Optimization Method & Optimal Guidance

Haichao Hong

AE8120

Homework

Install MOSEK in MATLAB and solve ANY convex QP problem

- Send to haichao.hong@sjtu.edu.cn
- Include “作业2” and YOUR NAME in the subject line

Contents

Sequential Convex Optimization Methods:

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

Sequential Convex Optimization Methods:

- Referring to a class of methods
- Philosophies are similar
- My approaches are introduced in the course

Trajectory Optimization

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

Constraints

The 1st step: Deal with the nonlinear dynamics

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

To determine a control sequence

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

Using Euler Forward: (Number of states is n)

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + h\boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ &= \boldsymbol{F}_k(\boldsymbol{x}_k, \boldsymbol{u}_k)\end{aligned}$$

Recall that we have an update process $\boldsymbol{u} = \boldsymbol{u}^p + d\boldsymbol{u}$

Recall that we have an update process $\boldsymbol{u} = \boldsymbol{u}^p + d\boldsymbol{u}$

Control correction

Similarly, we can have $\boldsymbol{x} = \boldsymbol{x}^p + d\boldsymbol{x}$

State increment

Considering the first-order approximation of the Taylor expansion

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= F_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \\
&= F_k\left(\boldsymbol{x}_k^p + d\boldsymbol{x}_k, \boldsymbol{u}_k^p + d\boldsymbol{u}_k\right) \\
&\approx F_k\left(\boldsymbol{x}_k^p, \boldsymbol{u}_k^p\right) + \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] d\boldsymbol{x}_k + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k \\
&= \boldsymbol{x}_{k+1}^p + \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] d\boldsymbol{x}_k + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k .
\end{aligned}$$

Sequential Convex Optimization Methods

$$
\begin{aligned}
\boldsymbol{x}_{k+1} &= F_k\left(\boldsymbol{x}_k,\, \boldsymbol{u}_k\right) \\
&= F_k\left(\boldsymbol{x}_k^p + d\boldsymbol{x}_k,\, \boldsymbol{u}_k^p + d\boldsymbol{u}_k\right) \\
&\approx F_k\left(\boldsymbol{x}_k^p,\, \boldsymbol{u}_k^p\right) + \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] d\boldsymbol{x}_k + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k \\
&= \boldsymbol{x}_{k+1}^p + \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] d\boldsymbol{x}_k + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k\,.
\end{aligned}
$$

The increment at k+1 can be written as

$$
d\boldsymbol{x}_{k+1} = \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] d\boldsymbol{x}_k + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k\,.
$$

Expanding $d\boldsymbol{x}_{k+1}$ for $j = k,\ k-1, \ldots, 1$ gives

$$
\begin{aligned}
d\boldsymbol{x}_{k+1} &= \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] d\boldsymbol{x}_k + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k \\
&= \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] \left[\frac{\partial F_{k-1}}{\partial \boldsymbol{x}_{k-1}}\right] d\boldsymbol{x}_{k-1} \\
&\quad + \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] \left[\frac{\partial F_{k-1}}{\partial \boldsymbol{u}_{k-1}}\right] d\boldsymbol{u}_{k-1} + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k \\
&\ \ \vdots
\end{aligned}
$$

The initial condition is assumed to be specified, so $d\boldsymbol{x}_1 = 0$

$$
\begin{aligned}
d\boldsymbol{x}_{k+1} &= \boldsymbol{B}_1^k d\boldsymbol{u}_1 + \boldsymbol{B}_2^k d\boldsymbol{u}_2 + \ldots + \boldsymbol{B}_k^k d\boldsymbol{u}_k \\
&= \sum_{j=1}^{k} \boldsymbol{B}_j^k d\boldsymbol{u}_j \ .
\end{aligned}
$$

Here, the superscript k denotes that the state increment at $k+1$ is constructed by k number of steps of input corrections.

Sequential Convex Optimization Methods

$$\boldsymbol{B}_j^k = \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right]\left[\frac{\partial F_{k-1}}{\partial \boldsymbol{x}_{k-1}}\right]\cdots\left[\frac{\partial F_{j+1}}{\partial \boldsymbol{x}_{j+1}}\right]\left[\frac{\partial F_j}{\partial \boldsymbol{u}_j}\right] \quad \text{for } j = 1, 2, \ldots, k-2$$

$$\boldsymbol{B}_{k-1}^k = \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right]\left[\frac{\partial F_{k-1}}{\partial \boldsymbol{u}_{k-1}}\right]$$

$$\boldsymbol{B}_k^k = \frac{\partial F_k}{\partial \boldsymbol{u}_k}$$

The computation of the sensitivity matrix B can be significantly simplified

$$\left(\boldsymbol{B}_k^k\right)^0 = \boldsymbol{I}_n$$

$$\left(\boldsymbol{B}_j^k\right)^0 = \left(\boldsymbol{B}_{j+1}^k\right)^0\left[\frac{\partial F_{j+1}}{\partial \boldsymbol{x}_{j+1}}\right], j = k-1, k-2, \ldots, 1$$

$$\boldsymbol{B}_j^k = \left(\boldsymbol{B}_j^k\right)^0\left[\frac{\partial F_j}{\partial \boldsymbol{u}_j}\right], \qquad j = k, k-1, \ldots, 1\,.$$

$$d\boldsymbol{x}_{k+1} = \sum_{j=1}^{k} \boldsymbol{B}_j^k d\boldsymbol{u}_j$$

Hence, we have

$$\begin{bmatrix} d\boldsymbol{x}_2 \\ d\boldsymbol{x}_3 \\ \vdots \\ d\boldsymbol{x}_N \end{bmatrix} = \boxed{\phantom{\begin{bmatrix} \quad \end{bmatrix}}} \begin{bmatrix} d\boldsymbol{u}_1 \\ d\boldsymbol{u}_2 \\ \vdots \\ d\boldsymbol{u}_{N-1} \end{bmatrix}$$

This means that we can adjust the state variables by correcting the controls.

