Robot Autonomy

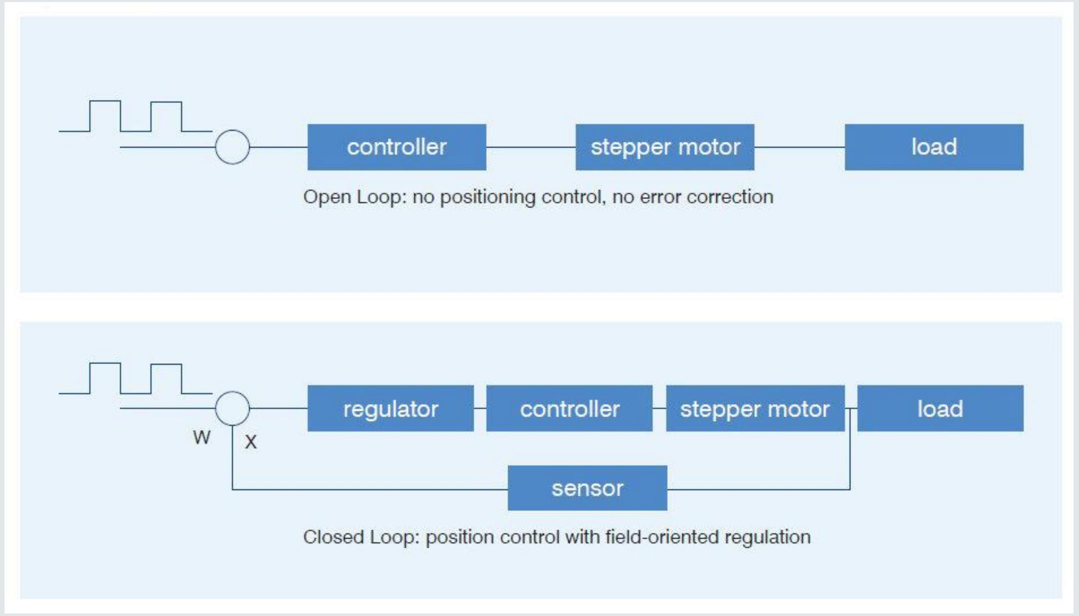
DR. RISMAN ADNAN MATTOTORANG

TELKOM UNIVERSITY

Principles of Robot Autonomy I

W3: Open-loop motion control and differential flatness

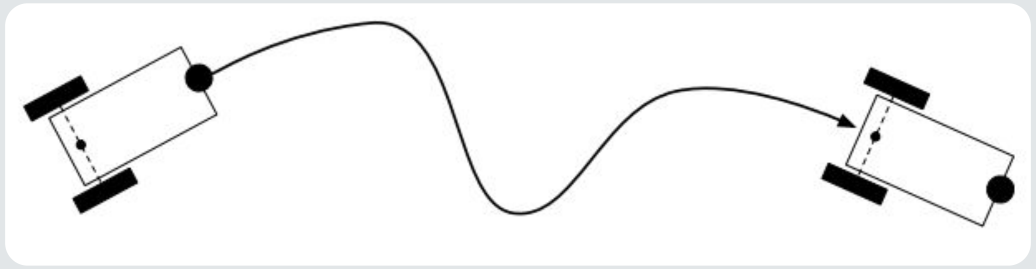
- Motion Control
- Kinematic/Dynamic Model
- Optimal Control Problem
- Open Loop Control
- Direct Methods
- Indirect Methods
- Differential Flatness


controller
stepper motor
load
Open Loop: no positioning control, no error correction
regulator
controller
stepper motor
load
W
X
sensor
Closed Loop: position control with field-oriented regulation

Open vs Close Loop Control

Motion Control

- Given a nonholonomic system, how to control its motion from an initial configuration to a final, desired configuration



- Aim
  - Learn about main techniques in optimal control and trajectory optimization
  - Learn about differential flatness and its use for trajectory optimization

- Further Readings (Home Work)
  - B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. Robotics: modelling, planning and control. 2010. Chapter 11.

