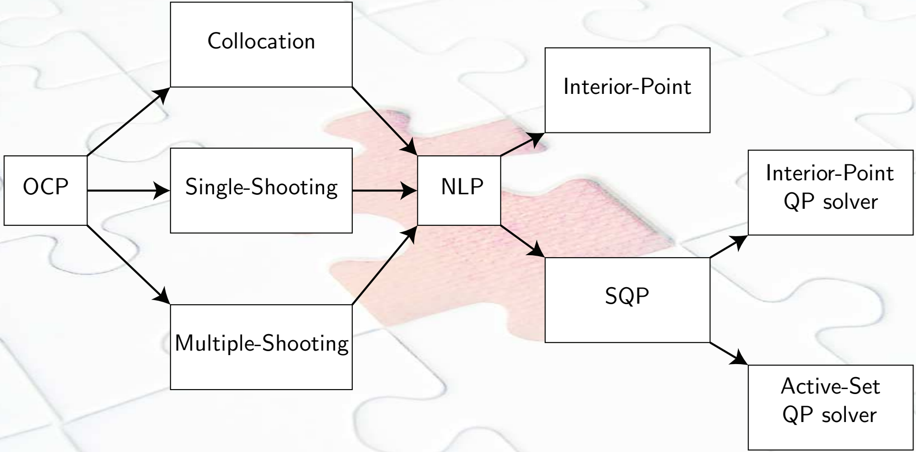
Numerical Optimal Control
Lecture 6: Direct Collocation

Sébastien Gros

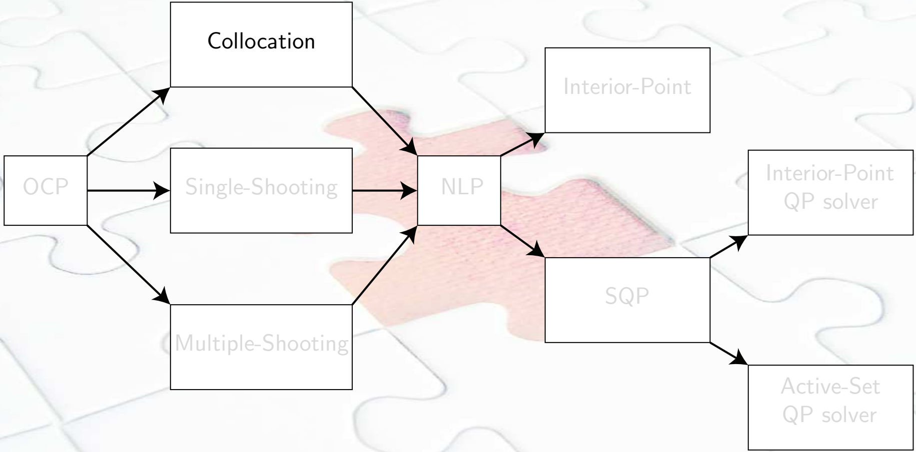
ITK, NTNU

NTNU PhD course

Survival map of Direct Optimal Control



Survival map of Direct Optimal Control



Another way for going from OCP to NLP

Outline

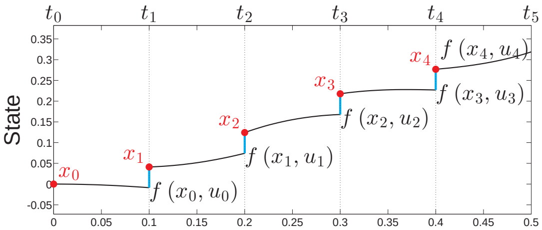
1. Polynomial interpolation
2. Collocation-based integration
3. Collocation in multiple-shooting
4. Direct Collocation
5. NLP from direct collocation

Outline

Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$



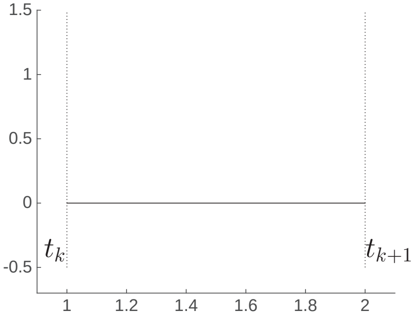
Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$



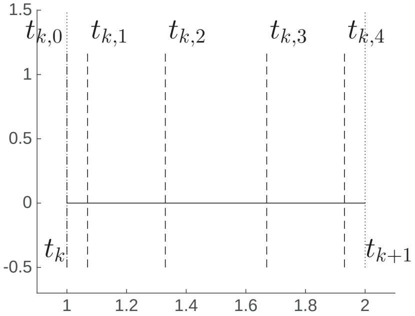
Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

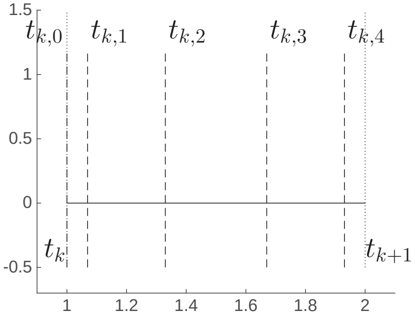
Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

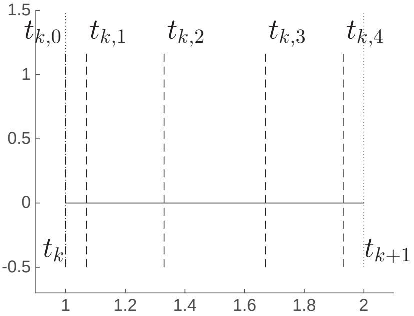
$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \dots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

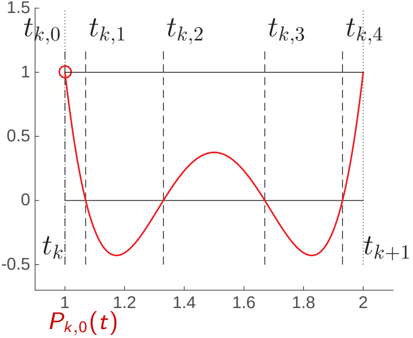
$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \dots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

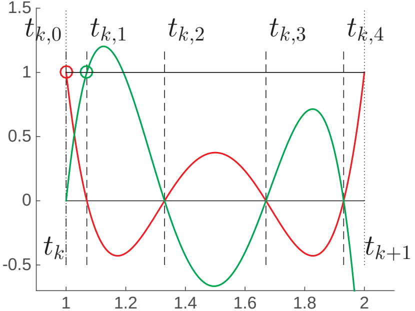
$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



$P_{k,0}(t)$, $P_{k,1}(t)$

Polynomial interpolation

Consider a time grid:

$$\{t_{k,0},\, ...,\, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

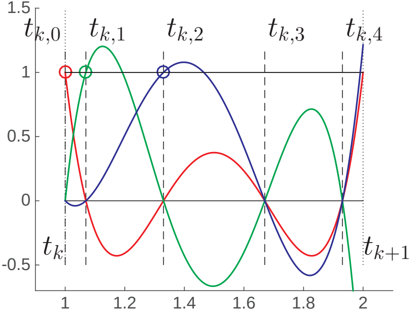
$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0},\, ...,\, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



$P_{k,0}(t)$, $P_{k,1}(t)$, $P_{k,2}(t)$

Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

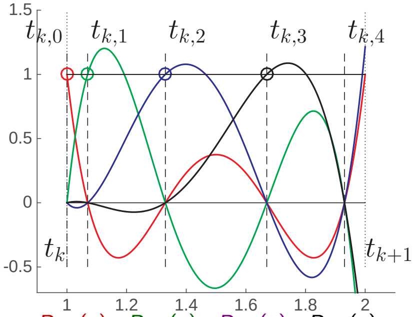
$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



$P_{k,0}(t)$, $P_{k,1}(t)$, $P_{k,2}(t)$, $P_{k,3}(t)$

Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \dots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

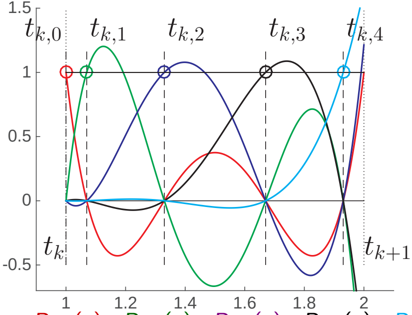
$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \dots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



$P_{k,0}(t)$, $P_{k,1}(t)$, $P_{k,2}(t)$, $P_{k,3}(t)$, $P_{k,4}(t)$

Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

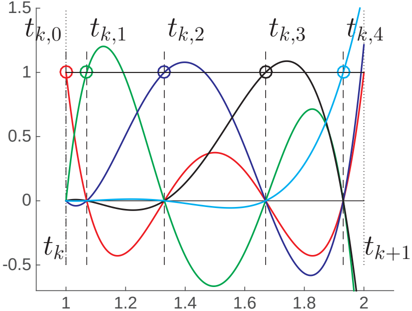
$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



$P_{k,0}(t)$, $P_{k,1}(t)$, $P_{k,2}(t)$, $P_{k,3}(t)$, $P_{k,4}(t)$

Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

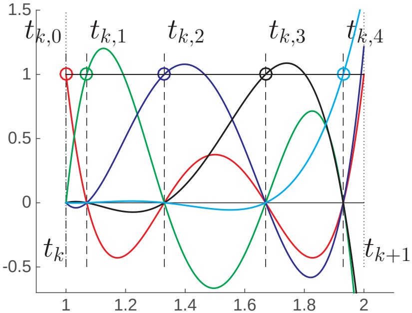
$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



$P_{k,0}(t)$, $P_{k,1}(t)$, $P_{k,2}(t)$, $P_{k,3}(t)$, $P_{k,4}(t)$

Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

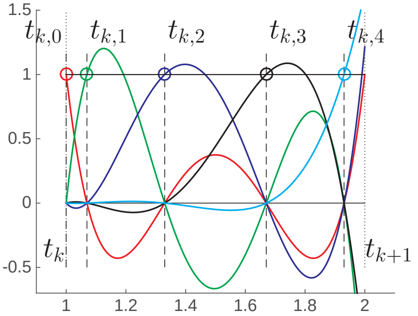
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

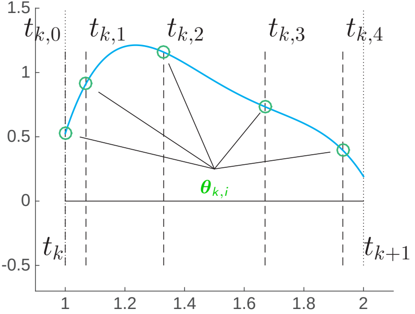
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \dots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

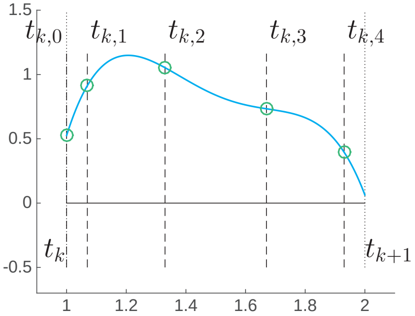
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \dots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

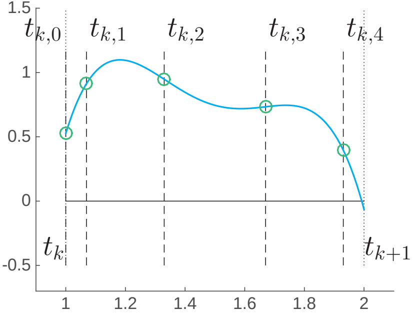
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

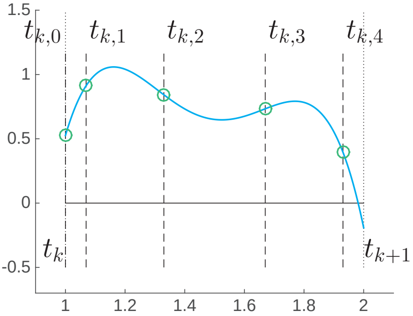
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

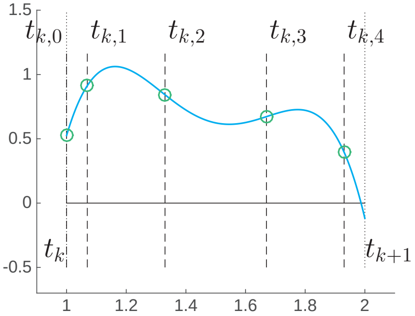
$$\mathrm{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathrm{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k, \ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

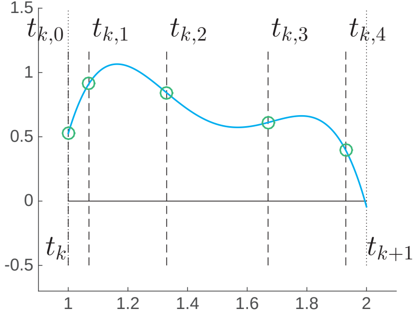
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1, \ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

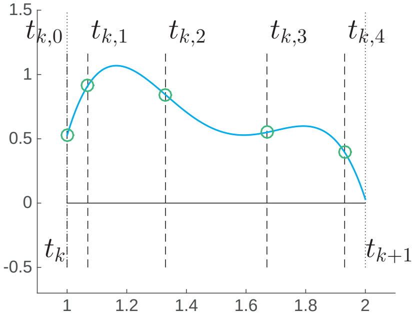
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \dots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

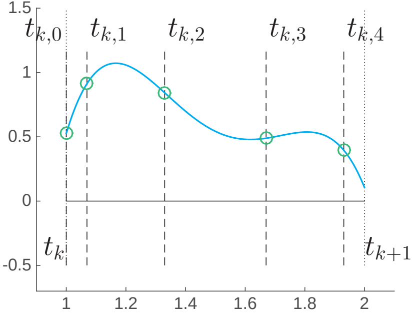
$$\mathrm{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathrm{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \dots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

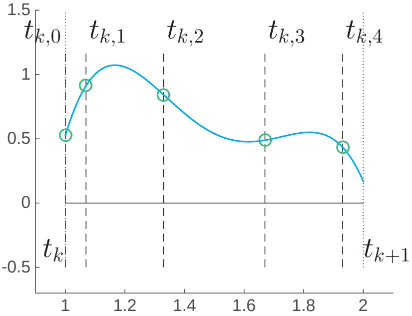
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

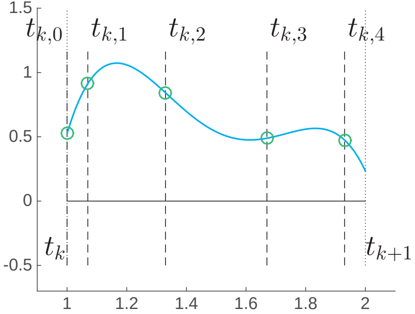
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

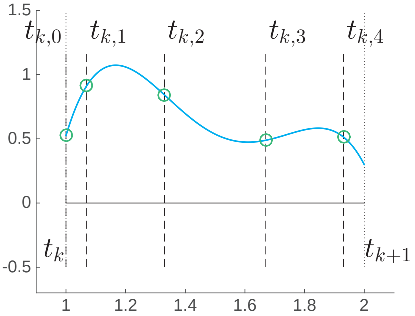
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \ldots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \dots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

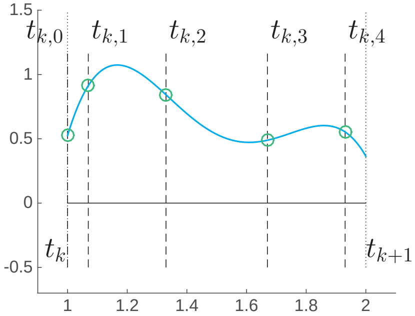
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \dots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



Polynomial interpolation

Consider a time grid:

$$\{t_{k,0}, \dots, t_{k,K}\} \in [t_k,\ t_{k+1}]$$

Lagrange Polynomials:

$$P_{k,i}(t) = \prod_{j=0,\, j\neq i}^{K} \frac{t - t_{k,j}}{t_{k,i} - t_{k,j}} \in \mathbb{R}$$

of order K, with property:

$$P_{k,i}(t_{k,l}) = \begin{cases} 1 & \text{if} \quad l = i \\ 0 & \text{if} \quad l \neq i \end{cases}$$

Interpolation with $\boldsymbol{\theta}_{k,i} \in \mathbb{R}^n$

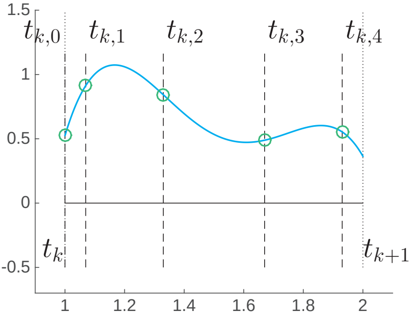
$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

having the property:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

E.g.