The dynamics are approximated by a linear equation

Sequential Convex Optimization Methods

$$\begin{bmatrix} d\boldsymbol{x}_2 \\ d\boldsymbol{x}_3 \\ \vdots \\ d\boldsymbol{x}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}_1^1 & 0 & \dots & 0 \\ \boldsymbol{B}_1^2 & \boldsymbol{B}_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{B}_1^{N-1} & \boldsymbol{B}_2^{N-1} & \dots & \boldsymbol{B}_{N-1}^{N-1} \end{bmatrix} \begin{bmatrix} d\boldsymbol{u}_1 \\ d\boldsymbol{u}_2 \\ \vdots \\ d\boldsymbol{u}_{N-1} \end{bmatrix}$$

1. We can adjusting the states by correcting the controls.
2. This relation approximates the dynamics.
3. It is LINEAR

How would it work?

Sequential Convex Optimization Methods

$$[d\boldsymbol{x}_N] = \begin{bmatrix} \boldsymbol{B}_1^{N-1} & \boldsymbol{B}_2^{N-1} & \cdots & \boldsymbol{B}_{N-1}^{N-1} \end{bmatrix} \begin{bmatrix} d\boldsymbol{u}_1 \\ d\boldsymbol{u}_2 \\ \vdots \\ d\boldsymbol{u}_{N-1} \end{bmatrix}$$

$$\Updownarrow$$

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

$$\boldsymbol{x}_N^p + d\boldsymbol{x}_N \rightarrow \boldsymbol{x}_f \quad \Rightarrow \quad d\boldsymbol{x}_N = \boldsymbol{x}_f - \boldsymbol{x}_N^p$$

Sequential Convex Optimization Methods

$$
\begin{aligned}
\underset{d\boldsymbol{u}_a}{\text{minimize}} \quad & J = \frac{1}{2}\left(d\boldsymbol{u}_a + \boldsymbol{u}_a^p\right)^T \boldsymbol{R}\left(d\boldsymbol{u}_a + \boldsymbol{u}_a^p\right) \\
\text{subject to} \quad & \begin{bmatrix} \boldsymbol{B}_1^{N-1} & \boldsymbol{B}_2^{N-1} & \ldots & \boldsymbol{B}_{N-1}^{N-1} \end{bmatrix} d\boldsymbol{u}_a = \boldsymbol{x}_f - \boldsymbol{x}_N^p \\
& \boldsymbol{u}_{a\min} - \boldsymbol{u}_a^p \le d\boldsymbol{u}_a \le \boldsymbol{u}_{a\max} - \boldsymbol{u}_a^p
\end{aligned}
$$

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

Sequential Convex Optimization Methods

$$\begin{bmatrix} d\boldsymbol{x}_2 \\ d\boldsymbol{x}_3 \\ \vdots \\ d\boldsymbol{x}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}_1^1 & 0 & \cdots & 0 \\ \boldsymbol{B}_1^2 & \boldsymbol{B}_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{B}_1^{N-1} & \boldsymbol{B}_2^{N-1} & \cdots & \boldsymbol{B}_{N-1}^{N-1} \end{bmatrix} \begin{bmatrix} d\boldsymbol{u}_1 \\ d\boldsymbol{u}_2 \\ \vdots \\ d\boldsymbol{u}_{N-1} \end{bmatrix}$$

$$d\boldsymbol{x}_a = \boldsymbol{B} d\boldsymbol{u}_a$$

$$\boldsymbol{g}(\boldsymbol{x}_a, \boldsymbol{u}_a) \leq 0 \Rightarrow \boldsymbol{g}(\boldsymbol{x}_a^p + d\boldsymbol{x}_a, \boldsymbol{u}_a^p + d\boldsymbol{u}_a) \leq 0$$

$$\Rightarrow \boldsymbol{g}(\boldsymbol{x}_a^p, \boldsymbol{u}_a^p, d\boldsymbol{u}_a) \leq 0$$

$$\Rightarrow \boldsymbol{g}(d\boldsymbol{u}_a) \leq 0$$

Sequential Convex Optimization Methods

$$\begin{aligned} \underset{\boldsymbol{u}_a}{\text{minimize}} \quad & J\left(\boldsymbol{x}_a, \boldsymbol{u}_a\right) \\ \text{subject to} \quad & \boldsymbol{x}_{k+1} = \boldsymbol{F}_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \\ & \boldsymbol{g}\left(\boldsymbol{x}_a, \boldsymbol{u}_a\right) \leq 0 \end{aligned}$$

$$\begin{aligned} \underset{d\boldsymbol{u}_a}{\text{minimize}} \quad & J\left(d\boldsymbol{u}_a\right) \\ \text{subject to} \quad & d\boldsymbol{x}_a = \boldsymbol{B} d\boldsymbol{u}_a \\ & \boldsymbol{g}\left(d\boldsymbol{u}_a\right) \leq 0 \end{aligned}$$

Note: Cost function and constraints must be convex

Sequential Convex Optimization Methods

Solve

$$\begin{aligned} \underset{d\boldsymbol{u}_a}{\text{minimize}} \quad & J\left(d\boldsymbol{u}_a\right) \\ \text{subject to} \quad & d\boldsymbol{x}_a = \boldsymbol{B} d\boldsymbol{u}_a \\ & \boldsymbol{g}\left(d\boldsymbol{u}_a\right) \leq 0 \end{aligned}$$

iteratively

Until stopping criteria are met.

