Optimization Method & Optimal Guidance

Haichao Hong

AE8120

Contents

What is Guidance?

Review

Keywords:

- Trajectory / Maneuvering
- Motion of Aircraft (Translational Dynamics)
- Select Commands
- Current State to Desired State
- Constraints

Note: Attitude is NOT irrelevant!

Control:

- Forces and torques
- Attitude or flight condition
- Stability

Guidance

- Realize the maneuver
- Output to control
- Velocity and accelerations

Computational Guidance*:

- Relying on numerical algorithms
- Relying on onboard computation
- Model-based or data-based, no reference (necessarily) needed

*Lu, P., "Introducing Computational Guidance and Control," Journal of Guidance, Control, and Dynamics, Vol. 40, No. 2, 2017

Relationship between

(Computational) Guidance and Trajectory Optimization

(Computational) Guidance	Trajectory Optimization
Determine a trajectory	Select a trajectory
Steer the aircraft	Design the maneuver
Need to consider constraints	Can consider constraints
Optimality is not fundamental	Optimality is fundamental
Computational guidance mostly onboard	Not necessarily onboard

Trajectory Optimization

- Performance index / Objective function / Cost function
- Equations of motion (often translational dynamics)
- Initial and terminal conditions
- Path constraints

Constraints

A generic formulation:

$$
\begin{aligned}
\underset{\boldsymbol{u}(t),\boldsymbol{p}}{\text{minimize}} \quad & J = \varphi(\boldsymbol{x}(t_f)) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \\
\text{subject to} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right), \boldsymbol{p}, t\right) , \\
& \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) = 0 , \\
& \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) \leq 0
\end{aligned}
$$

Note:

- Constraints are imposed when necessary.
- t_f might or might not be given.

Before solving it

$$\begin{aligned}
\underset{\boldsymbol{u}(t),\boldsymbol{p}}{\text{minimize}} \quad & J = \varphi(\boldsymbol{x}(t_f)) + \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t), t)dt \\
\text{subject to} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right), \boldsymbol{p}, t\right) \ , \\
& \boldsymbol{h}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) = 0 \, , \\
& \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{p}, t) \leq 0
\end{aligned}$$

1. Figure out your purpose
2. Know your objective (function)
3. Constraints = Practical concerns

Discretization methods

- Single Shooting
- Multiple Shooting
- Collocation – Full Discretization

A guidance problem:

$$
\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \boldsymbol{f}\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right), \boldsymbol{p}, t\right) \ , \\
\boldsymbol{x}(t_0) &= \boldsymbol{x}_1 \ , \\
\boldsymbol{x}(t_f) &= \boldsymbol{x}_f
\end{aligned}
$$

- Initial condition fixed
- Terminal condition fixed
- No path constraints
- To determine a control sequence

Review

With a known $\boldsymbol{x}_1$, using N discrete steps, step length h

To determine a control sequence

$$\boldsymbol{U} = [\boldsymbol{u}_1^T, \boldsymbol{u}_2^T, ..., \boldsymbol{u}_{N-1}^T]^T \in \mathbb{R}^{(N-1)m}$$

Using Euler Forward: (Number of states is n)

$$\begin{aligned}
\boldsymbol{x}_2 &= \boldsymbol{F}_1(\boldsymbol{x}_1, \boldsymbol{u}_1) \\
\boldsymbol{x}_3 &= \boldsymbol{F}_2(\boldsymbol{F}_1(\boldsymbol{x}_1, \boldsymbol{u}_1), \boldsymbol{u}_2) \\
&\vdots \\
\boldsymbol{x}_N &= \boldsymbol{F}_{N-1}(\boldsymbol{F}_{N-2}(...(\boldsymbol{F}_1(\boldsymbol{x}_1, \boldsymbol{u}_1), ...), \boldsymbol{u}_{N-2}), \boldsymbol{u}_{N-1})
\end{aligned}$$

Review

$$\boldsymbol{F}_{N-1}\left(\boldsymbol{F}_{N-2}\left(\ldots\left(\boldsymbol{F}_1\left(\boldsymbol{x}_1,\boldsymbol{u}_1\right),\ldots\right),\boldsymbol{u}_{N-2}\right),\boldsymbol{u}_{N-1}\right)-\boldsymbol{x}_f=0$$

Unknown: $\boldsymbol{U}=[\boldsymbol{u}_1^T,\boldsymbol{u}_2^T,\ldots,\boldsymbol{u}_{N-1}^T]^T\in\mathbb{R}^{(N-1)m}$

Number of equations: n (Number of states)

