Optimal Control

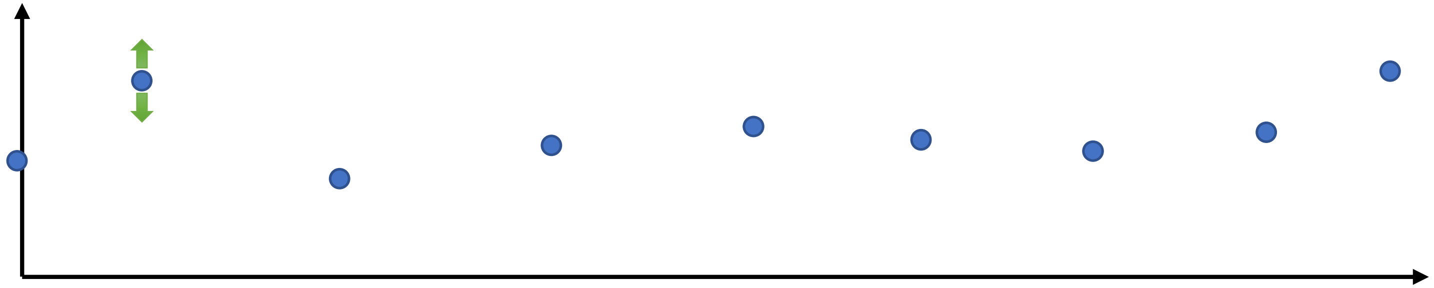
CMPT 882

Feb. 8

Nonlinear Optimization

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, i = 1, \dots, n \\ & h_j(x) = 0, j = 1, \dots, m \end{aligned}$$

- Nonlinear optimization:
  - Decision variable is $x \in \mathbb{R}^n$



Optimal Control

$$\underset{u(\cdot)}{\text{minimize}} \; \overbrace{l\big(x(t_f), t_f\big)}^{\text{Final cost}} + \overbrace{\int_0^{t_f} c(x(t), u(t), t)dt}^{\text{Running cost}}$$

Cost functional, $J\big(x(\cdot), u(\cdot)\big)$

$$\text{subject to } \dot{x}(t) = f\big(x(t), u(t)\big)$$

Dynamic model

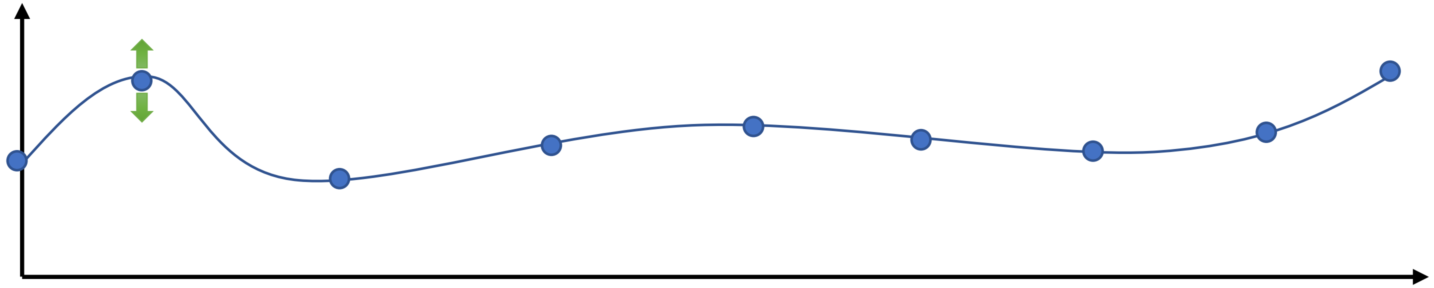
$$g\big(x(t), u(t)\big) \geq 0$$

Additional constraints

- Eg. actuation limits

$$x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, x(0) = x_0$$

- Nonlinear optimization:
  - Decision variable is $x \in \mathbb{R}^n$
- Optimal control:
  - Decision variable is a **function** $u(\cdot)$