- $t_k = 1,\ t_{k+1} = 2$
- $K = 4$
- $\{t_{k,0}, \dots, t_{k,K}\} = \{1.0, 1.0694, 1.33, 1.67, 1.931\}$



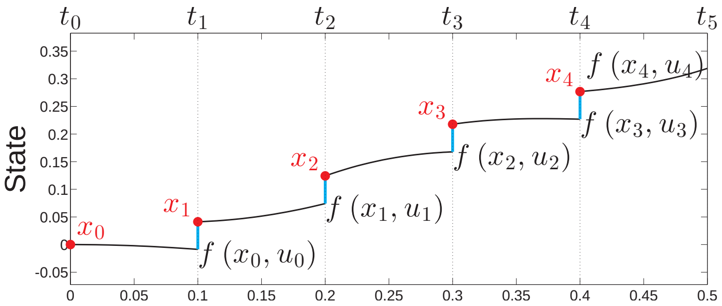
Note: the Lagrange polynomials are orthogonal, i.e.

$$\int_{t_k}^{t_{k+1}} P_{k,i}(t) P_{k,j}(t)\, \mathrm{d}t = 0, \quad \forall i \neq j$$

Outline

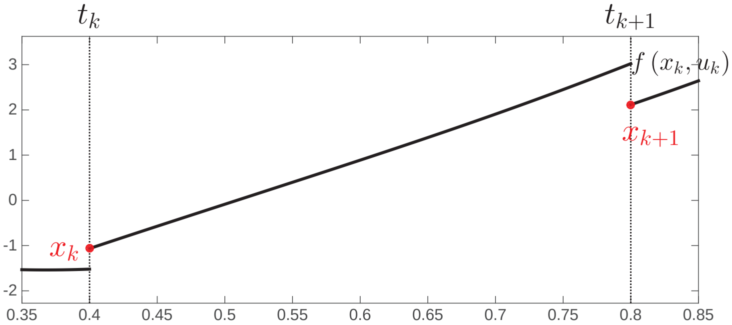
Collocation methods - key idea

Approximate state trajectory $\mathbf{s}(t)$ via polynomials (order K)



Collocation methods - key idea

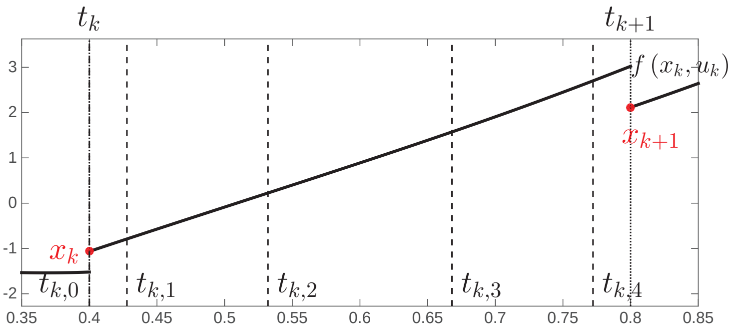
Approximate state trajectory $\mathbf{s}(t)$ via polynomials (order K)



Collocation methods - key idea

Approximate state trajectory $\mathbf{s}(t)$ via polynomials (order K)

- **Time grid**: $\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$

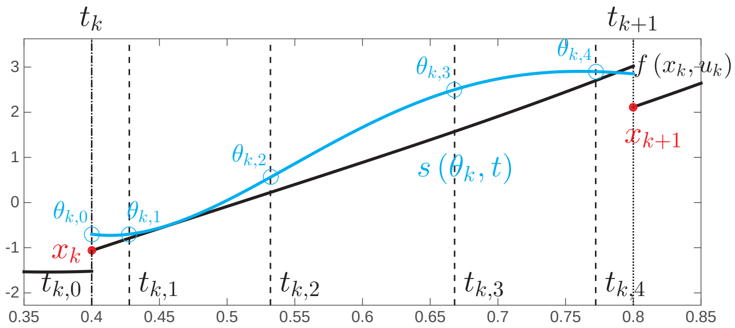


Collocation methods - key idea

Approximate state trajectory $\mathbf{s}(t)$ via polynomials (order K)

- **Time grid**: $\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$
- **Interpolate** on each interval $[t_k, t_{k+1}]$ using:

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$



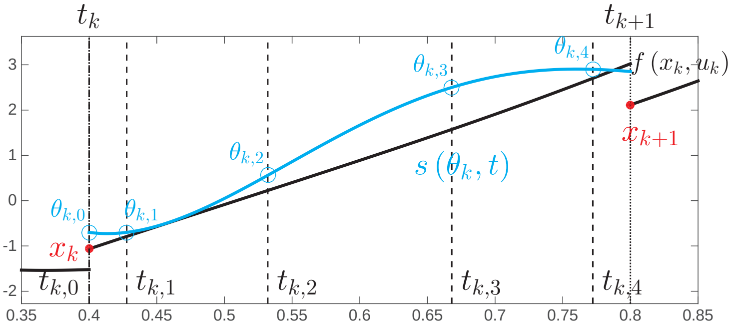
Collocation methods - key idea

Approximate state trajectory $\mathbf{s}(t)$ via polynomials (order K)

- **Time grid**: $\{t_{k,0}, \ldots, t_{k,K}\} \in [t_k,\ t_{k+1}]$
- **Interpolate** on each interval $[t_k, t_{k+1}]$ using:

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

- Integration: adjust $\boldsymbol{\theta}_{k,i}$ to approximate the dynamics $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u})$

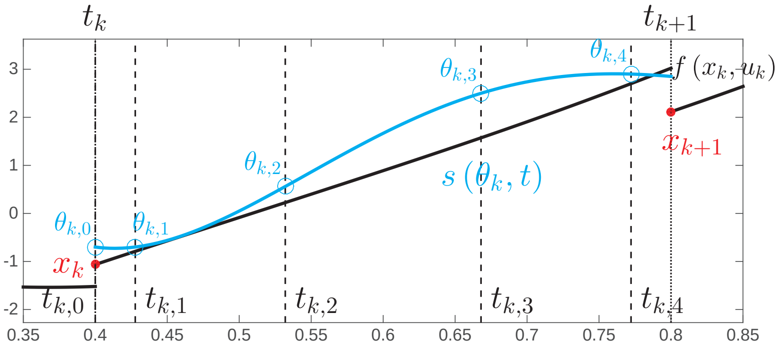


Collocation methods - how to adjust the $\boldsymbol{\theta}_{k,i}$?

On each interval $[t_k, t_{k+1}]$, approximate $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}_k)$ using

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}} \qquad \text{with} \qquad \mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

Note: we have $K+1$ degrees of freedom *per state*.



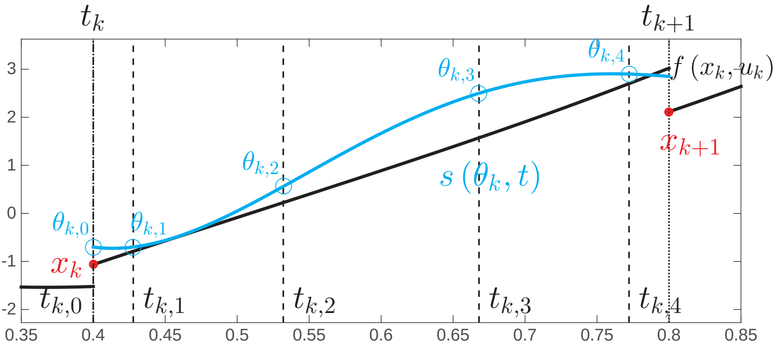
Collocation methods - how to adjust the $\boldsymbol{\theta}_{k,i}$?

On each interval $[t_k, t_{k+1}]$, approximate $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}_k)$ using

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}} \qquad \text{with} \qquad \mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

Note: we have $K+1$ degrees of freedom *per state*. Collocation uses the constraints:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_k) = \boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$



Collocation methods - how to adjust the $\boldsymbol{\theta}_{k,i}$?

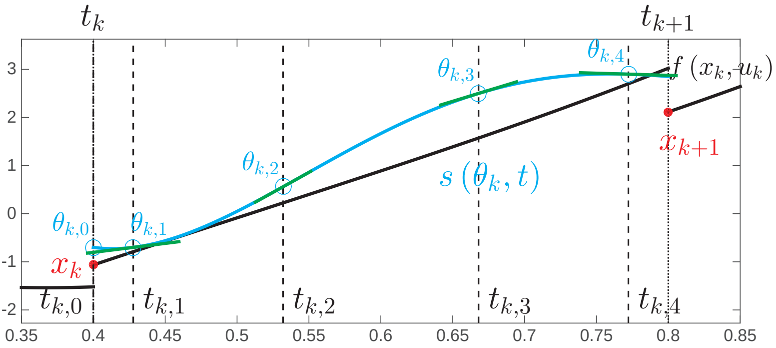
On each interval $[t_k, t_{k+1}]$, approximate $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}_k)$ using

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}} \qquad \text{with} \qquad \mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

Note: we have $K+1$ degrees of freedom *per state*. Collocation uses the constraints:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_k) = \boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \mathbf{F}(\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}), \mathbf{u}_k), \quad j = 1, ..., K$$



Collocation methods - how to adjust the $\boldsymbol{\theta}_{k,i}$?

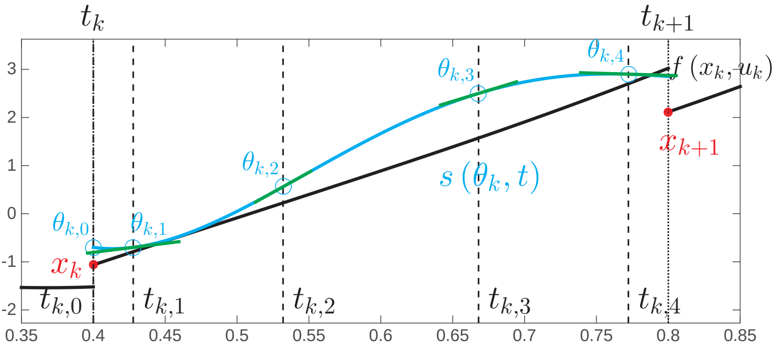
On each interval $[t_k, t_{k+1}]$, approximate $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}_k)$ using

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}} \qquad \text{with} \qquad \mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

Note: we have $K+1$ degrees of freedom *per state*. Collocation uses the constraints:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_k) = \boldsymbol{\theta}_{k,0} = \mathbf{x}_k \qquad \text{(note that } \mathbf{x}_k, \mathbf{u}_k \text{ are coming from the NLP !!)}$$

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \mathbf{F}(\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}), \mathbf{u}_k), \quad j = 1, ..., K$$



Collocation methods - how to adjust the $\boldsymbol{\theta}_{k,i}$?

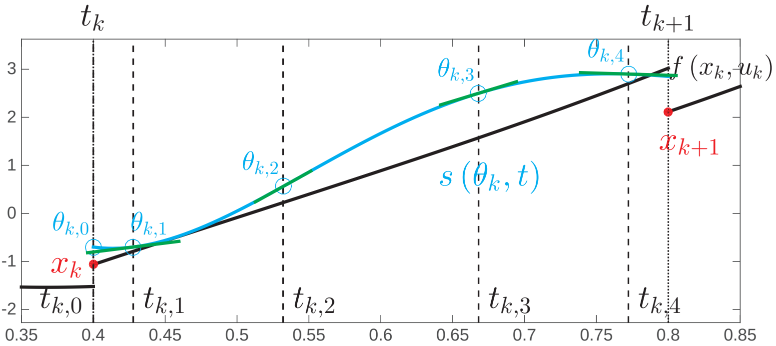
On each interval $[t_k, t_{k+1}]$, approximate $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}_k)$ using

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}} \qquad \text{with} \qquad \mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

Note: we have $K+1$ degrees of freedom *per state*. Collocation uses the constraints:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_k) = \boldsymbol{\theta}_{k,0} = \mathbf{x}_k \qquad \text{(note that } \mathbf{x}_k,\ \mathbf{u}_k \text{ are coming from the NLP !!)}$$

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \mathbf{F}(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k), \quad j = 1, ..., K$$



Collocation methods - how to adjust the $\boldsymbol{\theta}_{k,i}$?

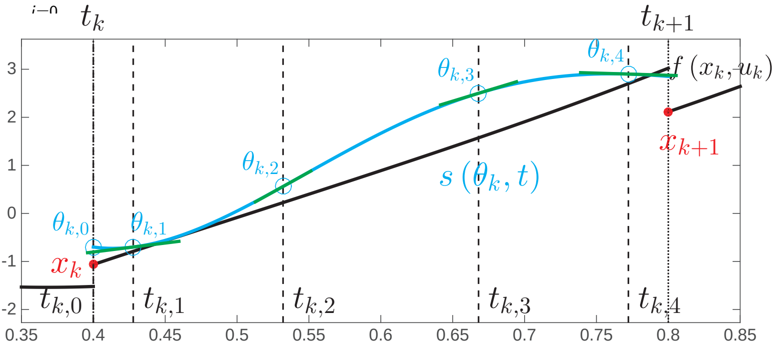
On each interval $[t_k, t_{k+1}]$, approximate $\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}_k)$ using

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}} \qquad \text{with} \qquad \mathbf{s}(\boldsymbol{\theta}_k, t_{k,j}) = \boldsymbol{\theta}_{k,j}$$

Note: we have $K+1$ degrees of freedom *per state*. Collocation uses the constraints:

$$\mathbf{s}(\boldsymbol{\theta}_k, t_k) = \boldsymbol{\theta}_{k,0} = \mathbf{x}_k \qquad \text{(note that } \mathbf{x}_k, \mathbf{u}_k \text{ are coming from the NLP !!)}$$

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \dot{P}_{k,i}(t_{k,j}) = \mathbf{F}(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k), \quad j = 1, ..., K$$



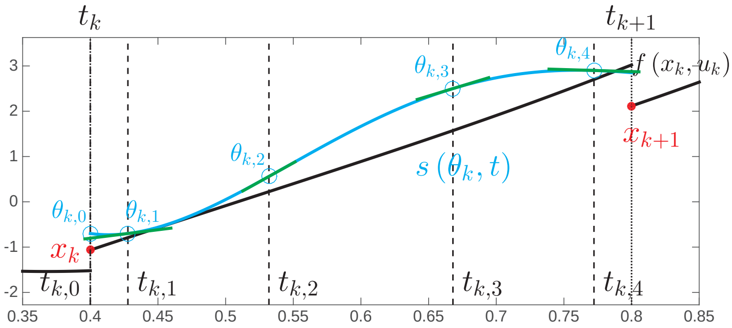
Collocation methods - Implementation

Collocation uses the constraints:

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \dot{P}_{k,i}(t_{k,j}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k\right)$$

for $j = 1, ..., K$.



