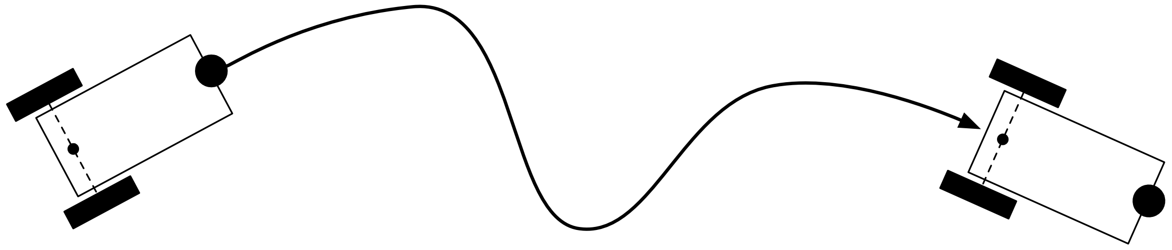
AA 274
Principles of Robotic Autonomy

Open-loop motion control and differential flatness

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
  - Appreciate relative benefits of open-loop control, closed-loop control and their combination (two-degree of freedom design)
- Readings
  - SNS: 3.4-3.6

Optimal control problem

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 $\mathcal{X} = R^n$; state constraints will be addressed in the context of motion planning
- Good reference: D. K. Kirk. Optimal Control Theory: An introduction. Dover Publications, 2004.

Form of optimal control

- If a functional relationship of the form

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

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

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

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

- A good compromise: two-step design

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

Open-loop control

- We want to find

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

- In general, two broad classes of methods:

1. Indirect methods: attempt to find a minimum point “indirectly,” by solving the necessary conditions of optimality $\Rightarrow$ “First optimize, then discretize”

2. Direct methods: transcribe infinite problem into finite dimensional, nonlinear programming (NLP) problem, and solve NLP $\Rightarrow$ “First discretize, then optimize”

Preliminaries: constrained optimization

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{subject to} \quad & h_i(\mathbf{x}) = 0, \qquad i = 1, \ldots, m \end{aligned}$$

- Form Lagrangian function L: $R^{n+m} \to R$

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i h_i(\mathbf{x})$$

- If $\boldsymbol{x}^*$a is a local minimum which is *regular*, the NOC conditions are

$$\nabla_{\mathbf{x}} L(\mathbf{x}^*, \lambda^*) = 0$$
$$\nabla_{\lambda} L(\mathbf{x}^*, \lambda^*) = 0$$

- First order condition represents a system of *n + m* equations with *n + m* unknowns

Indirect methods: NOC

Assume no state/control constraints

- Form Hamiltonian $H := g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t)[\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)]$
- Hamiltonian equations

$$\dot{\mathbf{x}}^*(t) = \frac{\partial H}{\partial \mathbf{p}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t)$$

$$\dot{\mathbf{p}}^*(t) = -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t)$$

$$\mathbf{0} = \frac{\partial H}{\partial \mathbf{u}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t)$$

- Boundary conditions: $\mathbf{x}^*(t_0) = \mathbf{x}_0$, and

$$\left[\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}^*(t_f), t_f) - \mathbf{p}^*(t_f)\right]^T \delta \mathbf{x}_f + \left[H(\mathbf{x}^*(t_f), \mathbf{u}^*(t_f), \mathbf{p}^*(t_f), t_f) + \frac{\partial h}{\partial t}(\mathbf{x}^*(t_f), t_f)\right] \delta t_f = 0$$

