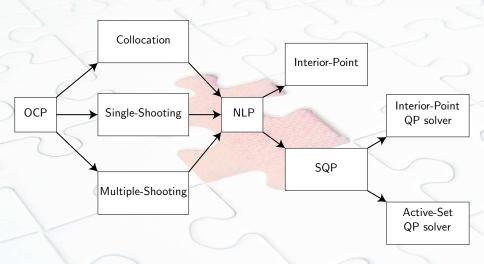
Numerical Optimal Control Lecture 5: Integrators with Sensitivities

Sébastien Gros

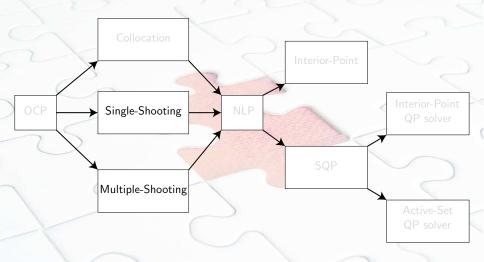
ITK, NTNU

NTNU PhD course

Survival map of Direct Optimal Control



Survival map of Direct Optimal Control



Shooting requires integrators, let's have a look at that...

Outline

- Introduction
- 2 Integration with sensitivities Variational approach
- 3 Integration with sensitivities Algorithmic Differentiation
- 4 Collocation-based integrators
- 5 Adjoint-mode sensitivity
- 6 Second-order sensitivity

Outline

- Introduction
- 2 Integration with sensitivities Variation approach
- 3 Integration with sensitivaties Alteria to the contraction
- 4 Collocation-based integrators
- 5 Adjoint-mode sensitivity
- **6** Second-order sensitivity

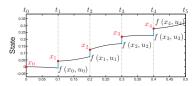
Multiple shooting yields an NLP:

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

s.t.
$$g(w) = 0$$

with $\mathbf{w} = \{x_0, \mathbf{u}_0, ..., x_{\mathcal{N}-1}, \mathbf{u}_{\mathcal{N}-1}, \mathbf{x}_{\mathcal{N}}\}$ and

$$\mathbf{g}\left(\mathbf{w}\right) = \begin{bmatrix} \mathbf{b}\left(\mathbf{x}_{0}, \mathbf{x}_{N}\right) \\ \mathbf{f}\left(\mathbf{x}_{0}, \mathbf{u}_{0}\right) - \mathbf{x}_{1} \\ \mathbf{f}\left(\mathbf{x}_{1}, \mathbf{u}_{1}\right) - \mathbf{x}_{2} \\ \dots \end{bmatrix}$$

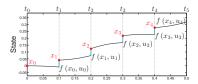


Multiple shooting yields an NLP:

s.t.
$$\mathbf{g}(\mathbf{w}) = 0$$

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

$$\mathbf{g}\left(\mathbf{w}\right) = \begin{bmatrix} \mathbf{b}\left(\mathbf{x}_{0}, \mathbf{x}_{N}\right) \\ \mathbf{f}\left(\mathbf{x}_{0}, \mathbf{u}_{0}\right) - \mathbf{x}_{1} \\ \mathbf{f}\left(\mathbf{x}_{1}, \mathbf{u}_{1}\right) - \mathbf{x}_{2} \\ \dots \end{bmatrix}$$



Function $f(x_k, u_k)$ is an integration of the ODE over the time interval $[t_k, t_{k+1}]$

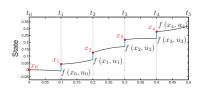
Multiple shooting yields an NLP:

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

s.t.
$$\mathbf{g}(\mathbf{w}) = 0$$

with $\mathbf{w} = \{x_0, \mathbf{u}_0, ..., x_{N-1}, \mathbf{u}_{N-1}, x_N\}$ and

$$\mathbf{g}\left(\mathbf{w}\right) = \begin{bmatrix} \mathbf{b}\left(\mathbf{x}_{0}, \mathbf{x}_{N}\right) \\ \mathbf{f}\left(\mathbf{x}_{0}, \mathbf{u}_{0}\right) - \mathbf{x}_{1} \\ \mathbf{f}\left(\mathbf{x}_{1}, \mathbf{u}_{1}\right) - \mathbf{x}_{2} \\ \dots \end{bmatrix}$$



Function $f(\mathbf{x}_k, \mathbf{u}_k)$ is an integration of the ODE over the time interval $[t_k, t_{k+1}]$

SQP, iterate:

QP problem (formed at w):

$$\begin{aligned} & \underset{\Delta \mathbf{w}}{\text{min}} & & \frac{1}{2} \Delta \mathbf{w}^{\top} B \Delta \mathbf{w} + \nabla \Phi^{\top} \Delta \mathbf{w} \\ & \text{s.t.} & & \nabla \mathbf{g}^{\top} \Delta \mathbf{w} + \mathbf{g} = 0 \end{aligned}$$

2
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \Delta \mathbf{w}, \quad \alpha \in [0, 1]$$

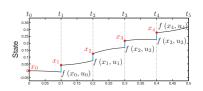
Multiple shooting yields an NLP:

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

s.t.
$$g(w) = 0$$

with $\mathbf{w} = \{x_0, \mathbf{u}_0, ..., x_{N-1}, \mathbf{u}_{N-1}, x_N\}$ and

$$\mathbf{g}\left(\mathbf{w}\right) = \left[\begin{array}{c} \mathbf{b}\left(\mathbf{x}_{0}, \mathbf{x}_{N}\right) \\ \mathbf{f}\left(\mathbf{x}_{0}, \mathbf{u}_{0}\right) - \mathbf{x}_{1} \\ \mathbf{f}\left(\mathbf{x}_{1}, \mathbf{u}_{1}\right) - \mathbf{x}_{2} \\ \dots \end{array}\right]$$



Function $f(x_k, u_k)$ is an integration of the ODE over the time interval $[t_k, t_{k+1}]$

SQP, iterate:

QP problem (formed at w):

$$\begin{aligned} & \underset{\Delta w}{\text{min}} & & \frac{1}{2} \Delta w^\top B \Delta w + \nabla \Phi^\top \Delta w \\ & \text{s.t.} & & \nabla \mathbf{g}^\top \Delta w + \mathbf{g} = 0 \end{aligned}$$

$$\mathbf{2} \ \mathbf{w} \leftarrow \mathbf{w} + \alpha \Delta \mathbf{w}, \quad \alpha \in]0,1]$$

Jacobian $\nabla g(\mathbf{w})$ requires the linearisation of the integrator, i.e. $\nabla f(\mathbf{x}_k, \mathbf{u}_k)$

NLP with multiple-shooting

$$\label{eq:standard_min_w} \begin{aligned} & \underset{w}{\text{min}} & \Phi\left(w\right) \\ & \text{s.t.} & & \mathbf{g}\left(w\right) = \left[\begin{array}{c} b\left(x_0, x_N\right) \\ f\left(x_0, u_0\right) - x_1 \\ f\left(x_1, u_1\right) - x_2 \\ & \dots \end{array} \right] \end{aligned}$$

NLP with multiple-shooting

$$\begin{aligned} & \underset{w}{\text{min}} & \Phi\left(w\right) \\ & \text{s.t.} & \mathbf{g}\left(w\right) = \left[\begin{array}{c} b\left(x_{0}, x_{N}\right) \\ f\left(x_{0}, u_{0}\right) - x_{1} \\ f\left(x_{1}, u_{1}\right) - x_{2} \\ & \dots \end{array} \right] \end{aligned}$$

where $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ are integrations of the ODE:

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

over the time intervals $[t_k, t_{k+1}]$

NLP with multiple-shooting

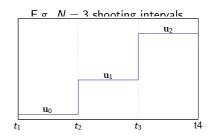
$$\min_{w} \quad \Phi\left(w\right)$$

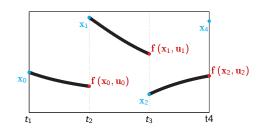
$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \left[\begin{array}{c} \mathbf{b}\left(\mathbf{x}_{0}, \mathbf{x}_{\mathcal{N}}\right) \\ \mathbf{f}\left(\mathbf{x}_{0}, \mathbf{u}_{0}\right) - \mathbf{x}_{1} \\ \mathbf{f}\left(\mathbf{x}_{1}, \mathbf{u}_{1}\right) - \mathbf{x}_{2} \\ \dots \end{array} \right]$$

where $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ are integrations of the ODE:

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

over the time intervals $[t_k, t_{k+1}]$





NLP with multiple-shooting

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

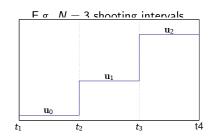
$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \left[\begin{array}{c} \mathbf{b}\left(\mathbf{x}_{0}, \mathbf{x}_{N}\right) \\ \mathbf{f}\left(\mathbf{x}_{0}, \mathbf{u}_{0}\right) - \mathbf{x}_{1} \\ \mathbf{f}\left(\mathbf{x}_{1}, \mathbf{u}_{1}\right) - \mathbf{x}_{2} \\ \dots \end{array} \right]$$

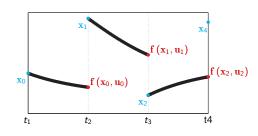
where $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ are integrations of the ODE:

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

over the time intervals $[t_k, t_{k+1}]$

NLP solver needs $\nabla_{\mathbf{x}_k} \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)$ and $\nabla_{\mathbf{u}_k} \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right)$





NLP with multiple-shooting

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

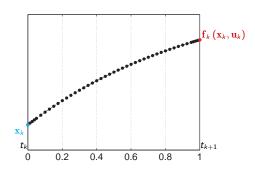
$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \left[\begin{array}{c} \mathbf{b}\left(\mathbf{x}_{0}, \mathbf{x}_{N}\right) \\ \mathbf{f}\left(\mathbf{x}_{0}, \mathbf{u}_{0}\right) - \mathbf{x}_{1} \\ \mathbf{f}\left(\mathbf{x}_{1}, \mathbf{u}_{1}\right) - \mathbf{x}_{2} \\ \dots \end{array} \right]$$

where $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ are integrations of the ODE:

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

over the time intervals $[t_k, t_{k+1}]$

NLP solver needs $\nabla_{\mathbf{x}_k} \mathbf{f} \left(\mathbf{x}_k, \mathbf{u}_k \right)$ and $\nabla_{\mathbf{u}_k} \mathbf{f} \left(\mathbf{x}_k, \mathbf{u}_k \right)$



NLP with multiple-shooting

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

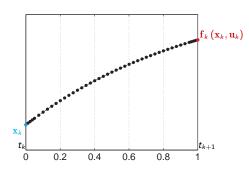
$$\text{s.t.} \quad \mathbf{g}\left(\mathbf{w}\right) = \left[\begin{array}{c} \mathbf{b}\left(\mathbf{x}_{0}, \mathbf{x}_{N}\right) \\ \mathbf{f}\left(\mathbf{x}_{0}, \mathbf{u}_{0}\right) - \mathbf{x}_{1} \\ \mathbf{f}\left(\mathbf{x}_{1}, \mathbf{u}_{1}\right) - \mathbf{x}_{2} \\ \dots \end{array} \right]$$

where $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ are integrations of the ODE:

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

over the time intervals $[t_k, t_{k+1}]$

NLP solver needs $\nabla_{\mathbf{x}_k} \mathbf{f} \left(\mathbf{x}_k, \mathbf{u}_k \right)$ and $\nabla_{\mathbf{u}_k} \mathbf{f} \left(\mathbf{x}_k, \mathbf{u}_k \right)$



How to compute $f(x_k, u_k)$ and:

$$\nabla_{\mathbf{x}_k} \mathbf{f} \left(\mathbf{x}_k, \mathbf{u}_k \right) \\ \nabla_{\mathbf{u}_k} \mathbf{f} \left(\mathbf{x}_k, \mathbf{u}_k \right)$$

?

Outline

- Introduction
- 2 Integration with sensitivities Variational approach
- 3 Integration with sensitivities Although 1996 Sentiation
- 4 Collocation-based integrator
- Adjoint-mode sensitivity
- 6 Second-order sensitivity

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}(t_{k+1})$.

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = s(t_{k+1})$.

Define $A(t) = \frac{\partial s(t)}{\partial x_k}$, such that $\nabla_{x_k} f(x_k, u_k)^\top = A(t_{k+1})$. Moreover:

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(x_k, u_k) = s(t_{k+1})$.

Define $A(t) = \frac{\partial s(t)}{\partial x_k}$, such that $\nabla_{x_k} \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)^{\top} = A(t_{k+1})$. Moreover:

$$\dot{A}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k}$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = s(t_{k+1})$.

Define $A(t) = \frac{\partial s(t)}{\partial x_k}$, such that $\nabla_{x_k} f(x_k, u_k)^\top = A(t_{k+1})$. Moreover:

$$\dot{A}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{x}_k}$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(x_k, u_k) = s(t_{k+1})$.

Define $A(t) = \frac{\partial s(t)}{\partial x_k}$, such that $\nabla_{x_k} f(x_k, u_k)^\top = A(t_{k+1})$. Moreover:

$$\dot{A}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{x}_k}$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = s(t_{k+1})$.

Define $A(t) = rac{\partial s(t)}{\partial x_k}$, such that $\nabla_{x_k} \mathbf{f}\left(x_k, \mathbf{u}_k\right)^{\top} = A(t_{k+1})$. Moreover:

$$\dot{A}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k}$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}(t_{k+1})$.