In this case $\boldsymbol{x}_N^p + d\boldsymbol{x}_N \to \boldsymbol{x}_f$

Sequential Convex Optimization Methods

$$\begin{aligned} \underset{d\boldsymbol{u}_a}{\text{minimize}} \quad & J\left(d\boldsymbol{u}_a\right) \\ \text{subject to} \quad & d\boldsymbol{x}_a = \boldsymbol{B}_t d\boldsymbol{u}_a \\ & \boldsymbol{g}\left(d\boldsymbol{u}_a\right) \leq 0 \end{aligned}$$

?

→ Fixed final time

→ Free final time

$$\begin{aligned}\boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+1}^{p} &\approx d\boldsymbol{x}_{k+1} \\ &= \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] d\boldsymbol{x}_k + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k + \dot{\boldsymbol{x}}_k dh\end{aligned}$$

where the first two terms on the right-hand side indicate the increment in $\boldsymbol{x}_{k+1}$ for time held fixed and the last term represents the incremental change in state induced by the change of time step length, dh.

Sequential Convex Optimization Methods

Expanding $d\boldsymbol{x}_{k+1}$ for $j = k,\ k-1, \ldots, 1$ gives

$$
\begin{aligned}
d\boldsymbol{x}_{k+1} &= \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] d\boldsymbol{x}_k + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k + \dot{\boldsymbol{x}}_k dh \\
&= \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right]\left[\frac{\partial F_{k-1}}{\partial \boldsymbol{x}_{k-1}}\right] d\boldsymbol{x}_{k-1} + \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right]\left[\frac{\partial F_{k-1}}{\partial \boldsymbol{u}_{k-1}}\right] d\boldsymbol{u}_{k-1} + \left[\frac{\partial F_k}{\partial \boldsymbol{x}_k}\right] \dot{\boldsymbol{x}}_{k-1} dh \\
&\quad + \left[\frac{\partial F_k}{\partial \boldsymbol{u}_k}\right] d\boldsymbol{u}_k + \dot{\boldsymbol{x}}_k dh \\
&\ \ \vdots \\
&= \boxed{}
\end{aligned}
$$

Expanding $d\boldsymbol{x}_{k+1}$ for $j = k, k-1, \ldots, 1$ gives As the initial condition is assumed to be specified, $d\boldsymbol{x}_1 = 0$.

$$d\boldsymbol{x}_{k+1} = \sum_{j=1}^{k} \boldsymbol{B}_j^k d\boldsymbol{u}_j + dh \sum_{j=1}^{k} \boldsymbol{C}_j^k$$

The superscript k indicates that the state increments at step $k+1$ are obtained by k number of expansions.

Sequential Convex Optimization Methods

The overall state vector and optimization variable vector are defined as

$$
\begin{aligned}
\boldsymbol{x}_a &= \left[\boldsymbol{x}_2^T, \boldsymbol{x}_3^T, \ldots, \boldsymbol{x}_N^T\right]^T \in \mathbb{R}^{(N-1)n} \\
\boldsymbol{u}_a &= \left[\boldsymbol{u}_1^T, \boldsymbol{u}_2^T, \ldots, \boldsymbol{u}_{N-1}^T, h\right]^T \in \mathbb{R}^{(N-1)m+1}
\end{aligned}
$$

Accordingly, the state increment vector $d\boldsymbol{x}_a = \left[d\boldsymbol{x}_2^T, d\boldsymbol{x}_3^T, \ldots, d\boldsymbol{x}_N^T\right]^T \in \mathbb{R}^{(N-1)n}$, and the variable correction vector $d\boldsymbol{u}_a = \left[d\boldsymbol{u}_1^T, d\boldsymbol{u}_2^T, \ldots, d\boldsymbol{u}_{N-1}^T, dh\right]^T \in \mathbb{R}^{(N-1)m+1}$ can be given.

$$d\boldsymbol{x}_a = \boldsymbol{B} d\boldsymbol{u}_a$$

$$d\boldsymbol{x}_{k+1} = \sum_{j=1}^{k} \boldsymbol{B}_j^k d\boldsymbol{u}_j + dh \sum_{j=1}^{k} \boldsymbol{C}_j^k$$

$$d\boldsymbol{u}_a = \left[d\boldsymbol{u}_1^T,\, d\boldsymbol{u}_2^T,\, \ldots,\, d\boldsymbol{u}_{N-1}^T,\, dh \right]^T$$

where

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_1^1 & 0 & 0 & \ldots & 0 & \boxed{} \\ \boldsymbol{B}_1^2 & \boldsymbol{B}_2^2 & 0 & \ldots & 0 & \\ \vdots & \vdots & \vdots & \ddots & \vdots & \\ \boldsymbol{B}_1^{N-1} & \boldsymbol{B}_2^{N-1} & \boldsymbol{B}_3^{N-1} & \ldots & \boldsymbol{B}_{N-1}^{N-1} & \end{bmatrix} \in \mathbb{R}^{(N-1)n \times \{(N-1)m+1\}}$$

Sequential Convex Optimization Methods

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

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

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

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

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

Thus, the updated state vector $\boldsymbol{x}_a$ and the updated optimization variable vector $\boldsymbol{u}_a$ are both linear functions of $d\boldsymbol{u}_a$.

$$\begin{aligned}\boldsymbol{x}_a &= d\boldsymbol{x}_a + \boldsymbol{x}_a^p = \boldsymbol{B}d\boldsymbol{u}_a + \boldsymbol{x}_a^p \\ \boldsymbol{u}_a &= d\boldsymbol{u}_a + \boldsymbol{u}_a^p\end{aligned}$$

Next: How linear inequality constraints and quadratic cost function are converted?