Optimal Control

Final cost — Running cost

$$\underset{u(\cdot)}{\text{minimize}}\ \underbrace{l\big(x(t_f), t_f\big)}_{\text{Final cost}} + \underbrace{\int_0^{t_f} c(x(t), u(t), t)dt}_{\text{Running cost}}$$

Cost functional, $J\big(x(\cdot), u(\cdot)\big)$

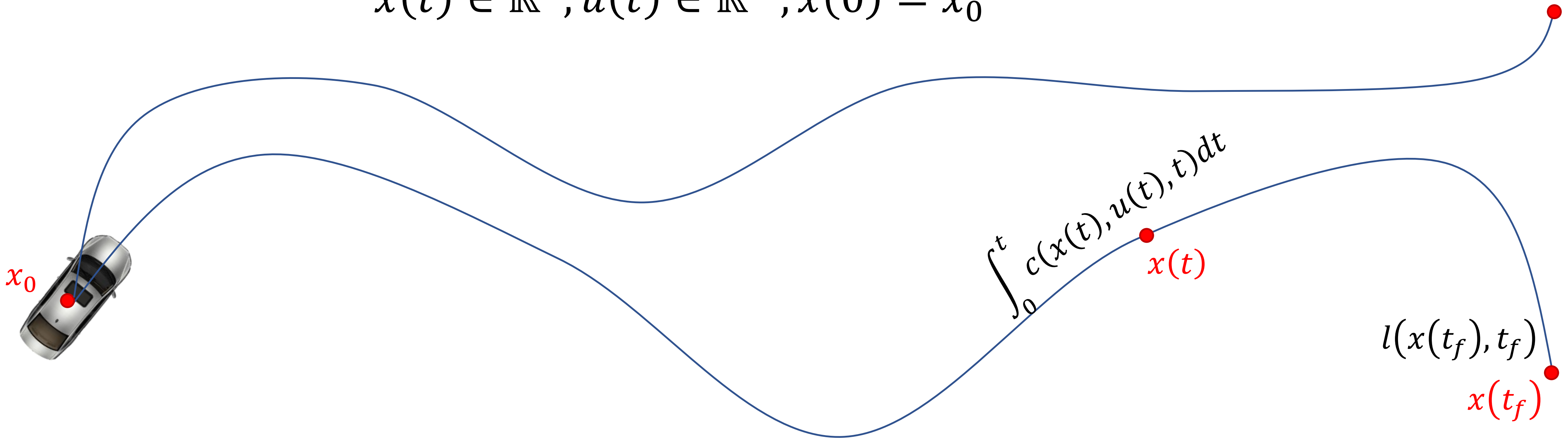
$$\text{subject to } \dot{x}(t) = f\big(x(t), u(t)\big)$$

Dynamic model

$$g\big(x(t), u(t)\big) \geq 0$$

Additional constraints
- Eg. actuation limits

$$x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, x(0) = x_0$$



Optimal Control

$$\underset{u(\cdot)}{\text{minimize}}\; l\big(x(t_f), t_f\big) + \int_0^{t_f} c(x(t), u(t), t)\,dt$$

$$\text{subject to } \dot{x}(t) = f\big(x(t), u(t)\big)$$

- Observation 1: Discretize time → nonlinear optimization problem
- Fact 1: Minimizing “cost” is same as maximizing “reward”
- Fact 2: Discretize time + maximizing reward → reinforcement learning problem

Optimal Control

$$\underset{u(\cdot)}{\text{minimize}}\ l\big(x(t_f), t_f\big) + \int_0^{t_f} c(x(t), u(t), t)dt$$