Indirect methods: NOC

Assume control inequality constraints: e.g., $|u_i| \leq \bar{u}_i$ for all *i*

- Form Hamiltonian $H := g(\mathbf{x}(t), \mathbf{u}(t), t) + \mathbf{p}^T(t)[\mathbf{a}(\mathbf{x}(t), \mathbf{u}(t), t)]$
- Hamiltonian equations

$$\dot{\mathbf{x}}^*(t) = \frac{\partial H}{\partial \mathbf{p}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t)$$

$$\dot{\mathbf{p}}^*(t) = -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t)$$

$$H(\mathbf{x}^*(t), \mathbf{u}^*(t), \mathbf{p}^*(t), t) \leq H(\mathbf{x}^*(t), \mathbf{u}(t), \mathbf{p}^*(t), t), \quad \forall \mathbf{u}(t) \in \mathcal{U}$$

- Boundary conditions: $\mathbf{x}^*(t_0) = \mathbf{x}_0$, and

$$\left[\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}^*(t_f), t_f) - \mathbf{p}^*(t_f)\right]^T \delta \mathbf{x}_f + \left[H(\mathbf{x}^*(t_f), \mathbf{u}^*(t_f), \mathbf{p}^*(t_f), t_f) + \frac{\partial h}{\partial t}(\mathbf{x}^*(t_f), t_f)\right] \delta t_f = 0$$

Substitutions for boundary conditions

Problem		Substitution
t_f	fixed	$\delta t_f = 0$
$\mathbf{x}(t_f)$	fixed	$\delta \mathbf{x}_f = 0$

BC

$$\mathbf{x}^*(t_0) = \mathbf{x}_0$$
$$\mathbf{x}^*(t_f) = \mathbf{x}_f$$

Problem		Substitution
t_f	fixed	$\delta t_f = 0$
$\mathbf{x}(t_f)$	free	$\delta \mathbf{x}_f$ arbitrary

BC

$$\mathbf{x}^*(t_0) = \mathbf{x}_0$$
$$\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}^*(t_f)) - \mathbf{p}^*(t_f) = \mathbf{0}$$

Problem		Substitution
t_f	free	δt_f arbitrary
$\mathbf{x}(t_f)$	fixed	$\delta \mathbf{x}_f = 0$

BC

$$\mathbf{x}^*(t_0) = \mathbf{x}_0$$
$$\mathbf{x}^*(t_f) = \mathbf{x}_f$$
$$H(\mathbf{x}^*(t_f), \mathbf{u}^*(t_f), \mathbf{p}^*(t_f), t_f) + \frac{\partial h}{\partial t}(\mathbf{x}^*(t_f), t_f) = 0$$

Problem		Substitution
t_f	free	δt_f arbitrary
$\mathbf{x}(t_f)$	free	$\delta \mathbf{x}_f$ arbitrary

BC

$$\mathbf{x}^*(t_0) = \mathbf{x}_0$$
$$\frac{\partial h}{\partial \mathbf{x}}(\mathbf{x}^*(t_f), t_f) - \mathbf{p}^*(t_f) = \mathbf{0}$$
$$H(\mathbf{x}^*(t_f), \mathbf{u}^*(t_f), \mathbf{p}^*(t_f), t_f) + \frac{\partial h}{\partial t}(\mathbf{x}^*(t_f), t_f) = 0$$

Indirect methods: practical aspects

Reference for NOC: D. K. Kirk. Optimal Control Theory: An introduction. Dover Publications, 2004.

In practice: To obtain solution to the necessary conditions for optimality, one needs to solve two-point boundary value problems

- For example, in Python: https://pythonhosted.org/scikits.bvp_solver/
- Allows to solve problem of the form

$$\dot{\mathbf{z}} = \mathbf{g}(\mathbf{z}, t), \qquad \mathbf{l}(\mathbf{z}(t_0), \mathbf{z}(t_f)) = \mathbf{0}$$

- Syntax: `solve(bvp_problem, solution_guess)`
- In Matlab: `bvp4c`

Example

$$\dot{z}_1(t) = z_2(t)$$
$$\dot{z}_2(t) = -|z_1(t)|$$
$$z_1(0) = 0$$
$$z_1(4) = -2$$

Extensions

- What about problems whose necessary conditions do not fit directly the "standard" form (e.g., free end time problems)?
- Handy tricks exist to convert problems into standard form: Ascher, U., & Russell, R. D. (1981). Reformulation of boundary value problems into "standard" form. SIAM review, 23(2), 238-254.