Collocation methods - Implementation

Collocation uses the constraints:

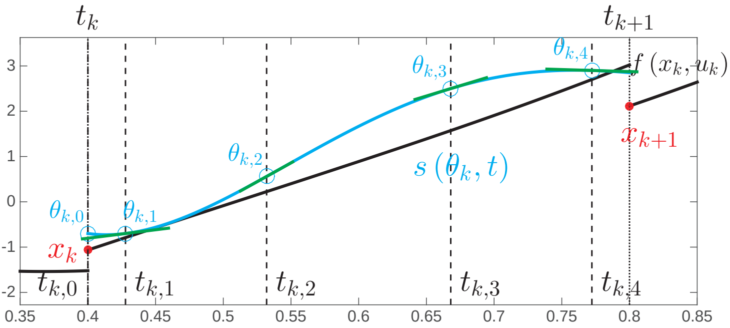
$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \dot{P}_{k,i}(t_{k,j}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k\right)$$

for $j = 1, ..., K$.

Solve for $\boldsymbol{\theta}_{k,i}$ using Newton

$$\mathbf{c} = \left[\begin{array}{c} \boldsymbol{\theta}_{k,0} - \mathbf{x}_k \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,1}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,1}, \mathbf{u}_k\right) \\ \vdots \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,K}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,K}, \mathbf{u}_k\right) \end{array}\right] = 0$$



Collocation methods - Implementation

Collocation uses the constraints:

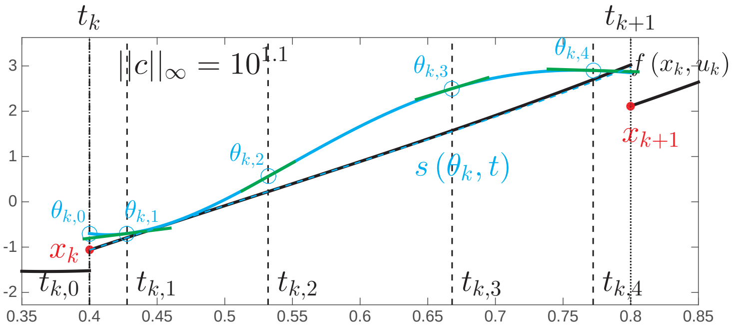
$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \dot{P}_{k,i}(t_{k,j}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k\right)$$

for $j = 1, ..., K$.

Solve for $\boldsymbol{\theta}_{k,i}$ using Newton

$$\mathbf{c} = \begin{bmatrix} \boldsymbol{\theta}_{k,0} - \mathbf{x}_k \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,1}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,1}, \mathbf{u}_k\right) \\ \vdots \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,K}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,K}, \mathbf{u}_k\right) \end{bmatrix} = 0$$



Collocation methods - Implementation

Collocation uses the constraints:

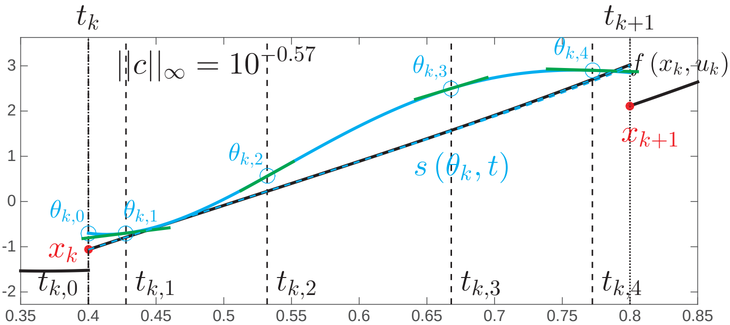
$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \dot{P}_{k,i}(t_{k,j}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k\right)$$

for $j = 1, ..., K$.

Solve for $\boldsymbol{\theta}_{k,i}$ using Newton

$$\mathbf{c} = \begin{bmatrix} \boldsymbol{\theta}_{k,0} - \mathbf{x}_k \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,1}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,1}, \mathbf{u}_k\right) \\ \vdots \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,K}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,K}, \mathbf{u}_k\right) \end{bmatrix} = 0$$



Collocation methods - Implementation

Collocation uses the constraints:

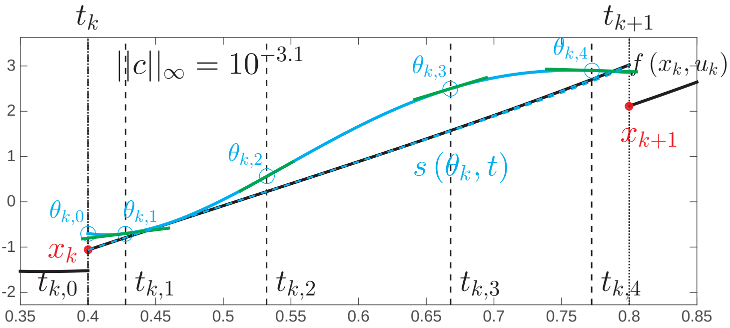
$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \dot{P}_{k,i}(t_{k,j}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k\right)$$

for $j = 1, ..., K$.

Solve for $\boldsymbol{\theta}_{k,i}$ using Newton

$$\mathbf{c} = \begin{bmatrix} \boldsymbol{\theta}_{k,0} - \mathbf{x}_k \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,1}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,1}, \mathbf{u}_k\right) \\ \vdots \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,K}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,K}, \mathbf{u}_k\right) \end{bmatrix} = 0$$



Collocation methods - Implementation

Collocation uses the constraints:

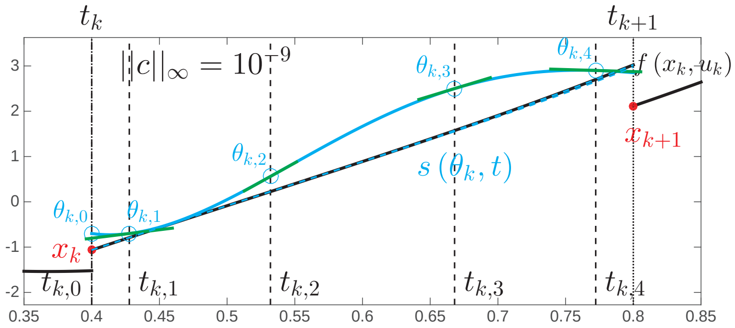
$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \dot{P}_{k,i}(t_{k,j}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k\right)$$

for $j = 1, ..., K$.

Solve for $\boldsymbol{\theta}_{k,i}$ using Newton

$$\mathbf{c} = \begin{bmatrix} \boldsymbol{\theta}_{k,0} - \mathbf{x}_k \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,1}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,1}, \mathbf{u}_k\right) \\ \vdots \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,K}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,K}, \mathbf{u}_k\right) \end{bmatrix} = 0$$



Collocation methods - Implementation

Collocation uses the constraints:

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

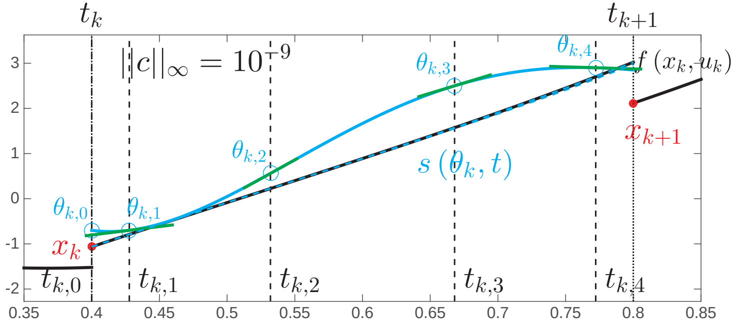
$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \dot{P}_{k,i}(t_{k,j}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k\right)$$

for $j = 1, ..., K$. End-state:

$$\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right) = \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot P_{k,i}(t_{k+1})$$

Solve for $\boldsymbol{\theta}_{k,i}$ using Newton

$$\mathbf{c} = \begin{bmatrix} \boldsymbol{\theta}_{k,0} - \mathbf{x}_k \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,1}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,1}, \mathbf{u}_k\right) \\ \vdots \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,K}) - \mathbf{F}\left(\boldsymbol{\theta}_{k,K}, \mathbf{u}_k\right) \end{bmatrix} = 0$$



Collocation methods - Implementation

Collocation uses the constraints:

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot \dot{P}_{k,i}(t_{k,j}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,j}, \mathbf{u}_k\right)$$

for $j = 1, ..., K$. End-state:

$$\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right) = \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot P_{k,i}(t_{k+1})$$

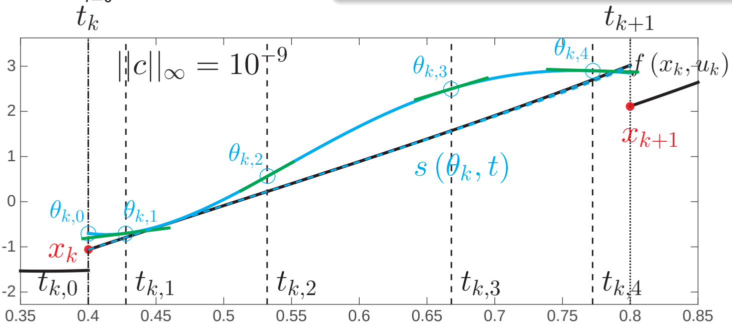
Shooting constraints

End-state reported to the NLP solver:

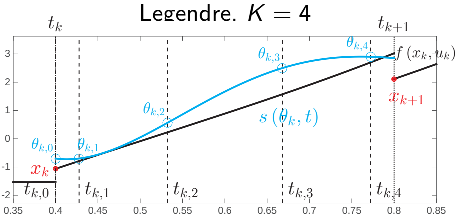
$$\underbrace{\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}_{=\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)} - \mathbf{x}_{k+1} = 0$$

Shooting constraints also reads as:

$$\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} P_{k,i}(t_{k+1}) - \mathbf{x}_{k+1} = 0$$



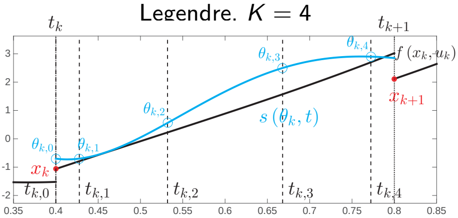
Selection of the time grid $t_{k,i}$



Selection of the time grid $t_{k,i}$

Collocation points on $[0, 1]$:

K	Legendre	Radau
1	0.5	1.0
2	0.211325 0.788675	0.333333 1.000000
3	0.112702 0.500000 0.887298	0.155051 0.644949 1.000000
4	0.069432 0.330009 0.669991 0.930568	0.088588 0.409467 0.787659 1.000000
5	0.046910 0.230765 0.500000 0.769235 0.953090	0.057104 0.276843 0.583590 0.860240 1.000000

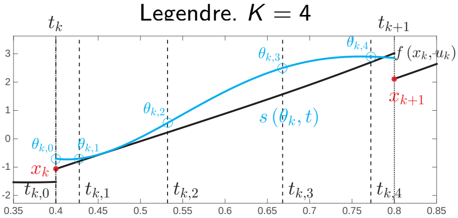


+one point at t_k to enforce continuity !

Selection of the time grid $t_{k,i}$

Collocation points on $[0, 1]$:

K	Legendre	Radau
1	0.5	1.0
2	0.211325	0.333333
	0.788675	1.000000
3	0.112702	0.155051
	0.500000	0.644949
	0.887298	1.000000
4	0.069432	0.088588
	0.330009	0.409467
	0.669991	0.787659
	0.930568	1.000000
5	0.046910	0.057104
	0.230765	0.276843
	0.500000	0.583590
	0.769235	0.860240
	0.953090	1.000000



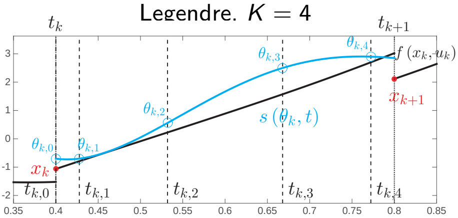
+one point at t_k to enforce continuity !

Selection of the time grid $t_{k,i}$

Collocation points on $[0,1]$:

K	Legendre	Radau
1	0.5	1.0
2	0.211325	0.333333
	0.788675	1.000000
3	0.112702	0.155051
	0.500000	0.644949
	0.887298	1.000000
4	0.069432	0.088588
	0.330009	0.409467
	0.669991	0.787659
	0.930568	1.000000
5	0.046910	0.057104
	0.230765	0.276843
	0.500000	0.583590
	0.769235	0.860240
	0.953090	1.000000

+one point at t_k to enforce continuity !



Why these points ?!? They deliver an exact integration for any polynomial $\mathbf{P}$ of order $< 2K$ (Legendre) and $< 2K - 1$ (Radau). I.e. for

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}) = \mathbf{P}(t)$$

the collocation equations deliver an exact solution, namely:

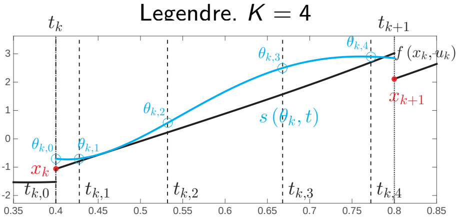
$$\mathbf{s}\left(t_{k+1}, \boldsymbol{\theta}_k\right) = \mathbf{x}_k + \int_{t_k}^{t_{k+1}} \mathbf{P}(\tau)\, \mathrm{d}\tau$$

Selection of the time grid $t_{k,i}$

Collocation points on $[0,1]$:

K	Legendre	Radau
1	0.5	1.0
2	0.211325 0.788675	0.333333 1.000000
3	0.112702 0.500000 0.887298	0.155051 0.644949 1.000000
4	0.069432 0.330009 0.669991 0.930568	0.088588 0.409467 0.787659 1.000000
5	0.046910 0.230765 0.500000 0.769235 0.953090	0.057104 0.276843 0.583590 0.860240 1.000000

+one point at t_k to enforce continuity !



Interval $[t_k,\ t_{k+1}]$??

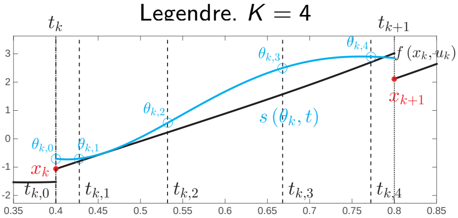
- **Rescale & translate** the collocation points to $[t_k,\ t_{k+1}]$

Selection of the time grid $t_{k,i}$

Collocation points on $[0,1]$:

K	Legendre	Radau
1	0.5	1.0
2	0.211325 0.788675	0.333333 1.000000
3	0.112702 0.500000 0.887298	0.155051 0.644949 1.000000
4	0.069432 0.330009 0.669991 0.930568	0.088588 0.409467 0.787659 1.000000
5	0.046910 0.230765 0.500000 0.769235 0.953090	0.057104 0.276843 0.583590 0.860240 1.000000

+one point at t_k to enforce continuity !



Interval $[t_k,\ t_{k+1}]$??

- **Rescale & translate** the collocation points to $[t_k,\ t_{k+1}]$, or...
- Modification of the collocation equations with $h_k = t_{k+1} - t_k$:

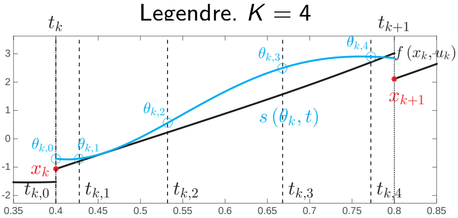
$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = h_k \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

Selection of the time grid $t_{k,i}$

Collocation points on $[0, 1]$:

K	Legendre	Radau
1	0.5	1.0
2	0.211325	0.333333
	0.788675	1.000000
3	0.112702	0.155051
	0.500000	0.644949
	0.887298	1.000000
4	0.069432	0.088588
	0.330009	0.409467
	0.669991	0.787659
	0.930568	1.000000
5	0.046910	0.057104
	0.230765	0.276843
	0.500000	0.583590
	0.769235	0.860240
	0.953090	1.000000

+one point at t_k to enforce continuity !



Interval $[t_k,\ t_{k+1}]$??