Sequential Convex Optimization Methods

J is a quadratic cost function of $d\boldsymbol{x}_a$ and $d\boldsymbol{u}_a$. J can be expressed as

$$
\begin{aligned}
J &= \begin{bmatrix} d\boldsymbol{x}_a \\ d\boldsymbol{u}_a \end{bmatrix}^T \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^T & \boldsymbol{R} \end{bmatrix} \begin{bmatrix} d\boldsymbol{x}_a \\ d\boldsymbol{u}_a \end{bmatrix} + \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{b} \end{bmatrix}^T \begin{bmatrix} d\boldsymbol{x}_a \\ d\boldsymbol{u}_a \end{bmatrix} \\
&= d\boldsymbol{u}_a^T \begin{bmatrix} \boldsymbol{B}^T \\ \boldsymbol{I}^T \end{bmatrix}^T \begin{bmatrix} \boldsymbol{Q} & \boldsymbol{S} \\ \boldsymbol{S}^T & \boldsymbol{R} \end{bmatrix} \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{I} \end{bmatrix} d\boldsymbol{u}_a + \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{b} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{I} \end{bmatrix} d\boldsymbol{u}_a \\
&= d\boldsymbol{u}_a^T \boldsymbol{H} d\boldsymbol{u}_a + \boldsymbol{q} d\boldsymbol{u}_a
\end{aligned}
$$

Similarly, the equality constraints can be given as,

$$\begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{D}_2 \\ \boldsymbol{E}_1 & \boldsymbol{E}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{u}_a \end{bmatrix} = \boldsymbol{e}$$

The above relation can be reformulated as,

$$\begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{D}_2 \\ \boldsymbol{E}_1 & \boldsymbol{E}_2 \end{bmatrix} \begin{bmatrix} d\boldsymbol{x}_a \\ d\boldsymbol{u}_a \end{bmatrix} = \boldsymbol{e} - \begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{D}_2 \\ \boldsymbol{E}_1 & \boldsymbol{E}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_a^p \\ \boldsymbol{u}_a^p \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{D}_2 \\ \boldsymbol{E}_1 & \boldsymbol{E}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{I}_{(N-1)m+1} \end{bmatrix} d\boldsymbol{u}_a = \boldsymbol{e} - \begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{D}_2 \\ \boldsymbol{E}_1 & \boldsymbol{E}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_a^p \\ \boldsymbol{u}_a^p \end{bmatrix}$$

$$\boldsymbol{E} d\boldsymbol{u}_a = \boldsymbol{p}$$

The QP problem can be formulated as

$$\text{Problem } \mathcal{O}: \begin{cases} \underset{d\boldsymbol{u}_a}{\text{minimize}} & J(d\boldsymbol{u}_a) = d\boldsymbol{u}_a^T \boldsymbol{H} d\boldsymbol{u}_a + \boldsymbol{q} d\boldsymbol{u}_a \\ \text{subject to} & \boldsymbol{G} d\boldsymbol{u}_a \leq \boldsymbol{h}, \quad \boldsymbol{E} d\boldsymbol{u}_a = \boldsymbol{p} \end{cases}$$

Because u_a include the time step, a free final time solution

It is a constrained QP problem. Feasibility?

Sequential Convex Optimization Methods

For clarity, an infeasible problem $\mathcal{O}$ is represented by

$$\text{Problem } \mathcal{O}_i : \begin{cases} \underset{d\boldsymbol{u}_a}{\text{minimize}} & J(d\boldsymbol{u}_a) \\ \text{subject to} & \boldsymbol{l} \leq \boldsymbol{M} d\boldsymbol{u}_a \leq \boldsymbol{u} \end{cases}$$

There always exists a shift $\boldsymbol{s}$ that relaxes the constraints so that the *shifted* problem becomes feasible given by

$$\text{Problem } \mathcal{O}_s : \begin{cases} \underset{d\boldsymbol{u}_a,\ \boldsymbol{s}}{\text{minimize}} & J(d\boldsymbol{u}_a) \\ \text{subject to} & \boldsymbol{l} \leq \boldsymbol{M} d\boldsymbol{u}_a + \boldsymbol{s} \leq \boldsymbol{u} \end{cases}$$

A revised augmented Lagrangian Algorithm

- To “solve” infeasible subproblem
- To let the iteration continue

  (Otherwise the sequence gets stuck)
- To find a minimum shift

A revised augmented Lagrangian Algorithm

For clarity, an infeasible problem $\mathcal{O}$ is represented by

$$\text{Problem } \mathcal{O}_i : \begin{cases} \underset{d\boldsymbol{u}_a}{\text{minimize}} & J(d\boldsymbol{u}_a) \\ \text{subject to} & \boldsymbol{l} \leq \boldsymbol{M} d\boldsymbol{u}_a \leq \boldsymbol{u} \end{cases}$$

