional, nonlinear programming (NLP) problem, and then the optimal solution to the NLP problem is found. In other words, we *"first discretize, then optimize"*. Several software packages including DIDO, PROPT, and GPOPS are available to solve optimal control problems with direct method.

### 3.4.2 Indirect Methods

Indirect methods first derive all necessary conditions of optimality (NCO). The condtions are then discretized, and the optimal solution was found by solving the resulting system. Indirect methods, therefore, can be summarized as *"first optimize, then discretize"*.

We will get more into in-depth discussion on direct and indirect methods in a couple lectures later. In the meantime, AA203 – Optimal and Learning-based Control, along with [Kir04] discusses direct and indirect methods to optimal control problems in a more thorough and rigorous manner.

## 3.5 Differential Flatness

Much of this section is a direct excerpt from [Mur].

The concept of differential flatness is extremely useful in controlling certain nonlinear systems, especially when explicit trajectory generation is required. We start with an example that appears to be difficult to solve at first, and examines how differential flatness in robot dynamics can simplify the optimal control problem.

Suppose we would like to find a trajectory $x_d(t)$ that steers the system from an initial condition $x_0$ to a final condition $x_f$. An real-life example is controlling the motion of a automobile so that it follows a given trajectory on the ground. We use a simple car model as shown in Figure 1, i.e.,

$$\begin{aligned} \dot{x} &= V\cos\theta \\ \dot{y} &= V\sin\theta \\ \dot{\theta} &= \frac{v}{L}\tan\phi, \end{aligned} \tag{9}$$

where $(x, y, \theta)$ is the position and orientation of the vehicle, $v$ is the velocity and $\phi$ is the steering angle, and $L$ is the wheelbase.

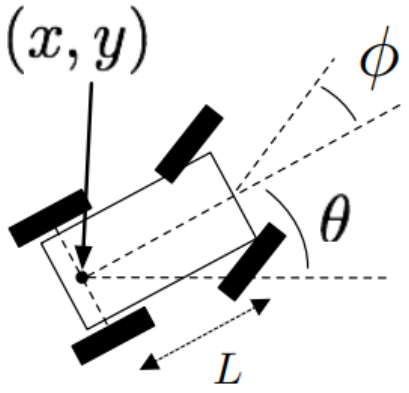


Figure 1: Simple model for an automobile.

We seek a feasible solution $(x_d(t), u_d(t))$ that satisfies system dynamics (9) and initial/final conditions,

$$\dot{\boldsymbol{x}} = \alpha(x, u), \boldsymbol{x}_d = x_0, \boldsymbol{x}_d(T) = x_f.$$

Formally, this problem is a two-point boundary value problem that can be quite difficult to solve in general. To solve this problem, we must find a solution to the differential equations (9) that satisfies the endpoint conditions. Given the nonlinear nature of the dynamics, it seems unlikely that one could find explicit solutions that satisfy the dynamics except in very special cases. The problem becomes intractable once we have additional constraints.

A closer inspection of this system shows that it is possible to understand the trajectories of the system by exploiting the particular structure of the dynamics. Suppose that we are given a trajectory for the rear wheels of the system, $x_d(t)$ and $y_d(t)$. From equation (9), we see that we can use this solution to solve for the angle of the car by writing

$$\frac{\dot{y}}{\dot{x}} = \frac{\sin\theta}{\cos\theta} \longrightarrow \theta_d = \tan^{-1}(\dot{y_d}/\dot{x_d}).$$

Furthermore, given $\theta$ we can solve for the velocity using the equation

$$\dot{x} = v\cos\theta \longrightarrow v_d = \dot{x}_d/\cos\theta_d,$$

assuming $\cos\theta_d \neq 0$ (if it is, use $v = \dot{y}/\sin\theta$). And given $\theta$, we can solve for $\phi$ using the relationship

$$\dot{\theta} = \frac{v}{l}\tan\phi \longrightarrow \phi_d = \tan^{-1}(\frac{L\dot{\theta}_d}{v_d}).$$

Hence all of the state variables and the inputs can be determined by the trajectory of the rear wheels and its derivatives. This property of a system is known as *differential flatness*.

**Definition 3.5.** *Differential Flatness a non-linear system*

$$\dot{\boldsymbol{x}} = \alpha(\boldsymbol{x}, \boldsymbol{u}), \tag{10}$$

*is differentially flat with flat output* $\boldsymbol{z}$ *if there exists a function* $\alpha$ *such that*

$$\boldsymbol{z} = \alpha(\boldsymbol{x}, \boldsymbol{u}, \dot{\boldsymbol{u}}, \ldots, \boldsymbol{u}^{(p)}), \tag{11}$$