- **Rescale & translate** the collocation points to $[t_k,\ t_{k+1}]$, or...
- Modification of the collocation equations with $h_k = t_{k+1} - t_k$:

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = h_k \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

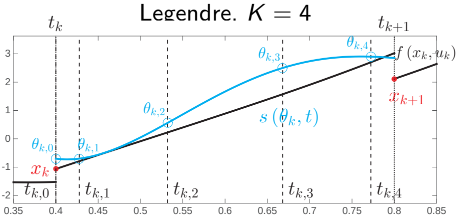
Careful if $\mathbf{F}$ is time-dependent !

Selection of the time grid $t_{k,i}$

Collocation points on $[0, 1]$:

K	Legendre	Radau
1	0.5	1.0
2	0.211325 0.788675	0.333333 1.000000
3	0.112702 0.500000 0.887298	0.155051 0.644949 1.000000
4	0.069432 0.330009 0.669991 0.930568	0.088588 0.409467 0.787659 1.000000
5	0.046910 0.230765 0.500000 0.769235 0.953090	0.057104 0.276843 0.583590 0.860240 1.000000

+one point at t_k to enforce continuity !



Interval $[t_k, \; t_{k+1}]$??

- **Rescale & translate** the collocation points to $[t_k, \; t_{k+1}]$, or...
- Modification of the collocation equations with $h_k = t_{k+1} - t_k$:

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = h_k \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

Careful if $\mathbf{F}$ is time-dependent !

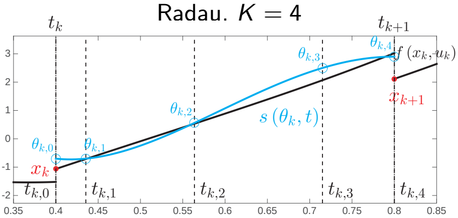
Note that Radau has a collocation point at the end of the interval, i.e. $\boldsymbol{\theta}_{k,K}$ provides the end-state of the integration !

Selection of the time grid $t_{k,i}$

Collocation points on $[0, 1]$:

K	Legendre	Radau
1	0.5	1.0
2	0.211325	0.333333
	0.788675	1.000000
3	0.112702	0.155051
	0.500000	0.644949
	0.887298	1.000000
4	0.069432	0.088588
	0.330009	0.409467
	0.669991	0.787659
	0.930568	1.000000
5	0.046910	0.057104
	0.230765	0.276843
	0.500000	0.583590
	0.769235	0.860240
	0.953090	1.000000

+one point at t_k to enforce continuity !



Interval $[t_k,\ t_{k+1}]$??

- **Rescale & translate** the collocation points to $[t_k,\ t_{k+1}]$, or...
- Modification of the collocation equations with $h_k = t_{k+1} - t_k$:

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = h_k \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

Careful if $\mathbf{F}$ is time-dependent !

Note that Radau has a collocation point at the end of the interval, i.e. $\boldsymbol{\theta}_{k,K}$ provides the end-state of the integration !

Order & Stability

- Collocation methods are **A-stable** (i.e. can handle stiff equations). They have no stability limitation on the time intervals $h = t_{k+1} - t_k$ for stiff problems. I.e. even large time steps $h = t_{k+1} - t_k$ allow for capturing steady state and slow dynamics correctly in the presence of very fast dynamics.

Order & Stability

- Collocation methods are **A-stable** (i.e. can handle stiff equations). They have no stability limitation on the time intervals $h = t_{k+1} - t_k$ for stiff problems. I.e. even large time steps $h = t_{k+1} - t_k$ allow for capturing steady state and slow dynamics correctly in the presence of very fast dynamics.

- Radau collocation is additionally **L-stable**. I.e. it can handle eigenvalues at $-\infty$.

Order & Stability

- Collocation methods are **A-stable** (i.e. can handle stiff equations). They have no stability limitation on the time intervals $h = t_{k+1} - t_k$ for stiff problems. I.e. even large time steps $h = t_{k+1} - t_k$ allow for capturing steady state and slow dynamics correctly in the presence of very fast dynamics.

- Radau collocation is additionally **L-stable**. I.e. it can handle eigenvalues at $-\infty$.

- On an interval $h = t_{k+1} - t_k$, the **integration error** is $O(h^{2K})$ for Legendre and $O(h^{2K-1})$ for Radau. Losing one order is the "price" for having a collocation point at t_{k+1}. Note: RK4 has order $O(h^4)$.

Order & Stability

- Collocation methods are **A-stable** (i.e. can handle stiff equations). They have no stability limitation on the time intervals $h = t_{k+1} - t_k$ for stiff problems. I.e. even large time steps $h = t_{k+1} - t_k$ allow for capturing steady state and slow dynamics correctly in the presence of very fast dynamics.

- Radau collocation is additionally **L-stable**. I.e. it can handle eigenvalues at $-\infty$.

- On an interval $h = t_{k+1} - t_k$, the **integration error** is $O(h^{2K})$ for Legendre and $O(h^{2K-1})$ for Radau. Losing one order is the "price" for having a collocation point at t_{k+1}. Note: RK4 has order $O(h^4)$.

- The integration error applies to the **end-state** of the integrator, but not to the intermediate points !

Order & Stability

- Collocation methods are **A-stable** (i.e. can handle stiff equations). They have no stability limitation on the time intervals $h = t_{k+1} - t_k$ for stiff problems. I.e. even large time steps $h = t_{k+1} - t_k$ allow for capturing steady state and slow dynamics correctly in the presence of very fast dynamics.

- Radau collocation is additionally **L-stable**. I.e. it can handle eigenvalues at $-\infty$.

- On an interval $h = t_{k+1} - t_k$, the **integration error** is $O(h^{2K})$ for Legendre and $O(h^{2K-1})$ for Radau. Losing one order is the "price" for having a collocation point at t_{k+1}. Note: RK4 has order $O(h^4)$.

- The integration error applies to the **end-state** of the integrator, but not to the intermediate points !

- Collocation-based integration is an **Implicit Runge-Kutta** scheme (e.g. an order 1 collocation scheme is implicit Euler !)

Collocation - Sensitivity

Collocation constraints...

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right), \quad i = 1, ..., K$$

Collocation - Sensitivity

Collocation constraints...

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right), \quad i = 1, \ldots, K$$

... can be seen as

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

Collocation - Sensitivity

Collocation constraints...

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right), \quad i = 1, ..., K$$

... can be seen as

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

Solved by iterating:

$$\Delta\boldsymbol{\theta}_k = -\frac{\partial \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)}{\partial \boldsymbol{\theta}_k}^{-1} \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)$$

Collocation - Sensitivity

Collocation constraints...

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right), \quad i = 1, ..., K$$

Integrator function given by:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

... can be seen as

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

Solved by iterating:

$$\Delta\boldsymbol{\theta}_k = -\frac{\partial \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)}{\partial \boldsymbol{\theta}_k}^{-1} \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)$$

Collocation - Sensitivity

Collocation constraints...

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right), \quad i = 1, ..., K$$

Integrator function given by:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

... can be seen as

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

Solved by iterating:

$$\Delta\boldsymbol{\theta}_k = -\frac{\partial \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)}{\partial \boldsymbol{\theta}_k}^{-1} \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)$$

Get sensitivities using:

$$\frac{\partial \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{x}_k}, \qquad \frac{\partial \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{u}_k} = \frac{\partial \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{u}_k}$$

Collocation - Sensitivity

Collocation constraints...

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right), \quad i = 1, ..., K$$

... can be seen as

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

Solved by iterating:

$$\Delta\boldsymbol{\theta}_k = -\frac{\partial \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)}{\partial \boldsymbol{\theta}_k}^{-1} \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)$$

Integrator function given by:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

Get sensitivities using:

$$\frac{\partial \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{x}_k}, \qquad \frac{\partial \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{u}_k} = \frac{\partial \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{u}_k}$$

Implicit function theorem states that

$$\frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{x}_k} + \frac{\partial \mathbf{c}}{\partial \mathbf{x}_k} = 0, \qquad \frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{c}}{\partial \mathbf{u}_k} = 0$$

Collocation - Sensitivity

Collocation constraints...

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right), \quad i = 1, ..., K$$

... can be seen as

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

Solved by iterating:

$$\Delta\boldsymbol{\theta}_k = -\frac{\partial \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)}{\partial \boldsymbol{\theta}_k}^{-1} \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)$$

Integrator function given by:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

Get sensitivities using:

$$\frac{\partial \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{x}_k}, \qquad \frac{\partial \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{u}_k} = \frac{\partial \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{u}_k}$$

Implicit function theorem states that

$$\frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{x}_k} + \frac{\partial \mathbf{c}}{\partial \mathbf{x}_k} = 0, \qquad \frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{c}}{\partial \mathbf{u}_k} = 0$$

Hence:

$$\frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{x}_k} = -\frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k}^{-1} \frac{\partial \mathbf{c}}{\partial \mathbf{x}_k}, \qquad \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{u}_k} = -\frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k}^{-1} \frac{\partial \mathbf{c}}{\partial \mathbf{u}_k}$$

Collocation - Sensitivity

Collocation constraints...

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right), \quad i = 1, ..., K$$

... can be seen as

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

Solved by iterating:

$$\Delta\boldsymbol{\theta}_k = -\frac{\partial \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)}{\partial \boldsymbol{\theta}_k}^{-1} \mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right)$$

Integrator function given by:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

Get sensitivities using:

$$\frac{\partial \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{x}_k}, \qquad \frac{\partial \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)}{\partial \mathbf{u}_k} = \frac{\partial \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{u}_k}$$

Implicit function theorem states that

$$\frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{x}_k} + \frac{\partial \mathbf{c}}{\partial \mathbf{x}_k} = 0, \qquad \frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k} \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{c}}{\partial \mathbf{u}_k} = 0$$

Hence:

$$\frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{x}_k} = -\frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k}^{-1} \frac{\partial \mathbf{c}}{\partial \mathbf{x}_k}, \qquad \frac{\partial \boldsymbol{\theta}_k}{\partial \mathbf{u}_k} = -\frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k}^{-1} \frac{\partial \mathbf{c}}{\partial \mathbf{u}_k}$$

Note that $\frac{\partial \mathbf{c}}{\partial \boldsymbol{\theta}_k}^{-1}$ is computed in the Newton iteration, i.e. it comes for free !!

Outline

Collocation-based integrators in Multiple-shooting

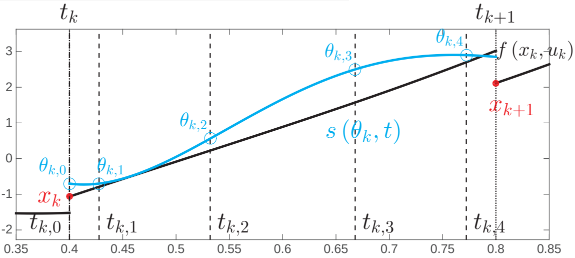
Collocation-based integrator solves:

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

on each time interval $[t_k,\ t_{k+1}]$, provides:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

with sensitivities.



Collocation-based integrators in Multiple-shooting

Collocation-based integrator solves:

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

on each time interval $[t_k,\ t_{k+1}]$, provides:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

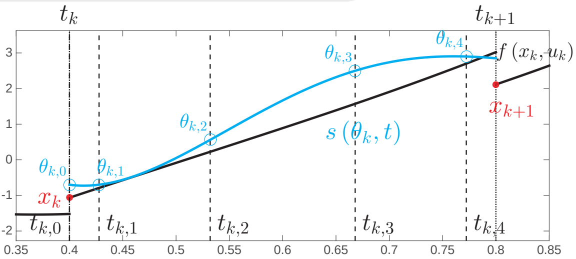
with sensitivities.

NLP with multiple-shooting

$$\min_{\mathbf{w}} \quad \Phi\left(\mathbf{w}\right)$$

$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \begin{bmatrix} \mathbf{x}_0 - \bar{\mathbf{x}}_0 \\ \mathbf{f}\left(\mathbf{x}_0, \mathbf{u}_0\right) - \mathbf{x}_1 \\ \dots \\ \mathbf{f}\left(\mathbf{x}_{N-1}, \mathbf{u}_{N-1}\right) - \mathbf{x}_N \end{bmatrix}$$

where $\mathbf{w} = \left\{\mathbf{x}_0,\ \mathbf{u}_0, \ldots, \mathbf{x}_{N-1},\ \mathbf{u}_{N-1}, \mathbf{x}_N\right\}$



Collocation-based integrators in Multiple-shooting

Collocation-based integrator solves:

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

on each time interval $[t_k,\ t_{k+1}]$, provides:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

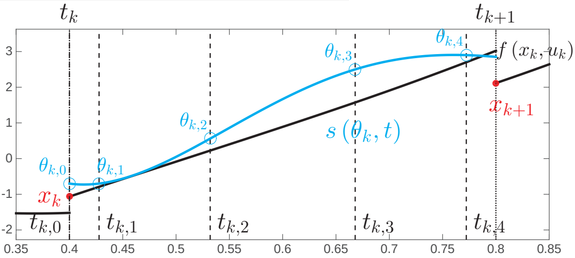
with sensitivities.

NLP with multiple-shooting

$$\min_{\mathbf{w}} \quad \Phi\left(\mathbf{w}\right)$$

$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \begin{bmatrix} \mathbf{x}_0 - \bar{\mathbf{x}}_0 \\ \mathbf{f}\left(\mathbf{x}_0, \mathbf{u}_0\right) - \mathbf{x}_1 \\ \dots \\ \mathbf{f}\left(\mathbf{x}_{N-1}, \mathbf{u}_{N-1}\right) - \mathbf{x}_N \end{bmatrix}$$

where $\mathbf{w} = \left\{\mathbf{x}_0,\ \mathbf{u}_0, \dots, \mathbf{x}_{N-1},\ \mathbf{u}_{N-1}, \mathbf{x}_N\right\}$



NLP:

$$\nabla_{\mathbf{w}} \mathcal{L}\left(\mathbf{w}, \boldsymbol{\lambda}\right) = 0$$

$$\mathbf{g}\left(\mathbf{w}\right) = 0$$

Collocation-based integrators in Multiple-shooting

Collocation-based integrator solves:

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

on each time interval $[t_k,\ t_{k+1}]$, provides:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

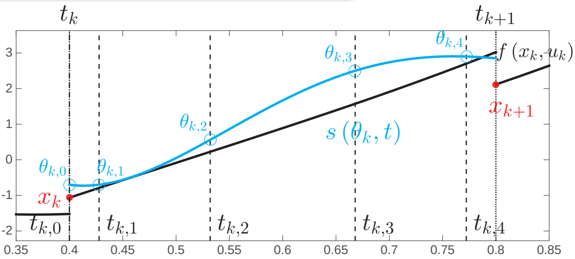
with sensitivities.

NLP with multiple-shooting

$$\min_{\mathbf{w}} \quad \Phi\left(\mathbf{w}\right)$$

$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \begin{bmatrix} \mathbf{x}_0 - \bar{\mathbf{x}}_0 \\ \mathbf{f}\left(\mathbf{x}_0, \mathbf{u}_0\right) - \mathbf{x}_1 \\ \dots \\ \mathbf{f}\left(\mathbf{x}_{N-1}, \mathbf{u}_{N-1}\right) - \mathbf{x}_N \end{bmatrix}$$

where $\mathbf{w} = \left\{\mathbf{x}_0,\ \mathbf{u}_0, \dots, \mathbf{x}_{N-1},\ \mathbf{u}_{N-1}, \mathbf{x}_N\right\}$



NLP:

$$\nabla_{\mathbf{w}}\mathcal{L}\left(\mathbf{w}, \boldsymbol{\lambda}\right) = 0$$
$$\mathbf{g}\left(\mathbf{w}\right) = 0$$

Each $\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)$ and $\nabla\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)$ is provided by the "collocation code"

Collocation-based integrators in Multiple-shooting

Collocation-based integrator solves:

$$\mathbf{c}\left(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k\right) = 0$$

on each time interval $[t_k,\ t_{k+1}]$, provides:

$$\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right)$$

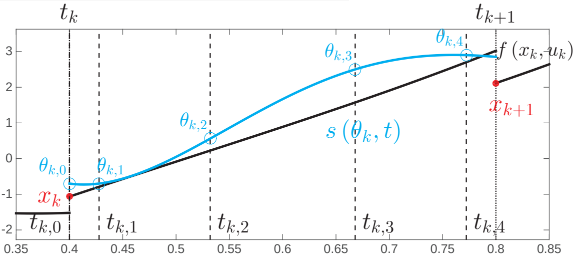
with sensitivities.