Kinematic / dynamic models

- In lecture 1 we saw how to derive models that describe the equations of motion of a robot in the form of differential equations (DE)

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

- DEs are equations relating the derivatives of an unknown function to the unknown function itself and known quantities. $\boldsymbol{x}$ can be thought as robot state in term of generalized coordinate, $\boldsymbol{u}$ is control input and $\boldsymbol{a}$ is the model.

- DEs can be integrated numerically, for example, via the **Euler method**

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \Delta t_i \mathbf{a}(\mathbf{x}_i, \mathbf{u}_i, t_i), \quad i = 0, \dots, N-1$$

where $\Delta t_i = t_{i+1} - t_i$, $\mathbf{u}_i = \mathbf{u}(t_i)$, and $\mathbf{x}_0 = \mathbf{x}(t_0)$

Optimal control problem

- An optimal control problem seeks an admissible control ***u(t)*** which causes the system to follow an admissible trajectory ***x(t)*** that minimizes a performance metric ***J(x(t), u(t), t)***.

The problem:

$$\begin{aligned} \min_{\mathbf{u}} \quad & h(\mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\, dt \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t) \\ & \mathbf{x}(t) \in \mathcal{X}, \quad \mathbf{u}(t) \in \mathcal{U} \end{aligned}$$

where $\boldsymbol{x}(t) \in R^n, \boldsymbol{u}(t) \in R^m$, and $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$

- We'll focus on the case $\chi = Rn$; state constraints will be addressed in the context of **motion planning**

Form of optimal control

- If a functional relationship of the form

$$u^*(t) = \pi(x(t), t)$$

can be found, then the optimal control is said to be in **closed-loop form**

- If the optimal control law is determined as a function of time for a specified initial state value

$$u^*(t) = f(x(t_0), t)$$

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

- A good compromise: two-step design

$$u^*(t) = \underset{\text{Reference control (open-loop)}}{u_d(t)} + \underset{\text{Trajectory-tracking law (closed-loop)}}{\pi}\big(x(t), \underbrace{x(t) - \overset{\text{Reference trajectory}}{xd}(t)}_{\text{Tracking error}}\big)$$