Define $A(t) = rac{\partial s(t)}{\partial x_k}$, such that $\nabla_{x_k} \mathbf{f}\left(x_k, \mathbf{u}_k\right)^{\top} = A(t_{k+1})$. Moreover:

$$\dot{A}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k}$$
$$= \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} A(t)$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}(t_{k+1})$.

Define $A(t) = rac{\partial s(t)}{\partial x_k}$, such that $\nabla_{x_k} \mathbf{f}\left(x_k, \mathbf{u}_k\right)^{\top} = A(t_{k+1})$. Moreover:

$$\dot{A}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k}$$
$$= \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} A(t)$$

with the initial conditions $A(t_k) = \frac{\partial s(t_k)}{\partial x_k} = \frac{\partial x_k}{\partial x_k} = I$.

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}(t_{k+1})$.

Define $A(t) = \frac{\partial s(t)}{\partial x_k}$, such that $\nabla_{x_k} \mathbf{f}\left(x_k, \mathbf{u}_k\right)^{\top} = A(t_{k+1})$. Moreover:

$$\dot{A}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{x}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{x}_k}$$
$$= \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} A(t)$$

with the initial conditions $A(t_k) = \frac{\partial s(t_k)}{\partial x_k} = \frac{\partial x_k}{\partial x_k} = I$.

Then

$$egin{array}{lcl}
abla_{\mathbf{x}_k}\mathbf{f}\left(\mathbf{x}_k,\mathbf{u}_k
ight)^{ op} &=& A\left(t_{k+1}
ight) & ext{with} \ \dot{\mathbf{s}}(t) &=& \mathbf{F}\left(\mathbf{s}(t),\mathbf{u}_k
ight), & \mathbf{s}(t_k) = \mathbf{x}_k \ \dot{A}(t) &=& rac{\partial \mathbf{F}\left(\mathbf{s}\left(t
ight),\mathbf{u}_k
ight)}{\partial \mathbf{s}\left(t
ight)}A(t), & A(t_k) = I \end{array}$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}(t_{k+1})$.

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t\right) = \mathbf{F}\left(\mathbf{s}\left(t\right), \mathbf{u}_{k}\right) \quad \text{and} \quad \mathbf{s}\left(t_{k}\right) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = s(t_{k+1})$.

Define $B\left(t\right) = \frac{\partial s(t)}{\partial \mathbf{u}_{k}}$, such that $\nabla_{\mathbf{u}_{k}}\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}\right)^{\top} = B\left(t_{k+1}\right)$.

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = s(t_{k+1})$.

Define $B\left(t\right)=rac{\partial s\left(t\right)}{\partial u_{k}}$, such that $\nabla_{u_{k}}\mathbf{f}\left(\mathbf{x}_{k},u_{k}\right)^{\top}=B\left(t_{k+1}\right)$. Moreover:

$$\dot{B}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{u}_k}$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = s(t_{k+1})$.

Define $B\left(t\right)=rac{\partial s\left(t\right)}{\partial \mathbf{u}_{k}}$, such that $\nabla_{\mathbf{u}_{k}}\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}\right)^{\top}=B\left(t_{k+1}\right)$. Moreover:

$$\dot{B}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{u}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{u}_k}$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t\right) = \mathbf{F}\left(\mathbf{s}\left(t\right), \mathbf{u}_{k}\right) \quad \text{and} \quad \mathbf{s}\left(t_{k}\right) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = s(t_{k+1})$.

Define $B\left(t\right)=rac{\partial s\left(t\right)}{\partial \mathbf{u}_{k}}$, such that $\nabla_{\mathbf{u}_{k}}\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}\right)^{\top}=B\left(t_{k+1}\right)$. Moreover:

$$\dot{B}(t) = \frac{d}{dt} \frac{\partial s(t)}{\partial u_k} = \frac{\partial \dot{s}(t)}{\partial u_k} = \frac{\partial F(s(t), u_k)}{\partial u_k} + \frac{\partial F(s(t), u_k)}{\partial s(t)} \frac{\partial s(t)}{\partial u_k}$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}(t_{k+1})$.

Define $B\left(t\right)=rac{\partial s\left(t\right)}{\partial \mathbf{u}_{k}}$, such that $\nabla_{\mathbf{u}_{k}}\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}\right)^{\top}=B\left(t_{k+1}\right)$. Moreover:

$$\dot{B}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{u}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{u}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{u}_k}$$

$$= \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} B(t)$$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(\mathbf{x}_k, \mathbf{u}_k) = s(t_{k+1})$.

Define $B\left(t\right)=rac{\partial s\left(t\right)}{\partial u_{k}}$, such that $\nabla_{u_{k}}\mathbf{f}\left(\mathbf{x}_{k},u_{k}\right)^{\top}=B\left(t_{k+1}\right)$. Moreover:

$$\dot{B}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{u}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{u}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{u}_k}$$

$$= \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} B(t)$$

with the initial conditions $B(t_k) = 0$

Let us define s(t) the state trajectory over the time interval $[t_k, t_{k+1}]$, i.e.

$$\dot{\mathbf{s}}\left(t
ight) = \mathbf{F}\left(\mathbf{s}\left(t
ight), \mathbf{u}_{k}
ight) \quad ext{and} \quad \mathbf{s}\left(t_{k}
ight) = \mathbf{x}_{k}$$

then $f(x_k, u_k) = s(t_{k+1})$.

Define $B\left(t\right)=rac{\partial s\left(t\right)}{\partial \mathbf{u}_{k}}$, such that $\nabla_{\mathbf{u}_{k}}\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}\right)^{\top}=B\left(t_{k+1}\right)$. Moreover:

$$\dot{B}(t) = \frac{d}{dt} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{u}_k} = \frac{\partial \dot{\mathbf{s}}(t)}{\partial \mathbf{u}_k} = \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} \frac{\partial \mathbf{s}(t)}{\partial \mathbf{u}_k}$$

$$= \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{u}_k} + \frac{\partial \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k)}{\partial \mathbf{s}(t)} B(t)$$

with the initial conditions $B(t_k) = 0$

Then

$$\nabla_{\mathbf{u}_{k}}\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}\right)^{\top} = B\left(t_{k+1}\right) \quad \text{with}$$

$$\dot{\mathbf{s}}(t) = \mathbf{F}\left(\mathbf{s}(t),\mathbf{u}_{k}\right), \quad \mathbf{s}(t_{k}) = \mathbf{x}_{k}$$

$$\dot{B}\left(t\right) = \frac{\partial \mathbf{F}\left(\mathbf{s}\left(t\right),\mathbf{u}_{k}\right)}{\partial \mathbf{u}_{k}} + \frac{\partial \mathbf{F}\left(\mathbf{s}\left(t\right),\mathbf{u}_{k}\right)}{\partial \mathbf{s}\left(t\right)}B\left(t\right), \quad B(t_{k}) = 0$$

Integrate forward over the time interval $[t_k, t_{k+1}]$:

State simulation:
$$\dot{\mathbf{s}}(t) = \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k), \qquad \mathbf{s}(t_k) = \mathbf{x}_k$$

State sensitivity:
$$\dot{A}(t) = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}(t),\mathbf{u}_k} A(t)$$
, $A(t_k) =$

State simulation:
$$\dot{\mathbf{s}}(t) = \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k),$$
 $\mathbf{s}(t_k) = \mathbf{x}_k$

State sensitivity: $\dot{A}(t) = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}\big|_{\mathbf{s}(t), \mathbf{u}_k} A(t),$ $A(t_k) = I$

Input sensitivity: $\dot{B}(t) = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}\big|_{\mathbf{s}(t), \mathbf{u}_k} B(t) + \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\big|_{\mathbf{s}(t), \mathbf{u}_k},$ $B(t_k) = 0$

Note: read
$$\left. \frac{\partial F}{\partial x} \right|_{s(t),u_k} = \left. \frac{\partial F(s(t),u_k)}{\partial s(t)} \right.$$
 and $\left. \frac{\partial F}{\partial u} \right|_{s(t),u_k} = \left. \frac{\partial F(s(t),u_k)}{\partial u_k} \right.$

Integrate forward over the time interval $[t_k, t_{k+1}]$:

State simulation:
$$\dot{\mathbf{s}}(t) = \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k), \qquad \mathbf{s}(t_k) = \mathbf{x}_k$$

State sensitivity:
$$\dot{A}(t) = \frac{\partial \mathbf{F}}{\partial x}|_{\mathbf{s}(t),\mathbf{u}_k} A(t),$$
 $A(t_k) =$

State simulation:
$$\dot{\mathbf{s}}(t) = \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k),$$
 $\mathbf{s}(t_k) = \mathbf{x}_k$

State sensitivity: $\dot{A}(t) = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}(t), \mathbf{u}_k} A(t),$ $A(t_k) = I$

Input sensitivity: $\dot{B}(t) = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}(t), \mathbf{u}_k} B(t) + \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s}(t), \mathbf{u}_k},$ $B(t_k) = 0$

Note: read
$$\frac{\partial F}{\partial x}\big|_{s(t),u_k} = \frac{\partial F(s(t),u_k)}{\partial s(t)}$$
 and $\frac{\partial F}{\partial u}\big|_{s(t),u_k} = \frac{\partial F(s(t),u_k)}{\partial u_k}$

Integrator evaluation with sensitivities

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

$$\nabla_{\mathbf{x}_{k}} \mathbf{f}\left(\mathbf{x}_{k}, \mathbf{u}_{k}\right) = A\left(t_{k+1}\right)^{\top},$$

$$\nabla_{\mathbf{u}_k}\mathbf{f}\left(\mathbf{x}_k,\mathbf{u}_k\right)=B\left(t_{k+1}\right)^{\top},$$

Integrate forward over the time interval $[t_k, t_{k+1}]$:

State simulation:
$$\dot{\mathbf{s}}(t) = \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k),$$
 $\mathbf{s}(t_k) = \mathbf{x}_k$

State simulation:
$$\dot{\mathbf{s}}(t) = \mathbf{F}(\mathbf{s}(t), \mathbf{u}_k),$$
 $\mathbf{s}(t_k) = \mathbf{x}_k$

State sensitivity: $\dot{A}(t) = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}(t), \mathbf{u}_k} A(t),$ $A(t_k) = I$

Input sensitivity:
$$\dot{B}\left(t\right) = \frac{\partial F}{\partial x}\big|_{\mathbf{s}\left(t\right),\mathbf{u}_{k}} B\left(t\right) + \frac{\partial F}{\partial \mathbf{u}}\big|_{\mathbf{s}\left(t\right),\mathbf{u}_{k}}, \qquad B\left(t_{k}\right) = 0$$

Note: read
$$\frac{\partial F}{\partial x}\big|_{s(t),u_k} = \frac{\partial F(s(t),u_k)}{\partial s(t)}$$
 and $\frac{\partial F}{\partial u}\big|_{s(t),u_k} = \frac{\partial F(s(t),u_k)}{\partial u_k}$

Integrator evaluation with sensitivities

$$\mathbf{f}\left(\mathbf{x}_{k}, \mathbf{u}_{k}\right) = \mathbf{s}\left(t_{k+1}\right), \
abla_{\mathbf{x}_{k}} \mathbf{f}\left(\mathbf{x}_{k}, \mathbf{u}_{k}\right) = A\left(t_{k+1}\right)^{\top}, \
abla_{\mathbf{u}_{k}} \mathbf{f}\left(\mathbf{x}_{k}, \mathbf{u}_{k}\right) = B\left(t_{k+1}\right)^{\top}, \$$

Pros: can use your favourite integrator to handle s, A, B jointly

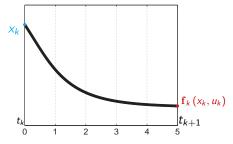
Cons: inexact derivative (because of inexact integration) !!