The problem $\mathcal{O}_i$ can be equivalently reformulated by introducing an auxiliary vector $\boldsymbol{z}$ as

$$\text{Problem } \mathcal{A} : \begin{cases} \underset{d\boldsymbol{u}_a,\ \boldsymbol{z}}{\text{minimize}} & J(d\boldsymbol{u}_a) \\ \text{subject to} & \boldsymbol{M} d\boldsymbol{u}_a = \boldsymbol{z} \\ & \boldsymbol{l} \leq \boldsymbol{z} \leq \boldsymbol{u} \end{cases}$$

A revised augmented Lagrangian Algorithm

From Problem $\mathcal{A}$, the augmented Lagrangian can be given as

$$L\left(d\boldsymbol{u}_a, \boldsymbol{z}, \boldsymbol{\lambda}\right) = J(d\boldsymbol{u}_a) + \boldsymbol{\lambda}^T\left(\boldsymbol{M}d\boldsymbol{u}_a - \boldsymbol{z}\right) + \frac{r}{2}\left\|\boldsymbol{M}d\boldsymbol{u}_a - \boldsymbol{z}\right\|^2$$

The standard augmented Lagrangian method solves

$$\text{Problem } \mathcal{A}_a \ : \begin{cases} \underset{d\boldsymbol{u}_a, \boldsymbol{z}}{\text{minimize}} & L\left(d\boldsymbol{u}_a, \boldsymbol{z}, \boldsymbol{\lambda}\right) \\ \text{subject to} & \boldsymbol{l} \le \boldsymbol{z} \le \boldsymbol{u} \end{cases}$$

and updates the multiplier and the augmentation parameter until $\boldsymbol{s} = \boldsymbol{z} - \boldsymbol{M}d\boldsymbol{u}_a \simeq 0$.

A revised augmented Lagrangian Algorithm

From Problem $\mathcal{A}$, the augmented Lagrangian can be given as

$$L\left(d\boldsymbol{u}_a, \boldsymbol{z}, \boldsymbol{\lambda}\right) = J(d\boldsymbol{u}_a) + \boldsymbol{\lambda}^T \left(\boldsymbol{M}d\boldsymbol{u}_a - \boldsymbol{z}\right) + \frac{r}{2}\left\|\boldsymbol{M}d\boldsymbol{u}_a - \boldsymbol{z}\right\|^2$$

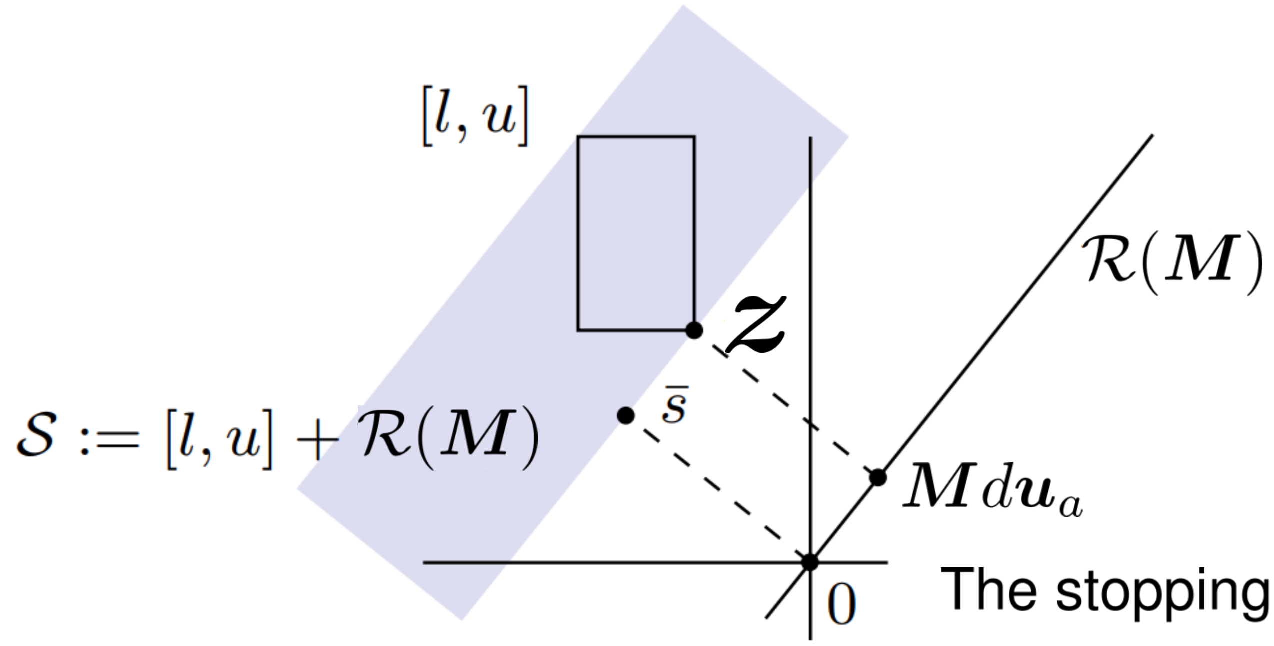
The standard augmented Lagrangian method solves

$$\text{Problem } \mathcal{A}_a : \begin{cases} \underset{d\boldsymbol{u}_a, \boldsymbol{z}}{\text{minimize}} & L\left(d\boldsymbol{u}_a, \boldsymbol{z}, \boldsymbol{\lambda}\right) \\ \text{subject to} & \boldsymbol{l} \leq \boldsymbol{z} \leq \boldsymbol{u} \end{cases}$$

and updates the multiplier and the augmentation parameter until $\boldsymbol{s} = \boldsymbol{z} - \boldsymbol{M}d\boldsymbol{u}_a \simeq 0$.