It is very likely that $(N-1)m\gg n$

$$\boldsymbol{G}\left(\boldsymbol{U}\right)=0$$

It is an underdetermined system! But Nonlinear

Review

To find the root of a nonlinear underdetermined system

$$\boldsymbol{G}(\boldsymbol{U}) = 0$$

Newton's method is available for this task:

$$\boldsymbol{U} = \boldsymbol{U}^p + d\boldsymbol{U} \qquad\qquad \boldsymbol{r} - \boldsymbol{G}'(\boldsymbol{U}^p)d\boldsymbol{U} = 0$$

where $\boldsymbol{G}'$ is the Jacobian matrix, $d\boldsymbol{U}$ is the Newton step,

and $\boldsymbol{r}$ is the residual from the previous iteration.

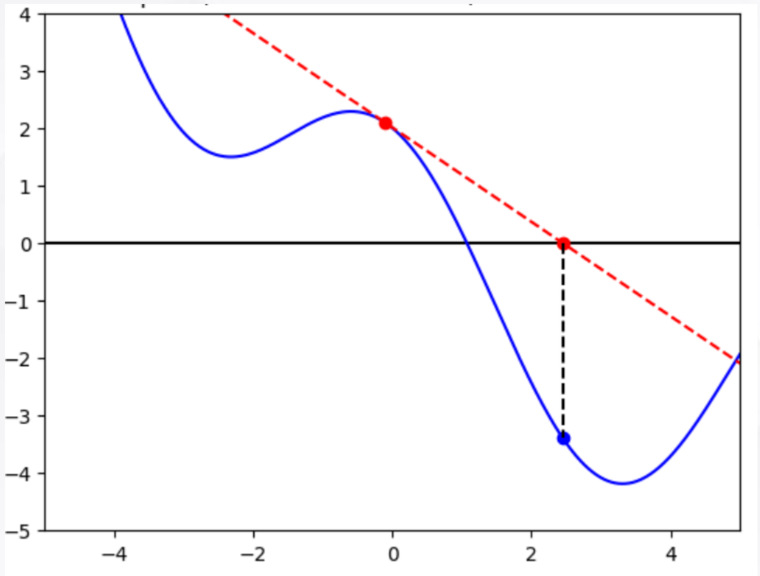
Review

One more step!

Line Search

Newton iterations $\boldsymbol{U} = \boldsymbol{U}^p + s \cdot d\boldsymbol{U}$

Improve convergence



Algorithm 1 Line-Search Strategy

1: Get $\boldsymbol{U}^p$, $d\boldsymbol{U}$, $s = 1$, $\kappa \in (0, 1]$
2: **while** $\left|\boldsymbol{G}_n\left(\boldsymbol{U}^p + s \cdot d\boldsymbol{U}\right)\right|_\infty > \left|\boldsymbol{G}_n\left(\boldsymbol{U}^p\right)\right|_\infty$ **do**
3: $\quad s \leftarrow \kappa s$
4: **end while**
5: $\boldsymbol{U} \leftarrow \boldsymbol{U}^p + s \cdot d\boldsymbol{U}$
6: **return** $\boldsymbol{U}$

(Many) Optimization Toolboxes

- Rely on external solvers
- Automatic transcription takes time
- Easy to work with
- GPOPS, FALCON.m, CVX

Solvers/Optimizers

- Have rather fixed interface
- Manual transcription takes effort
- Hard to develop from scratch
- IPOPT, SNOPT, MOSEK

Many of them have academic licenses

Sequential Convex Optimization Methods:

- Trending in computational guidance
- Convert the original nonlinear trajectory optimization problem into convex optimization subproblems
- Good computational efficiency

The 1st step: Deal with the nonlinear dynamics

Review

$$(\dot{\boldsymbol{x}})^{(j+1)} = \boldsymbol{F}_{\boldsymbol{x}}^{(j)}\left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) + \boldsymbol{F}_{\boldsymbol{u}}^{(j)}\left(\boldsymbol{u}^{(j+1)} - \boldsymbol{u}^{(j)}\right) + \boldsymbol{f}^{(j)}$$

$$d\boldsymbol{x}^{(j+1)} := \boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\,,$$
$$d\boldsymbol{u}^{(j+1)} := \boldsymbol{u}^{(j+1)} - \boldsymbol{u}^{(j)}\,.$$

$$\boldsymbol{x}_{k+1}^{(j+1)} = \boldsymbol{x}_k^{(j+1)} + h\,(\dot{\boldsymbol{x}})_k^{(j+1)}$$