> First differentiate, then discretize (i.e. "integrate")

ODE: $\dot{x} = -\sin(x) + u$

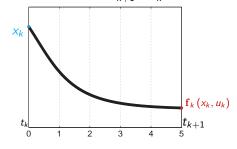
ODE:
$$\dot{x} = -\sin(x) + u$$

Using ode45.m, integrate the ODE over a time interval $t_{k+1} - t_k = 5 \,\mathrm{s}$

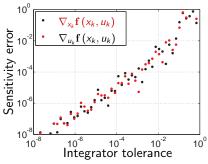


ODE:
$$\dot{x} = -\sin(x) + u$$

Using ode45.m, integrate the ODE over a time interval $t_{k+1} - t_k = 5 \,\mathrm{s}$

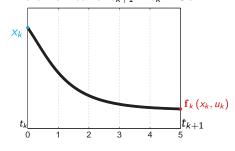


Error in the sensitivities (baseline is numerical differentiation):

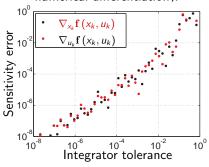


ODE:
$$\dot{x} = -\sin(x) + u$$

Using ode45.m, integrate the ODE over a time interval $t_{k+1} - t_k = 5 \,\mathrm{s}$



Error in the sensitivities (baseline is numerical differentiation):



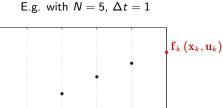
An NLP solver fed with inaccurate derivatives will struggle when reaching a KKT residual in the range of the derivative inaccuracy. If using a variational approach, make sure that the integrator tolerance << tolerance of the NLP solver (at least 1 order of magnitude)

Outline

- 1 Introduction
- 2 Integration with sensitivities Variation approach
- 3 Integration with sensitivities Algorithmic Differentiation
- 4 Collocation-based integrate
- 5 Adjoint-mode sensitivity
- 6 Second-order sensitivity

Key idea: first descritize, then differentiate

Key idea: first descritize, then differentiate



 \mathbf{x}_k

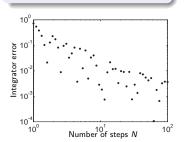
0.2

0.4

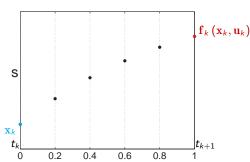
0.6

0.8

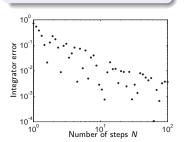
Key idea: first descritize, then differentiate



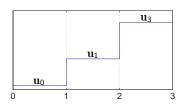
E.g. with N = 5, $\Delta t = 1$

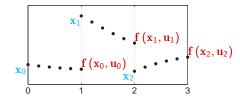


Key idea: first descritize, then differentiate



E.g. N = 5, with 3 shooting intervals



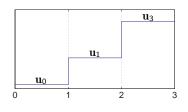


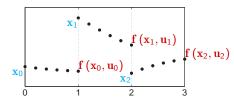
Key idea: first descritize, then differentiate

Reminder:

$$\mathbf{g}\left(\mathbf{w}\right) = \left[\begin{array}{c} \mathbf{b}\left(\mathbf{x}_{0}, \mathbf{x}_{N}\right) \\ \mathbf{x}_{1} - \mathbf{f}\left(\mathbf{x}_{0}, \mathbf{u}_{0}\right) \\ \dots \\ \mathbf{x}_{N} - \mathbf{f}\left(\mathbf{x}_{N-1}, \mathbf{u}_{N-1}\right) \end{array} \right]$$

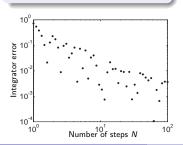
E.g. N = 5, with 3 shooting intervals





Let's differentiate Explicit Euler

Key idea: first descritize, then differentiate



Algorithmic Differentiation

Algorithm: Explicit Euler with forward sensitivities

Input:
$$\mathbf{x}_{k}$$
, \mathbf{u}_{k} , $[t_{k}, t_{k+1}]$, N $\mathbf{s} = \mathbf{x}_{k}$, $\Delta t = t_{k+1} - t_{k}$ for $i = 0 : N - 1$ do

$$\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{N} \mathbf{F} \left(\mathbf{s}, \mathbf{u}_k \right)$$

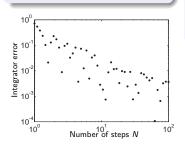
return

$$f(\mathbf{x}_k,\mathbf{u}_k)=\mathbf{s}$$

13 / 27

Let's differentiate Explicit Euler

Key idea: first descritize, then differentiate



Algorithmic Differentiation

Algorithm: Explicit Euler with forward sensitivities

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_k$, $\Delta t = t_{k+1} - t_k$, $A = I$
for $i = 0 : N - 1$ do

$$tor \ i = 0 : N - 1 \ do$$

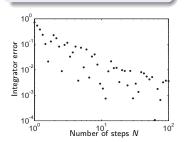
$$\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{N} \mathbf{F} \left(\mathbf{s}, \mathbf{u}_k \right)$$

$$f(\mathbf{x}_k,\mathbf{u}_k)=\mathbf{s}$$

$$A = \frac{\partial \mathbf{s}}{\partial \mathbf{x}}$$

Let's differentiate Explicit Euler

Key idea: first descritize, then differentiate



Algorithmic Differentiation

Algorithm: Explicit Euler with forward sensitivities

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_k$, $\Delta t = t_{k+1} - t_k$, $A = I$
for $i = 0 : N - 1$ do
$$A \leftarrow \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_k}\right) A$$

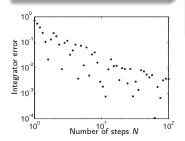
$$\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{N} \mathbf{F} \left(\mathbf{s}, \mathbf{u}_k \right)$$

$$\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}\right)=\mathbf{s}$$

$$A = \frac{\partial \mathbf{s}}{\partial \mathbf{x}_{k}}$$

Let's differentiate Explicit Euler

Key idea: first descritize, then differentiate



Algorithmic Differentiation

Algorithm: Explicit Euler with forward sensitivities

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_k$, $\Delta t = t_{k+1} - t_k$, $A = I$
for $i = 0 : N - 1$ do

$$A \leftarrow \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}}\big|_{\mathbf{s}, \mathbf{u}_k}\right) A$$

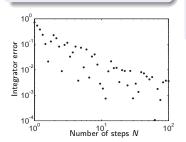
$$\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{N} \mathbf{F} \left(\mathbf{s}, \mathbf{u}_k \right)$$

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}, \ \nabla_{\mathbf{x}_k} \mathbf{f}_k = A^{\top}$$

$$A = \frac{\partial \mathbf{s}}{\partial \mathbf{x}_{k}}$$

Let's differentiate Explicit Euler

Key idea: first descritize, then differentiate



Algorithmic Differentiation

Algorithm: Explicit Euler with forward sensitivities

Input:
$$\mathbf{x}_{k}$$
, \mathbf{u}_{k} , $[t_{k}, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_{k}$, $\Delta t = t_{k+1} - t_{k}$, $A = I$, $B = 0$

for
$$i = 0 : N - 1$$
 do

$$A \leftarrow \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}}\big|_{\mathbf{s}, \mathbf{u}_k}\right) A$$

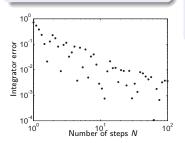
$$\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{N} \mathbf{F} \left(\mathbf{s}, \mathbf{u}_k \right)$$

$$\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}\right)=\mathbf{s},\ \nabla_{\mathbf{x}_{k}}\mathbf{f}_{k}=\mathbf{A}^{\top}$$

$$A = \frac{\partial \mathbf{s}}{\partial \mathbf{x}_k}, \qquad B = \frac{\partial \mathbf{s}}{\partial \mathbf{u}_k}$$

Let's differentiate Explicit Euler

Key idea: first descritize, then differentiate



Algorithmic Differentiation

Algorithm: Explicit Euler with forward sensitivities

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_k$, $\Delta t = t_{k+1} - t_k$, $A = I$, $B = 0$

for
$$i = 0 : N - 1$$
 do

$$A \leftarrow \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \Big|_{\mathbf{s}, \mathbf{u}_k}\right) A$$

$$B \leftarrow \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \Big|_{\mathbf{s}, \mathbf{u}_k}\right) B + \frac{\partial \mathbf{F}}{\partial \mathbf{u}_k} \Big|_{\mathbf{s}, \mathbf{u}_k}$$

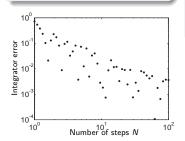
$$\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{N} \mathbf{F} \left(\mathbf{s}, \mathbf{u}_k\right)$$

$$\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}
ight)=\mathbf{s},\
abla_{\mathbf{x}_{k}}\mathbf{f}_{k}=A^{ op}$$

$$A = \frac{\partial \mathbf{s}}{\partial \mathbf{x}_k}, \qquad B = \frac{\partial \mathbf{s}}{\partial \mathbf{u}_k}$$

Let's differentiate Explicit Euler

Key idea: first descritize, then differentiate



Algorithmic Differentiation

Algorithm: Explicit Euler with forward sensitivities

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_k$, $\Delta t = t_{k+1} - t_k$, $A = I$, $B = 0$
for $i = 0$: $N - 1$ do

$$A \leftarrow \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_{k}}\right) A$$

$$B \leftarrow \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_{k}}\right) B + \frac{\partial \mathbf{F}}{\partial \mathbf{u}_{k}}|_{\mathbf{s}, \mathbf{u}_{k}}$$

$$\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{N} \mathbf{F} \left(\mathbf{s}, \mathbf{u}_k \right)$$

return

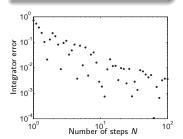
$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}, \ \nabla_{\mathbf{x}_k} \mathbf{f}_k = A^\top, \ \nabla_{\mathbf{u}_k} \mathbf{f}_k = B^\top$$

$$A = \frac{\partial \mathbf{s}}{\partial \mathbf{x}_k}, \qquad B = \frac{\partial \mathbf{s}}{\partial \mathbf{u}_k}$$

◆ロト ◆御 ト ◆ 恵 ト ◆ 恵 ・ り へ で

Let's differentiate Explicit Euler

Key idea: first descritize, then differentiate



Algorithmic Differentiation

Algorithm: Explicit Euler with forward sensitivities

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_k$, $\Delta t = t_{k+1} - t_k$, $A = I$, $B = 0$
for $i = 0$: $N - 1$ do

$$A \leftarrow \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_k}\right) A$$

$$B \leftarrow \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \Big|_{\mathbf{s}, \mathbf{u}_k}\right) B + \left. \frac{\partial \mathbf{F}}{\partial \mathbf{u}_k} \right|_{\mathbf{s}, \mathbf{u}_k}$$
$$\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{N} \mathbf{F}(\mathbf{s}, \mathbf{u}_k)$$

return

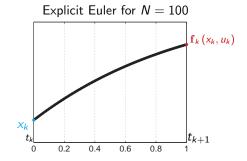
$$\mathbf{f}\left(\mathbf{x}_{k},\mathbf{u}_{k}\right)=\mathbf{s},\,\nabla_{\mathbf{x}_{k}}\mathbf{f}_{k}=A^{\top},\,\nabla_{\mathbf{u}_{k}}\mathbf{f}_{k}=B^{\top}$$

Pros: "exact" derivatives, extremely simple code

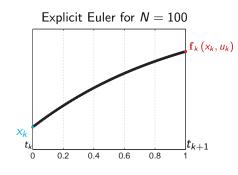
Cons: explicit 1st-order has a poor computational efficiency (flops vs. accuracy)

ODE: $\dot{x} = -\sin(x) + u$

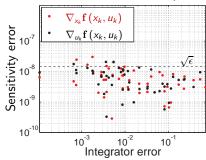
ODE: $\dot{x} = -\sin(x) + u$



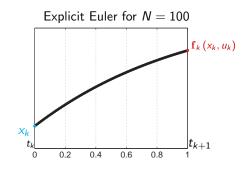
ODE:
$$\dot{x} = -\sin(x) + u$$



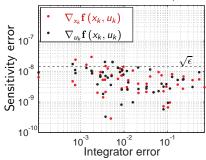
Error in the sensitivities (baseline is numerical differentiation):



ODE:
$$\dot{x} = -\sin(x) + u$$



Error in the sensitivities (baseline is numerical differentiation):



Algorithmic Differentiation (AD) allows for working with derivatives at machine precision (i.e. $\epsilon=10^{-16}$), regardless of how accurate the integrator is. AD is a generic principle that can be deployed on any (locally) smooth algorithm, i.e. one can use it on more efficient integrators than Explicit Euler.

Algorithm: ERK4 with sensitivities

Input: \mathbf{x}_k , \mathbf{u}_k , $[t_k, t_{k+1}]$, N

$$\mathbf{s} = \mathbf{x}_k$$

for
$$i = 0 : N do$$

$$\mathbf{k}_{1}:=\mathbf{F}\left(\mathbf{s},\mathbf{u}_{k}\right)$$

$$\mathbf{k}_2 := \mathbf{F}\left(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k\right)$$

$$\mathbf{k}_3 := \mathbf{F}\left(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k\right)$$

$$\mathbf{k}_4 := \mathbf{F}\left(\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_3, \mathbf{u}_k\right)$$

$$s \leftarrow s + \frac{\Delta \mathit{t}}{6\textit{N}} \left(k_1 + 2k_2 + 2k_3 + k_4 \right)$$

return $f(x_k, u_k) = s$

The ERK4 - Explicit Runge-Kutta of 4th-order

Algorithm: ERK4 with sensitivities

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N

$$\mathbf{s} = \mathbf{x}_k$$

for
$$i = 0 : N$$
 do

$$egin{aligned} \mathbf{k}_1 &:= \mathbf{F}\left(\mathbf{s}, \mathbf{u}_k
ight) \ rac{\partial \mathbf{k}_1}{\partial \mathbf{s}} &= rac{\partial \mathbf{F}}{\partial \mathbf{x}}ig|_{\mathbf{s}, \mathbf{u}_k} \end{aligned}$$

$$\mathbf{k}_2 := \mathbf{F}\left(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k\right)$$

$$\mathbf{k}_3 := \mathbf{F}\left(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k\right)$$

$$\mathbf{k}_4 := \mathbf{F}\left(\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_3, \mathbf{u}_k\right)$$

$$\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{6N} \left(\mathbf{k}_1 + 2 \mathbf{k}_2 + 2 \mathbf{k}_3 + \mathbf{k}_4 \right)$$

return
$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}$$

 $\frac{\partial \mathbf{k}_1}{\partial \mathbf{n}_i} = \frac{\partial \mathbf{F}}{\partial \mathbf{u}} \Big|_{\mathbf{S}, \mathbf{u}_i}$