A revised augmented Lagrangian Algorithm

To find the minimum shift



$$\text{Problem } \mathcal{A}: \begin{cases} \underset{d\boldsymbol{u}_a,\ \boldsymbol{z}}{\text{minimize}} & J(d\boldsymbol{u}_a) \\ \text{subject to} & \boldsymbol{M} d\boldsymbol{u}_a = \boldsymbol{z} \\ & \boldsymbol{l} \le \boldsymbol{z} \le \boldsymbol{u} \end{cases}$$

The stopping criteria are given by

$$\boldsymbol{M}^T \left(\boldsymbol{M} d\boldsymbol{u}_a - \boldsymbol{z}\right) \simeq 0 \ \&\ P_{[l,u]} \left(\boldsymbol{M} d\boldsymbol{u}_a\right) \simeq \boldsymbol{z}$$

A revised augmented Lagrangian Algorithm for Guidance

A terminally constrained problem's subproblem:

$$
\text{Problem } \mathcal{G}_o : \begin{cases} \underset{d\boldsymbol{u}_a}{\text{minimize}} & J(d\boldsymbol{u}_a) \\ \text{subject to} & \boldsymbol{B}^{N-1} d\boldsymbol{u}_a = \Delta \boldsymbol{x}_N \end{cases}
$$

$\mathcal{G}_o$ is always feasible.

A revised augmented Lagrangian Algorithm for Guidance

The inequalities and equalities excluding the terminal constraints are denoted by $\boldsymbol{M}_s$ which is a subset of $\boldsymbol{M}$. They can be added to Problem $\mathcal{G}_o$ by following the same relaxing pattern using an auxiliary vector. Then, a shifted QP problem for guidance command generation can be expressed as

$$\text{Problem } \mathcal{G}_s: \begin{cases} \underset{d\boldsymbol{u}_a, \boldsymbol{z}, \boldsymbol{s}}{\text{minimize}} & J(d\boldsymbol{u}_a) \\ \text{subject to} & \boldsymbol{B}^{N-1} d\boldsymbol{u}_a = \Delta \boldsymbol{x}_N \\ & \boldsymbol{M}_s d\boldsymbol{u}_a + \boldsymbol{s} = \boldsymbol{z} \\ & \boldsymbol{l} \leq \boldsymbol{z} \leq \boldsymbol{u} \end{cases} \tag{1}$$

- G_s is also always FEASIBLE
- We can find the minimum shift

Summary

- Constrained Guidance Problem
- Dealt with the Nonlinear Dynamics by finding the relation between the state increments and the control corrections
- Fixed-final-time and Free-final-time solutions
- Dealt with Infeasible Subproblems

Sequential Convex Optimization Methods

$$\begin{aligned} \underset{d\boldsymbol{u}_a}{\text{minimize}} \quad & J\left(d\boldsymbol{u}_a\right) \\ \text{subject to} \quad & d\boldsymbol{x}_a = \boldsymbol{B}_t d\boldsymbol{u}_a \\ & \boldsymbol{g}\left(d\boldsymbol{u}_a\right) \leq 0 \end{aligned}$$

?

Single Shooting
Controls are variables

Full Discretization
States and Controls are variables

The continuous nonlinear system dynamics can be expressed by

$$\dot{\boldsymbol{x}} = \boldsymbol{f}\left(\boldsymbol{x}, \boldsymbol{u}\right)$$

The successive linearization converts the nonlinear dynamics to a linear form. In an iterative process, the dynamics at the $(j+1)$-th iteration can be linearized with respect to the (j)-th solution, which can be written as

$$\begin{aligned}
\left(\dot{\boldsymbol{x}}\right)^{(j+1)} &= \boldsymbol{F}_{\boldsymbol{x}}\left(\boldsymbol{x}^{(j)}, \boldsymbol{u}^{(j)},\right)\boldsymbol{x}^{(j+1)} + \boldsymbol{F}_{\boldsymbol{u}}\left(\boldsymbol{x}^{(j)}, \boldsymbol{u}^{(j)}\right)\boldsymbol{u}^{(j+1)} \\
&\quad + \boldsymbol{f}\left(\boldsymbol{x}^{(j)}, \boldsymbol{u}^{(j)}\right) - \boldsymbol{F}_{\boldsymbol{x}}\left(\boldsymbol{x}^{(j)}, \boldsymbol{u}^{(j)}\right)\boldsymbol{x}^{(j)} - \boldsymbol{F}_{\boldsymbol{u}}\left(\boldsymbol{x}^{(j)}, \boldsymbol{u}^{(j)}\right)\boldsymbol{u}^{(j)} \\
&= \boldsymbol{F}_{\boldsymbol{x}}^{(j)}\left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) + \boldsymbol{F}_{\boldsymbol{u}}^{(j)}\left(\boldsymbol{u}^{(j+1)} - \boldsymbol{u}^{(j)}\right) + \boldsymbol{f}^{(j)}
\end{aligned}$$

The incremental changes of the state and input vectors over iterations are defined as

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

The forward Euler method:

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

Sequential Convex Optimization Methods

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

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

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

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

Sequential Convex Optimization Methods

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

It is to be noted that $\boldsymbol{x}_{k+1}^{(j)}$ on the left-hand side is not equal to $h\boldsymbol{f}_k^{(j)} + \boldsymbol{x}_k^{(j)}$ on the right-hand side.