Open-loop control

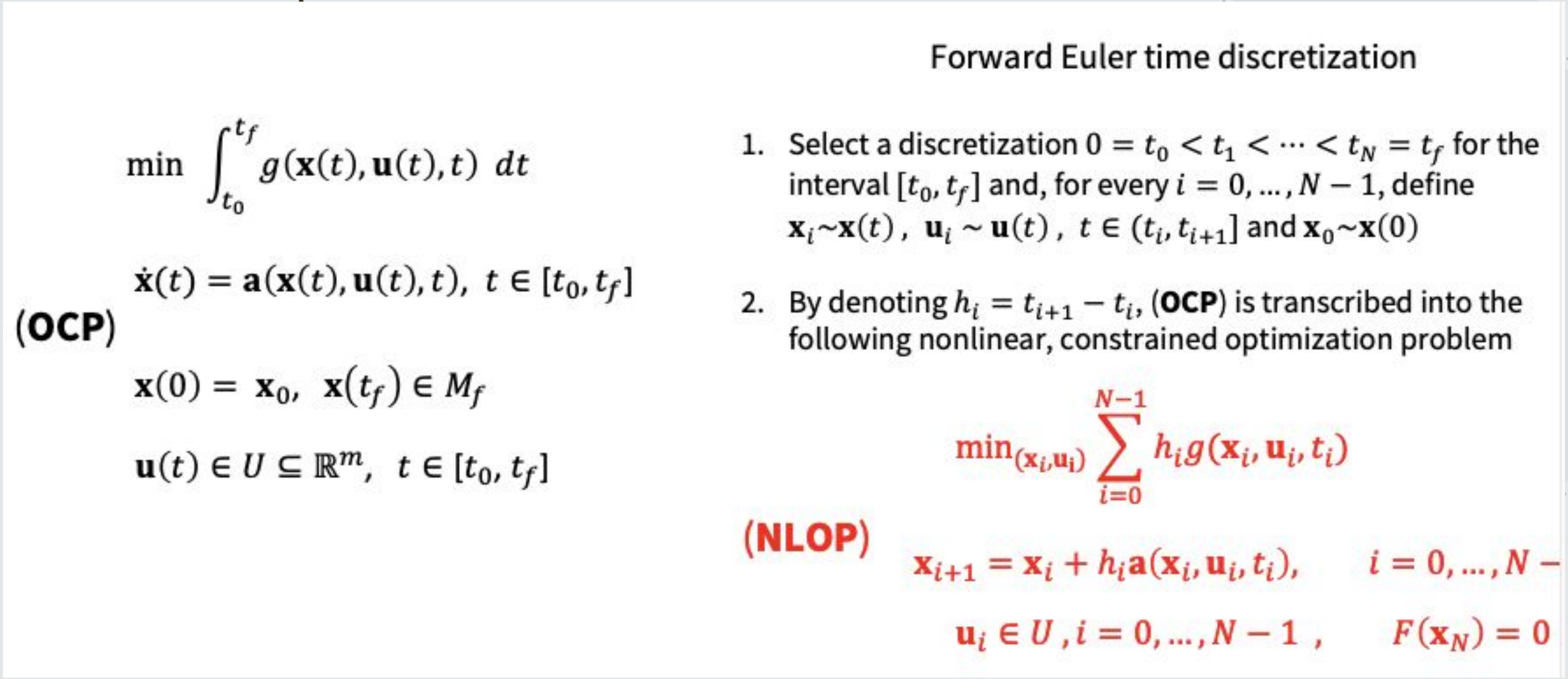
We want to find

$$u^{*}(t) = f(x(t_0), t)$$

In general, two broad classes of methods:

1. **Direct methods:** transcribe infinite problem into finite dimensional, nonlinear programming (NLP) problem, and solve NLP ⇒ "First discretize, then optimize"
2. **Indirect methods:** attempt to find a minimum point "indirectly," by solving the necessary conditions of optimality ⇒ "First optimize, then discretize"

Direct methods - nonlinear programming transcription



(OCP)

$$\min \int_{t_0}^{t_f} g(\mathbf{x}(t), \mathbf{u}(t), t)\ dt$$

$$\dot{\mathbf{x}}(t) = \mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t),\ t \in [t_0, t_f]$$

$$\mathbf{x}(0) = \mathbf{x}_0,\ \ \mathbf{x}(t_f) \in M_f$$

$$\mathbf{u}(t) \in U \subseteq \mathbb{R}^m,\ \ t \in [t_0, t_f]$$

Forward Euler time discretization

1. Select a discretization $0 = t_0 < t_1 < \cdots < t_N = t_f$ for the interval $[t_0, t_f]$ and, for every $i = 0, \dots, N-1$, define $\mathbf{x}_i \sim \mathbf{x}(t)$, $\mathbf{u}_i \sim \mathbf{u}(t)$, $t \in (t_i, t_{i+1}]$ and $\mathbf{x}_0 \sim \mathbf{x}(0)$
2. By denoting $h_i = t_{i+1} - t_i$, (**OCP**) is transcribed into the following nonlinear, constrained optimization problem