$$\begin{aligned} d\boldsymbol{x}_{k+1}^{(j+1)} + \boldsymbol{x}_{k+1}^{(j)} &= h\left[(\boldsymbol{F}_{\boldsymbol{x}})_k^{(j)} d\boldsymbol{x}_k^{(j+1)} + (\boldsymbol{F}_{\boldsymbol{u}})_k^{(j)} d\boldsymbol{u}_k^{(j+1)} + \boldsymbol{f}_k^{(j)}\right] + \left[\boldsymbol{x}_k^{(j)} + d\boldsymbol{x}_k^{(j+1)}\right] \\ &= \left[h(\boldsymbol{F}_{\boldsymbol{x}})_k^{(j)} + \boldsymbol{I}_n\right] d\boldsymbol{x}_k^{(j+1)} + h(\boldsymbol{F}_{\boldsymbol{u}})_k^{(j)} d\boldsymbol{u}_k^{(j+1)} + h\boldsymbol{f}_k^{(j)} + \boldsymbol{x}_k^{(j)} \end{aligned}$$

States dynamics with normalized time

The derivative of the state vector with respect to the normalized time τ can be defined using the chain rule as:

$$\boldsymbol{x}' := \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\tau} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau} = t_f \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}$$

It can be noticed that $\boldsymbol{x}'$ is a function of $\boldsymbol{x}$, $\boldsymbol{u}$, and t_f as

$$\boldsymbol{x}'\left(\boldsymbol{x}\left(\tau\right), \boldsymbol{u}\left(\tau\right), t_f\right) = t_f \boldsymbol{f}\left(\boldsymbol{x}\left(\tau\right), \boldsymbol{u}\left(\tau\right)\right)$$

Review

Linear inequality constraints on states and optimization variables can be expressed as

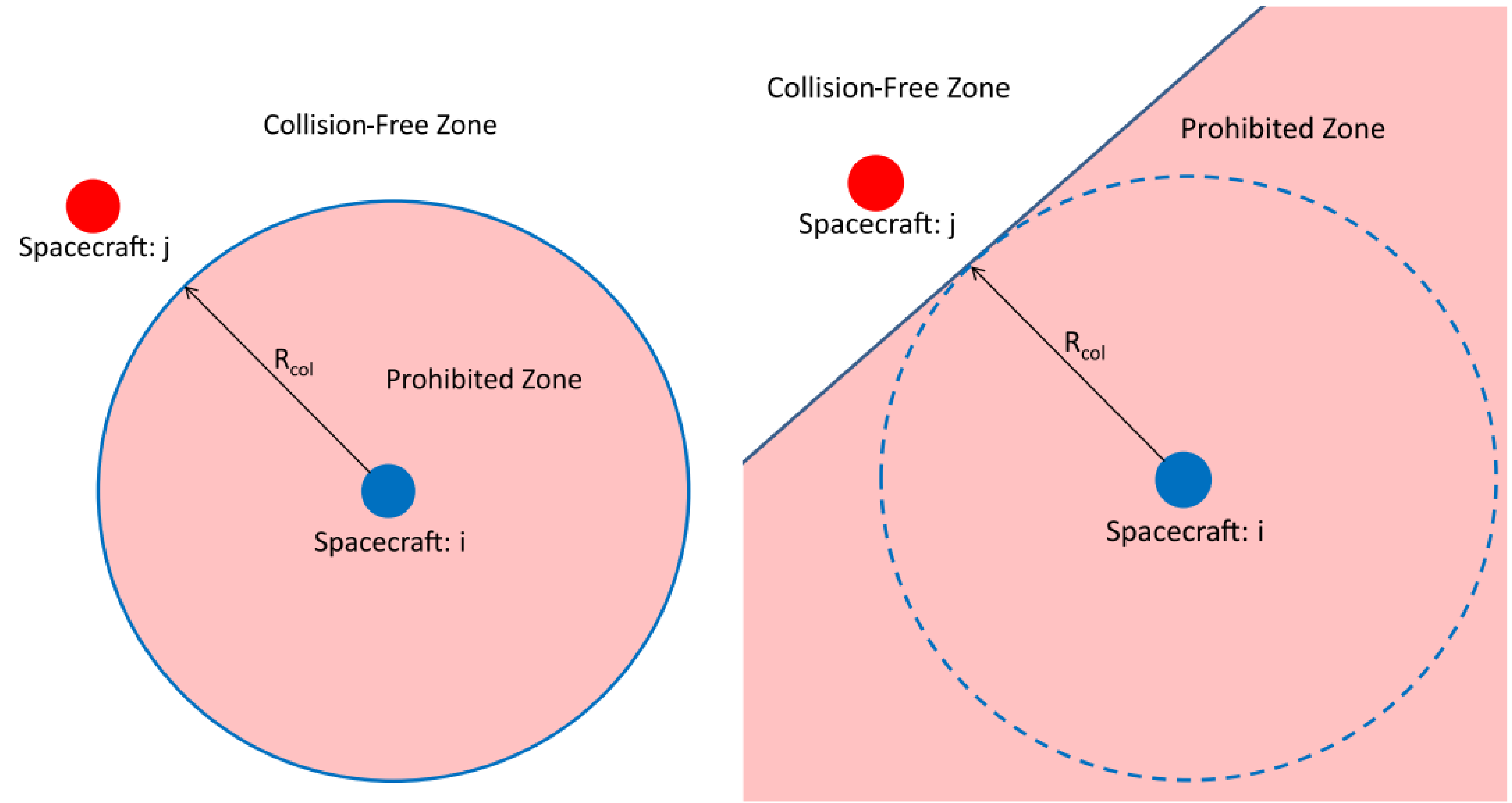
$$\begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{L}_2 \\ \boldsymbol{G}_1 & \boldsymbol{G}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{u}_a \end{bmatrix} \le \boldsymbol{l}$$