The ERK4 - Explicit Runge-Kutta of 4th-order

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_k$
for $i = 0$: N do
$$\begin{vmatrix}
\mathbf{k}_1 := \mathbf{F}(\mathbf{s}, \mathbf{u}_k) \\
\frac{\partial \mathbf{k}_1}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_k} \\
\mathbf{k}_2 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k) \\
\frac{\partial \mathbf{k}_2}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} (I + \frac{\Delta t}{2N}\frac{\partial \mathbf{k}_1}{\partial \mathbf{s}}) & \frac{\partial \mathbf{k}_2}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} + \frac{\Delta t}{2N}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} \frac{\partial \mathbf{k}_1}{\partial \mathbf{u}_k} \\
\mathbf{k}_3 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k) \\
\mathbf{k}_4 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_3, \mathbf{u}_k) \\
\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{6N}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

return
$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}$$

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_k$
for $i = 0$: N do
$$\begin{vmatrix}
\mathbf{k}_1 := \mathbf{F}(\mathbf{s}, \mathbf{u}_k) \\
\frac{\partial \mathbf{k}_1}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_k} & \frac{\partial \mathbf{k}_1}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s}, \mathbf{u}_k} \\
\mathbf{k}_2 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k) \\
\frac{\partial \mathbf{k}_2}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} (I + \frac{\Delta t}{2N}\frac{\partial \mathbf{k}_1}{\partial \mathbf{s}}) & \frac{\partial \mathbf{k}_2}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} + \frac{\Delta t}{2N}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} & \frac{\partial \mathbf{k}_1}{\partial \mathbf{u}_k} \\
\mathbf{k}_3 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k) & \frac{\partial \mathbf{k}_3}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{s}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k} (I + \frac{\Delta t}{2N}\frac{\partial \mathbf{k}_2}{\partial \mathbf{s}}) & \frac{\partial \mathbf{k}_3}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k} + \frac{\Delta t}{2N}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k} & \frac{\partial \mathbf{k}_2}{\partial \mathbf{u}_k} \\
\mathbf{k}_4 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_3, \mathbf{u}_k) & \\
\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{6N}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

return
$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}$$

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N

$$\mathbf{s} = \mathbf{x}_k$$
for $i = 0 : N$ do
$$\begin{vmatrix}
\mathbf{k}_1 := \mathbf{F}(\mathbf{s}, \mathbf{u}_k) \\
\frac{\partial \mathbf{k}_1}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_k} \\
\mathbf{k}_2 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k) \\
\frac{\partial \mathbf{k}_2}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{s}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} \\
\mathbf{k}_3 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k) \\
\frac{\partial \mathbf{k}_3}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{r}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} (I + \frac{\Delta t}{2N}\frac{\partial \mathbf{k}_2}{\partial \mathbf{s}}) \quad \frac{\partial \mathbf{k}_2}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} + \frac{\Delta t}{2N}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_1, \mathbf{u}_k} \frac{\partial \mathbf{k}_1}{\partial \mathbf{u}_k} \\
\mathbf{k}_4 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k) (I + \frac{\Delta t}{2N}\frac{\partial \mathbf{k}_2}{\partial \mathbf{s}}) \quad \frac{\partial \mathbf{k}_3}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k} + \frac{\Delta t}{2N}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_2, \mathbf{u}_k} \frac{\partial \mathbf{k}_2}{\partial \mathbf{u}_k} \\
\mathbf{k}_4 := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_3, \mathbf{u}_k) \\
\frac{\partial \mathbf{k}_4}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_3, \mathbf{u}_k} (I + \frac{\Delta t}{N}\frac{\partial \mathbf{k}_3}{\partial \mathbf{s}}) \quad \frac{\partial \mathbf{k}_4}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_3, \mathbf{u}_k} + \frac{\Delta t}{N}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_3, \mathbf{u}_k} \frac{\partial \mathbf{k}_3}{\partial \mathbf{u}_k} \\
\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{6N}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

return
$$f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}$$

Algorithm: ERK4 with sensitivities

Input:
$$\mathbf{x}_{k}$$
, \mathbf{u}_{k} , $[t_{k}, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_{k}$, $A = I$
for $i = 0$: N do
$$\begin{vmatrix}
\mathbf{k}_{1} := \mathbf{F}(\mathbf{s}, \mathbf{u}_{k}) \\
\frac{\partial \mathbf{k}_{1}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{1}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \\
\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{k}_{3}}{\partial \mathbf{$$

return $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}$

Input:
$$\mathbf{x}_{k}$$
, \mathbf{u}_{k} , $[t_{k}, t_{k+1}]$, N

$$\mathbf{s} = \mathbf{x}_{k}$$
, $A = I$, $B = 0$

for $i = 0 : N$ do
$$\begin{vmatrix}
\mathbf{k}_{1} := \mathbf{F}(\mathbf{s}, \mathbf{u}_{k}) \\
\frac{\partial \mathbf{k}_{1}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_{k}} \\
k_{2} := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}) \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}} (I + \frac{\Delta t}{2N}\frac{\partial \mathbf{k}_{1}}{\partial \mathbf{s}}) & \frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}} + \frac{\Delta t}{2N}\frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}} & \frac{\partial \mathbf{k}_{1}}{\partial \mathbf{u}_{k}} \\
\mathbf{k}_{3} := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}) \\
\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} (I + \frac{\Delta t}{2N}\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}}) & \frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} + \frac{\Delta t}{2N}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} & \frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} \\
\mathbf{k}_{4} := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}) & \frac{\partial \mathbf{k}_{3}}{\partial \mathbf{s}} & \frac{\partial \mathbf{k}_{4}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} + \frac{\Delta t}{N}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} & \frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{k}} \\
\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{6N}(\mathbf{k}_{1} + 2\mathbf{k}_{2} + 2\mathbf{k}_{3} + \mathbf{k}_{4}) \\
M \leftarrow I + \frac{\Delta t}{6N}(\frac{\partial \mathbf{k}_{1}}{\partial \mathbf{s}} + 2\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} + 2\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{s}} + 2\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{s}} + \frac{\partial \mathbf{k}_{4}}{\partial \mathbf{s}}) \\
A \leftarrow MA$$

return
$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}, \quad \nabla_{\mathbf{x}_k} \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k) = A^{\top},$$

Algorithm: ERK4 with sensitivities

Input:
$$\mathbf{x}_k$$
, \mathbf{u}_k , $[t_k, t_{k+1}]$, N

$$\mathbf{s} = \mathbf{x}_k$$
, $A = I$, $B = 0$
for $i = 0 : N$ do
$$\begin{vmatrix}
\mathbf{k}_1 := \mathbf{F}(\mathbf{s}, \mathbf{u}_k) \\
\frac{\partial \mathbf{k}_1}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}} |_{\mathbf{s}, \mathbf{u}_k} \\
k_2 := \mathbf{F}\left(\mathbf{s} + \frac{\Delta t}{2N} \mathbf{k}_1, \mathbf{u}_k\right) \\
\frac{\partial \mathbf{k}_2}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}} |_{\mathbf{s} + \frac{\Delta t}{2N} \mathbf{k}_1, \mathbf{u}_k} \left(I + \frac{\Delta t}{2N} \frac{\partial \mathbf{k}_1}{\partial \mathbf{s}}\right) \quad \frac{\partial \mathbf{k}_2}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}} |_{\mathbf{s} + \frac{\Delta t}{2N} \mathbf{k}_1, \mathbf{u}_k} + \frac{\Delta t}{2N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} |_{\mathbf{s} + \frac{\Delta t}{2N} \mathbf{k}_1, \mathbf{u}_k} \frac{\partial \mathbf{k}_1}{\partial \mathbf{u}_k} \\
k_3 := \mathbf{F}\left(\mathbf{s} + \frac{\Delta t}{2N} \mathbf{k}_2, \mathbf{u}_k\right) \\
\frac{\partial \mathbf{k}_3}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}} |_{\mathbf{s} + \frac{\Delta t}{2N} \mathbf{k}_2, \mathbf{u}_k} \left(I + \frac{\Delta t}{2N} \frac{\partial \mathbf{k}_2}{\partial \mathbf{s}}\right) \quad \frac{\partial \mathbf{k}_3}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}} |_{\mathbf{s} + \frac{\Delta t}{2N} \mathbf{k}_2, \mathbf{u}_k} + \frac{\Delta t}{2N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} |_{\mathbf{s} + \frac{\Delta t}{2N} \mathbf{k}_2, \mathbf{u}_k} \frac{\partial \mathbf{k}_2}{\partial \mathbf{u}_k} \\
k_4 := \mathbf{F}\left(\mathbf{s} + \frac{\Delta t}{N} \mathbf{k}_3, \mathbf{u}_k\right) \\
\frac{\partial \mathbf{k}_4}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{r}} |_{\mathbf{s} + \frac{\Delta t}{N} \mathbf{k}_3, \mathbf{u}_k} \left(I + \frac{\Delta t}{N} \frac{\partial \mathbf{k}_3}{\partial \mathbf{s}}\right) \quad \frac{\partial \mathbf{k}_4}{\partial \mathbf{u}_k} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}} |_{\mathbf{s} + \frac{\Delta t}{N} \mathbf{k}_3, \mathbf{u}_k} + \frac{\Delta t}{N} \frac{\partial \mathbf{F}}{\partial \mathbf{x}} |_{\mathbf{s} + \frac{\Delta t}{N} \mathbf{k}_3, \mathbf{u}_k} \frac{\partial \mathbf{k}_3}{\partial \mathbf{u}_k} \\
\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{6N} \left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right) \\
M \leftarrow I + \frac{\Delta t}{6N} \left(\frac{\partial \mathbf{k}_1}{\partial \mathbf{s}} + 2\frac{\partial \mathbf{k}_2}{\partial \mathbf{s}} + 2\frac{\partial \mathbf{k}_3}{\partial \mathbf{s}} + \frac{\partial \mathbf{k}_4}{\partial \mathbf{s}}\right) \\
A \leftarrow MA \\
B \leftarrow MB + \frac{\Delta t}{6N} \left(\frac{\partial \mathbf{k}_1}{\partial \mathbf{u}} + 2\frac{\partial \mathbf{k}_2}{\partial \mathbf{u}} + 2\frac{\partial \mathbf{k}_3}{\partial \mathbf{u}} + 2\frac{\partial \mathbf{k}_4}{\partial \mathbf{u}}\right)$$

return $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}, \quad \nabla_{\mathbf{x}_k} \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k) = A^{\top},$

Algorithm: ERK4 with sensitivities

Input:
$$\mathbf{x}_{k}$$
, \mathbf{u}_{k} , $[t_{k}, t_{k+1}]$, N
 $\mathbf{s} = \mathbf{x}_{k}$, $A = I$, $B = 0$

for $i = 0 : N$ do

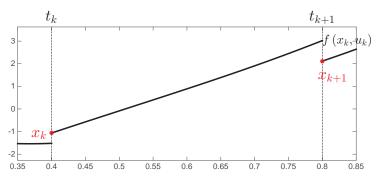
$$\begin{vmatrix}
\mathbf{k}_{1} := \mathbf{F}(\mathbf{s}, \mathbf{u}_{k}) \\
\frac{\partial \mathbf{k}_{1}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s}, \mathbf{u}_{k}} \\
k_{2} := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}) \\
\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}} (I + \frac{\Delta t}{2N}\frac{\partial \mathbf{k}_{1}}{\partial \mathbf{s}}) \quad \frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}} + \frac{\Delta t}{2N}\frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{1}, \mathbf{u}_{k}} \frac{\partial \mathbf{k}_{1}}{\partial \mathbf{u}_{k}} \\
\mathbf{k}_{3} := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}) \\
\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} (I + \frac{\Delta t}{2N}\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}}) \quad \frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} + \frac{\Delta t}{2N}\frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{2N}\mathbf{k}_{2}, \mathbf{u}_{k}} \frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{k}} \\
\mathbf{k}_{4} := \mathbf{F}(\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}) \\
\frac{\partial \mathbf{k}_{4}}{\partial \mathbf{s}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} (I + \frac{\Delta t}{N}\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{s}}) \quad \frac{\partial \mathbf{k}_{4}}{\partial \mathbf{u}_{k}} &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} + \frac{\Delta t}{N}\frac{\partial \mathbf{F}}{\partial \mathbf{x}}|_{\mathbf{s} + \frac{\Delta t}{N}\mathbf{k}_{3}, \mathbf{u}_{k}} \frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{k}} \\
\mathbf{s} \leftarrow \mathbf{s} + \frac{\Delta t}{6N}(\mathbf{k}_{1} + 2\mathbf{k}_{2} + 2\mathbf{k}_{3} + \mathbf{k}_{4}) \\
M \leftarrow I + \frac{\Delta t}{6N}(\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{s}} + 2\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{s}} + 2\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{s}} + 2\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{s}} + \frac{\partial \mathbf{k}_{4}}{\partial \mathbf{s}}) \\
A \leftarrow MA \\
B \leftarrow MB + \frac{\Delta t}{6N}(\frac{\partial \mathbf{k}_{1}}{\partial \mathbf{u}_{1}} + 2\frac{\partial \mathbf{k}_{2}}{\partial \mathbf{u}_{2}} + 2\frac{\partial \mathbf{k}_{3}}{\partial \mathbf{u}_{1}} + \frac{\partial \mathbf{k}_{4}}{\partial \mathbf{u}})$$

return $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{s}, \quad \nabla_{\mathbf{x}_k} \mathbf{f}_k (\mathbf{x}_k, \mathbf{u}_k) = A^\top, \quad \nabla_{\mathbf{u}_k} \mathbf{f}_k (\mathbf{x}_k, \mathbf{u}_k) = B^\top$