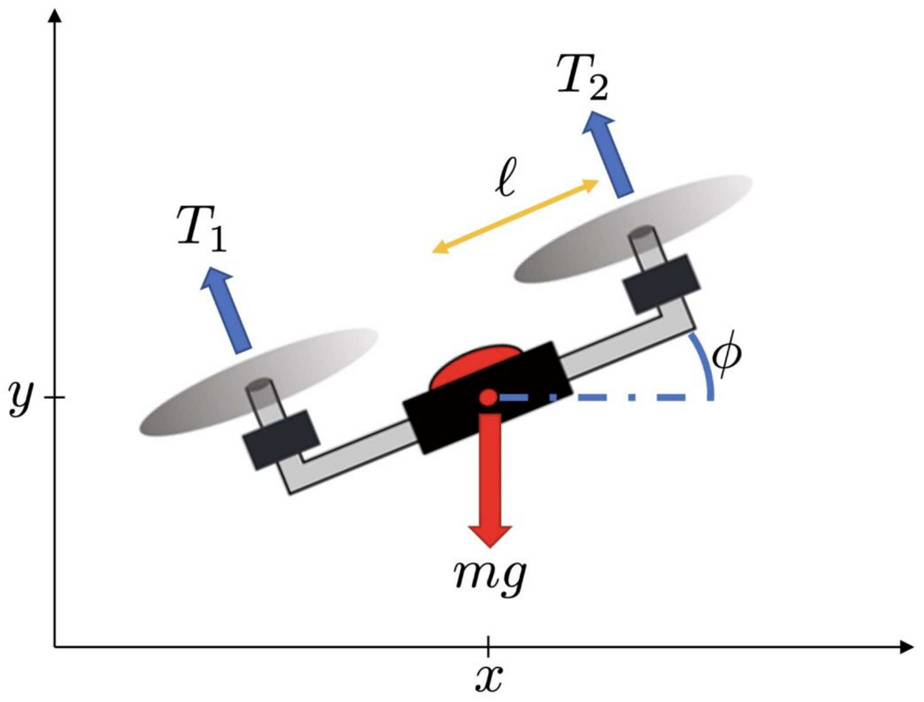
(NLOP)

$$\min_{(\mathbf{x}_i, \mathbf{u}_i)} \sum_{i=0}^{N-1} h_i g(\mathbf{x}_i, \mathbf{u}_i, t_i)$$

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h_i \mathbf{a}(\mathbf{x}_i, \mathbf{u}_i, t_i), \qquad i = 0, \dots, N -$$

$$\mathbf{u}_i \in U\ , i = 0, \dots, N-1\ , \qquad F(\mathbf{x}_N) = 0$$

Illustrative Example: Planar Quadrotor



$$\min \int_0^{t_f} T_1(t)^2 + T_2(t)^2 dt$$

(energy objective)

subject to dynamics

$$\begin{bmatrix} \dot{x} \\ \dot{v}_x \\ \dot{y} \\ \dot{v}_y \\ \dot{\phi} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v_x \\ \frac{-(T_1+T_2)\sin\phi}{m} \\ v_y \\ \frac{(T_1+T_2)\cos\phi}{m} - g \\ \omega \\ \frac{(T_2-T_1)\ell}{I_{zz}} \end{bmatrix}$$

Direct methods – software packages

Some software packages:
- DIDO: http://www.elissarglobal.com/academic/products/
- PROPT: http://tomopt.com/tomlab/products/propt/
- GPOPS: http://www.gpops2.com/
- CasADi: https://github.com/casadi/casadi/wiki
- ACADO: http://acado.github.io/

For an in-depth study of direct and indirect methods, see AA203 “Optimal and Learning-based Control” (Spring 2020)

Indirect methods – main ideas

Indirect methods entail three main steps:

1. Derive necessary conditions of optimality
   - leading to a two-point boundary value problem
2. Discretize such conditions
3. Solve the resulting system

For an in-depth study of direct and indirect methods, see AA203 “Optimal and Learning-based Control” (Spring 2020)