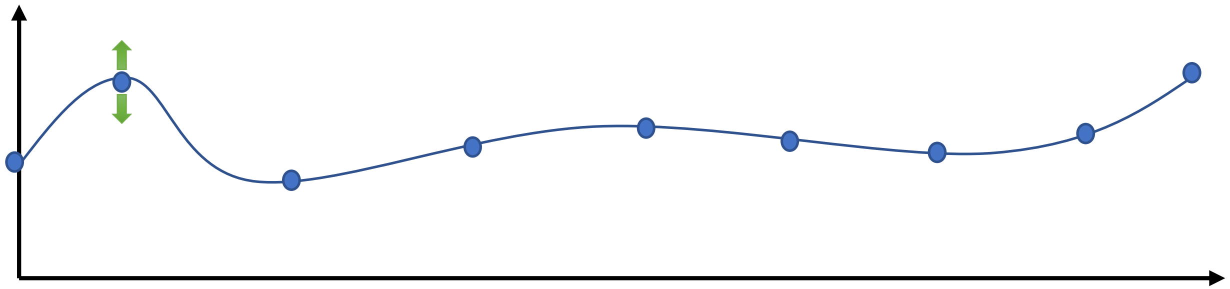
$$\text{subject to } \dot{x}(t) = f\big(x(t), u(t)\big)$$

- Observation 1: Discretize time → nonlinear optimization problem
- Fact 1: Minimizing "cost" is same as maximizing "reward"
- Fact 2: Discretize time + maximizing reward → reinforcement learning problem

Optimal Control

$$\underset{u(\cdot)}{\text{minimize}} \; l\big(x(t_f), t_f\big) + \int_0^{t_f} c(x(t), u(t), t)\,dt$$

$$\text{subject to } \dot{x}(t) = f\big(x(t), u(t)\big)$$



- Open-loop control
  - Find $u(t)$ for $t \in [0, t_f]$
  - Scalable, but errors will add up
- Closed-loop control
  - Find $u(t, x)$ for $t \in [0, t_f]$, $x \in \mathbb{R}^n$
  - Not scalable, but robust
  - “Special” techniques needed (eg. Reinforcement learning) for large n
- Receding horizon control:
  - Find $u(t)$ for $t \in [0, T]$, use $u(t)$ for $t \in [0, h]$, then find $u(t)$ for $t \in [h, T + h]$ and repeat
  - Has features of both open- and closed-loop control

Optimal Control

- For now: Deterministic systems, continuous time, continuous state
- Other variations:
  - Stochastic
  - Discrete time
  - Discrete state

Outline – Open-Loop Control

- Optimal Control Problems
- Differential flatness
- Direct Methods (Numerical Methods)
  - Shooting methods
  - Collocation
  - CasADi Matlab toolbox

Optimal Control

$$\underset{u(\cdot)}{\text{minimize}}\ \overbrace{l\left(x(t_f), t_f\right)}^{\text{Final cost}} + \overbrace{\int_0^{t_f} c(x(t), u(t), t)dt}^{\text{Running cost}}$$