A MAJOR difference with the shooting methods:
States are variables

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

Defining

$$
\begin{aligned}
\tilde{\boldsymbol{x}}_{k+1}^{(j)} &:= h\boldsymbol{f}_k^{(j)} + \boldsymbol{x}_k^{(j)} \\
\boldsymbol{e}_{k+1}^{(j)} &:= \tilde{\boldsymbol{x}}_{k+1}^{(j)} - \boldsymbol{x}_{k+1}^{(j)}
\end{aligned}
$$

Integration Residual

$$
d\boldsymbol{x}_{k+1}^{(j+1)} - \left(\tilde{\boldsymbol{F}}_{\boldsymbol{x}}\right)_k^{(j)} d\boldsymbol{x}_k^{(j+1)} - \left(\tilde{\boldsymbol{F}}_{\boldsymbol{u}}\right)_k^{(j)} d\boldsymbol{u}_k^{(j+1)} = \boldsymbol{e}_{k+1}^{(j)},
$$

$$d\boldsymbol{x}_{k+1}^{(j+1)} - \left(\tilde{\boldsymbol{F}}_{\boldsymbol{x}}\right)_k^{(j)} d\boldsymbol{x}_k^{(j+1)} - \left(\tilde{\boldsymbol{F}}_{\boldsymbol{u}}\right)_k^{(j)} d\boldsymbol{u}_k^{(j+1)} = \boldsymbol{e}_{k+1}^{(j)},$$

$d\boldsymbol{x}_a$ is the total increment vector

$$d\boldsymbol{x}_a = \left[(d\boldsymbol{x}_2)^\mathsf{T}, (d\boldsymbol{x}_3)^\mathsf{T}, \ldots, (d\boldsymbol{x}_N)^\mathsf{T}, (d\boldsymbol{u}_1)^\mathsf{T}, (d\boldsymbol{u}_2)^\mathsf{T}, \ldots, (d\boldsymbol{u}_{N-1})^\mathsf{T},\right]^\mathsf{T},$$

and $\boldsymbol{e}_a$ is the total residual vector

$$\boldsymbol{e}_a = \left[(\boldsymbol{e}_2)^\mathsf{T}, (\boldsymbol{e}_3)^\mathsf{T}, \ldots, (\boldsymbol{e}_{N_t})^\mathsf{T}\right]^\mathsf{T}.$$

Sequential Convex Optimization Methods

$$d\boldsymbol{x}_{k+1}^{(j+1)} - \left(\tilde{\boldsymbol{F}}_{\boldsymbol{x}}\right)_k^{(j)} d\boldsymbol{x}_k^{(j+1)} - \left(\tilde{\boldsymbol{F}}_{\boldsymbol{u}}\right)_k^{(j)} d\boldsymbol{u}_k^{(j+1)} = \boldsymbol{e}_{k+1}^{(j)}$$

$$\boldsymbol{D}^{(j)} d\boldsymbol{x}_a^{(j+1)} = \boldsymbol{e}_a^{(j)}$$

where

$$\boldsymbol{D}^{(j)} = \begin{bmatrix} \boldsymbol{I}_n & 0 & \cdots & 0 & 0 & -\left(\tilde{\boldsymbol{F}}_{\boldsymbol{u}}\right)_1^{(j)} & 0 & \cdots & 0 \\ -\left(\tilde{\boldsymbol{F}}_{\boldsymbol{x}}\right)_2^{(j)} & \boldsymbol{I}_n & \cdots & 0 & 0 & 0 & -\left(\tilde{\boldsymbol{F}}_{\boldsymbol{u}}\right)_2^{(j)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -\left(\tilde{\boldsymbol{F}}_{\boldsymbol{x}}\right)_{N_t-1}^{(j)} & \boldsymbol{I}_n & 0 & 0 & \cdots & -\left(\tilde{\boldsymbol{F}}_{\boldsymbol{u}}\right)_{N_t-1}^{(j)} \end{bmatrix}$$

States dynamics with respect to climb distance

Provided that r has a strict monotonicity, it can be considered as an independent variable, with respect to whom the dynamics are reformulated using the chain rule as

$$\boldsymbol{x}' := \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}r} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}r} = \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t}\frac{1}{V\cos(\gamma)},$$

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:

It can be noticed that x' is a function of x, u, and t_f as

Sequential Convex Optimization Methods

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

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

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

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

Sequential Convex Optimization Methods

Using the trapezoidal rule, a discrete form of system dynamics can be expressed as

$$\boldsymbol{x}_{k+1} = \left(\boldsymbol{x}'_k + \boldsymbol{x}'_{k+1}\right) d\tau/2 + \boldsymbol{x}_k$$

where $k = 1,\, 2\, \ldots, N-1$ are the discrete grid points in the normalized cycle.