Differential flatness

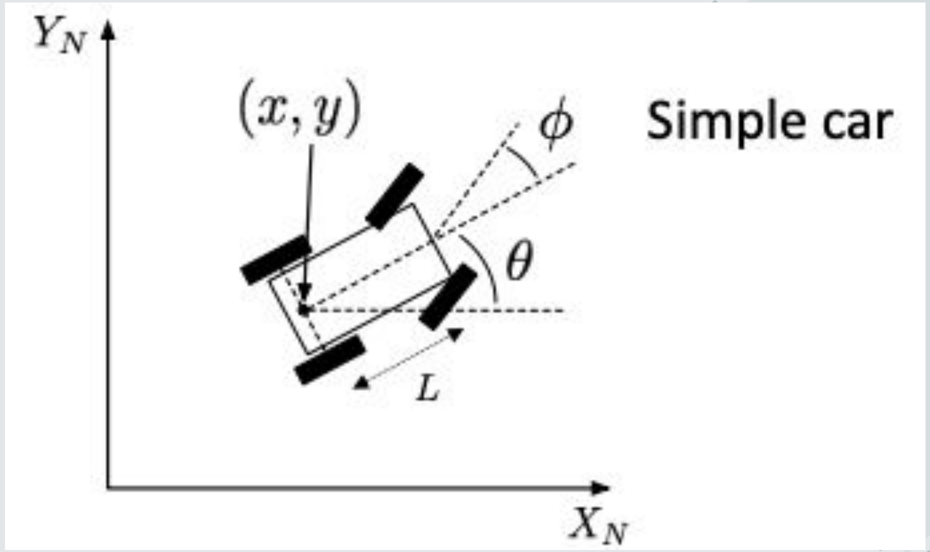
- Computing “good” feasible trajectories is often sufficient for trajectory generation purposes, and typically much faster than computing optimal ones

- A class of systems for which trajectory generation is particularly easy are the so-called **differentially flat systems**

- Reference: M. J. Van Nieuwstadt and R. M. Murray. Real-time trajectory generation for differentially flat systems. 1998.

Motivating example: simple car

- Consider the problem of finding a feasible solution that satisfies the dynamics:
  $$\dot{x} = a(x,u), \quad x(0) = x_0, \quad x(t_f) = x_f$$
- Example: simple car steering
  $$\dot{x} = \cos\theta \; v, \quad \dot{y} = \sin\theta \, v, \quad \theta = \frac{v}{L}\tan\phi$$
- Stated: (x, y, θ)
- Input: (v, ϕ)



Structure of the dynamics for simple car steering

- Suppose we are given a (smooth) trajectory for the rear wheels of the system, x(t) and y(t)

1. we can use this solution to solve for the angle of the car by writing
$$\frac{\dot{y}}{x} = \frac{\sin\theta}{\cos\theta} \rightarrow \theta = \tan^{-1}\left(\frac{\dot{y}}{\dot{x}}\right)$$

2. we can solve for the velocity
$$\dot{x} = v\ \cos\theta\ \rightarrow\ v = \ \dot{x}/\cos\theta\ (\text{or } v = \ \dot{x}/\sin\theta)$$

3. and finally
$$\dot{\theta} = \frac{v}{L}\tan\phi\ \rightarrow\ \phi = \ \tan^{-1}\left(\frac{L\dot{\theta}}{v}\right)$$

Structure of the dynamics for simple car steering

- **Bottom line:** all of the state variables and the inputs can be determined by the trajectory of the rear wheels and its derivatives!
- We say that the system is *differentially flat with flat output* $z = (x, y)$
- This provides a dramatic simplification for the purposes of trajectory generation (more on this later)

Differential flatness

- **Differential flatness:** A nonlinear system $x = \dot{a}(x, u)$ is differentially flat if there exists a function α such that
$$z = \alpha(x, u, \ldots, u^{(p)})$$

- and we can write the solutions of the nonlinear system as functions of ▯ and a finite number of derivatives
$$x = \beta(z, \dot{z}, \ldots, z^{(q)})$$
$$u = \gamma(z, \dot{z}, \ldots, z^{(q)})$$

- In words, a system is differentially flat if we can find a set of outputs (equal in number to the number of inputs) such that all states and inputs can be determined from these outputs *without integration*