$$\begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{L}_2 \\ \boldsymbol{G}_1 & \boldsymbol{G}_2 \end{bmatrix} \begin{bmatrix} d\boldsymbol{x}_a \\ d\boldsymbol{u}_a \end{bmatrix} \le \boldsymbol{l} - \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{L}_2 \\ \boldsymbol{G}_1 & \boldsymbol{G}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_a^p \\ \boldsymbol{u}_a^p \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{L}_2 \\ \boldsymbol{G}_1 & \boldsymbol{G}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{I}_{(N-1)m+1} \end{bmatrix} d\boldsymbol{u}_a \le \boldsymbol{l} - \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{L}_2 \\ \boldsymbol{G}_1 & \boldsymbol{G}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_a^p \\ \boldsymbol{u}_a^p \end{bmatrix}$$

$$\boldsymbol{G} d\boldsymbol{u}_a \le \boldsymbol{h}$$

Inexact



a) Nonconvex prohibited zone

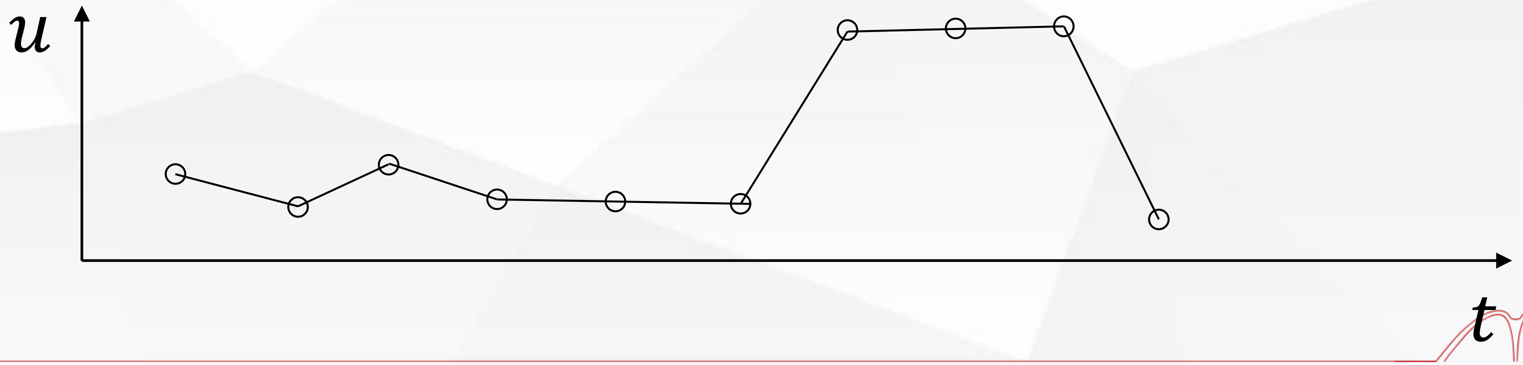
b) Convex approximation of prohibited zone