Outline

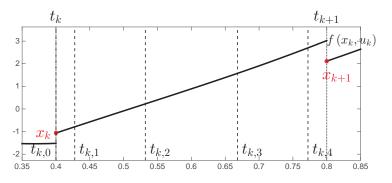
- 1 Introduction
- 2 Integration with sensitivities Variation approach
- 3 Integration with sensitivities. Algebra to the sentiation
- 4 Collocation-based integrators
- 5 Adjoint-mode sensitivity
- 6 Second-order sensitivity

Approximate state trajectory s(t) via polynomials (order K)



Approximate state trajectory s(t) via polynomials (order K)

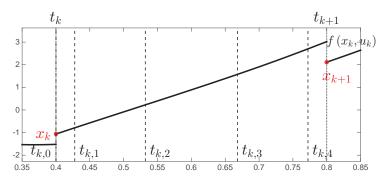
• Pick K + 1 time nodes $t_{k,i} \in [t_k, t_{k+1}]$



Approximate state trajectory s(t) via polynomials (order K)

- Pick K + 1 time nodes $t_{k,i} \in [t_k, t_{k+1}]$
- Approximate on each interval $[t_k, t_{k+1}]$:

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

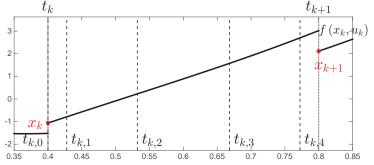


Approximate state trajectory s(t) via polynomials (order K)

- Pick K + 1 time nodes $t_{k,i} \in [t_k, t_{k+1}]$
- Approximate on each interval $[t_k, t_{k+1}]$:

$$\mathbf{s}\left(oldsymbol{ heta}_{k},t
ight) = \sum_{i=0}^{K} \underbrace{oldsymbol{ heta}_{k,i}}_{ ext{parameters}} \cdot \underbrace{oldsymbol{ heta}_{k,i}(t)}_{ ext{polynomials}}$$

ullet Pick the $P_{k,i}(t)$ so that $\mathbf{s}\left(oldsymbol{ heta}_k,t_{k,i}
ight)=oldsymbol{ heta}_{k,i}$,



◆ロト ◆御 ト ◆ 恵 ト ◆ 恵 ・ 夕 Q C

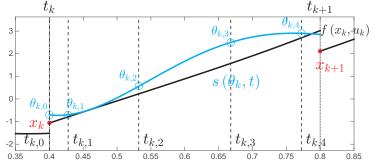
Collocation methods - key idea

Approximate state trajectory s(t) via polynomials (order K)

- Pick K + 1 time nodes $t_{k,i} \in [t_k, t_{k+1}]$
- Approximate on each interval $[t_k, t_{k+1}]$:

$$\mathbf{s}\left(oldsymbol{ heta}_{k},t
ight) = \sum_{i=0}^{K} \underbrace{oldsymbol{ heta}_{k,i}}_{ ext{parameters}} \cdot \underbrace{oldsymbol{ heta}_{k,i}(t)}_{ ext{polynomials}}$$

• Pick the $P_{k,i}(t)$ so that $s(\theta_k, t_{k,i}) = \theta_{k,i}$,



◆ロト ◆御 ト ◆ 恵 ト ◆ 恵 ・ り へ ②

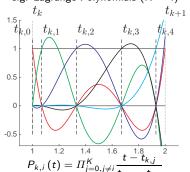
Collocation methods - key idea

Approximate state trajectory s(t) via polynomials (order K)

- Pick K + 1 time nodes $t_{k,i} \in [t_k, t_{k+1}]$
- Approximate on each interval $[t_k, t_{k+1}]$:

$$\mathrm{s}\left(oldsymbol{ heta}_{k},t
ight) = \sum_{i=0}^{K} \underbrace{oldsymbol{ heta}_{k,i}}_{\mathsf{parameters}} \cdot \underbrace{oldsymbol{ heta}_{k,i}(t)}_{\mathsf{polynomials}}$$

• Pick the $P_{k,i}(t)$ so that $\mathrm{s}\left(m{ heta}_k,t_{k,i}
ight) = m{ heta}_{k,i},$ e.g. Lagrange Polynomials (K=4)



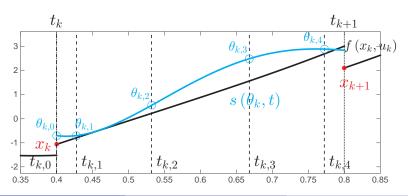
□ ▶ ◀鬪 ▶ ◀불 ▶ ◀불 ▶ · 볼 · ∽ Q ()

Collocation methods - how to interpolate?

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

$$\mathbf{s}\left(oldsymbol{ heta}_{k},t
ight) = \sum_{i=0}^{K} \underbrace{oldsymbol{ heta}_{k,i}}_{ ext{parameters}} \cdot \underbrace{oldsymbol{P}_{k,i}(t)}_{ ext{polynomials}} \qquad ext{with} \qquad \mathbf{s}\left(oldsymbol{ heta}_{k},t_{k,i}
ight) = oldsymbol{ heta}_{k,i}$$

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



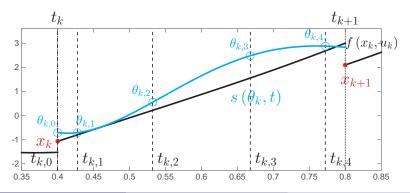
Collocation methods - how to interpolate?

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

$$\mathbf{s}\left(oldsymbol{ heta}_{k},t
ight) = \sum_{i=0}^{K} \underbrace{oldsymbol{ heta}_{k,i}}_{ ext{parameters}} \cdot \underbrace{oldsymbol{P}_{k,i}(t)}_{ ext{polynomials}} \qquad ext{with} \qquad \mathbf{s}\left(oldsymbol{ heta}_{k},t_{k,i}
ight) = oldsymbol{ heta}_{k,i}$$

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

$$\mathbf{S}\left(\boldsymbol{\theta}_{k},t_{k}\right)=\mathbf{X}_{k},\quad \text{(note that }\mathbf{x}_{k}\text{ is coming from the NLP !!)}$$



Collocation methods - how to interpolate?

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

$$\mathbf{s}\left(oldsymbol{ heta}_{k},t
ight) = \sum_{i=0}^{K} \underbrace{oldsymbol{ heta}_{k,i}}_{ ext{parameters}} \cdot \underbrace{oldsymbol{P}_{k,i}(t)}_{ ext{polynomials}} \qquad ext{with} \qquad \mathbf{s}\left(oldsymbol{ heta}_{k},t_{k,i}
ight) = oldsymbol{ heta}_{k,i}$$

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

$$\mathbf{s}\left(\boldsymbol{\theta}_{k},t_{k}\right) = \mathbf{x}_{k}, \quad \text{(note that } \mathbf{x}_{k} \text{ is coming from the NLP !!)}$$

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

$$t_{k}$$

$$t_{k+1}$$

$$\theta_{k,3}$$

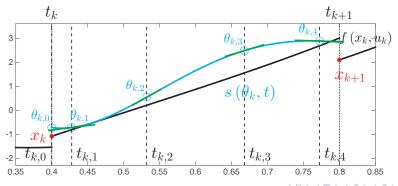
$$\theta_{k,4}$$

Collocation uses the constraints:

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

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

with i = 1, ..., K.

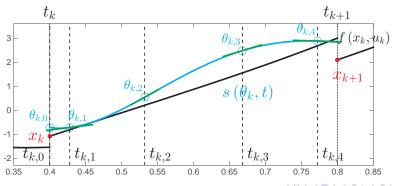


Collocation uses the constraints:

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

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

with i = 1, ..., K.



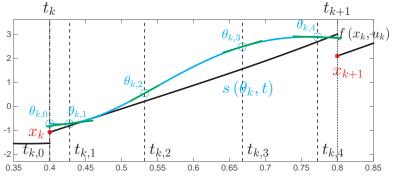
Collocation uses the constraints:

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

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

with i = 1, ..., K. Note:

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_{k},t\right)=\sum_{j=0}^{K}\boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t)$$



4 D > 4 B > 4 E > 4 E > 9 Q C

Collocation uses the constraints:

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

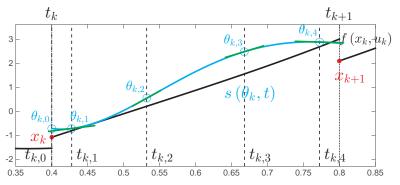
$$\frac{\partial}{\partial t} \mathbf{s}(\boldsymbol{\theta}_{k}, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k}),$$

with i = 1, ..., K. Note:

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k,t) = \sum_{j=0}^K \boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t)$$

$$\sum_{i=0}^K \frac{\boldsymbol{\theta}_{k,i} P_{k,i}(t_k) = \mathbf{x}_k}{\mathbf{x}_k}$$

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



Collocation uses the constraints:

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

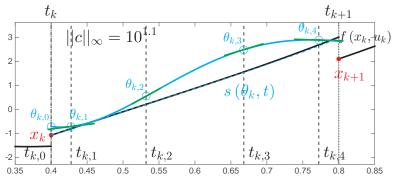
$$\frac{\partial}{\partial t} \mathbf{s}(\boldsymbol{\theta}_{k}, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k}),$$

with i = 1, ..., K. Note:

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k,t) = \sum_{j=0}^K \boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t)$$

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

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



Collocation uses the constraints:

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

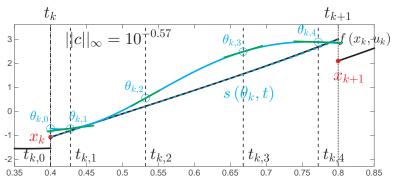
$$\frac{\partial}{\partial t} \mathbf{s}(\boldsymbol{\theta}_{k}, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k}),$$

with i = 1, ..., K. Note:

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_{k},t\right)=\sum_{j=0}^{K}\boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t)$$

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

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



Collocation uses the constraints:

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

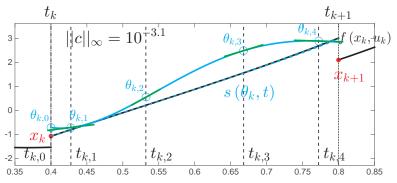
$$\frac{\partial}{\partial t} \mathbf{s}(\boldsymbol{\theta}_{k}, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k}),$$

with i = 1, ..., K. Note:

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k,t) = \sum_{j=0}^K \boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t)$$

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

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



Collocation uses the constraints:

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

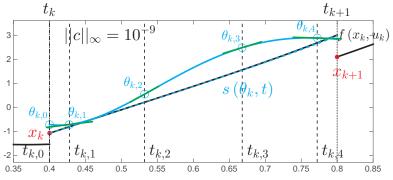
$$\frac{\partial}{\partial t} \mathbf{s}(\boldsymbol{\theta}_{k}, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k}),$$

with i = 1, ..., K. Note:

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k,t) = \sum_{j=0}^K \boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t)$$

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

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



Collocation uses the constraints:

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

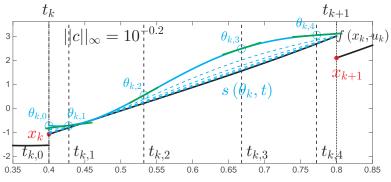
$$\frac{\partial}{\partial t} \mathbf{s}(\boldsymbol{\theta}_{k}, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k}),$$

with i = 1, ..., K. Note:

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_{k},t\right)=\sum_{j=0}^{K}\boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t)$$

$$\sum_{i=0}^K \theta_{k,i} P_{k,i}(t_k) = \mathbf{x}_k$$

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



Collocation uses the constraints:

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

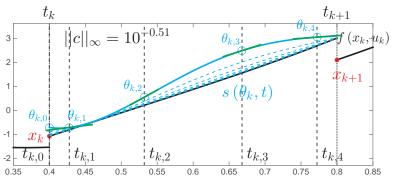
$$\frac{\partial}{\partial t} \mathbf{s}(\boldsymbol{\theta}_{k}, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k}),$$

with i = 1, ..., K. Note:

$$\frac{\partial}{\partial t}\mathbf{s}\left(\boldsymbol{\theta}_{k},t\right)=\sum_{j=0}^{K}\boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t)$$

$$\sum_{i=0}^K \theta_{k,i} P_{k,i}(t_k) = \mathbf{x}_k$$

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



Collocation uses the constraints:

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

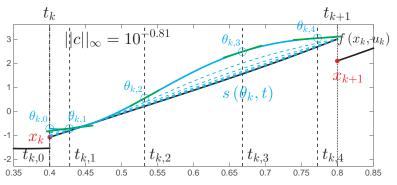
$$\frac{\partial}{\partial t} \mathbf{s}(\boldsymbol{\theta}_{k}, t_{k,i}) = \mathbf{F}(\boldsymbol{\theta}_{k,i}, \mathbf{u}_{k}),$$

with i = 1, ..., K. Note:

$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k,t) = \sum_{j=0}^K \boldsymbol{\theta}_{k,j}\dot{P}_{k,j}(t)$$

$$\sum_{i=0}^K \frac{\theta_{k,i}}{P_{k,i}}(t_k) = \mathbf{x}_k$$

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



Collocation uses the constraints:

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

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

with i = 1, ..., K. Note:

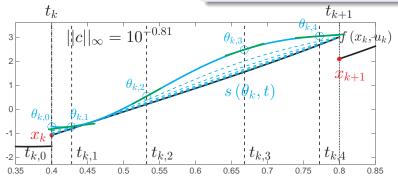
$$\frac{\partial}{\partial t}\mathbf{s}(\boldsymbol{\theta}_k,t) = \sum_{j=0}^K \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t)$$

Shooting constraints

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

becomes:

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



Collocation constraints...