Cost functional, $J\left(x(\cdot), u(\cdot)\right)$

$$\text{subject to } \dot{x}(t) = f\left(x(t), u(t)\right)$$

Dynamic model

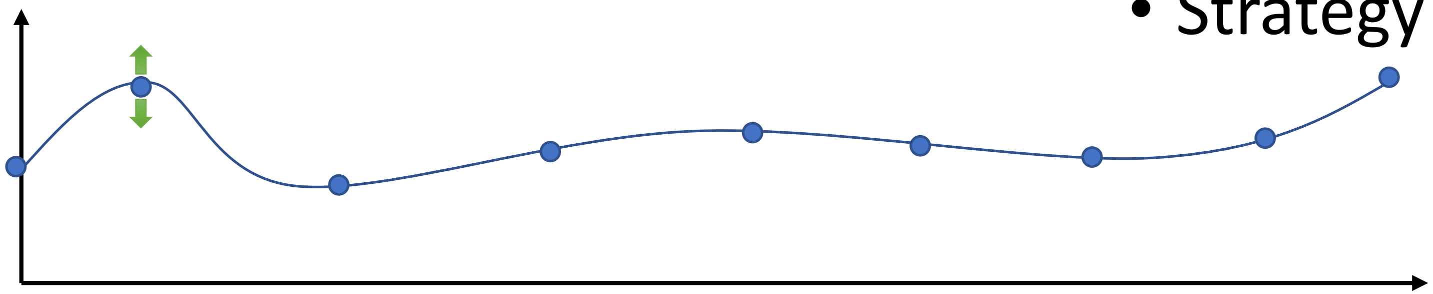
$$g\left(x(t), u(t)\right) \geq 0$$

Additional constraints

- Eg. actuation limits

$$x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, x(0) = x_0$$

- Optimal control:
  - Decision variable is a function $u(\cdot)$



- Strategy 1: Optimality conditions
- Strategy 2: Discretize first → nonlinear optimization
- Strategy 3: Use differential flatness (if lucky)

Differential Flatness

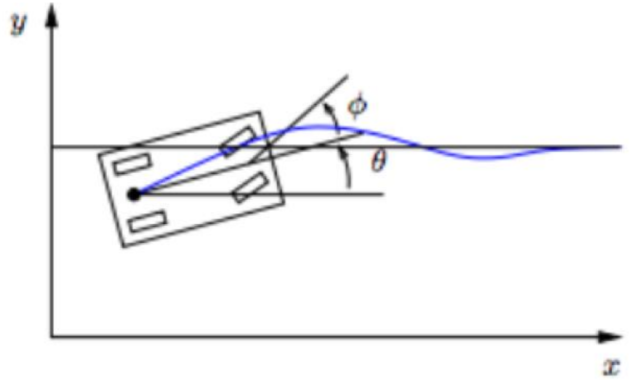
- Problem: find a $u(\cdot)$ such that

$$\dot{x}(t) = f(x, u)$$
$$x(0) = x_0$$
$$x(T) = x_f$$

  - Worry about feasibility for now, and ignore cost

- Example: vehicle steering
  - State: (x, y, θ)
  - Inputs: (v, ϕ)

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



Use Special Structure

Dynamics:

$$\dot{x} = v\cos\theta$$

$$\dot{y} = v\sin\theta$$

$$\dot{\theta} = \frac{v}{l}\tan\phi$$

- First, suppose $x(t), y(t)$ are smooth and **given**.

1. Obtain heading:

$$\frac{\dot{y}}{\dot{x}} = \frac{\sin\theta}{\cos\theta} \Rightarrow \theta = \arctan\left(\frac{\dot{y}}{\dot{x}}\right)$$

2. Obtain speed

$$\dot{x} = v\cos\theta \Rightarrow v = \frac{\dot{x}}{\cos\theta}$$

3. Obtain steering angle

$$\dot{\theta} = \frac{v}{l}\tan\phi \Rightarrow \phi = \arctan\left(\frac{l\dot{\theta}}{v}\right)$$

- All state variables and control inputs can be determined from the given trajectory!

Differential Flatness Definition

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

$$z = \alpha\left(x, u, \dots, u^{(p)}\right)$$

and we can write the solutions of the nonlinear system as functions of z and a finite number of derivatives

$$x = \beta\left(z, \dot{z}, \dots, z^{(q)}\right)$$
$$u = \gamma\left(z, \dot{z}, \dots, z^{(q)}\right)$$

Differential Flatness Definition

Generic system		**Kinematic car** $x(t), y(t)$ are smooth and given
$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, u)$	$\longrightarrow$	$\dot{x} = v\cos\theta$ $\dot{y} = v\sin\theta$ $\dot{\theta} = \frac{v}{l}\tan\phi$
$z = \alpha\left(\boldsymbol{x}, u, \dots, u^{(p)}\right)$	$\longrightarrow$	$z = (x, y)$
$\boldsymbol{x} = \beta\left(z, \dot{z}, \dots, z^{(q)}\right)$	$\longrightarrow$	$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} x \\ y \\ \arctan\left(\frac{\dot{x}}{\dot{y}}\right) \end{bmatrix}$
$u = \gamma\left(z, \dot{z}, \dots, z^{(q)}\right)$	$\longrightarrow$	$\begin{bmatrix} v \\ \phi \end{bmatrix} = \begin{bmatrix} \frac{\dot{x}}{\cos\theta} \\ \arctan\left(\frac{l\dot{\theta}}{v}\right) \end{bmatrix}$