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

Sequential Convex Optimization Methods

$$
\begin{aligned}
d\boldsymbol{x}_{k+1} = & \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_k^p d\boldsymbol{x}_k + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{u}}\right]_k^p d\boldsymbol{u}_k + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_k^p dt_f \\
& + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_{k+1}^p d\boldsymbol{x}_{k+1} + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{u}}\right]_{k+1}^p d\boldsymbol{u}_{k+1} + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_{k+1}^p dt_f \\
& + d\boldsymbol{x}_k + \boldsymbol{e}_{k+1}
\end{aligned}
$$

where $d\boldsymbol{x}$, $d\boldsymbol{u}$, and dt_f are the incremental changes of states, controls, and final time, respectively.

$$
\tilde{\boldsymbol{x}}_{k+1}^p = \left((\boldsymbol{x}_k')^p + (\boldsymbol{x}_{k+1}')^p \right) d\tau/2 + \boldsymbol{x}_k^p
$$

$$
\boldsymbol{e}_{k+1} = \tilde{\boldsymbol{x}}_{k+1}^p - \boldsymbol{x}_{k+1}^p
$$

$$d\boldsymbol{x}_{k+1} = \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_k^p d\boldsymbol{x}_k + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{u}}\right]_k^p d\boldsymbol{u}_k + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_k^p dt_f$$
$$+ \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_{k+1}^p d\boldsymbol{x}_{k+1} + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{u}}\right]_{k+1}^p d\boldsymbol{u}_{k+1} + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_{k+1}^p dt_f$$
$$+ d\boldsymbol{x}_k + \boldsymbol{e}_{k+1}$$

$$\left(\boldsymbol{I}_n - \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_{k+1}^p\right) d\boldsymbol{x}_{k+1} - \left(\boldsymbol{I}_n + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_k^p\right) d\boldsymbol{x}_k$$
$$- \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{u}}\right]_k^p d\boldsymbol{u}_k - \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{u}}\right]_{k+1}^p d\boldsymbol{u}_{k+1}$$
$$- \frac{d\tau}{2}\left(\left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_k^p + \left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_{k+1}^p\right) dt_f = \boldsymbol{e}_{k+1}$$

Sequential Convex Optimization Methods

$$\left(\boldsymbol{I}_n - \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_{k+1}^{p}\right) d\boldsymbol{x}_{k+1} - \left(\boldsymbol{I}_n + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_{k}^{p}\right) d\boldsymbol{x}_{k}$$
$$- \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{u}}\right]_{k}^{p} d\boldsymbol{u}_{k} - \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{u}}\right]_{k+1}^{p} d\boldsymbol{u}_{k+1}$$
$$- \frac{d\tau}{2}\left(\left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_{k}^{p} + \left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_{k+1}^{p}\right) dt_f = \boldsymbol{e}_{k+1}$$

$$\left[\boldsymbol{F}_{\boldsymbol{x}}^{-}\right]_k = \boldsymbol{I}_n - \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_{k}^{p}$$

$$\left[\boldsymbol{F}_{\boldsymbol{x}}^{+}\right]_k = \boldsymbol{I}_n + \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{x}}\right]_{k}^{p}$$

$$\left[\boldsymbol{F}_{\boldsymbol{u}}\right]_k = \frac{d\tau}{2}\left[\frac{\partial \boldsymbol{x}'}{\partial \boldsymbol{u}}\right]_{k}^{p}$$

$$\left[\boldsymbol{F}_{t_f}\right]_k = \frac{d\tau}{2}\left(\left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_{k}^{p} + \left[\frac{\partial \boldsymbol{x}'}{\partial t_f}\right]_{k+1}^{p}\right)$$

Sequential Convex Optimization Methods

$$
\begin{bmatrix}
-[\boldsymbol{F}_{\boldsymbol{x}}^{+}]_1 & [\boldsymbol{F}_{\boldsymbol{x}}^{-}]_2 & 0 & \cdots & 0 \\
0 & -[\boldsymbol{F}_{\boldsymbol{x}}^{+}]_2 & [\boldsymbol{F}_{\boldsymbol{x}}^{-}]_3 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & -[\boldsymbol{F}_{\boldsymbol{x}}^{+}]_{N-1} & [\boldsymbol{F}_{\boldsymbol{x}}^{-}]_N
\end{bmatrix}
\begin{bmatrix} d\boldsymbol{x}_1 \\ d\boldsymbol{x}_2 \\ \vdots \\ d\boldsymbol{x}_N \end{bmatrix}
- \begin{bmatrix}
[\boldsymbol{F}_{\boldsymbol{u}}]_1 & [\boldsymbol{F}_{\boldsymbol{u}}]_2 & 0 & \cdots & 0 \\
0 & [\boldsymbol{F}_{\boldsymbol{u}}]_2 & [\boldsymbol{F}_{\boldsymbol{u}}]_3 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & [\boldsymbol{F}_{\boldsymbol{u}}]_{N-1} & [\boldsymbol{F}_{\boldsymbol{u}}]_N
\end{bmatrix}
\begin{bmatrix} d\boldsymbol{u}_1 \\ d\boldsymbol{u}_2 \\ \vdots \\ d\boldsymbol{u}_N \end{bmatrix}
- \begin{bmatrix} [\boldsymbol{F}_{t_f}]_1 \\ [\boldsymbol{F}_{t_f}]_2 \\ \vdots \\ [\boldsymbol{F}_{t_f}]_{N-1} \end{bmatrix} dt_f
= \begin{bmatrix} \boldsymbol{e}_2 \\ \boldsymbol{e}_3 \\ \vdots \\ \boldsymbol{e}_N \end{bmatrix}
$$

Course Project

Solve any computational guidance or trajectory optimization problem

- Group of max. 4 people
- Presentation of 20 to 30 minutes
- Starting the 15^{th} or the 16^{th} week till the 17^{th} week
- Let me know your group member by the end of next Thursday

Evaluation:

- Technical details 50%
- Presentation performance 50%