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

$$\sum_{j=0}^{K} \frac{\theta_{k,j} \dot{P}_{k,j}(t_{k,i})}{\sum_{j=0}^{K} \left(\theta_{k,i}, \mathbf{u}_{k}\right), \ i = 1, ..., K$$

Collocation constraints...

$$\sum_{i=0}^{K} \theta_{k,i} P_{k,i}(t_k) = \mathbf{x}_k$$

$$\sum_{j=0}^{\mathcal{K}} oldsymbol{ heta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F}\left(oldsymbol{ heta}_{k,i}, \mathbf{u}_{k}
ight), \; i=1,...,\mathcal{K}$$

.. can be seen as

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

Collocation constraints...

$$\begin{split} &\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} P_{k,i}(t_k) = \mathbf{x}_k \\ &\sum_{j=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F} \left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k \right), \ i = 1, ..., K \end{split}$$

... can be seen as

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

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

Collocation constraints...

$$\begin{split} & \sum_{i=0}^{N} \boldsymbol{\theta}_{k,i} P_{k,i}(t_k) = \mathbf{x}_k \\ & \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} \dot{P}_{k,i}(t_{k,i}) = \mathbf{F} \left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k \right), \ i = 1, ..., K \end{split}$$

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

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

$$\Delta oldsymbol{ heta}_k = -rac{\partial \mathbf{c} \left(\mathbf{x}_k, oldsymbol{ heta}_k, \mathbf{u}_k
ight)^{-1} \mathbf{c} \left(\mathbf{x}_k, oldsymbol{ heta}_k, \mathbf{u}_k
ight)}{\partial oldsymbol{ heta}_k}$$

Collocation constraints...

$$\begin{split} \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} P_{k,i}(t_k) &= \mathbf{x}_k \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) &= \mathbf{F} \left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k \right), \ i = 1, ..., K \end{split}$$

Integrator function given by:

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

Get sensitivities using:

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

can be seen as

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

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

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

Collocation constraints...

$$\begin{split} \sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} P_{k,i}(t_k) &= \mathbf{x}_k \\ \sum_{i=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) &= \mathbf{F} \left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k \right), \ i = 1, ..., K \end{split}$$

Integrator function given by:

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

Get sensitivities using:

$$\frac{\partial f\left(x_{k}, u_{k}\right)}{\partial x_{k}} = \frac{\partial s\left(\theta_{k}, t_{k+1}\right)}{\partial \theta_{k}} \frac{\partial \theta_{k}}{\partial x_{k}}, \qquad \frac{\partial f\left(x_{k}, u_{k}\right)}{\partial u_{k}} = \frac{\partial s\left(\theta_{k}, t_{k+1}\right)}{\partial \theta_{k}} \frac{\partial \theta_{k}}{\partial u_{k}}$$

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

Implicit function theorem states that

$$\frac{\partial c}{\partial \theta_k} \frac{\partial \theta_k}{\partial x_k} + \frac{\partial c}{\partial x_k} = 0, \qquad \frac{\partial c}{\partial \theta_k} \frac{\partial \theta_k}{\partial u_k} + \frac{\partial c}{\partial u_k} = 0$$

... can be seen as

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

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

Collocation constraints...

$$\sum_{i=0}^{K} \frac{\theta_{k,i} P_{k,i}(t_k) = \mathbf{x}_k}{\sum_{i=0}^{K} \frac{\theta_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F} \left(\theta_{k,i}, \mathbf{u}_k\right), i = 1, ..., K}$$

... can be seen as

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

Solved by iterating:

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

Integrator function given by:

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

Get sensitivities using:

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

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

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 constraints

$$\begin{split} &\sum_{i=0}^{K} \boldsymbol{\theta}_{k,i} P_{k,i}(t_k) = \mathbf{x}_k \\ &\sum_{i=0}^{K} \boldsymbol{\theta}_{k,j} \dot{P}_{k,j}(t_{k,i}) = \mathbf{F} \left(\boldsymbol{\theta}_{k,i}, \mathbf{u}_k \right), \ i = 1, ..., K \end{split}$$

... can be seen as

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

Solved by iterating:

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

Integrator function given by:

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

Get sensitivities using:

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

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

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:

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

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

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

Outline

- 1 Introduction
- 2 Integration with sensitivities Variation approach
- 3 Integration with sensitivities Altered the Sentiation
- 4 Collocation-based integrator
- 6 Adjoint-mode sensitivity
- 6 Second-order sensitivity

Consider f(x, u) given by

Algorithm: function f(x, u)

Input: $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$

s = x

for i = 1.N do

 $\mathbf{s} \leftarrow \boldsymbol{\xi} \left(\mathbf{s}, \mathbf{u} \right)$

return f(x, u) = s

Note: e.g. ξ reads as $s + \Delta tF$ in Euler

Consider f(x, u) given by

Algorithm: function f(x, u)

Input: $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ s = xfor i = 1.N do

 $\mathbf{s} \leftarrow \boldsymbol{\xi} \left(\mathbf{s}, \mathbf{u} \right)$ return f(x, u) = s

Note: e.g. ξ reads as $s + \Delta tF$ in Fuler

Forward sensitivity computation:

Algorithm: function f(x, u) with forward sensitivity

Input: $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$

s = x, A = I, B = 0

for i = 1 : N do

$$A \leftarrow \frac{\partial \boldsymbol{\xi}(\mathbf{s},\mathbf{u})}{\partial \mathbf{s}} A$$

$$\begin{bmatrix} A \leftarrow \frac{\partial \boldsymbol{\xi}(\mathbf{s}, \mathbf{u})}{\partial \mathbf{s}} A \\ B \leftarrow \frac{\partial \boldsymbol{\xi}(\mathbf{s}, \mathbf{u})}{\partial \mathbf{s}} B + \frac{\partial \boldsymbol{\xi}(\mathbf{s}, \mathbf{u})}{\partial \mathbf{u}} \\ \mathbf{s} \leftarrow \boldsymbol{\zeta}(\mathbf{s}, \mathbf{u}) \end{bmatrix}$$

return
$$f(x, u) = s$$
, $\frac{\partial f(x, u)}{\partial x} = A$, $\frac{\partial f(x, u)}{\partial u} = B$

Consider f(x, u) given by

Algorithm: function f(x, u)

Input: $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ s = x

for i = 1.N do $\mathbf{s} \leftarrow \boldsymbol{\xi} \left(\mathbf{s}, \mathbf{u} \right)$

return f(x, u) = s

Note: e.g. ξ reads as $s + \Delta tF$ in Fuler

Forward sensitivity computation:

Algorithm: function f(x, u) with forward sensitivity

Input: $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ s = x, A = I, B = 0

for i = 1 : N do

$$\begin{bmatrix} A \leftarrow \frac{\partial \boldsymbol{\xi}(\mathbf{s}, \mathbf{u})}{\partial \mathbf{s}} A \\ B \leftarrow \frac{\partial \boldsymbol{\xi}(\mathbf{s}, \mathbf{u})}{\partial \mathbf{s}} B + \frac{\partial \boldsymbol{\xi}(\mathbf{s}, \mathbf{u})}{\partial \mathbf{u}} \\ \mathbf{s} \leftarrow \boldsymbol{\zeta}(\mathbf{s}, \mathbf{u}) \end{bmatrix}$$

return f(x, u) = s, $\frac{\partial f(x, u)}{\partial x} = A$, $\frac{\partial f(x, u)}{\partial u} = B$

- \bullet $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, often dense
- Memory footprint n(n+m) floating point numbers
- N dense matrix-matrix multiplications to build A, complexity Nn^3
- N dense matrix-matrix multiplications to build B, complexity Nn^2m
- Total complexity $Nn^2(n+m)$

Consider f(x, u) given by

Algorithm: function f(x, u)

Input:
$$\mathbf{x} \in \mathbb{R}^n$$
, $\mathbf{u} \in \mathbb{R}^m$
 $\mathbf{s} = \mathbf{x}$
for $i = 1:N$ do

$$\mathbf{return} \ \mathbf{f} \left(\mathbf{x}, \mathbf{u} \right) = \mathbf{s}$$

Note: e.g. ξ reads as $s + \Delta tF$ in Fuler

Forward sensitivity computation:

Algorithm: function f(x, u) with forward sensitivity

Input:
$$\mathbf{x} \in \mathbb{R}^n$$
, $\mathbf{u} \in \mathbb{R}^m$
 $\mathbf{s} = \mathbf{x}$, $A = I$, $B = 0$

for
$$i = 1 : N$$
 do

$$\begin{bmatrix} A \leftarrow \frac{\partial \boldsymbol{\xi}(\mathbf{s}, \mathbf{u})}{\partial \mathbf{s}} A \\ B \leftarrow \frac{\partial \boldsymbol{\xi}(\mathbf{s}, \mathbf{u})}{\partial \mathbf{s}} B + \frac{\partial \boldsymbol{\xi}(\mathbf{s}, \mathbf{u})}{\partial \mathbf{u}} \\ \mathbf{s} \leftarrow \boldsymbol{\zeta}(\mathbf{s}, \mathbf{u}) \end{bmatrix}$$

$$\begin{bmatrix} B \leftarrow \frac{\neg \varsigma}{\partial s} B + \frac{\neg \varsigma}{\partial u} \\ s \leftarrow \zeta(s, u) \end{bmatrix}$$

return
$$f(x, u) = s$$
, $\frac{\partial f(x, u)}{\partial x} = A$, $\frac{\partial f(x, u)}{\partial u} = B$

- \bullet $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, often dense
- Memory footprint n(n+m) floating point numbers
- N dense matrix-matrix multiplications to build A, complexity Nn^3
- N dense matrix-matrix multiplications to build B, complexity Nn^2m
- Total complexity $Nn^2(n+m)$

What if we have a function $\zeta: \mathbb{R}^n \to \mathbb{R}$, and care only about $\frac{\partial \zeta(f(x,u))}{\partial x} \in \mathbb{R}^n$, $\frac{\partial \zeta(f(x,u))}{\partial x} \in \mathbb{R}^m$?

Consider T(x, u) given by

Algorithm: function T(x, u)

$$\label{eq:solution} \begin{split} & \overline{\textbf{Input:}} \ \mathbf{s} \in \mathbb{R}^n, \quad \mathbf{u} \in \mathbb{R}^m \\ & \mathbf{s}_0 = \mathbf{x} \\ & \text{for } i = 1:N \ \mathbf{do} \\ & \mid \quad \mathbf{s}_i = \boldsymbol{\xi} \left(\mathbf{s}_{i-1}, \mathbf{u} \right) \end{split}$$

return $T(\mathbf{x}, \mathbf{u}) = \zeta(\mathbf{s}_N)$

Can we compute $\frac{\partial T(x,u)}{\partial x}$, $\frac{\partial T(x,u)}{\partial u}$ more efficiently than by doing forward sensitivity ? Yes, use the adjoint mode.

Consider $T(\mathbf{x}, \mathbf{u})$ given by

Define $\lambda_i^{\top} = \frac{\partial \zeta(\mathbf{s}_N)}{\partial \mathbf{s}_i}$

Algorithm: function T(x, u)

Input: $\mathbf{s} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$

 $\mathbf{s}_0 = \mathbf{x}$

for i = 1:N do

 $\textbf{return}\ \ \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)=\zeta\left(\mathbf{s}_{\textit{N}}\right)$

Can we compute $\frac{\partial T(x,u)}{\partial x}$, $\frac{\partial T(x,u)}{\partial u}$ more efficiently than by doing forward sensitivity ? Yes, use the adjoint mode.

Consider T(x, u) given by

Define $\lambda_i^{\top} = \frac{\partial \zeta(\mathbf{s}_N)}{\partial \mathbf{s}_i}$, then

Algorithm: function T(x, u)

 $oldsymbol{\lambda}_{N}^{ op}=rac{\partial\zeta\left(\mathbf{s}_{N}
ight)}{\partial\mathbf{s}_{N}}$

$$\label{eq:solution} \begin{split} &\overline{\text{Input:}} \ s \in \mathbb{R}^n, \quad \mathbf{u} \in \mathbb{R}^m \\ & s_0 = \mathbf{x} \\ & \text{for } i = 1 \text{:} N \ \text{do} \\ & \quad \ \ \, \bigsqcup \ s_i = \boldsymbol{\xi} \left(s_{i-1}, \mathbf{u} \right) \end{split}$$

return $T(\mathbf{x}, \mathbf{u}) = \zeta(\mathbf{s}_N)$

Can we compute $\frac{\partial \mathcal{T}(x,u)}{\partial x}$, $\frac{\partial \mathcal{T}(x,u)}{\partial u}$ more efficiently than by doing forward sensitivity ? Yes, use the adjoint mode.