Review

A general class of constrained trajectory optimization formulations:

$$
\begin{aligned}
\underset{\boldsymbol{u}(t),\boldsymbol{p}}{\text{minimize}} \quad & J\left(\boldsymbol{x}\left(t\right),\boldsymbol{u}\left(t\right),\boldsymbol{p}\right) \\
\text{subject to} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}\left(t\right),\boldsymbol{u}\left(t\right),\boldsymbol{p}\right), \\
& \boldsymbol{g}_{lb}\left(t\right) \leq \boldsymbol{g}\left(\boldsymbol{x}\left(t\right),\boldsymbol{u}\left(t\right),\boldsymbol{p}\right) \leq \boldsymbol{g}_{ub}\left(t\right)
\end{aligned}
\tag{1}
$$

Problem (1) does not always correspond to a smooth control history



The i-th control component is expressed by

$$u^{(i)}(t) = a_0^{(i)} + \sum_{n=1}^{N} \left(a_n^{(i)} \cos\left(\frac{n\pi}{T_f} t\right) + b_n^{(i)} \sin\left(\frac{n\pi}{T_f} t\right)\right)$$

$$= \boldsymbol{s}_N^{(i)}(t)\, \boldsymbol{c}^{(i)}, \quad i = 1, 2, \ldots, m\,,$$

where

$$\boldsymbol{s}_N^{(i)}(t) = \left[1, \cos\left(\frac{\pi}{T_f} t\right), \ldots, \cos\left(\frac{N\pi}{T_f} t\right), \right.$$

$$\left. \sin\left(\frac{\pi}{T_f} t\right), \ldots, \sin\left(\frac{N\pi}{T_f} t\right)\right]$$

$$\boldsymbol{c}^{(i)} = \left[a_0^{(i)}, \ldots, a_N^{(i)}, b_1^{(i)}, \ldots, b_N^{(i)}\right]^{\mathrm{T}}.$$

Constraints on derivatives are represented by

$$\boldsymbol{l}_{lb}\left(t\right) \leq \boldsymbol{l}\left(\boldsymbol{c}, t\right) \leq \boldsymbol{l}_{ub}\left(t\right) \ .$$

The original problem is re-written as

$$\begin{aligned}
\underset{\boldsymbol{c},\boldsymbol{p}}{\text{minimize}} \quad & J\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right), \boldsymbol{p}\right) \\
\text{subject to} \quad & \dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right), \boldsymbol{p}\right) \ , \\
& \boldsymbol{u}\left(t\right) = \boldsymbol{S}_N\left(t\right)\boldsymbol{c} \, , \\
& \boldsymbol{g}_{lb}\left(t\right) \leq \boldsymbol{g}\left(\boldsymbol{x}\left(t\right), \boldsymbol{u}\left(t\right), \boldsymbol{p}\right) \leq \boldsymbol{g}_{ub}\left(t\right) \ , \\
& \boldsymbol{l}_{lb}\left(t\right) \leq \boldsymbol{l}\left(\boldsymbol{c}, t\right) \leq \boldsymbol{l}_{ub}\left(t\right) \ .
\end{aligned}$$

Review

The trigonometric polynomial can be defined with respect to other independent variables.

Here, the climb distance r

$$u_i(r) = (a_0)_i + \sum_{n=1}^{N}\left[(a_n)_i \cos\left(\frac{2n\pi}{\kappa x_f} r\right) + (b_n)_i \sin\left(\frac{2n\pi}{\kappa x_f} r\right)\right]$$
$$= \boldsymbol{s}_{Ni}(r)\,\boldsymbol{c}_i, \quad i = 1, 2, \ldots, p,$$

Review

$$u^{(i)}(\tau) = \boldsymbol{s}_N^{(i)}(\tau)\,\boldsymbol{c}^{(i)}, \quad \tau \in [0,1]$$

$$\overset{(n)}{\boldsymbol{u}}(0) = \overset{(n)}{\boldsymbol{u}}(1), \qquad n \in \mathbb{N}^0$$

The period of $\boldsymbol{s}_N^{(i)}(\tau)$ is 1.

Review

$$u^{(i)}(\tau) = (a_0)^{(i)} + \sum_{n=1}^{N}\left[(a_n)^{(i)}\cos(2\pi n\tau) + (b_n)^{(i)}\sin(2\pi n\tau)\right]$$

$$= \boldsymbol{s}_N{}^{(i)}(\tau)\,\boldsymbol{c}^{(i)}, \quad i = 1, 2, \ldots, q$$

$$\dot{\boldsymbol{u}}(t) = \frac{\mathrm{d}\boldsymbol{u}(\tau)}{\mathrm{d}\tau}\frac{\mathrm{d}\tau}{\mathrm{d}t} = \left[\frac{\mathrm{d}\boldsymbol{S}_N(\tau)}{\mathrm{d}\tau}\right]\frac{1}{t_f}\boldsymbol{c}$$

Also, $\boldsymbol{u}$ being a linear function of $\boldsymbol{c}$ leads that the incremental change of $\boldsymbol{u}$ and $\dot{\boldsymbol{u}}$ are linear to the increments of $\boldsymbol{c}$.