Trajectory generation

- Problem: find a feasible solution that satisfies

$$\dot{x}(t) = f\big(x(t), u(t)\big)$$
$$x(0) = x_0$$
$$x(T) = x_f$$

- Differential flatness: $\boldsymbol{x} = \beta\big(z, \dot{z}, \dots, z^{(q)}\big)$

$$\Rightarrow x(0) = \beta\left(z(0), \dot{z}(0), \dots, z^{(q)}(0)\right) = x_0$$
$$x(T) = \beta\left(z(T), \dot{z}(T), \dots, z^{(q)}(T)\right) = x_f$$

- Let $z(t) = \sum_{i=1}^{N} b_i \psi_i(t) \Rightarrow \dot{z}(t) = \sum_{i=1}^{N} b_i \dot{\psi}_i(t)$

$$\vdots$$

$$z^{(q)}(t) = \sum_{i=1}^{N} b_i \psi_i^{(q)}(t)$$

Trajectory generation

- Differential flatness:

$$x(0) = \beta\left(z(0), \dot{z}(0), \ldots, z^{(q)}(0)\right) = x_0$$
$$x(T) = \beta\left(z(T), \dot{z}(T), \ldots, z^{(q)}(T)\right) = x_f$$

$$z(t) = \sum_{i=1}^{N} b_i \psi_i(t), \quad \dot{z}(t) = \sum_{i=1}^{N} b_i \dot{\psi}_i(t), \quad \cdots \quad z^{(q)}(t) = \sum_{i=1}^{N} b_i \phi_i^{(q)}(t)$$

$$\Rightarrow \begin{bmatrix} \psi_1(0) & \psi_2(0) & \cdots & \psi_N(0) \\ & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} z_1(0) \\ \\ \\ \\ \end{bmatrix}$$

Trajectory generation

- Differential flatness:

$$x(0) = \beta\left(z(0), \dot{z}(0), \dots, z^{(q)}(0)\right) = x_0$$
$$x(T) = \beta\left(z(T), \dot{z}(T), \dots, z^{(q)}(T)\right) = x_f$$

$$z(t) = \sum_{i=1}^{N} b_i \psi_i(t), \quad \dot{z}(t) = \sum_{i=1}^{N} b_i \dot{\psi}_i(t), \quad \cdots \quad z^{(q)}(t) = \sum_{i=1}^{N} b_i \phi_i^{(q)}(t)$$

$$\Rightarrow \begin{bmatrix} \psi_1(0) & \psi_2(0) & \cdots & \psi_N(0) \\ \dot{\psi}_1(0) & \dot{\psi}_2(0) & \cdots & \dot{\psi}_N(0) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1^{(q)}(0) & \psi_2(0)^{(q)} & \cdots & \psi_N^{(q)}(0) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} z_1(0) \\ \dot{z}_1(0) \\ \vdots \\ z_1^{(q)}(0) \end{bmatrix}$$

Trajectory generation

- Differential flatness:

$$x(0) = \beta\left(z(0), \dot{z}(0), \ldots, z^{(q)}(0)\right) = x_0$$
$$x(T) = \beta\left(z(T), \dot{z}(T), \ldots, z^{(q)}(T)\right) = x_f$$

$$z(t) = \sum_{i=1}^{N} b_i \psi_i(t), \quad \dot{z}(t) = \sum_{i=1}^{N} b_i \dot{\psi}_i(t), \quad \cdots \quad z^{(q)}(t) = \sum_{i=1}^{N} b_i \phi_i^{(q)}(t)$$