NLP with multiple-shooting

$$\min_{\mathbf{w}} \quad \Phi\left(\mathbf{w}\right)$$

$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \begin{bmatrix} \mathbf{x}_0 - \bar{\mathbf{x}}_0 \\ \mathbf{f}\left(\mathbf{x}_0, \mathbf{u}_0\right) - \mathbf{x}_1 \\ \ldots \\ \mathbf{f}\left(\mathbf{x}_{N-1}, \mathbf{u}_{N-1}\right) - \mathbf{x}_N \end{bmatrix}$$

where $\mathbf{w} = \left\{\mathbf{x}_0,\ \mathbf{u}_0, \ldots, \mathbf{x}_{N-1},\ \mathbf{u}_{N-1}, \mathbf{x}_N\right\}$



NLP:

$$\nabla_{\mathbf{w}}\mathcal{L}\left(\mathbf{w}, \boldsymbol{\lambda}\right) = 0$$
$$\mathbf{g}\left(\mathbf{w}\right) = 0$$

Each $\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)$ and $\nabla\mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)$ is provided by the "collocation code"

Collocation-based integrator inside the NLP becomes a two-level Newton scheme !!

Collocation-based integrators in Multiple-shooting (cont')

NLP solver
$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = 0$$
$$\mathbf{g}(\mathbf{w}) = 0$$

$$\mathbf{w} = \{\mathbf{x}_0, \mathbf{u}_0, ..., \mathbf{x}_{N-1}, \mathbf{u}_{N-1}, \mathbf{x}_N\}$$

Collocation-based integrators in Multiple-shooting (cont')

NLP solver

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = 0$$

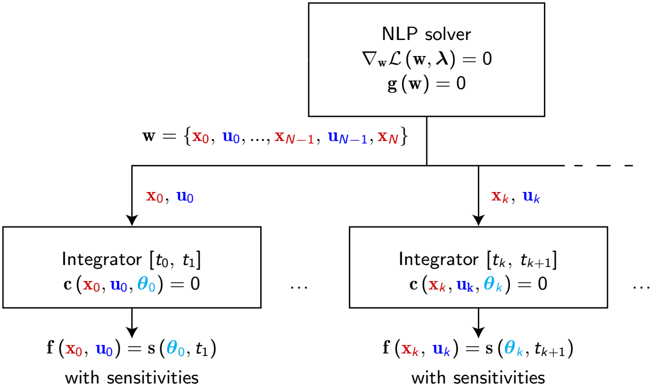
$$\mathbf{g}(\mathbf{w}) = 0$$

$$\mathbf{w} = \{\mathbf{x}_0,\, \mathbf{u}_0, \ldots, \mathbf{x}_{N-1},\, \mathbf{u}_{N-1}, \mathbf{x}_N\}$$

NLP level

- Constraints $\mathbf{g} = 0$
- Newton iterations (SQP/IP)

Collocation-based integrators in Multiple-shooting (cont')



NLP level

- Constraints $\mathbf{g} = 0$
- Newton iterations (SQP/IP)

Collocation-based integrators in Multiple-shooting (cont')

NLP solver
$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = 0$$
$$\mathbf{g}(\mathbf{w}) = 0$$

$$\mathbf{w} = \{\mathbf{x}_0,\ \mathbf{u}_0, \dots, \mathbf{x}_{N-1},\ \mathbf{u}_{N-1}, \mathbf{x}_N\}$$

$\mathbf{x}_0,\ \mathbf{u}_0$

Integrator $[t_0,\ t_1]$
$$\mathbf{c}(\mathbf{x}_0, \mathbf{u}_0, \boldsymbol{\theta}_0) = 0$$

$$\mathbf{f}(\mathbf{x}_0,\ \mathbf{u}_0) = \mathbf{s}(\boldsymbol{\theta}_0, t_1)$$
with sensitivities

...

$\mathbf{x}_k,\ \mathbf{u}_k$

Integrator $[t_k,\ t_{k+1}]$
$$\mathbf{c}(\mathbf{x}_k, \mathbf{u}_k, \boldsymbol{\theta}_k) = 0$$

$$\mathbf{f}(\mathbf{x}_k,\ \mathbf{u}_k) = \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1})$$
with sensitivities

...

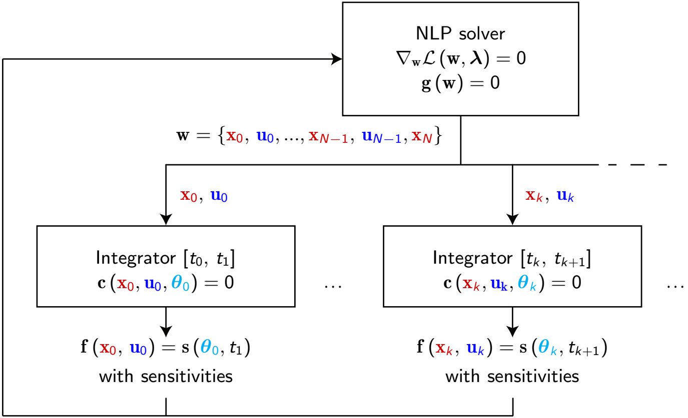
NLP level

- Constraints $\mathbf{g} = 0$
- Newton iterations (SQP/IP)

Integrator level

- Constraints $\mathbf{c} = 0$
- Newton iterations

Collocation-based integrators in Multiple-shooting (cont')



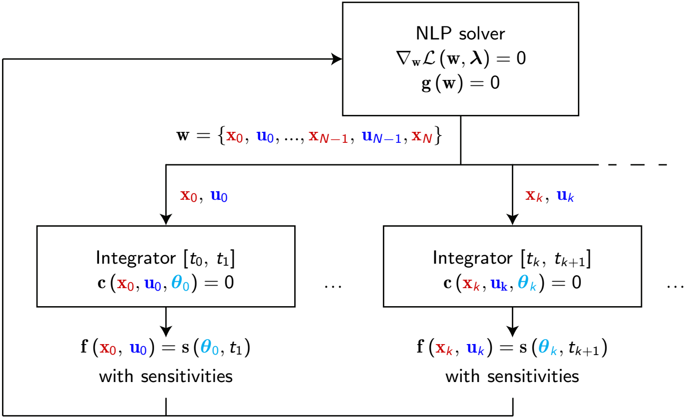
NLP level

- Constraints $\mathbf{g} = 0$
- Newton iterations (SQP/IP)

Integrator level

- Constraints $\mathbf{c} = 0$
- Newton iterations

Collocation-based integrators in Multiple-shooting (cont')



NLP level

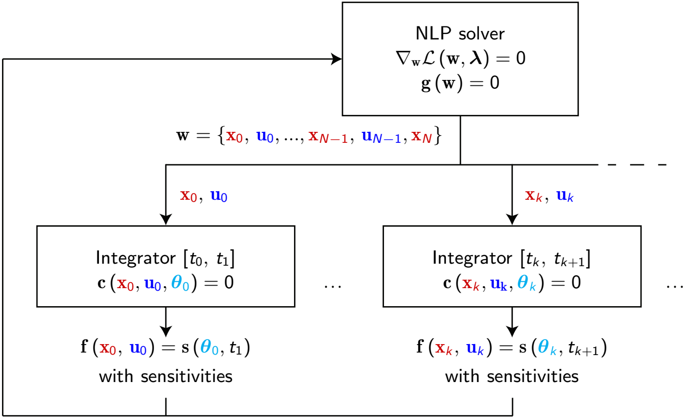
- Constraints $\mathbf{g} = 0$
- Newton iterations (SQP/IP)

Integrator level

- Constraints $\mathbf{c} = 0$
- Newton iterations

Constraints are solved at the NLP and at the integrator level separately !!

Collocation-based integrators in Multiple-shooting (cont')



NLP level

- Constraints $\mathbf{g}=0$
- Newton iterations (SQP/IP)

Integrator level

- Constraints $\mathbf{c}=0$
- Newton iterations

Constraints are solved at the NLP and at the integrator level separately !!

... what about handling them **altogether** in the NLP ?!?

Outline

Direct collocation - Give all constraints to the NLP solver

On each interval $[t_k, t_{k+1}]$

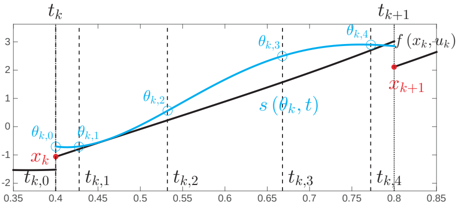
$$\dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, \mathbf{u}_k\right)$$

is approximated using:

$$\mathrm{s}\left(\boldsymbol{\theta}_k, t\right) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

Note:

- $\mathrm{s}\left(\boldsymbol{\theta}_{k,i}, t_{k,i}\right) = \boldsymbol{\theta}_{k,i}$
- $K+1$ degrees of freedom per state.



Direct collocation - Give all constraints to the NLP solver

On each interval $[t_k, t_{k+1}]$

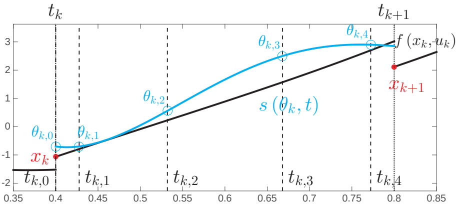
$$\dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, \mathbf{u}_k\right)$$

is approximated using:

$$\mathbf{s}\left(\boldsymbol{\theta}_k, t\right) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

Note:

- $\mathbf{s}\left(\boldsymbol{\theta}_{k,i}, t_{k,i}\right) = \boldsymbol{\theta}_{k,i}$
- $K+1$ degrees of freedom per state.



Integration constraints $(i = 1, ..., K)$

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k,i}\right) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

i.e.

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

Direct collocation - Give all constraints to the NLP solver

On each interval $[t_k, t_{k+1}]$

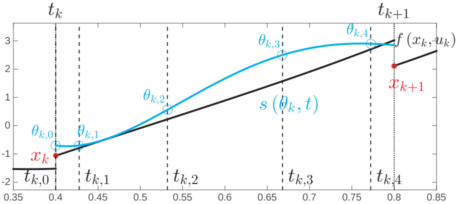
$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}_k)$$

is approximated using:

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

Note:

- $\mathbf{s}(\boldsymbol{\theta}_{k,i}, t_{k,i}) = \boldsymbol{\theta}_{k,i}$
- $K+1$ degrees of freedom per state.



NLP with direct collocation

$$\min_{\mathbf{w}} \quad \Phi(\mathbf{w})$$

$$\text{s.t.} \quad \mathbf{g}(\mathbf{w}) = \left[\quad \right]$$

Integration constraints ($i = 1, ..., K$)

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k)$$

i.e.

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k)$$

Direct collocation - Give all constraints to the NLP solver

On each interval $[t_k, t_{k+1}]$

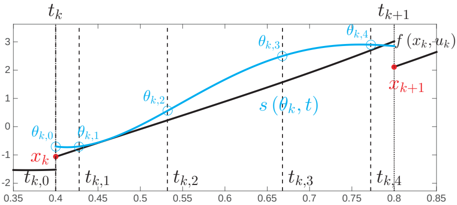
$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}_k)$$

is approximated using:

$$\mathrm{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

Note:

- $\mathrm{s}(\boldsymbol{\theta}_{k,i}, t_{k,i}) = \boldsymbol{\theta}_{k,i}$
- $K+1$ degrees of freedom per state.



NLP with direct collocation

$$\min_{\mathbf{w}} \quad \Phi(\mathbf{w})$$

$$\text{s.t.} \quad \mathbf{g}(\mathbf{w}) = \begin{bmatrix} \boldsymbol{\theta}_{0,0} - \bar{\mathbf{x}}_0 \\ \\ \\ \\ \end{bmatrix}$$

Initial conditions $\bar{\mathbf{x}}_0$

Integration constraints $(i = 1, \dots, K)$

$$\frac{\partial}{\partial t}\mathrm{s}(\boldsymbol{\theta}_k, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k)$$

i.e.

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k)$$

Direct collocation - Give all constraints to the NLP solver

On each interval $[t_k, t_{k+1}]$

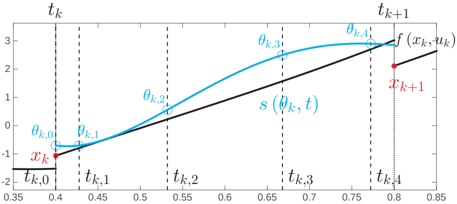
$$\dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, \mathbf{u}_k\right)$$

is approximated using:

$$\mathbf{s}\left(\boldsymbol{\theta}_k, t\right) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

Note:

- $\mathbf{s}\left(\boldsymbol{\theta}_{k,i}, t_{k,i}\right) = \boldsymbol{\theta}_{k,i}$
- $K+1$ degrees of freedom per state.



NLP with direct collocation

$$\min_{\mathbf{w}} \quad \Phi\left(\mathbf{w}\right)$$

$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \begin{bmatrix} \boldsymbol{\theta}_{0,0} - \bar{\mathbf{x}}_0 \\ \mathbf{s}\left(\boldsymbol{\theta}_0, t_1\right) - \boldsymbol{\theta}_{1,0} \\ \\ \\ \\ \end{bmatrix}$$

Continuity constraints ($\equiv$ shooting gaps)

Integration constraints ($i = 1, \ldots, K$)

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k,i}\right) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

i.e.

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

Direct collocation - Give all constraints to the NLP solver

On each interval $[t_k, t_{k+1}]$

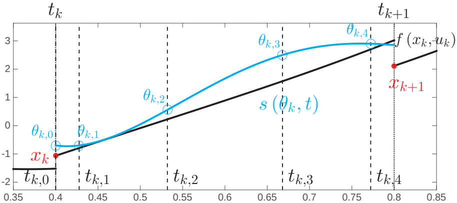
$$\dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, \mathbf{u}_k\right)$$

is approximated using:

$$\mathbf{s}\left(\boldsymbol{\theta}_k, t\right) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

Note:

- $\mathbf{s}\left(\boldsymbol{\theta}_{k,i}, t_{k,i}\right) = \boldsymbol{\theta}_{k,i}$
- $K+1$ degrees of freedom per state.



NLP with direct collocation

$$\min_{\mathbf{w}} \quad \Phi\left(\mathbf{w}\right)$$

$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \begin{bmatrix} \boldsymbol{\theta}_{0,0} - \bar{\mathbf{x}}_0 \\ \mathbf{s}\left(\boldsymbol{\theta}_0, t_1\right) - \boldsymbol{\theta}_{1,0} \\ \mathbf{F}\left(\boldsymbol{\theta}_{0,i}, \mathbf{u}_0\right) - \sum_{j=0}^{K} \boldsymbol{\theta}_{0,j} \dot{P}_{0,j}(t_{0,i}) \\ \\ \\ \\ \end{bmatrix}$$

Integration constraints for $k = 0$

Integration constraints $(i = 1, \ldots, K)$

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k,i}\right) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

i.e.

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

Direct collocation - Give all constraints to the NLP solver

On each interval $[t_k, t_{k+1}]$

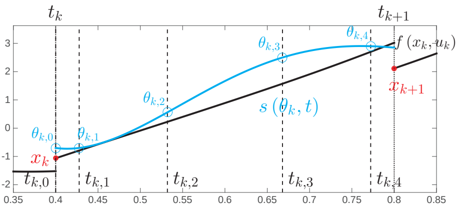
$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, \mathbf{u}_k)$$

is approximated using:

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

Note:

- $\mathbf{s}(\boldsymbol{\theta}_{k,i}, t_{k,i}) = \boldsymbol{\theta}_{k,i}$
- $K+1$ degrees of freedom per state.