Impact-time Control Problem

$$r_i = V \int_0^{t_f} \cos\varepsilon(t)\,\mathrm{d}t$$

To eliminate the nonlinearities, we parameterize

$$\cos\varepsilon(t) = \boldsymbol{s}_N(t)\,\boldsymbol{c}$$

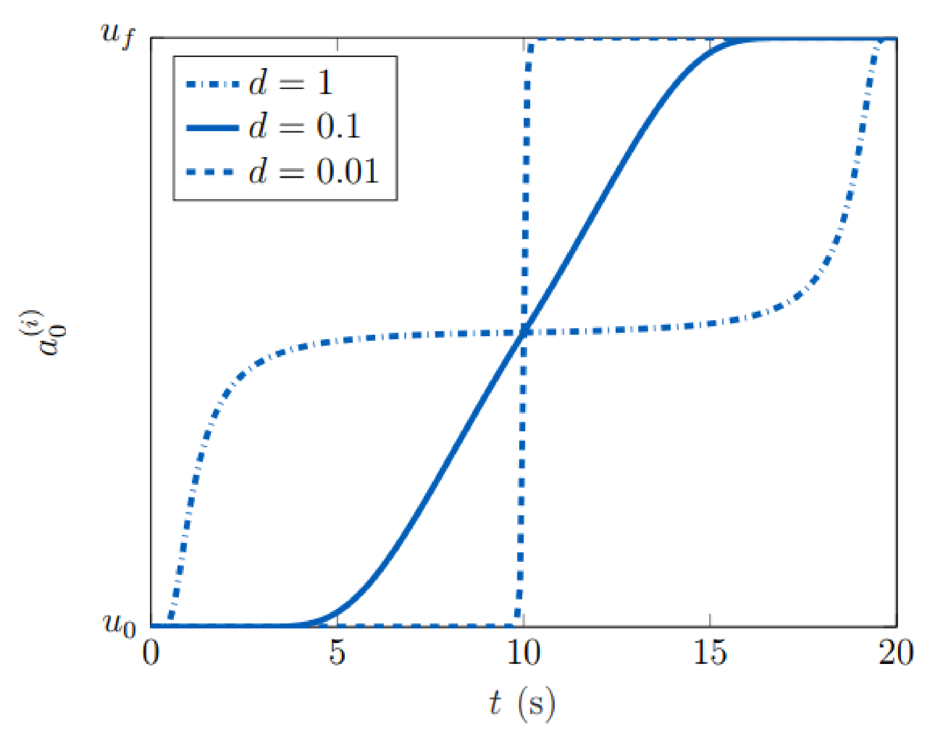
Its definite integral is still a linear function of the coefficient vector:

$$\int_0^{t_f} \cos\varepsilon(t)\,\mathrm{d}t = \int_0^{t_f} \boldsymbol{s}_N(t)\,\boldsymbol{c}\,\mathrm{d}t = \left(\int_0^{t_f} \boldsymbol{s}_N(t)\,\mathrm{d}t\right)\boldsymbol{c} = \tilde{\boldsymbol{s}}_N\boldsymbol{c}$$

$$r_i = V \int_0^{t_f} \cos\varepsilon(t)\,\mathrm{d}t \quad \Rightarrow \quad \tilde{\boldsymbol{s}}_N\boldsymbol{c} = r_i/V$$

Therefore, a_0 also needs to be parameterized.

$$a_0^{(i)}(t) = \begin{cases} u_0^{(i)} & t = 0 \\ \frac{(u_f^{(i)} - u_0^{(i)})/2}{\exp(-1/(d_1 t_f/2)^2)} \exp\left(\frac{-1}{(d_1 t)^2}\right) + u_0^{(i)} & t \in (0, t_f/2] \\ -\frac{(u_f^{(i)} - u_0^{(i)})/2}{\exp(-1/(d_1 t_f/2)^2)} \exp\left(\frac{-1}{(d_1(t - t_f))^2}\right) + u_f^{(i)} & t \in (t_f/2, t_f) \\ u_f^{(i)} & t = t_f \end{cases}$$



The hierarchical parameterization is summarized as

$$u^{(i)}(t) = a_0^{(i)}(t) + \hat{\boldsymbol{s}}_N^{(i)}(t)\, \boldsymbol{H}_v^{(i)}(t)\, \tilde{\boldsymbol{c}}^{(i)}$$