*and such that the solutions to the system* $\boldsymbol{x}$ *and* $\boldsymbol{u}$ *can then be written as functions of the flat output* $\boldsymbol{z}$ *and a finite number of its derivatives:*

$$\begin{aligned} \boldsymbol{x} &= \beta(\boldsymbol{z}, \dot{\boldsymbol{z}}, \ldots, \boldsymbol{z}^{(q)}) \\ \boldsymbol{u} &= \gamma(\boldsymbol{z}, \dot{\boldsymbol{z}}, \ldots, \boldsymbol{z}^{(q)}). \end{aligned} \tag{12}$$

For a differentially flat system, all of the feasible trajectories for the system can be written as functions of a flat output $\boldsymbol{z}(\cdot)$ and its derivatives. Simply put, when a nonlinear system is differentially flat, the trajectory designer can plan in the flat output space and then directly map those to the appropriate control inputs without having to worry about violating the dynamics equations for the system. The number of flat outputs is always equal to the number of system inputs [Mur].

Differentially flat systems are useful in situations where explicit trajectory generation is required. Since the behavior of a flat system is determined by the flat outputs, we can plan trajectories in output space, and then map these to appropriate inputs. Suppose we wish to generate a feasible trajectory for the the nonlinear system

$$\dot{\boldsymbol{x}} = \alpha(\boldsymbol{x}, \boldsymbol{u}), \quad \boldsymbol{x}_d(0) = \boldsymbol{x}_0, \quad \boldsymbol{x}_d(T) = \boldsymbol{x}_f.$$

If the system is differentially flat then

$$\begin{aligned} \boldsymbol{x}(0) &= \beta(\boldsymbol{z}(0), \dot{\boldsymbol{z}}(0), \ldots, \boldsymbol{z}^{(q)}(0)) = \boldsymbol{x}_0 \\ \boldsymbol{x}(T) &= \beta(\boldsymbol{z}(T), \dot{\boldsymbol{z}}(T), \ldots, \boldsymbol{z}^{(q)}(T)) = \boldsymbol{x}_f \end{aligned} \tag{13}$$

and any trajectory for $\boldsymbol{z}$ that satisfies these boundary conditions can be mapped to a feasible trajectory for the nonlinear system. This means that $\boldsymbol{z}$ can be defined however the designer chooses. A common choice is to use a set of smooth basis functions $\boldsymbol{\psi}(t)$

$$\boldsymbol{z}(t) = \sum_{i=1}^{N} \alpha_i \boldsymbol{\psi}_i(t), \quad \alpha_i \in \mathbb{R} \tag{14}$$

For example for a scalar flat output $z$, one choice could be to use polynomial basis functions $\psi_1(t) = 1$, $\psi_2(t) = t$, $\psi_3(t) = t^2$, and so on. Choosing a set of smooth basis functions leads to representation that is linear in the coefficients $\alpha_i$, and thus can be easily solved algebraically.

Taking derivatives of the representation in Eq. 14 gives

$$\begin{aligned} \dot{z}(t) &= \sum_{i=1}^{N} \alpha_i \dot{\psi}_i(t) \\ &\vdots \\ z^{(q)}(t) &= \sum_{i=1}^{N} \alpha_i \psi_i^{(q)}(t). \end{aligned} \tag{15}$$

Given the initial and final conditions $z(0)$, $\dot{z}(0)$, $\ldots, z^{(q)}(0)$ and $z(T)$, $\dot{z}(T)$, $\ldots, z^{(q)}(T)$ that satisfy the boundary conditions in Eq. 13, the coefficients $\alpha_i$ can be found by solving the following linear system, assuming the matrix is full rank.

$$\begin{bmatrix} \psi_1(0) & \psi_2(0) & \ldots & \psi_N(0) \\ \dot{\psi}_1(0) & \dot{\psi}_2(0) & \ldots & \dot{\psi}_N(0) \\ \vdots & \vdots & & \vdots \\ \psi_1^{(q)}(0) & \psi_2^{(q)}(0) & \ldots & \psi_N^{(q)}(0) \\ \psi_1(T) & \psi_2(T) & \ldots & \psi_N(T) \\ \dot{\psi}_1(T) & \dot{\psi}_2(T) & \ldots & \dot{\psi}_N(T) \\ \vdots & \vdots & & \vdots \\ \psi_1^{(q)}(T) & \psi_2^{(q)}(T) & \ldots & \psi_N^{(q)}(T) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix} = \begin{bmatrix} z(0) \\ \dot{z}(0) \\ \vdots \\ z^{(q)}(0) \\ z(T) \\ \dot{z}(T) \\ \vdots \\ z^{(q)}(T). \end{bmatrix} \tag{16}$$

To summarize, for differentially flat nonlinear systems, we work in terms of the *flat outputs* of the system to solve for a feasible trajectory that satisfies the boundary constraints in a computationally efficiennt manner. We can then use the flat trajectory to back out the state trajectories and control inputs.

# References


[Kir04] Donald E. Kirk. *Optimal Control Theory: An Introduction (Dover Books on Electrical Engineering)*. Dover Publications, 2004.

[Mur] Richard M Murray. Optimization-Based Control. page 111.