Important case: free final time (Problem 1 in pset)

1. Rescale time so that $\tau = t/t_f$, then $\tau \in [0,1]$
2. Change derivatives $\frac{d}{d\tau} := t_f \frac{d}{dt}$
3. Introduce dummy state *r* that corresponds to t_f with dynamics $\dot{r} = 0$
4. Replace all instances of t_f with *r*

Example

- Dynamics:

$$\ddot{x} = u,\ x(0) = 10,\ \dot{x}(0) = 0,\ x(t_f) = 0,\ \dot{x}(t_f) = 0$$

- Cost:

$$J = \frac{1}{2}\alpha t_f^2 + \frac{1}{2}\int_{t_0}^{t_f} b\, u^2(t)\, dt$$

- Analytical solution gives:

$$t_f = (1800b/\alpha)^{1/5}$$

Example (solution)

- Define state as $\boldsymbol{z} = [\boldsymbol{x}, \boldsymbol{p}, r]$
- BC are:

$$x_1(0) = 10,\ x_2(0) = 0,\ x_1(t_f) = 0,\ x_2(t_f) = 0,$$
$$-0.5b(-p_2(t_f)/b)^2 + \alpha t_f = 0$$

- BVP becomes

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\frac{d\mathbf{z}}{d\tau} = t_f \frac{d\mathbf{z}}{dt} = z_5 \begin{bmatrix} A & -B\,[0\ 1]/b & 0 \\ 0 & -A' & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{z}$$

$$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- BC become

$$z_1(0) = 10,\ z_2(0) = 0,\ z_1(1) = 0,\ z_2(1) = 0,$$
$$-0.5b(-z_4(1)/b)^2 + \alpha z_5(1) = 0$$

Direct methods

Main methods:

- control parameterization methods
  - direct shooting
- state and control parameterization methods
  - local collocation (e.g., orthogonal collocation)
  - global collocation (e.g., pseudospectral methods)

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/

Special case: differential flatness

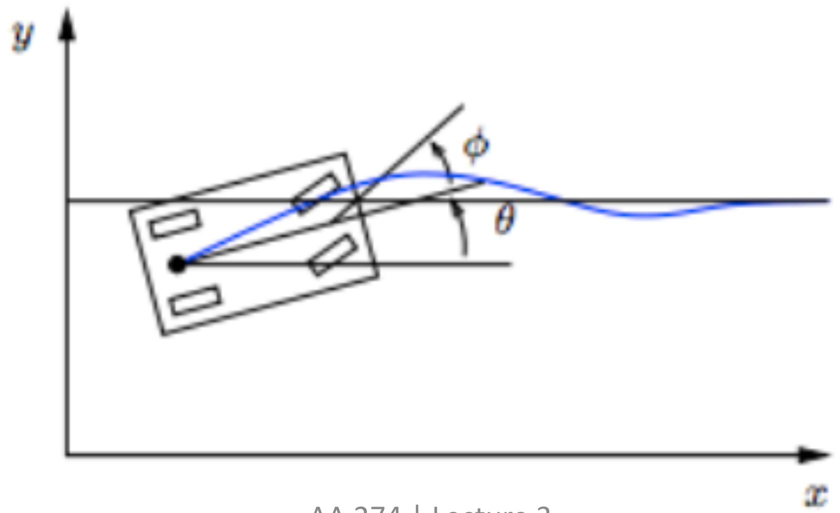
Consider the problem of finding a *feasible* solution that satisfies the dynamics:

$$\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}, \mathbf{u}), \qquad \mathbf{x}(0) = \mathbf{x}_0, \qquad \mathbf{x}(t_f) = \mathbf{x}_f$$

Example: vehicle steering

$$\dot{x} = \cos\theta\, v \qquad \dot{y} = \sin\theta\, v, \qquad \dot{\theta} = \frac{v}{l}\tan\phi$$