By construction, the following conditions are satisfied:

$$u^{(i)}(0) = u_0^{(i)}$$

$$u^{(i)}(t_f) = u_f^{(i)}$$

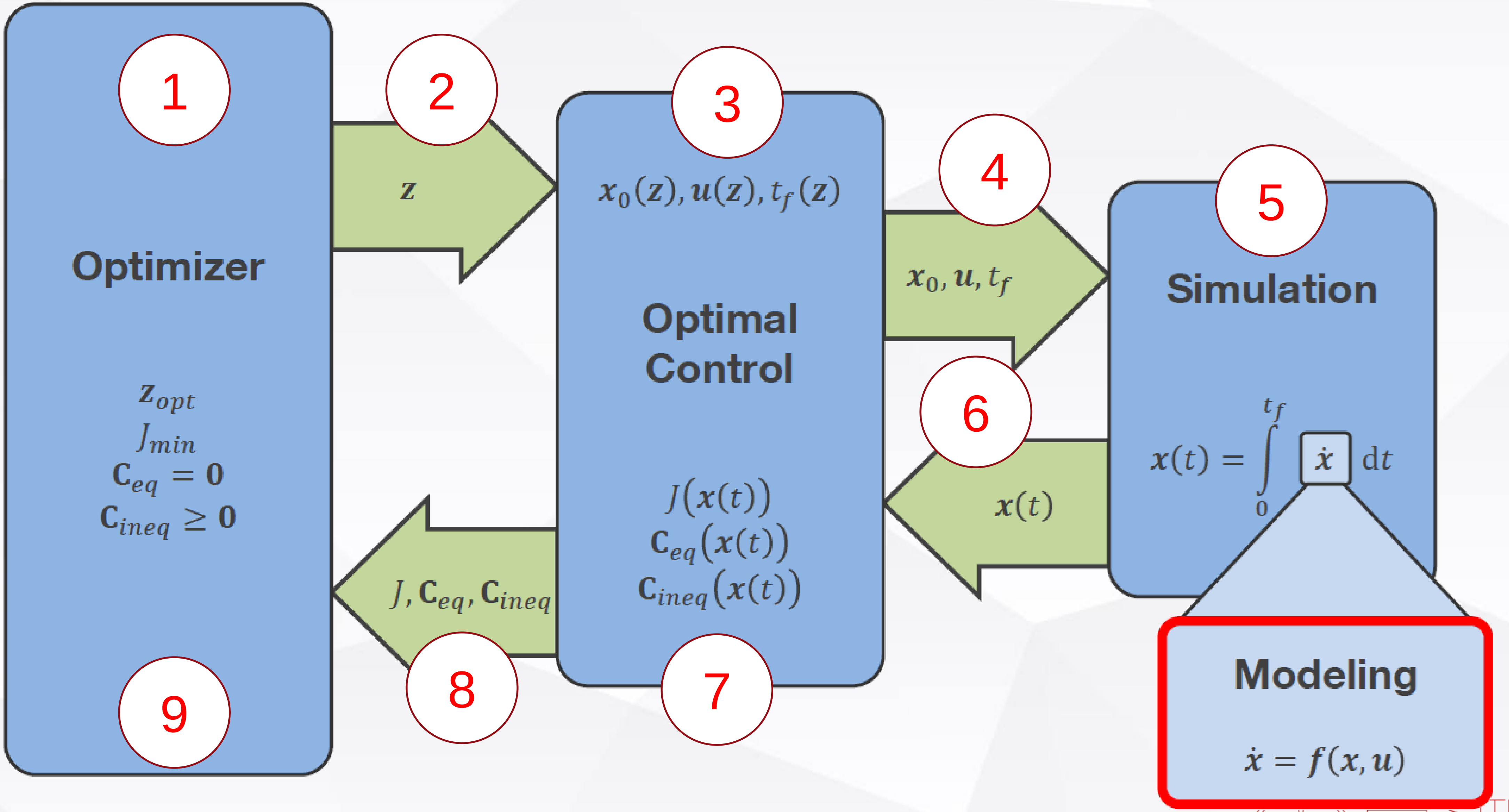
$$\overset{(r)}{u}{}^{(i)}(0) = \overset{(r)}{u}{}^{(i)}(t_f) = 0, \quad r \in \mathbb{N}^+ .$$

Trigonometric-polynomial parameterization

- Infinite differentiability
- Linear with respect to the coefficients
- Given as function of time, or other independent variables
- Derivative / Integral

FALCON.m is the *FSD optimAL CONtrol tool for MATLAB* that has been developed at the Institute of *Flight System Dynamics* of *Technische Universität München*.