$$\Rightarrow \begin{bmatrix} \psi_1(0) & \psi_2(0) & \cdots & \psi_N(0) \\ \dot{\psi}_1(0) & \dot{\psi}_2(0) & \cdots & \dot{\psi}_N(0) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1^{(q)}(0) & \psi_2(0)^{(q)} & \cdots & \psi_N^{(q)}(0) \\ \psi_1(T) & \psi_2(T) & \cdots & \psi_N(T) \\ \dot{\psi}_1(T) & \dot{\psi}_2(T) & \cdots & \dot{\psi}_N(T) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1^{(q)}(T) & \psi_2^{(q)}(T) & \cdots & \psi_N^{(q)}(T) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} z_1(0) \\ \dot{z}_1(0) \\ \vdots \\ z_1^{(q)}(0) \\ z_1(T) \\ \dot{z}_1(T) \\ \vdots \\ z_1^{(q)}(T) \end{bmatrix}$$

What to do with $\boldsymbol{b}$?

$$\begin{bmatrix} \psi_1(0) & \psi_2(0) & \cdots & \psi_N(0) \\ \dot{\psi}_1(0) & \dot{\psi}_2(0) & \cdots & \dot{\psi}_N(0) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1^{(q)}(0) & \psi_2(0)^{(q)} & \cdots & \psi_N^{(q)}(0) \\ \psi_1(T) & \psi_2(T) & \cdots & \psi_N(T) \\ \dot{\psi}_1(T) & \dot{\psi}_2(T) & \cdots & \dot{\psi}_N(T) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1^{(q)}(T) & \psi_2^{(q)}(T) & \cdots & \psi_N^{(q)}(T) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} z_1(0) \\ \dot{z}_1(0) \\ \vdots \\ z_1^{(q)}(0) \\ z_1(T) \\ \dot{z}_1(T) \\ \vdots \\ z_1^{(q)}(T) \end{bmatrix}$$

$$z(t) = \sum_{i=1}^{N} b_i \psi_i(t), \quad \dot{z}(t) = \sum_{i=1}^{N} b_i \dot{\psi}_i(t), \quad \cdots \quad z^{(q)}(t) = \sum_{i=1}^{N} b_i \phi_i^{(q)}(t)$$

What to do with $\boldsymbol{b}$?

$$\begin{bmatrix} \psi_1(0) & \psi_2(0) & \cdots & \psi_N(0) \\ \dot{\psi}_1(0) & \dot{\psi}_2(0) & \cdots & \dot{\psi}_N(0) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1^{(q)}(0) & \psi_2(0)^{(q)} & \cdots & \psi_N^{(q)}(0) \\ \psi_1(T) & \psi_2(T) & \cdots & \psi_N(T) \\ \dot{\psi}_1(T) & \dot{\psi}_2(T) & \cdots & \dot{\psi}_N(T) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1^{(q)}(T) & \psi_2^{(q)}(T) & \cdots & \psi_N^{(q)}(T) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} z_1(0) \\ \dot{z}_1(0) \\ \vdots \\ z_1^{(q)}(0) \\ z_1(T) \\ \dot{z}_1(T) \\ \vdots \\ z_1^{(q)}(T) \end{bmatrix}$$

$$z(t) = \sum_{i=1}^{N} b_i \psi_i(t), \quad \dot{z}(t) = \sum_{i=1}^{N} b_i \dot{\psi}_i(t), \quad \cdots \quad z^{(q)}(t) = \sum_{i=1}^{N} b_i \phi_i^{(q)}(t)$$

$$\boldsymbol{x} = \boldsymbol{\beta}\left(\boldsymbol{z}, \dot{\boldsymbol{z}}, \dots, \boldsymbol{z}^{(q)}\right)$$

$$\boldsymbol{u} = \boldsymbol{\gamma}\left(\boldsymbol{z}, \dot{\boldsymbol{z}}, \dots, \boldsymbol{z}^{(q)}\right)$$

Key points

$$\underset{u(\cdot)}{\text{minimize}}\ l\left(x\left(t_f\right), t_f\right) + \int_0^{t_f} c(x(t), u(t), t)dt$$

$$\text{subject to}\ \dot{x}(t) = f\left(x(t), u(t)\right)$$

- Trajectory generation via solving algebraic equations
- Other constraints can be transformed into z space
- Cost/performance index also transformed into z space

- Quadrotors are differentially flat
  - D. Mellinger and V. Kumar. *Minimum snap trajectory generation and control for quadrotors*, ICRA 2011.