NLP with direct collocation

$$\min_{\mathbf{w}} \quad \Phi(\mathbf{w})$$

$$\text{s.t.} \quad \mathbf{g}(\mathbf{w}) = \begin{bmatrix} \boldsymbol{\theta}_{0,0} - \bar{\mathbf{x}}_0 \\ \mathbf{s}(\boldsymbol{\theta}_0, t_1) - \boldsymbol{\theta}_{1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{0,i}, \mathbf{u}_0) - \sum_{j=0}^{K} \boldsymbol{\theta}_{0,j} \dot{P}_{0,j}(t_{0,i}) \\ \dots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \boldsymbol{\theta}_{k+1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \dots \end{bmatrix}$$

Remaining integration constraints $k = 1, ..., N-1$

Integration constraints $(i = 1, ..., K)$

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k)$$

i.e.

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k)$$

Direct collocation - Give all constraints to the NLP solver

On each interval $[t_k, t_{k+1}]$

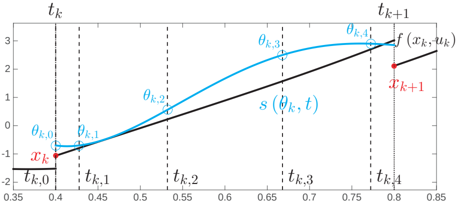
$$\dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, \mathbf{u}_k\right)$$

is approximated using:

$$\mathbf{s}\left(\boldsymbol{\theta}_k, t\right) = \sum_{i=0}^{K} \underbrace{\boldsymbol{\theta}_{k,i}}_{\text{parameters}} \cdot \underbrace{P_{k,i}(t)}_{\text{polynomials}}$$

Note:

- $\mathbf{s}\left(\boldsymbol{\theta}_{k,i}, t_{k,i}\right) = \boldsymbol{\theta}_{k,i}$
- $K+1$ degrees of freedom per state.



NLP with direct collocation

$$\min_{\mathbf{w}} \quad \Phi\left(\mathbf{w}\right)$$

$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \left[\begin{array}{c} \boldsymbol{\theta}_{0,0} - \bar{\mathbf{x}}_0 \\ \mathbf{s}\left(\boldsymbol{\theta}_0, t_1\right) - \boldsymbol{\theta}_{1,0} \\ \mathbf{F}\left(\boldsymbol{\theta}_{0,i}, \mathbf{u}_0\right) - \sum_{j=0}^{K} \boldsymbol{\theta}_{0,j} \dot{P}_{0,j}(t_{0,i}) \\ \dots \\ \mathbf{s}\left(\boldsymbol{\theta}_k, t_{k+1}\right) - \boldsymbol{\theta}_{k+1,0} \\ \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \dots \end{array}\right]$$

Decision variables:

$$\mathbf{w} = \left\{\boldsymbol{\theta}_{0,0}, \dots, \boldsymbol{\theta}_{0,K}, \mathbf{u}_0, \dots, \boldsymbol{\theta}_{N-1,0}, \dots, \boldsymbol{\theta}_{N-1,K}, \mathbf{u}_{N-1}\right\}$$

Integration constraints $(i = 1, \dots, K)$

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k,i}\right) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

i.e.

$$\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

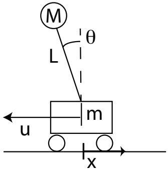
Direct Collocation - Example: swing-up of a pendulum

OCP

$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, u_k\right), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}, \quad \mathbf{x}\left(t_{\mathrm{f}}\right) = 0$$



$$\mathbf{x} = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^\top$$

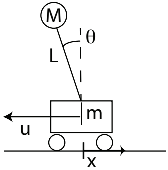
Direct Collocation - Example: swing-up of a pendulum

OCP

$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, u_k), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}, \quad \mathbf{x}(t_f) = 0$$



$$\mathbf{x} = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^\top$$

$N = 20$
$K = 4$ with Legendre, order 8 !!
420 variables
404 constraints

Reminder:

$$\mathbf{s}(\boldsymbol{\theta}_k, t) = \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \cdot P_{k,i}(t)$$

$$\mathbf{s}(\boldsymbol{\theta}_k, t_{k,i}) = \boldsymbol{\theta}_{k,i}$$

NLP with direct collocation

$$\min_{\mathbf{w}} \quad \sum_{k=0}^{N-1} u_k^2$$

$$\text{s.t.} \quad \mathbf{g}(\mathbf{w}) = \begin{bmatrix} \boldsymbol{\theta}_{0,0} - \bar{\mathbf{x}}_0 \\ \mathbf{s}(\boldsymbol{\theta}_0, t_1) - \boldsymbol{\theta}_{1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{0,i}, \mathbf{u}_0) - \sum_{j=0}^{K} \boldsymbol{\theta}_{0,j} \dot{P}_{0,j}(t_{0,i}) \\ \dots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \boldsymbol{\theta}_{k+1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \dots \\ \mathbf{s}(\boldsymbol{\theta}_{N-1}, t_N) \end{bmatrix} = 0$$

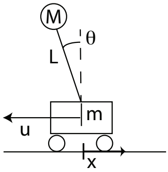
Direct Collocation - Example: swing-up of a pendulum

OCP

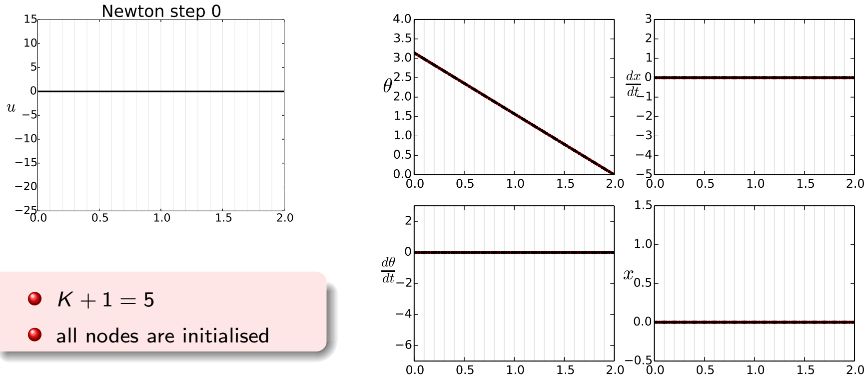
$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, u_k), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}, \quad \mathbf{x}(t_f) = 0$$



$$\mathbf{x} = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^\top$$



- $K + 1 = 5$
- all nodes are initialised

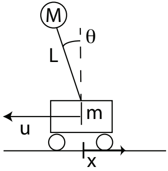
Direct Collocation - Example: swing-up of a pendulum

OCP

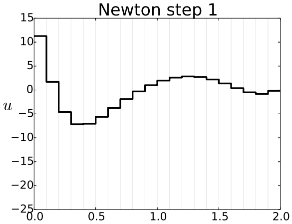
$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

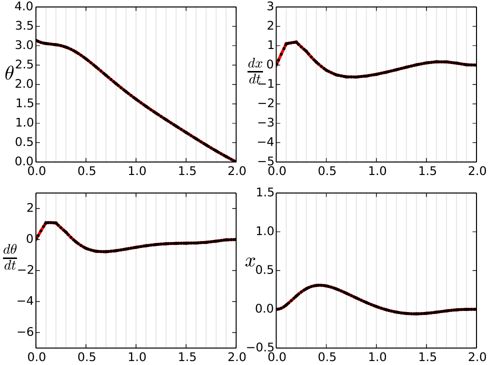
$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, u_k), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}, \quad \mathbf{x}(t_f) = 0$$



$$\mathbf{x} = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^\top$$





- $K + 1 = 5$
- all nodes are initialised

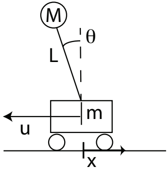
Direct Collocation - Example: swing-up of a pendulum

OCP

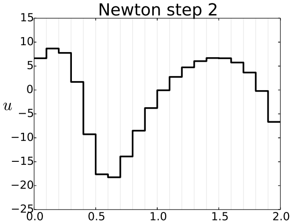
$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

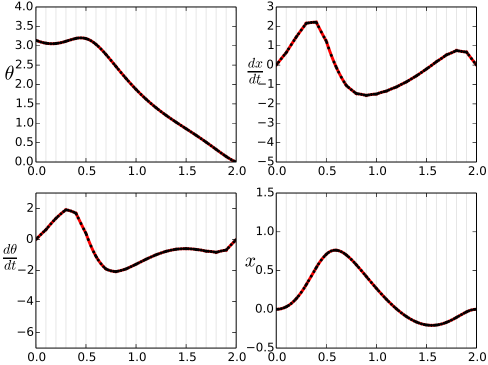
$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, u_k\right), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}, \quad \mathbf{x}\left(t_{\mathrm{f}}\right) = 0$$



$$\mathbf{x} = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^\top$$





- $K + 1 = 5$
- all nodes are initialised

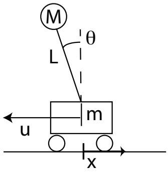
Direct Collocation - Example: swing-up of a pendulum

OCP

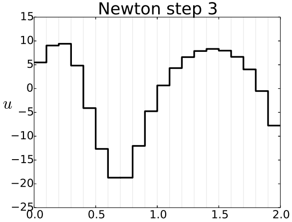
$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

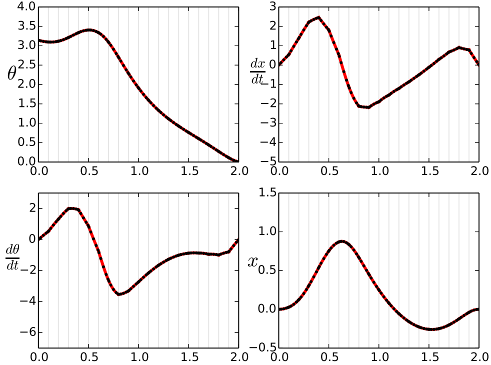
$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, u_k\right), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}, \quad \mathbf{x}\left(t_\mathrm{f}\right) = 0$$



$$\mathbf{x} = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^\top$$





- $K + 1 = 5$
- all nodes are initialised

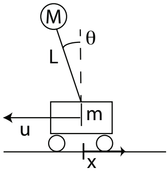
Direct Collocation - Example: swing-up of a pendulum

OCP

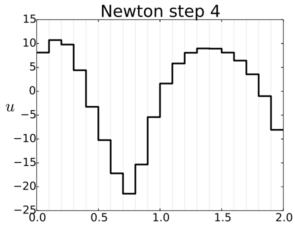
$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

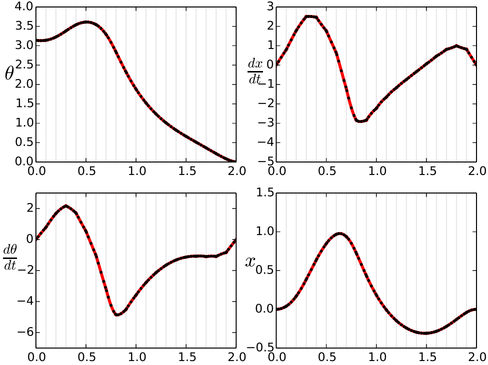
$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, u_k\right), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \left[\begin{array}{cccc} 0 & \pi & 0 & 0 \end{array}\right], \quad \mathbf{x}\left(t_\mathrm{f}\right) = 0$$



$$\mathbf{x} = \left[\begin{array}{cccc} x & \theta & \dot{x} & \dot{\theta} \end{array}\right]^\top$$





- $K + 1 = 5$
- all nodes are initialised

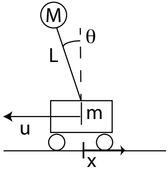
Direct Collocation - Example: swing-up of a pendulum

OCP

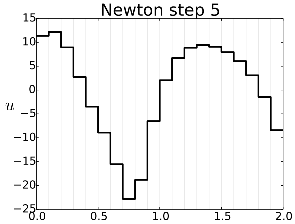
$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

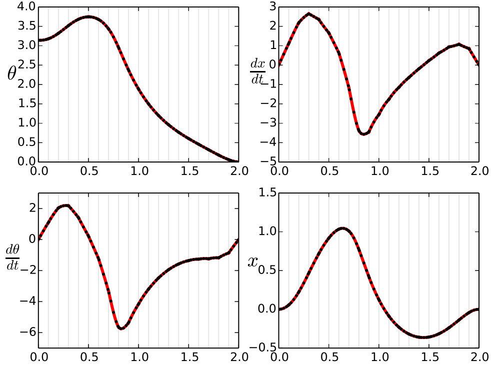
$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, u_k), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \left[\begin{array}{cccc} 0 & \pi & 0 & 0 \end{array}\right], \quad \mathbf{x}(t_f) = 0$$



$$\mathbf{x} = \left[\begin{array}{cccc} x & \theta & \dot{x} & \dot{\theta} \end{array}\right]^\top$$





- $K + 1 = 5$
- all nodes are initialised

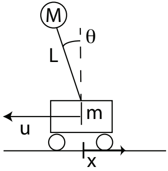
Direct Collocation - Example: swing-up of a pendulum

OCP

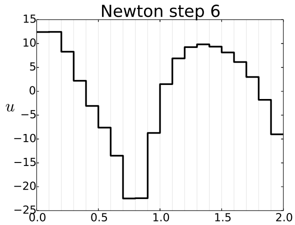
$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

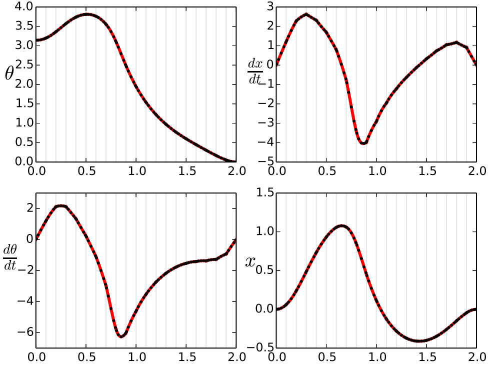
$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, u_k\right), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}, \quad \mathbf{x}\left(t_\mathrm{f}\right) = 0$$



$$\mathbf{x} = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^\top$$





- $K + 1 = 5$
- all nodes are initialised

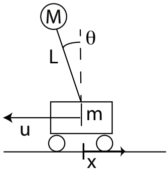
Direct Collocation - Example: swing-up of a pendulum

OCP

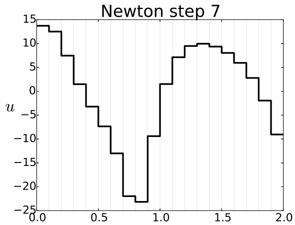
$$\min_{u_0,\ldots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

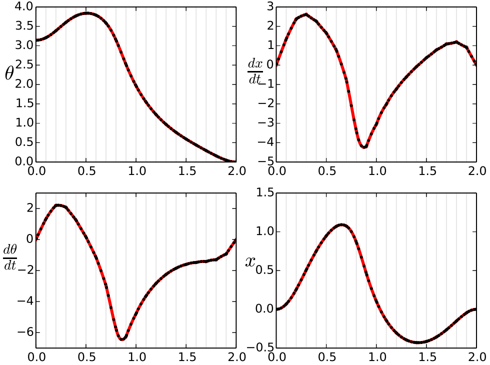
$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}, u_k), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}, \quad \mathbf{x}(t_f) = 0$$



$$\mathbf{x} = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^\top$$





- $K+1=5$
- all nodes are initialised

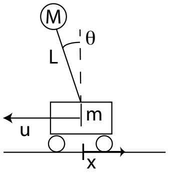
Direct Collocation - Example: swing-up of a pendulum

OCP

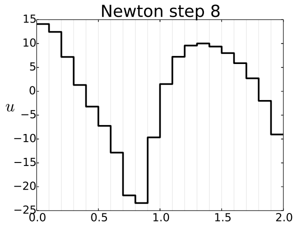
$$\min_{u_0,\dots,u_{N-1}} \quad \sum_{k=0}^{N-1} u_k^2$$