Consider $T(\mathbf{x}, \mathbf{u})$ given by

Algorithm: function T(x, u)

Input: $\mathbf{s} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$

 $\mathbf{s}_0 = \mathbf{x}$

for i = 1:N do

 $\mathbf{return}\ \, \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)=\zeta\left(\mathbf{s}_{N}\right)$

Can we compute $\frac{\partial T(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}$, $\frac{\partial T(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}$ more efficiently than by doing forward sensitivity ? Yes, use the adjoint mode.

Define
$$\lambda_{i}^{\top} = \frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{i}}$$
, then
$$\lambda_{N}^{\top} = \frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{N}}$$

$$\lambda_{i-1}^{\top} = \frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{i-1}} = \underbrace{\frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{i}}}_{\underbrace{\partial \mathbf{s}_{i}}} \cdot \underbrace{\frac{\partial \mathbf{s}_{i}}{\partial \mathbf{s}_{i-1}}}_{\underbrace{\partial \mathbf{s}_{i-1}}}$$

Consider T(x, u) given by

Algorithm: function T(x, u)

Input: $\mathbf{s} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$

 $\mathbf{s}_0 = \mathbf{x}$

for i = 1:N do

$$\mathbf{L} \mathbf{s}_i = \boldsymbol{\xi} \left(\mathbf{s}_{i-1}, \mathbf{u} \right)$$

 $\underline{\mathsf{return}\ T\left(\mathbf{x},\mathbf{u}\right) = \zeta\left(\mathbf{s}_{N}\right)}$

Can we compute $\frac{\partial T(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}$, $\frac{\partial T(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}$ more efficiently than by doing forward sensitivity ? Yes, use the adjoint mode.

Define
$$\lambda_{i}^{\top} = \frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{i}}$$
, then
$$\lambda_{N}^{\top} = \frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{N}}$$
$$\lambda_{i-1}^{\top} = \frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{i-1}} = \underbrace{\frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{i}}}_{\lambda_{i}^{\top}} \cdot \underbrace{\frac{\partial \mathbf{s}_{i}}{\partial \mathbf{s}_{i-1}}}$$

Consider $T(\mathbf{x}, \mathbf{u})$ given by

Algorithm: function T(x, u)

Input: $\mathbf{s} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$

 $\mathbf{s}_0 = \mathbf{x}$

for i = 1:N do

Can we compute $\frac{\partial T(x,u)}{\partial x}$, $\frac{\partial T(x,u)}{\partial u}$ more efficiently than by doing forward sensitivity ? Yes, use the adjoint mode.

Define
$$\lambda_{i}^{\top} = \frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{i}}$$
, then
$$\lambda_{N}^{\top} = \frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{N}}$$

$$\lambda_{i-1}^{\top} = \frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{i-1}} = \underbrace{\frac{\partial \zeta(\mathbf{s}_{N})}{\partial \mathbf{s}_{i}}}_{\lambda_{i}^{\top}} \cdot \underbrace{\frac{\partial \mathbf{s}_{i}}{\partial \mathbf{s}_{i-1}}}_{\frac{\partial \xi(\mathbf{s}_{i-1},\mathbf{u})}{\partial \mathbf{s}_{i}}}$$

Adjoint-mode AD (reverse mode) - key idea

Consider T(x, u) given by

Algorithm: function T(x, u)

Input:
$$\mathbf{s} \in \mathbb{R}^n$$
, $\mathbf{u} \in \mathbb{R}^m$

$$\mathbf{s}_0 = \mathbf{x}$$

for
$$i = 1:N$$
 do

Can we compute $\frac{\partial T(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}}$, $\frac{\partial T(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}}$ more efficiently than by doing forward sensitivity ? Yes, use the adjoint mode.

Define $oldsymbol{\lambda}_i^{ op} = rac{\partial \zeta(\mathbf{s}_N)}{\partial \mathbf{s}_i}$, then

$$\boldsymbol{\lambda}_{N}^{\top} = \frac{\partial \zeta \left(\mathbf{s}_{N} \right)}{\partial \mathbf{s}_{N}}$$

$$\boldsymbol{\lambda}_{i-1}^{\top} = \frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{s}_{i-1}} = \underbrace{\frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{s}_{i}}}_{\boldsymbol{\lambda}_{i}^{\top}} \cdot \underbrace{\frac{\partial \mathbf{s}_{i}}{\partial \mathbf{s}_{i-1}}}_{\partial \boldsymbol{s}_{i}(\mathbf{s}_{i-1})}$$

Moreover

$$\frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{u}} = \sum_{i=1}^{N} \frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{s}_{i}} \frac{\partial \mathbf{s}_{i}}{\partial \mathbf{u}} =$$

$$\sum_{i=1}^{N} \frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{s}_{i}} \frac{\partial \boldsymbol{\xi}\left(\mathbf{s}_{i-1}, \mathbf{u}\right)}{\partial \mathbf{u}} = \sum_{i=1}^{N} \boldsymbol{\lambda}_{i}^{\top} \frac{\partial \boldsymbol{\xi}\left(\mathbf{s}_{i-1}, \mathbf{u}\right)}{\partial \mathbf{u}}$$

Adjoint-mode AD (reverse mode) - key idea

Consider T(x, u) given by

Algorithm: function T(x, u)

Input: $\mathbf{s} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$

 $\mathbf{s}_0 = \mathbf{x}$

for i = 1:N do

$$\mathbf{L} \mathbf{s}_i = \boldsymbol{\xi}(\mathbf{s}_{i-1}, \mathbf{u})$$

 $\underline{\mathsf{return}\ T\left(\mathbf{x},\mathbf{u}\right) = \zeta\left(\mathbf{s}_{\mathit{N}}\right)}$

Can we compute $\frac{\partial T(x,u)}{\partial x}$, $\frac{\partial T(x,u)}{\partial u}$, more efficiently than by doing forward sensitivity ? Yes, use the adjoint mode.

Define $\lambda_i^{ op} = \frac{\partial \zeta(\mathbf{s}_N)}{\partial \mathbf{s}_i}$, then

$$oldsymbol{\lambda}_{N}^{ op} = rac{\partial \zeta \left(\mathbf{s}_{N}
ight)}{\partial \mathbf{s}_{N}}$$

$$\boldsymbol{\lambda}_{i-1}^{\top} = \frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{s}_{i-1}} = \underbrace{\frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{s}_{i}}}_{\boldsymbol{\lambda}_{i}^{\top}} \cdot \underbrace{\frac{\partial \mathbf{s}_{i}}{\partial \mathbf{s}_{i-1}}}_{\frac{\partial \boldsymbol{\varepsilon}\left(\mathbf{s}_{i-1},\mathbf{u}\right)}{\partial \mathbf{s}_{i}}}$$

Moreover

$$\frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{u}} = \sum_{i=1}^{N} \frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{s}_{i}} \frac{\partial \mathbf{s}_{i}}{\partial \mathbf{u}} =$$

$$\sum_{i=1}^{N} \frac{\partial \zeta\left(\mathbf{s}_{N}\right)}{\partial \mathbf{s}_{i}} \frac{\partial \boldsymbol{\xi}\left(\mathbf{s}_{i-1},\mathbf{u}\right)}{\partial \mathbf{u}} = \sum_{i=1}^{N} \boldsymbol{\lambda}_{i}^{\top} \frac{\partial \boldsymbol{\xi}\left(\mathbf{s}_{i-1},\mathbf{u}\right)}{\partial \mathbf{u}}$$

$$\begin{split} &\frac{\partial \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)}{\partial \mathbf{x}} = \boldsymbol{\lambda}_{0}^{\top} \\ &\frac{\partial \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)}{\partial \mathbf{u}} = \sum_{i=1}^{N} \boldsymbol{\lambda}_{i}^{\top} \frac{\partial \boldsymbol{\xi}\left(\mathbf{s}_{i-1},\mathbf{u}\right)}{\partial \mathbf{u}} \end{split}$$

$$oldsymbol{\lambda}_{i}^{ op} = oldsymbol{\lambda}_{i+1}^{ op} rac{\partial \xi \left(\mathbf{s}_{i}, \mathbf{u}
ight)}{\partial \mathbf{s}_{i}} \ oldsymbol{\lambda}_{N}^{ op} = rac{\partial \zeta \left(\mathbf{s}_{N}
ight)}{\partial \mathbf{s}_{N}}$$

Consider $T(\mathbf{x}, \mathbf{u})$ given by

Algorithm: function T(x, u)

Input: $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$

 $s_0 = x$

for
$$i = 1:N$$
 do

return

$$\mathcal{T}\left(\mathbf{x},\mathbf{u}\right)=\zeta\left(\mathbf{s}_{\textit{N}}\right),\ \mathbf{s}_{0,...,\textit{N}}$$

$$\begin{split} &\frac{\partial \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)}{\partial \mathbf{x}} = \boldsymbol{\lambda}_{0}^{\top} \\ &\frac{\partial \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)}{\partial \mathbf{u}} = \sum_{i=1}^{N} \boldsymbol{\lambda}_{i}^{\top} \frac{\partial \boldsymbol{\xi}\left(\mathbf{s}_{i-1},\mathbf{u}\right)}{\partial \mathbf{u}} \end{split}$$

$$\boldsymbol{\lambda}_{i}^{\top} = \frac{\partial \boldsymbol{\xi} \left(\mathbf{s}_{i}, \mathbf{u} \right)}{\partial \mathbf{s}_{i}} \boldsymbol{\lambda}_{i+1}^{\top}$$
$$\boldsymbol{\lambda}_{N}^{\top} = \frac{\partial \zeta \left(\mathbf{s}_{N} \right)}{\partial \mathbf{s}_{N}}$$

$$\mathbf{\lambda}_{N}^{\top} = \frac{\partial \zeta \left(\mathbf{s}_{N} \right)}{\partial \mathbf{s}_{N}}$$

Consider $T(\mathbf{x}, \mathbf{u})$ given by

Algorithm: function T(x, u)

Input:
$$\mathbf{x} \in \mathbb{R}^n$$
, $\mathbf{u} \in \mathbb{R}^m$

$$\mathbf{s}_0 = \mathbf{x}$$

for
$$i = 1:N$$
 do $\mathbf{s}_i = \boldsymbol{\xi}(\mathbf{s}_{i-1}, \mathbf{u})$

return

$$T(\mathbf{x}, \mathbf{u}) = \zeta(\mathbf{s}_N), \ \mathbf{s}_{0,\dots,N}$$

Algorithm: adjoint-mode sensitivity of function

$$T(\mathbf{x}, \mathbf{u})$$

Input:
$$s_{0,...,N}$$
, u

$$\lambda_N = \nabla \zeta(\mathbf{s}_N), \ \boldsymbol{\sigma} = 0$$

for
$$i = N-1:0$$
 do

$$\sigma = \sigma \perp \nabla$$

$$egin{aligned} oldsymbol{\sigma} &= oldsymbol{\sigma} +
abla_{\mathbf{u}} oldsymbol{\xi}\left(\mathbf{s}_i, \mathbf{u}
ight) oldsymbol{\lambda}_{i+1} \ oldsymbol{\lambda}_i &=
abla_{\mathbf{s}_i} oldsymbol{\xi}\left(\mathbf{s}_i, \mathbf{u}
ight) oldsymbol{\lambda}_{i+1} \end{aligned}$$

$$\mathsf{return} \; \nabla_{\mathbf{x}} \mathcal{T} \left(\mathbf{x}, \mathbf{u} \right) = \boldsymbol{\lambda}_0, \quad \nabla_{\mathbf{u}} \mathcal{T} \left(\mathbf{x}, \mathbf{u} \right) = \boldsymbol{\sigma}$$

$$\begin{split} &\frac{\partial \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)}{\partial \mathbf{x}} = \boldsymbol{\lambda_0}^\top \\ &\frac{\partial \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)}{\partial \mathbf{u}} = \sum_{i=1}^{N} \boldsymbol{\lambda_i}^\top \frac{\partial \boldsymbol{\xi}\left(\mathbf{s}_{i-1},\mathbf{u}\right)}{\partial \mathbf{u}} \end{split}$$

$$oldsymbol{\lambda}_i^{ op} = rac{\partial oldsymbol{\xi}\left(\mathbf{s}_i,\mathbf{u}
ight)}{\partial \mathbf{s}_i} oldsymbol{\lambda}_{i+1}^{ op}$$

$$oldsymbol{\lambda}_{N}^{ op} = rac{\partial \zeta \left(\mathbf{s}_{N}
ight)}{\partial \mathbf{s}_{N}}$$

Consider $T(\mathbf{x}, \mathbf{u})$ given by

Algorithm: function T(x, u)