Direct methods

- Differential flatness
  - Algebraic method for special system dynamics
- **Direct shooting**
  - **Parametrize control**
  - **Numerical example with CasADi**
- Collocation
  - Parametrize both state and control

Single shooting

$$\min_{u(\cdot)} h\big(x(t_f), t_f\big) + \int_{t_0}^{t_f} g(x(t), u(t), t)dt$$

subject to $\dot{x}(t) = a\big(x(t), u(t)\big)$

$c\big(x(t), u(t)\big) \geq 0, t \in \big[t_0, t_f\big]$

where $x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, x(t_0) = x_0$

- Discretize: $t_0 < t_1 < \cdots < t_N \coloneqq t_f$

  $u(t) = q_i$ for $t \in [t_i, t_{i+1}]$

Single shooting

$$\min_{u(\cdot)} h\big(x(t_f), t_f\big) + \int_{t_0}^{t_f} g(x(t), u(t), t)dt$$

subject to $\dot{x}(t) = a\big(x(t), u(t)\big)$

$c\big(x(t), u(t)\big) \geq 0, t \in [t_0, t_f]$

where $x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, x(t_0) = x_0$

- Discretize: $t_0 < t_1 < \cdots < t_N \coloneqq t_f$

  $u(t) = q_i$ for $t \in [t_i, t_{i+1}]$

- Numerically integrate dynamics and cost:
  - Simple example:

Dynamics: $x(t_{i+1}) \approx x(t_i) + a(x(t_i), q_i)(t_{i+1} - t_i)$

Cost: $$\int_{t_0}^{t_f} g(x(t), u(t), t)dt \approx \sum_{i=0}^{N-1} g(x(t_i), q_i, t_i)(t_{i+1} - t_i)$$

Single shooting

$$\min_{u(\cdot)} h\big(x(t_f), t_f\big) + \int_{t_0}^{t_f} g(x(t), u(t), t)dt$$

subject to $\dot{x}(t) = a\big(x(t), u(t)\big)$

$c\big(x(t), u(t)\big) \geq 0, t \in [t_0, t_f]$

- Discretized problem:

$$\min_{q} h(x(t_N), t_N) + \sum_{i=0}^{N-1} g(x(t_i), q_i, t_i)(t_{i+1} - t_i)$$

subject to $\forall i \in \{0,1, \ldots, N-1\}$,

$$x(t_{i+1}) = x(t_i) + a(x(t_i), q_i)(t_{i+1} - t_i)$$

$$c(x(t_i), q_i) \geq 0$$

Introduction to CasADi

- Numerical optimization software
  - Tailored towards transcription of optimal control problems into NLPs
  - Eg. single shooting, multiple shooting, collocation
- Interfaces
  - MATLAB, Python, C++, etc.
  - We will use MATLAB
- Tools:
  - NLP solvers (eg. IPOPT, SNOPT, etc.)
  - Convex solvers (eg. Gurobi, Cplex, etc.)
  - Symbolic integrators

Introduction to CasADi

- Home page
  - https://github.com/casadi/casadi/wiki

- Installation
  - https://github.com/casadi/casadi/wiki/InstallationInstructions

- User guide
  - http://casadi.sourceforge.net/v3.4.0/users_guide/casadi-users_guide.pdf

Coding example

$$\min_{u(\cdot)} \int_0^{10} (x_1^2 + x_2^2 + u^2) dt$$

subject to

$$\dot{x}_1 = (1 - x_2^2)x_1 - x_2 + u$$
$$\dot{x}_2 = x_1$$
$$x_1 \geq -0.25$$
$$-1 \leq u \leq 1$$

- Reformulation as NLP
  - $N = 100$ with uniform time intervals
  - Forward Euler integration for dynamics
  - First-order integration