Review



1
2
3
4
5
6
7
8
9
Optimizer
z
$x_0(z), u(z), t_f(z)$
Optimal Control
x_0, u, t_f
Simulation
z_{opt}
J_{min}
$C_{eq} = 0$
$C_{ineq} \geq 0$
$x(t) = \int_0^{t_f} \dot{x} \, dt$
$x(t)$
$J(x(t))$
$C_{eq}(x(t))$
$C_{ineq}(x(t))$
J, C_{eq}, C_{ineq}
Modeling
$\dot{x} = f(x, u)$

Numerical Benchmark – 2D Time Optimal Trajectory

Variable	LB	UB	Scaling	Final
x	0	100	0.01	100
y	0	100	0.01	100
V	0	5	1	5
chi	-2*pi	2*pi	1	0
Vdot	-0.1	0.1	1	n/a
chidot	-pi/8	pi/8	1	n/a

Tasks:

- **Complete x_vec**
- **Complete u_vec**
- **Complete source_car**
- **Add final boundary constraints**

The dynamic model is given as

$$\dot{x}(t) = V(t)\cos\chi(t),$$
$$\dot{y}(t) = V(t)\sin\chi(t),$$
$$\dot{V}(t) = \dot{V}_{cmd}(t),$$
$$\dot{\chi}(t) = \dot{\chi}_{cmd}(t),$$

and the state and control vectors are defined as

$$\boldsymbol{x} = [x, y, V, \chi]^{\mathrm{T}}$$

$$\boldsymbol{u} = \left[\dot{V}_{cmd}, \dot{\chi}_{cmd}\right]^{\mathrm{T}}.$$

Jacobian Matrix

$$\boldsymbol{J} = \begin{bmatrix} \dfrac{\partial \boldsymbol{f}}{\partial x_1} & \cdots & \dfrac{\partial \boldsymbol{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^{\mathrm{T}} f_1 \\ \vdots \\ \nabla^{\mathrm{T}} f_m \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}$$

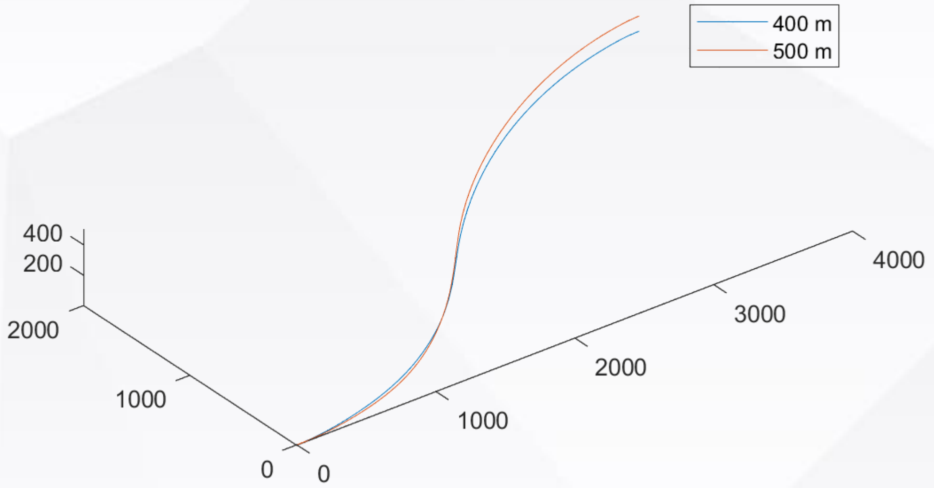
Note: MATLAB symbolic toolbox can help

Sequential Convex Optimization: Mission A

To climb 500 m instead of 400 m

Task part I:

- **Open main_fixed.m**
- **Read the code**
- **Complete the forward Euler integration**
- **Complete the lower-triangular matrix**



Task part II:

- **Open scp_solve_fixed.m**
- **Convert the box constraints**
- **Run the optimization**
- **Plot the velocity profile**

$$\boldsymbol{u}_{a\min} \le \boldsymbol{u}_a \le \boldsymbol{u}_{a\max}$$

$$\boldsymbol{u}_{a\min} - \boldsymbol{u}_a^p \le d\boldsymbol{u}_a \le \boldsymbol{u}_{a\max} - \boldsymbol{u}_a^p$$

Final Exam

17th week, January 6th, 14:00 – 15:40

- Questions given on Canvas
- Monitoring via Tencent Meeting
- Front and rear cameras
- Prepare (A4) white papers for writing answers