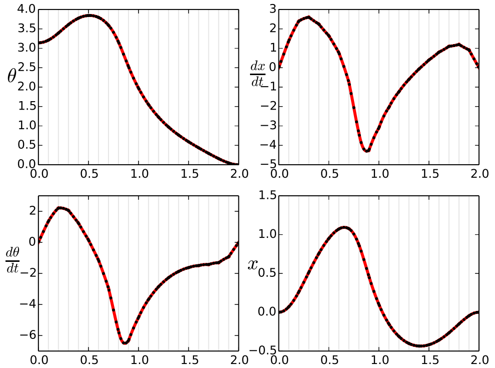
$$\text{s.t.} \quad \dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, u_k\right), \quad \forall t \in [t_k, t_{k+1}]$$

$$\mathbf{x}(0) = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}, \quad \mathbf{x}\left(t_{\mathrm{f}}\right) = 0$$



$$\mathbf{x} = \begin{bmatrix} x & \theta & \dot{x} & \dot{\theta} \end{bmatrix}^\top$$



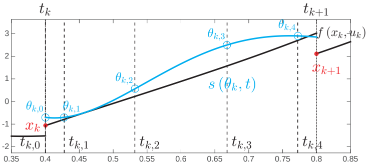


- $K + 1 = 5$
- all nodes are initialised

Cost and constraints discretisation in Direct Collocation

OCP:

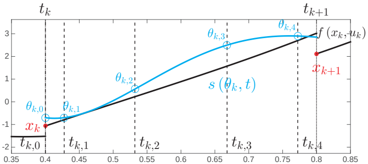
$$\begin{aligned} \min \quad & T\left(\mathbf{x}\left(t_{\mathrm{f}}\right)\right)+\int_{0}^{t_{\mathrm{f}}} L\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right)\right) dt \\ \text{s.t.} \quad & \dot{\mathbf{x}}=\mathbf{F}\left(\mathbf{x},\mathbf{u}\right) \\ & \mathbf{h}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right)\right) \leq 0 \end{aligned}$$



Cost and constraints discretisation in Direct Collocation

OCP:

$$\begin{aligned} \min \quad & T\left(\mathbf{x}\left(t_\mathrm{f}\right)\right) + \int_0^{t_\mathrm{f}} L\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) dt \\ \text{s.t.} \quad & \dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, \mathbf{u}\right) \\ & \mathbf{h}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) \leq 0 \end{aligned}$$



- Inequality constraints: $\mathbf{h}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) \leq 0$ can be enforced on all collocation nodes:

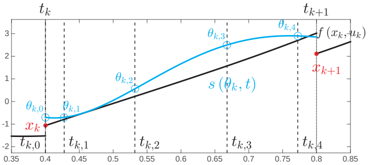
$$\mathbf{h}\left(\boldsymbol{\theta}_k, t_{k,i}, \mathbf{u}_k\right) \leq 0, \quad \forall\, k = 0, \dots, N-1, \quad i = 0, \dots, K$$

but often only on the "shooting" nodes $t_{0,0}$, $t_{1,0}$, ..., $t_{N,0}$

Cost and constraints discretisation in Direct Collocation

OCP:

$$\begin{aligned} \min \quad & T\left(\mathbf{x}\left(t_{\mathrm{f}}\right)\right) + \int_0^{t_{\mathrm{f}}} L\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) dt \\ \text{s.t.} \quad & \dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, \mathbf{u}\right) \\ & \mathbf{h}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) \leq 0 \end{aligned}$$



- Inequality constraints: $\mathbf{h}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) \leq 0$ can be enforced on all collocation nodes:

$$\mathbf{h}\left(\boldsymbol{\theta}_k, t_{k,i}, \mathbf{u}_k\right) \leq 0, \quad \forall\, k = 0, ..., N-1, \quad i = 0, ..., K$$

  but often only on the "shooting" nodes $t_{0,0}$, $t_{1,0}$, ..., $t_{N,0}$

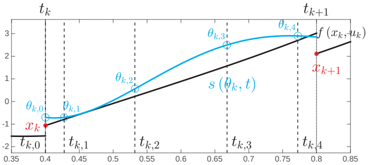
- Cost function often approximated as (rectangular quadrature):

$$T\left(\mathbf{x}\left(\boldsymbol{\theta}_{N-1}, t_{N-1,K}\right)\right) + \sum_{k=0}^{N-1} \left(t_{k+1} - t_k\right) L\left(\boldsymbol{\theta}_{k,0}, \mathbf{u}_k\right)$$

Cost and constraints discretisation in Direct Collocation

OCP:

$$\begin{aligned}\min \quad & T\left(\mathbf{x}\left(t_{\mathrm{f}}\right)\right)+\int_0^{t_{\mathrm{f}}} L\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right)\right) dt \\ \text{s.t.} \quad & \dot{\mathbf{x}}=\mathbf{F}\left(\mathbf{x},\mathbf{u}\right) \\ & \mathbf{h}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right)\right)\leq 0\end{aligned}$$



- Inequality constraints: $\mathbf{h}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right)\right)\leq 0$ can be enforced on all collocation nodes:

$$\mathbf{h}\left(\boldsymbol{\theta}_k, t_{k,i}, \mathbf{u}_k\right)\leq 0, \quad \forall\, k=0,\dots,N-1, \quad i=0,\dots,K$$

  but often only on the "shooting" nodes $t_{0,0}$, $t_{1,0}$, ..., $t_{N,0}$

- Cost function often approximated as (rectangular quadrature):

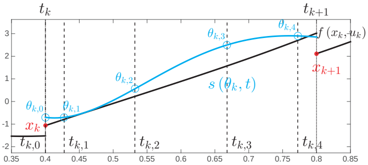
$$T\left(\mathbf{x}\left(\boldsymbol{\theta}_{N-1}, t_{N-1,K}\right)\right)+\sum_{k=0}^{N-1}\left(t_{k+1}-t_k\right) L\left(\boldsymbol{\theta}_{k,0}, \mathbf{u}_k\right)$$

Careful: if you want to use $\boldsymbol{\theta}_{k,i}$ for $i=1,\dots,K$, the time grid is not uniform !!

Cost and constraints discretisation in Direct Collocation

OCP:

$$\begin{aligned} \min \quad & T\left(\mathbf{x}\left(t_\mathrm{f}\right)\right) + \int_0^{t_\mathrm{f}} L\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) dt \\ \text{s.t.} \quad & \dot{\mathbf{x}} = \mathbf{F}\left(\mathbf{x}, \mathbf{u}\right) \\ & \mathbf{h}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) \leq 0 \end{aligned}$$



- Inequality constraints: $\mathbf{h}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) \leq 0$ can be enforced on all collocation nodes:

$$\mathbf{h}\left(\boldsymbol{\theta}_k, t_{k,i}, \mathbf{u}_k\right) \leq 0, \quad \forall\, k = 0, ..., N-1, \quad i = 0, ..., K$$

but often only on the "shooting" nodes $t_{0,0}$, $t_{1,0}$, ..., $t_{N,0}$

- Cost function often approximated as (rectangular quadrature):

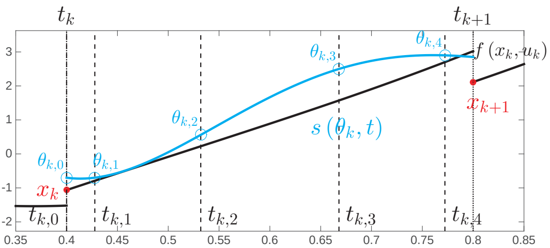
$$T\left(\mathbf{x}\left(\boldsymbol{\theta}_{N-1}, t_{N-1,K}\right)\right) + \sum_{k=0}^{N-1} \left(t_{k+1} - t_k\right) L\left(\boldsymbol{\theta}_{k,0}, \mathbf{u}_k\right)$$

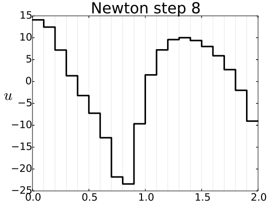
Careful: if you want to use $\boldsymbol{\theta}_{k,i}$ for $i = 1, ..., K$, the time grid is not uniform !!

- Quadratic term in cost function $L\left(\mathbf{x}, \mathbf{u}\right) = \frac{1}{2}\mathbf{x}^\top Q\mathbf{x} + ...$ can be implemented using:

$$\int_{t_k}^{t_{k+1}} \frac{1}{2}\mathbf{x}\left(t\right)^\top Q\mathbf{x}\left(t\right)\,\mathrm{d}t = \frac{1}{2}\sum_{l=0}^{K}\sum_{j=0}^{K} \boldsymbol{\theta}_{k,l}{}^\top Q\boldsymbol{\theta}_{k,j} \underbrace{\int_{t_k}^{t_{k+1}} P_{k,l}(t)P_{k,j}(t)dt}_{=\alpha_j\delta_{l,j}\ (P\text{:s are orthogonal})} = \frac{1}{2}\sum_{j=0}^{K} \alpha_j \boldsymbol{\theta}_{k,j}{}^\top Q\boldsymbol{\theta}_{k,j}$$

Some remarks



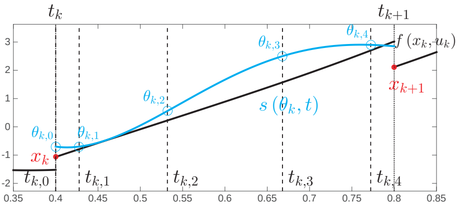


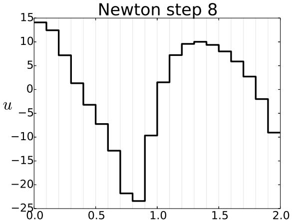
- Direct collocation is a "fully simultaneous" approach, as the **integration and the optimization are performed together** in the NLP solver.
- The decision variables are:

$$\mathbf{w} = \{\boldsymbol{\theta}_{0,0}, ..., \boldsymbol{\theta}_{0,K}, \mathbf{u}_0, ..., \boldsymbol{\theta}_{N-1,0}, ..., \boldsymbol{\theta}_{N-1,K}, \mathbf{u}_{N-1}\}$$

Observe that $\boldsymbol{\theta}_{k,i}$, i.e. the state at the collocation point $t_{k,i}$ of the interval $[t_k, \ t_{k+1}]$ is in $\mathbb{R}^n$ (size of the state). Manipulating these variables properly in a computer code can be tricky.

Refining the input discretization





- Input $\mathbf{u}(t)$ is usually chosen piecewise-constant, i.e. constant in every $[t_k,\ t_{k+1}]$

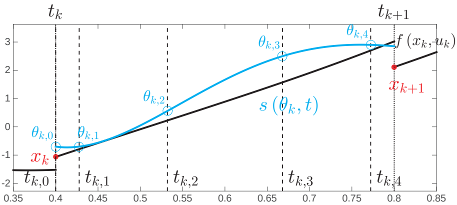
Collocation constraints:

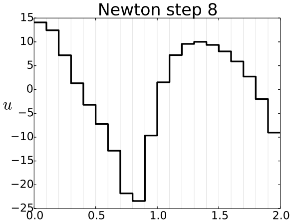
$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k,i}\right) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

for $i = 1, \ldots, K$

Refining the input discretization





- Input $\mathbf{u}(t)$ is usually chosen piecewise-constant, i.e. constant in every $[t_k,\ t_{k+1}]$
- However one can pick a different input $\mathbf{u}_{k,i}$ for each collocation time $t_{k,i}$. Gives K input vector per collocation interval, i.e. $\mathbf{u}_{k,1}, ..., \mathbf{u}_{k,K}$

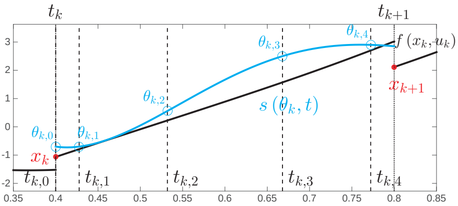
Collocation constraints:

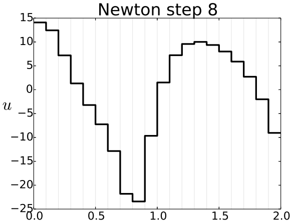
$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k,i}\right) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k\right)$$

for $i = 1, ..., K$

Refining the input discretization





- Input $\mathbf{u}(t)$ is usually chosen piecewise-constant, i.e. constant in every $[t_k, \ t_{k+1}]$
- However one can pick a different input $\mathbf{u}_{k,i}$ for each collocation time $t_{k,i}$. Gives K input vector per collocation interval, i.e. $\mathbf{u}_{k,1}, ..., \mathbf{u}_{k,K}$

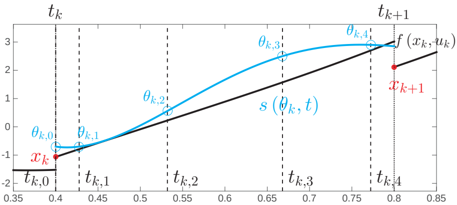
Collocation constraints:

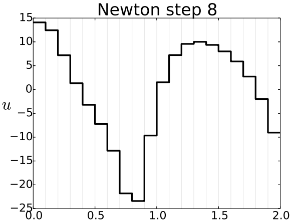
$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k,i})$$

for $i = 1, ..., K$

Refining the input discretization





- Input $\mathbf{u}(t)$ is usually chosen piecewise-constant, i.e. constant in every $[t_k, \ t_{k+1}]$
- However one can pick a different input $\mathbf{u}_{k,i}$ for each collocation time $t_{k,i}$. Gives K input vector per collocation interval, i.e. $\mathbf{u}_{k,1}, ..., \mathbf{u}_{k,K}$
- The continuous input is then given by the $K-1^{\text{th}}$ order polynomial interpolation of $\mathbf{u}_{k,1}, \ ..., \ \mathbf{u}_{k,K}$

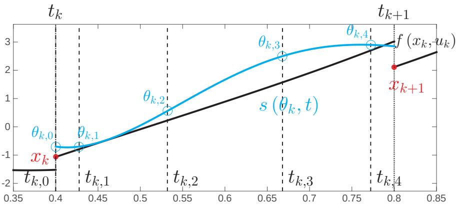
Collocation constraints:

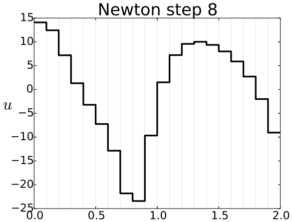
$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k,i}\right) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k,i}\right)$$

for $i = 1, ..., K$

Refining the input discretization





- Input $\mathbf{u}(t)$ is usually chosen piecewise-constant, i.e. constant in every $[t_k,\ t_{k+1}]$
- However one can pick a different input $\mathbf{u}_{k,i}$ for each collocation time $t_{k,i}$. Gives K input vector per collocation interval, i.e. $\mathbf{u}_{k,1}, ..., \mathbf{u}_{k,K}$
- The continuous input is then given by the $K-1^{\text{th}}$ order polynomial interpolation of $\mathbf{u}_{k,1},\ ...,\ \mathbf{u}_{k,K}$
- Drawbacks: **1.** the input profile can present important "oscillations", **2.** the linear algebra tends to loose conditioning

Collocation constraints:

$$\boldsymbol{\theta}_{k,0} = \mathbf{x}_k$$

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_k, t_{k,i}\right) = \mathbf{F}\left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k,i}\right)$$

for $i = 1, ..., K$

Outline

1. Polynomial interpolation
2. Collocation-based integration
3. Collocation in multiple-shooting
4. Direct Collocation
5. NLP from direct collocation

Hessian in Direct Collocation

Lagrange function:

$$\mathcal{L}\left(\mathbf{w}, \boldsymbol{\lambda}\right) = \Phi\left(\mathbf{w}\right) + \boldsymbol{\lambda}^{\top}\mathbf{g}\left(\mathbf{w}\right) + \boldsymbol{\mu}^{\top}\mathbf{h}\left(\mathbf{w}\right)$$