Adapted from example in casADi user manual

Coding example

$$\min_{u(\cdot)} \int_0^{10} (x_1^2 + x_2^2 + u^2)dt$$

subject to
$$\dot{x}_1 = (1 - x_2^2)x_1 - x_2 + u$$
$$\dot{x}_2 = x_1$$
$$x_1 \geq -0.25$$
$$-1 \leq u \leq 1$$

1. Installation and basic test
2. Preliminary setup
3. Integrating dynamics
4. NLP formulation
5. Solve and plot

Adapted from example in casADi user manual

https://github.com/casadi/casadi/wiki/InstallationInstructions

Installing CasADi

Option 1: Binary installation (recommended)

Install CasADi 3.4.3

For Python users: `pip install casadi` (you must have `pip --version` >= 8.1!)

Grab a binary from the table (for MATLAB, use the newest compatible version below):

	Windows	Linux	Mac
Matlab	R2014b or later, R2014a, R2013a or R2013b	R2014b or later, R2014a	R2015a or later, R2014b, R2014a
Octave	4.2.2 (32bit / 64bit)	4.2.2	4.2.2
Python	Py27 (32bit[1,2] / 64bit[2]), Py35 (32bit[2] / 64bit[2]), Py36 (32bit[2] / 64bit[2])	Py27, Py35, Py36	Py27, Py35, Py36

Coding example

$$\min_{u(\cdot)} \int_0^{10} (x_1^2 + x_2^2 + u^2)dt$$

subject to
$$\dot{x}_1 = (1 - x_2^2)x_1 - x_2 + u$$
$$\dot{x}_2 = x_1$$
$$x_1 \geq -0.25$$
$$-1 \leq u \leq 1$$

1. Installation and basic test
2. Preliminary setup
3. Integrating dynamics
4. NLP formulation
5. Solve and plot

Adapted from example in casADi user manual

Single shooting

$$\min_{q} h(x(t_N), t_N) + \sum_{i=0}^{N-1} g(x(t_i), q_i, t_i)(t_{i+1} - t_i)$$

subject to $\forall i \in \{0,1,\ldots,N-1\}$,

$$x(t_{i+1}) = x(t_i) + a(x(t_i), q_i)(t_{i+1} - t_i)$$

$$c(x(t_i), q_i) \geq 0$$

- Main disadvantage: integration error

```
>> Xk

Xk =

F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(F(
>> 
```

Adapted from example in casADi user manual

Multiple shooting

$$\min_{q} h(\boldsymbol{x}(\boldsymbol{t_N}), t_N) + \sum_{i=0}^{N-1} g(\boldsymbol{x}(\boldsymbol{t_i}), q_i, t_i)(t_{i+1} - t_i)$$

subject to $\forall i \in \{0,1,\dots,N-1\}$,

$$\boldsymbol{x}(\boldsymbol{t_{i+1}}) = \boldsymbol{x}(\boldsymbol{t_i}) + a(\boldsymbol{x}(\boldsymbol{t_i}), q_i)(t_{i+1} - t_i)$$

$$c(\boldsymbol{x}(\boldsymbol{t_i}), q_i) \geq 0$$

⬇

$$\min_{\boldsymbol{s},q} h(\boldsymbol{s_N}, t_N) + \sum_{i=0}^{N-1} g(\boldsymbol{s_i}, q_i, t_i)(t_{i+1} - t_i)$$

subject to $\forall i \in \{0,1,\dots,N-1\}$,

$$\boldsymbol{s_{i+1}} = \boldsymbol{s_i} + a(\boldsymbol{s_i}, q_i)(t_{i+1} - t_i)$$

$$c(\boldsymbol{s_i}, q_i) \geq 0$$

Shooting method disadvantages

- Numerical integration
  - Potentially slow
  - Numerical errors

Direct methods

- Differential flatness
  - Algebraic method for special system dynamics
- Direct shooting
  - Parametrize control
  - Numerical example with CasADi
- Collocation
  - Parametrize both state and control