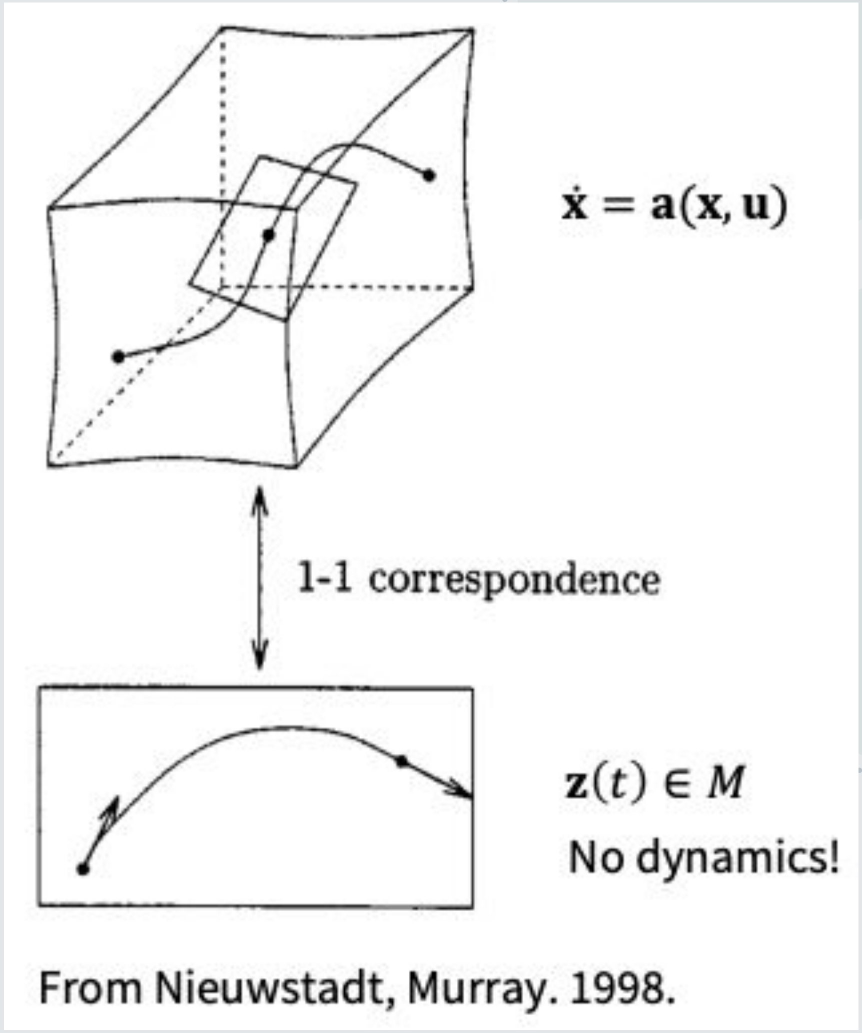
Differential flatness

- Implication for trajectory generation: to every curve $t \rightarrow z(t)$ enough differentiable, there corresponds a trajectory

$$t \rightarrow \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} = \begin{pmatrix} \beta(z(t), \dot{z}(t), \ldots, z^{(q)}(t)) \\ \gamma(z(t), \dot{z}(t), \ldots, z^{(q)}(t)) \end{pmatrix}$$

that identically satisfies the system equations

- The simple car is differentially flat with the position of the rear wheels as the flat output



From Nieuwstadt, Murray. 1998.

Practical implications

- This leads to a simple, yet effective strategy for trajectory generation

1. Find the initial and final conditions for the flat output:

Given	Find

2. Build a smooth curve $t \to z(t)$ for $t \in [t_0, t_f]$ by interpolation, possibly satisfying further constraints
3. Deduce the corresponding trajectory $t \to (x(t), u(t))$

More on Step 2

- We can parameterize the flat output trajectory using a set of smooth basis functions $\psi_i(t)$