Hessian in Direct Collocation

Lagrange function:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \Phi(\mathbf{w}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{w}) + \boldsymbol{\mu}^\top \mathbf{h}(\mathbf{w})$$

Hessian:

$$\nabla^2_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \nabla^2 \Phi + \nabla^2_{\mathbf{w}} \left(\boldsymbol{\lambda}^\top \mathbf{g}\right) + \nabla^2_{\mathbf{w}} \left(\boldsymbol{\mu}^\top \mathbf{h}\right)$$

Hessian in Direct Collocation

Lagrange function:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \Phi(\mathbf{w}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{w}) + \boldsymbol{\mu}^\top \mathbf{h}(\mathbf{w})$$

Hessian:

$$\nabla^2_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \nabla^2 \Phi + \nabla^2_{\mathbf{w}} \left(\boldsymbol{\lambda}^\top \mathbf{g}\right) + \nabla^2_{\mathbf{w}} \left(\boldsymbol{\mu}^\top \mathbf{h}\right)$$

Reminder: dynamics yield

$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \boldsymbol{\theta}_{0,0} - \bar{\mathbf{x}}_0 \\ \mathbf{s}(\boldsymbol{\theta}_0, t_1) - \boldsymbol{\theta}_{1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{0,i}, \mathbf{u}_0) - \sum_{j=0}^{K} \boldsymbol{\theta}_{0,j} \dot{P}_{0,j}(t_{0,i}) \\ \dots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \boldsymbol{\theta}_{k+1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \dots \end{bmatrix}$$

Hessian in Direct Collocation

Lagrange function:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \Phi(\mathbf{w}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{w}) + \boldsymbol{\mu}^\top \mathbf{h}(\mathbf{w})$$

Hessian:

$$\nabla^2_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \nabla^2 \Phi + \nabla^2_{\mathbf{w}} \left(\boldsymbol{\lambda}^\top \mathbf{g}\right) + \nabla^2_{\mathbf{w}} \left(\boldsymbol{\mu}^\top \mathbf{h}\right)$$

Reminder: dynamics yield

$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \boldsymbol{\theta}_{0,0} - \bar{\mathbf{x}}_0 \\ \mathbf{s}(\boldsymbol{\theta}_0, t_1) - \boldsymbol{\theta}_{1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{0,i}, \mathbf{u}_0) - \sum_{j=0}^{K} \boldsymbol{\theta}_{0,j} \dot{P}_{0,j}(t_{0,i}) \\ \dots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \boldsymbol{\theta}_{k+1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \dots \end{bmatrix}$$

Nonlinear contributions of the dynamics:

$$\nabla^2_{\mathbf{w}} \left(\boldsymbol{\lambda}^\top \mathbf{g}\right) = \nabla^2_{\mathbf{w}} \left[\sum_{k=0,\dots,N-1} \sum_{i=1,..,K} \boldsymbol{\lambda}_{k,i}^\top \left(\mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \right) \right]$$

Hessian in Direct Collocation

Lagrange function:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \Phi(\mathbf{w}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{w}) + \boldsymbol{\mu}^\top \mathbf{h}(\mathbf{w})$$

Hessian:

$$\nabla^2_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \nabla^2 \Phi + \nabla^2_{\mathbf{w}} \left(\boldsymbol{\lambda}^\top \mathbf{g}\right) + \nabla^2_{\mathbf{w}} \left(\boldsymbol{\mu}^\top \mathbf{h}\right)$$

Reminder: dynamics yield

$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \boldsymbol{\theta}_{0,0} - \bar{\mathbf{x}}_0 \\ \mathbf{s}(\boldsymbol{\theta}_0, t_1) - \boldsymbol{\theta}_{1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{0,i}, \mathbf{u}_0) - \sum_{j=0}^{K} \boldsymbol{\theta}_{0,j} \dot{P}_{0,j}(t_{0,i}) \\ \dots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \boldsymbol{\theta}_{k+1,0} \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \dots \end{bmatrix}$$

Nonlinear contributions of the dynamics:

$$\nabla^2_{\mathbf{w}} \left(\boldsymbol{\lambda}^\top \mathbf{g}\right) = \nabla^2_{\mathbf{w}} \left[\sum_{k=0,\dots,N-1} \sum_{i=1,..,K} \boldsymbol{\lambda}_{k,i}^\top \left(\mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i})\right)\right]$$

$$= \sum_{k=0,\dots,N-1} \sum_{i=1,..,K} \nabla^2_{\mathbf{w}} \left(\boldsymbol{\lambda}_{k,i}^\top \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k)\right)$$

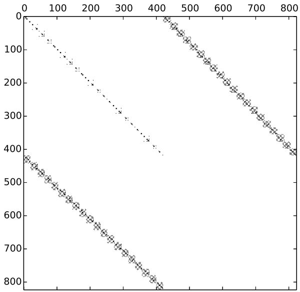
Function $\mathbf{F}$ is simply your ODE. With $\mathbf{w} = \{\boldsymbol{\theta}_{0,0}, \dots, \boldsymbol{\theta}_{0,K}, \mathbf{u}_0, \dots, \boldsymbol{\theta}_{N-1,0}, \dots, \boldsymbol{\theta}_{N-1,K}, \mathbf{u}_{N-1}\}$, the contributions

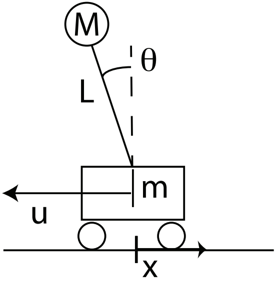
$$\nabla^2_{\mathbf{w}} \left(\boldsymbol{\lambda}_{k,i}^\top \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k)\right)$$

are very sparse and trivial to compute !! (e.g. CasADi)

Sparsity pattern

For the pendulum, KKT matrix M is:



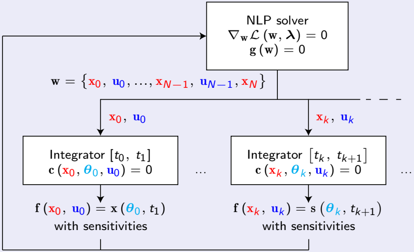


$$M = \begin{bmatrix} H & \nabla \mathbf{g} \\ \nabla \mathbf{g}^\top & 0 \end{bmatrix}$$

- Direct collocation yields **very large but very sparse** NLPs. Typically not a problem for dedicated NLP solvers (e.g. ipopt)
- Exact Hessian is inexpensive to build and compute, unlike in multiple-shooting

Multiple-shooting vs. Direct Collocation

NLP with multiple-shooting & collocation integrators



NLP with direct collocation

$$\begin{aligned}\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\boldsymbol{\lambda}) &= 0\\ \mathbf{g}(\mathbf{w}) &= 0\end{aligned}$$

where

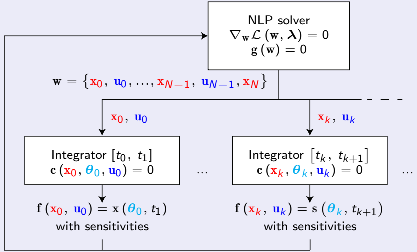
$$\mathbf{w} = \{\boldsymbol{\theta}_{0,0}, \dots, \boldsymbol{\theta}_{0,K}, \mathbf{u}_0, \dots, \boldsymbol{\theta}_{N-1,0}, \dots, \boldsymbol{\theta}_{N-1,K}, \mathbf{u}_{N-1}\}$$

and

$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \mathbf{s}(\boldsymbol{\theta}_0, t_0) - \bar{\mathbf{x}}_0 \\ \dots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \mathbf{s}(\boldsymbol{\theta}_{k+1}, t_{k+1}) \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \dots \end{bmatrix} = 0$$

Multiple-shooting vs. Direct Collocation

NLP with multiple-shooting & collocation integrators



NLP with direct collocation

$$
\begin{aligned}
\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) &= 0 \\
\mathbf{g}(\mathbf{w}) &= 0
\end{aligned}
$$

where

$$
\mathbf{w} = \{\boldsymbol{\theta}_{0,0}, \ldots, \boldsymbol{\theta}_{0,K}, \mathbf{u}_0, \ldots, \boldsymbol{\theta}_{N-1,0}, \ldots, \boldsymbol{\theta}_{N-1,K}, \mathbf{u}_{N-1}\}
$$

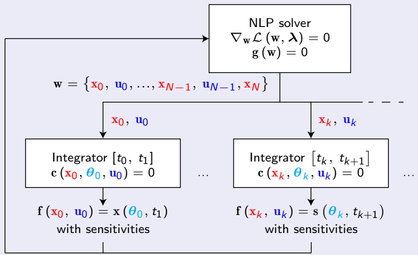
and

$$
\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \mathbf{s}(\boldsymbol{\theta}_0, t_0) - \bar{\mathbf{x}}_0 \\ \ldots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \mathbf{s}(\boldsymbol{\theta}_{k+1}, t_{k+1}) \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \ldots \end{bmatrix} = 0
$$

- NLP has $n_{\mathrm{x}}(N+1) + n_{\mathrm{u}}N$ variables
- N integrators with $n_{\mathrm{x}}(K+1)$ variables, **parallelizable**

Multiple-shooting vs. Direct Collocation

NLP with multiple-shooting & collocation integrators



- NLP has $n_{\mathrm{x}}(N+1) + n_{\mathrm{u}}N$ variables
- N integrators with $n_{\mathrm{x}}(K+1)$ variables, **parallelizable**

NLP with direct collocation

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\boldsymbol{\lambda}) = 0$$
$$\mathbf{g}(\mathbf{w}) = 0$$

where

$$\mathbf{w} = \left\{\boldsymbol{\theta}_{0,0}, \ldots, \boldsymbol{\theta}_{0,K},\, \mathbf{u}_0, \ldots, \boldsymbol{\theta}_{N-1,0}, \ldots, \boldsymbol{\theta}_{N-1,K},\, \mathbf{u}_{N-1}\right\}$$

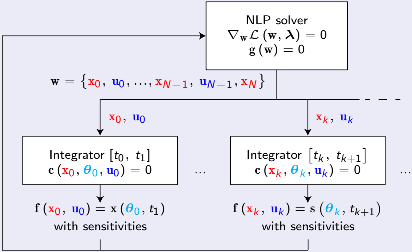
and

$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \mathbf{s}(\boldsymbol{\theta}_0, t_0) - \bar{\mathbf{x}}_0 \\ \ldots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \mathbf{s}(\boldsymbol{\theta}_{k+1}, t_{k+1}) \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \ldots \end{bmatrix} = 0$$

- NLP has $n_{\mathrm{x}}N(K+1) + n_{\mathrm{u}}N$ variables
- **Integration** performed in the NLP, converges together with optimization

Multiple-shooting vs. Direct Collocation

NLP with multiple-shooting & collocation integrators



NLP with direct collocation

$$
\begin{aligned}
\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) &= 0 \\
\mathbf{g}(\mathbf{w}) &= 0
\end{aligned}
$$

where

$$\mathbf{w} = \{\boldsymbol{\theta}_{0,0}, \ldots, \boldsymbol{\theta}_{0,K}, \mathbf{u}_0, \ldots, \boldsymbol{\theta}_{N-1,0}, \ldots, \boldsymbol{\theta}_{N-1,K}, \mathbf{u}_{N-1}\}$$

and

$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \mathbf{s}(\boldsymbol{\theta}_0, t_0) - \bar{\mathbf{x}}_0 \\ \ldots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \mathbf{s}(\boldsymbol{\theta}_{k+1}, t_{k+1}) \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \ldots \end{bmatrix} = 0$$

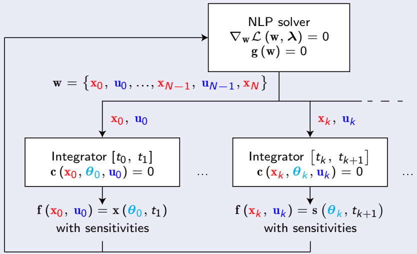
- NLP has $n_{\mathrm{x}}(N+1) + n_{\mathrm{u}}N$ variables
- N integrators with $n_{\mathrm{x}}(K+1)$ variables, **parallelizable**

- NLP has $n_{\mathrm{x}}N(K+1) + n_{\mathrm{u}}N$ variables
- **Integration** performed in the NLP, converges together with optimization

"Multiple-shooting + collocation integrators" does not converge as efficiently as direct collocation...

Multiple-shooting vs. Direct Collocation

NLP with multiple-shooting & collocation integrators



- NLP has $n_{\mathrm{x}}(N+1) + n_{\mathrm{u}}N$ variables
- N integrators with $n_{\mathrm{x}}(K+1)$ variables, **parallelizable**

NLP with direct collocation

$$\begin{aligned} \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w},\boldsymbol{\lambda}) &= 0 \\ \mathbf{g}(\mathbf{w}) &= 0 \end{aligned}$$

where

$$\mathbf{w} = \{\boldsymbol{\theta}_{0,0}, \ldots, \boldsymbol{\theta}_{0,K},\, \mathbf{u}_0, \ldots, \boldsymbol{\theta}_{N-1,0}, \ldots, \boldsymbol{\theta}_{N-1,K},\, \mathbf{u}_{N-1}\}$$

and

$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \mathbf{s}(\boldsymbol{\theta}_0,\, t_0) - \bar{\mathbf{x}}_0 \\ \ldots \\ \mathbf{s}(\boldsymbol{\theta}_k,\, t_{k+1}) - \mathbf{s}(\boldsymbol{\theta}_{k+1},\, t_{k+1}) \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t_{k,i}) \\ \ldots \end{bmatrix} = 0$$

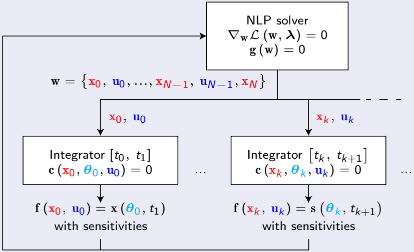
- NLP has $n_{\mathrm{x}}N(K+1) + n_{\mathrm{u}}N$ variables
- **Integration** performed in the NLP, converges together with optimization

"Multiple-shooting + collocation integrators" does not converge as efficiently as direct collocation... Unless:

Lifted implicit integrators for direct optimal control, R. Quirynen, S. Gros, M. Diehl, NMPC Workshop 2015
Lifted implicit integrators for NMPC based on Multiple Shooting, R. Quirynen, S. Gros, M. Diehl, CDC 2015
Lifted Collocation Integrators for Direct Optimal Control in ACADO Toolkit, R. Quirynen, S. Gros, B. Houska, M. Diehl, Journal of Math. Prog. Comp. 2017

Multiple-shooting vs. Direct Collocation

NLP with multiple-shooting & collocation integrators



- NLP has $n_{\mathrm{x}}(N+1) + n_{\mathrm{u}}N$ variables
- N integrators with $n_{\mathrm{x}}(K+1)$ variables, **parallelizable**

NLP with direct collocation

$$\begin{aligned} \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) &= 0 \\ \mathbf{g}(\mathbf{w}) &= 0 \end{aligned}$$

where

$$\mathbf{w} = \{\boldsymbol{\theta}_{0,0}, \dots, \boldsymbol{\theta}_{0,K}, \mathbf{u}_0, \dots, \boldsymbol{\theta}_{N-1,0}, \dots, \boldsymbol{\theta}_{N-1,K}, \mathbf{u}_{N-1}\}$$

and

$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \mathbf{s}(\boldsymbol{\theta}_0, t_0) - \bar{\mathbf{x}}_0 \\ \dots \\ \mathbf{s}(\boldsymbol{\theta}_k, t_{k+1}) - \mathbf{s}(\boldsymbol{\theta}_{k+1}, t_{k+1}) \\ \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k) - \sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) \\ \dots \end{bmatrix} = 0$$

- NLP has $n_{\mathrm{x}}N(K+1) + n_{\mathrm{u}}N$ variables
- **Integration** performed in the NLP, converges together with optimization

Consequence: there is a systematic parallel linear algebra for Direct Collocation !!