- State: (x, y, θ)
- Inputs: (v, ϕ)



Structure of the dynamics for vehicle steering

- Suppose we are given a 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

$$\frac{\dot{y}}{\dot{x}} = \frac{\sin\theta}{\cos\theta} \qquad \Rightarrow \qquad \theta = \tan^{-1}\left(\frac{\dot{y}}{\dot{x}}\right)$$

  2. we can solve for the velocity

$$\dot{x} = v\cos\theta \qquad \Rightarrow \qquad v = \dot{x}/\cos\theta \qquad (\text{or } v = \dot{y}/\sin\theta)$$

  3. and finally

$$\dot{\theta} = \frac{v}{l}\tan\phi \qquad \Rightarrow \qquad \phi = \tan^{-1}\left(\frac{l\dot{\theta}}{v}\right)$$

Differential flatness

Bottom line: all of the state variables and the inputs can be determined by the trajectory of the rear wheels and its derivatives!

Differential flatness: A nonlinear system $\dot{\boldsymbol{x}} = \boldsymbol{a}(\boldsymbol{x}, \boldsymbol{u})$ is differentially flat if there exists a function α such that

$$\mathbf{z} = \alpha(\mathbf{x}, \mathbf{u}, \ldots, \mathbf{u}^{(p)})$$

and we can write the solutions of the nonlinear system as functions of $\boldsymbol{z}$ and a finite number of derivatives

$$\mathbf{x} = \beta(\mathbf{z}, \dot{\mathbf{z}}, \ldots, \mathbf{z}^{(q)})$$

$$\mathbf{u} = \gamma(\mathbf{z}, \dot{\mathbf{z}}, \ldots, \mathbf{z}^{(q)})$$

Practical implications (1/2)

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

Application to trajectory generation (Problem 2 in pset):

- If the system is differentially flat then

$$\begin{aligned}\mathbf{x}(0) &= \beta(\mathbf{z}(0), \dot{\mathbf{z}}(0), \ldots, \mathbf{z}^{(q)}(0)) = \mathbf{x}_0 \\ \mathbf{x}(t_f) &= \beta(\mathbf{z}(t_f), \dot{\mathbf{z}}(t_f), \ldots, \mathbf{z}^{(q)}(t_f)) = \mathbf{x}_f\end{aligned}$$

- Thus any trajectory for $\boldsymbol{z}(\cdot)$ that satisfies these boundary conditions will be a feasible trajectory for the system!

Practical implications (2/2)

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

$$\mathbf{z}(t) = \sum_{i=1}^{N} \alpha_i \psi_i(t)$$

- and then solve

$$\begin{bmatrix} \psi_1(0) & \psi_2(0) & \dots & \psi_N(0) \\ \dot{\psi}_1(0) & \dot{\psi}_2(0) & \dots & \dot{\psi}_N(0) \\ \vdots & \vdots & & \vdots \\ \psi_1^{(q)}(0) & \psi_2^{(q)}(t_f) & \dots & \psi_N^{(q)}(t_f) \\ \psi_1(t_f) & \psi_2(t_f) & \dots & \psi_N(t_f) \\ \dot{\psi}_1(t_f) & \dot{\psi}_2(t_f) & \dots & \dot{\psi}_N(t_f) \\ \vdots & \vdots & & \vdots \\ \psi_1^{(q)}(t_f) & \psi_2^{(q)}(t_f) & \dots & \psi_N^{(q)}(t_f) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1(0) \\ \dot{\mathbf{z}}_1(0) \\ \vdots \\ \mathbf{z}_1^{(q)}(0) \\ \mathbf{z}_1(t_f) \\ \dot{\mathbf{z}}_1(t_f) \\ \vdots \\ \mathbf{z}_1^{(q)}(t_f) \end{bmatrix}$$

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 2 in pset)

- 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

Next time: closed-loop control and sensors