$$z_j(t) = \sum_{i=1}^{N} \alpha_i^{[j]} \psi_i(t)$$

- and then solve **(Problem 1 in pset)**

$$\begin{bmatrix} \psi_1(t_0) & \psi_2(t_0) & \dots & \psi_N(t_0) \\ \dot\psi_1(t_0) & \dot\psi_2(t_0) & \dots & \dot\psi_N(t_0) \\ \vdots & \vdots & & \vdots \\ \psi_1^{(q)}(t_0) & \psi_2^{(q)}(t_0) & \dots & \psi_N^{(q)}(t_0) \\ \psi_1(t_f) & \psi_2(t_f) & \dots & \psi_N(t_f) \\ \dot\psi_1(t_f) & \dot\psi_2(t_f) & \dots & \dot\psi_N(t_f) \\ \vdots & \vdots & & \vdots \\ \psi_1^{(q)}(t_f) & \psi_2^{(q)}(t_f) & \dots & \psi_N^{(q)}(t_f) \end{bmatrix} \begin{bmatrix} \alpha_1^{[j]} \\ \alpha_2^{[j]} \\ \vdots \\ \alpha_N^{[j]} \end{bmatrix} = \begin{bmatrix} z_j(t_0) \\ \dot z_j(t_0) \\ \vdots \\ z_j^{(q)}(t_0) \\ z_j(t_f) \\ \dot z_j(t_f) \\ \vdots \\ z_j^{(q)}(t_f) \end{bmatrix}$$

For more details see: “Optimization-Based Control” by Richard Murray

Key points

- Nominal trajectories and inputs can be computed in a computationally-efficient way (solving a set of *algebraic equations*)

- Other constraints on the system, such as input bounds, can be transformed into the flat output space and (typically) become limits on the curvature or higher order derivative properties of the curve
  - Alternative: **time scaling,** i.e., break down trajectory planning in (1) finding a path (via differential flatness) and (2) defining a timing law on the path **(Problem 1 in pset)** -- more on this next time

- If there is a performance index for the system, this index can be transformed and becomes a functional depending on the flat outputs and their derivatives up to some order

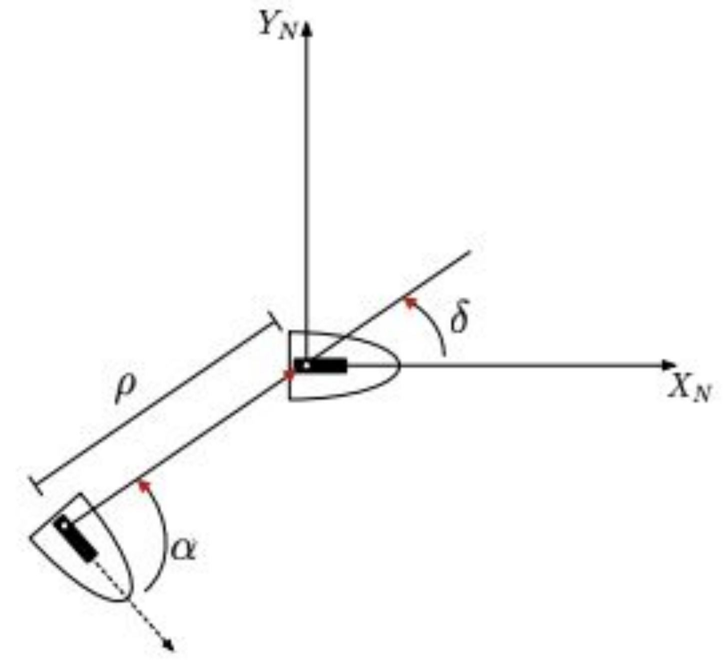
When is a system differentially flat?

- The existence of a general, computable criterion so as to decide if the dynamical system $\dot{x} = a(x, u)$ is differentially flat remains open
- Some results in this direction are, however, available

- Further readings:
- Application to trajectory optimization:
  1. M. J. Van Nieuwstadt and R. M. Murray. Real-time trajectory generation for differentially flat systems. 1998
  2. R. M. Murray, M. Rathinam, and W. Sluis. Differential flatness of mechanical control systems: A catalog of prototype systems. 1995
  3. B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. Robotics: modelling, planning and control. 2010
- Theory:
  1. J. Levine. Analysis and control of nonlinear systems: A flatness-based approach. 2009
  2. G. G. Rigatos, Gerasimos. Nonlinear control and filtering using differential flatness approaches: applications to electromechanical systems. 2015

Next Lecture

- Trajectory tracking and closed-loop control


Y_N
X_N
ρ
δ
α