Input: $\mathbf{x} \in \mathbb{R}^n$. $\mathbf{u} \in \mathbb{R}^m$

$$s_0 = x$$

for $i = 1:N$ do

for
$$i = 1:N$$
 do
 $\mathbf{s}_i = \boldsymbol{\xi}(\mathbf{s}_{i-1}, \mathbf{u})$

return

$$T(\mathbf{x}, \mathbf{u}) = \zeta(\mathbf{s}_N), \ \mathbf{s}_{0,\dots,N}$$

Algorithm: adjoint-mode sensitivity of function

$$T(\mathbf{x}, \mathbf{u})$$

$$\lambda = \nabla \zeta \left(\mathbf{s}_{N} \right), \ \boldsymbol{\sigma} = 0$$

for
$$i = N-1:0$$
 do

$$egin{aligned} oldsymbol{\sigma} \leftarrow oldsymbol{\sigma} +
abla_{\mathbf{u}} oldsymbol{\xi} \left(\mathbf{s}_i, \mathbf{u}
ight) oldsymbol{\lambda} \ oldsymbol{\lambda} \leftarrow
abla_{\mathbf{s}_i} oldsymbol{\xi} \left(\mathbf{s}_i, \mathbf{u}
ight) oldsymbol{\lambda} \end{aligned}$$

$$\lambda \leftarrow \nabla_{\mathbf{s}_i} \boldsymbol{\xi} \left(\mathbf{s}_i, \mathbf{u} \right) \lambda$$

$$\mathsf{return}\ \nabla_{x}\,\mathcal{T}\left(x,u\right) = \boldsymbol{\lambda},\quad \nabla_{u}\,\mathcal{T}\left(x,u\right) = \boldsymbol{\sigma}$$

$$\begin{split} &\frac{\partial \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)}{\partial \mathbf{x}} = \boldsymbol{\lambda}_{0}^{\top} \\ &\frac{\partial \mathcal{T}\left(\mathbf{x},\mathbf{u}\right)}{\partial \mathbf{u}} = \sum_{i=1}^{N} \boldsymbol{\lambda}_{i}^{\top} \frac{\partial \boldsymbol{\xi}\left(\mathbf{s}_{i-1},\mathbf{u}\right)}{\partial \mathbf{u}} \end{split}$$

$$oldsymbol{\lambda}_i^ op = rac{\partial oldsymbol{\xi}\left(\mathbf{s}_i,\mathbf{u}
ight)}{\partial \mathbf{s}_i} oldsymbol{\lambda}_{i+1}^ op$$

$$oldsymbol{\lambda}_{N}^{ op} = rac{\partial \zeta \left(\mathbf{s}_{N}
ight)}{\partial \mathbf{s}_{N}}$$

Consider $T(\mathbf{x}, \mathbf{u})$ given by

Algorithm: function T(x, u)

Input: $\mathbf{x} \in \mathbb{R}^n$. $\mathbf{u} \in \mathbb{R}^m$

 $s_0 = x$

for i = 1:N do

 $\mathbf{s}_i = \boldsymbol{\xi} \left(\mathbf{s}_{i-1}, \mathbf{u} \right)$

return

$$T\left(\mathbf{x},\mathbf{u}\right) = \zeta\left(\mathbf{s}_{N}\right), \ \mathbf{s}_{0,...,N}$$

Algorithm: adjoint-mode sensitivity of function

 $T(\mathbf{x}, \mathbf{u})$

Input: s_0, \dots, N , u

 $\lambda = \nabla \zeta(\mathbf{s}_N), \ \boldsymbol{\sigma} = 0$

for i = N-1.0 do

 $egin{aligned} oldsymbol{\sigma} \leftarrow oldsymbol{\sigma} +
abla_{\mathbf{u}} oldsymbol{\xi}\left(\mathbf{s}_{i}, \mathbf{u}
ight) oldsymbol{\lambda} \ oldsymbol{\lambda} \leftarrow
abla_{\mathbf{s}_{i}} oldsymbol{\xi}\left(\mathbf{s}_{i}, \mathbf{u}
ight) oldsymbol{\lambda} \end{aligned}$

return $\nabla_{\mathbf{x}} T(\mathbf{x}, \mathbf{u}) = \lambda$, $\nabla_{\mathbf{u}} T(\mathbf{x}, \mathbf{u}) = \sigma$

- Two passes algorithm !! First do the forward sweep (compute $s_0 = N$), then do the backward sweep (compute λ , σ)
- Forward sweep needs to store s_0 , memory footprint Nn.
- Forward sweep inexpensive
- Backward sweep requires N matrix-vector multiplication to build λ , complexity n^2
- Backward sweep requires N matrix-vector multiplication to build σ , complexity np
- Total complexity Nn(n+p), i.e. n times less than the forward mode

$$\frac{\partial T(\mathbf{x}, \mathbf{u})}{\partial \mathbf{x}} = \boldsymbol{\lambda}_0^{\top}$$

$$\frac{\partial \mathbf{x}}{\partial \mathbf{u}} = \sum_{i=1}^{N} \lambda_{i}^{\top} \frac{\partial \boldsymbol{\xi} \left(\mathbf{s}_{i-1}, \mathbf{u} \right)}{\partial \mathbf{u}}$$

$$\boldsymbol{\lambda}_{i}^{\top} = \frac{\partial \boldsymbol{\xi}\left(\mathbf{s}_{i}, \mathbf{u}\right)}{\partial \mathbf{s}_{i}} \boldsymbol{\lambda}_{i+1}^{\top}$$

$$oldsymbol{\lambda}_{N}^{ op} = rac{\partial \zeta \left(\mathbf{s}_{N}
ight)}{\partial \mathbf{s}_{N}}$$

Outline

- 1 Introduction
- 2 Integration with sensitivities Variation approach
- 3 Integration with sensitivities Alteriation
- 4 Collocation-based integrators
- 6 Adjoint-mode sensitivity
- 6 Second-order sensitivity

NLP with multiple-shooting

$$\begin{aligned} & \underset{w}{\text{min}} \quad \Phi\left(w\right) \\ & \text{s.t.} \quad g\left(w\right) = \left[\begin{array}{c} x_0 - \bar{x}_0 \\ f\left(x_0, u_0\right) - x_1 \\ f\left(x_1, u_1\right) - x_2 \end{array} \right] \end{aligned}$$

$$\mathbf{w} = \{x_0, \mathbf{u}_0, ..., x_{N-1}, \mathbf{u}_{N-1}, x_N\}$$

NLP with multiple-shooting

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

s.t.
$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} \mathbf{x}_0 - \overline{\mathbf{x}}_0 \\ \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) - \mathbf{x}_1 \\ \mathbf{f}(\mathbf{x}_1, \mathbf{u}_1) - \mathbf{x}_2 \\ \dots \end{bmatrix}$$

with

$$\mathbf{w} = \{x_0, \mathbf{u}_0, ..., x_{N-1}, \mathbf{u}_{N-1}, x_N\}$$

SQP iterates:

$$\min_{\Delta \mathbf{w}} \quad \frac{1}{2} \Delta \mathbf{w}^{\top} B \Delta \mathbf{w} + \nabla \Phi^{\top} \Delta \mathbf{w}$$

s.t.
$$\nabla \mathbf{g}^{\top} \Delta \mathbf{w} + \mathbf{g} = 0$$

where
$$B = \nabla_{\mathbf{w}}^2 \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda})$$



NLP with multiple-shooting

$$\begin{aligned} & \underset{w}{\text{min}} \quad \Phi\left(w\right) \\ & \text{s.t.} \quad g\left(w\right) = \left[\begin{array}{c} x_0 - \overline{x}_0 \\ f\left(x_0, u_0\right) - x_1 \\ f\left(x_1, u_1\right) - x_2 \end{array} \right] \end{aligned}$$

$$\mathcal{L}\left(\mathbf{w}, \boldsymbol{\lambda}\right) = \phi\left(\mathbf{w}\right) + \sum_{k=0}^{N-1} \boldsymbol{\lambda}_{k+1}^{\top} \left(\mathbf{f}\left(\mathbf{x}_{k}, \mathbf{u}_{k}\right) - \mathbf{x}_{k+1}\right) + \boldsymbol{\lambda}_{0}^{\top} \left(\mathbf{x}_{0} - \bar{\mathbf{x}}_{0}\right)$$

with

$$\mathbf{w} = \{x_0, \mathbf{u}_0, ..., x_{N-1}, \mathbf{u}_{N-1}, x_N\}$$

SQP iterates:

$$\min_{\Delta \mathbf{w}} \quad \frac{1}{2} \Delta \mathbf{w}^{\top} B \Delta \mathbf{w} + \nabla \Phi^{\top} \Delta \mathbf{w}
\text{s.t.} \quad \nabla \mathbf{g}^{\top} \Delta \mathbf{w} + \mathbf{g} = 0$$

where
$$B = \nabla_{\mathbf{w}}^2 \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda})$$

NLP with multiple-shooting

$$\label{eq:started_problem} \begin{aligned} & \underset{w}{\text{min}} \quad \Phi\left(w\right) \\ & \text{s.t.} \quad g\left(w\right) = \left[\begin{array}{c} x_0 - \overline{x}_0 \\ f\left(x_0, u_0\right) - x_1 \\ f\left(x_1, u_1\right) - x_2 \end{array} \right] \end{aligned}$$

with

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

Reminder:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \phi(\mathbf{w}) + \sum_{k=0}^{N-1} \boldsymbol{\lambda}_{k+1}^{\top} (\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_{k+1}) + \boldsymbol{\lambda}_0^{\top} (\mathbf{x}_0 - \bar{\mathbf{x}}_0)$$

Then

$$\begin{split} \nabla_{\mathbf{w}_{k}}^{2} \mathcal{L}\left(\mathbf{w}, \boldsymbol{\lambda}\right) &= \nabla_{\mathbf{w}_{k}}^{2} \phi\left(\mathbf{w}\right) \\ &+ \sum_{k=0}^{N-1} \nabla_{\mathbf{w}_{k}}^{2} \left(\boldsymbol{\lambda}_{k+1}^{\mathsf{T}} \mathbf{f}\left(\mathbf{x}_{k}, \mathbf{u}_{k}\right)\right) \end{split}$$

SQP iterates:

$$\begin{aligned} & \underset{\Delta \mathbf{w}}{\text{min}} & \frac{1}{2} \Delta \mathbf{w}^{\top} B \Delta \mathbf{w} + \nabla \Phi^{\top} \Delta \mathbf{w} \\ & \text{s.t.} & \nabla \mathbf{g}^{\top} \Delta \mathbf{w} + \mathbf{g} = 0 \end{aligned}$$

where
$$B = \nabla_{\mathbf{w}}^2 \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda})$$

with $\mathbf{w}_k = \left\{\mathbf{x}_k, \mathbf{u}_k \right\}, \ k = 0, ..., N-1$

NLP with multiple-shooting

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

s.t.
$$\mathbf{g}(\mathbf{w}) = \begin{bmatrix} x_0 - x_0 \\ f(x_0, \mathbf{u}_0) - x_1 \\ f(x_1, \mathbf{u}_1) - x_2 \\ \dots \end{bmatrix}$$

with

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

SQP iterates:

$$\min_{\Delta \mathbf{w}} \quad \frac{1}{2} \Delta \mathbf{w}^{\top} B \Delta \mathbf{w} + \nabla \Phi^{\top} \Delta \mathbf{w}
\text{s.t.} \quad \nabla \mathbf{g}^{\top} \Delta \mathbf{w} + \mathbf{g} = 0$$

where
$$B = \nabla_{\mathbf{w}}^2 \mathcal{L}(\mathbf{w}, \boldsymbol{\lambda})$$

Reminder:

$$\mathcal{L}(\mathbf{w}, \boldsymbol{\lambda}) = \phi(\mathbf{w}) + \sum_{k=0}^{N-1} \boldsymbol{\lambda}_{k+1}^{\mathsf{T}} (\mathbf{f}(\mathbf{x}_{k}, \mathbf{u}_{k}) - \mathbf{x}_{k+1}) + \boldsymbol{\lambda}_{0}^{\mathsf{T}} (\mathbf{x}_{0} - \bar{\mathbf{x}}_{0})$$

Then

$$\begin{split} \nabla_{\mathbf{w}_{k}}^{2} \mathcal{L}\left(\mathbf{w}, \boldsymbol{\lambda}\right) &= \nabla_{\mathbf{w}_{k}}^{2} \phi\left(\mathbf{w}\right) \\ &+ \sum_{k=0}^{N-1} \nabla_{\mathbf{w}_{k}}^{2} \left(\boldsymbol{\lambda}_{k+1}^{\mathsf{T}} \mathbf{f}\left(\mathbf{x}_{k}, \mathbf{u}_{k}\right)\right) \end{split}$$

with $\mathbf{w}_k = \{\mathbf{x}_k, \mathbf{u}_k\}, \ k = 0, ..., N-1$

To deploy an SQP method with exact Hessian, we need to compute

$$\nabla^2_{\mathbf{w}_k} \left(\boldsymbol{\lambda}_{k+1}^{\mathsf{T}} \mathbf{f} \left(\mathbf{x}_k, \mathbf{u}_k \right) \right)$$

for
$$k = 0, ..., N - 1 !!$$

To make it simple we ignore the input $\mathbf{u}.$ Let's

$$T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\mu}^{\top} \mathbf{f}(\mathbf{x}) \quad (\in \mathbb{R})$$

We can compute the sensitivity of T using the adjoint mode, then:

$$\nabla_{\mathbf{x}} T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\lambda}$$

and

$$\nabla_{\mathbf{x}}^{2}T\left(\mathbf{x},\boldsymbol{\mu}\right)=\nabla_{\mathbf{x}}\boldsymbol{\lambda}$$

To make it simple we ignore the input u. Let's

look at:
$$T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\mu}^{\top} \mathbf{f}(\mathbf{x}) \quad (\in \mathbb{R})$$

We can compute the sensitivity of T using the adjoint mode, then:

$$\nabla_{\mathbf{x}} T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\lambda}$$

and

$$\nabla_{\mathbf{x}}^{2}T(\mathbf{x},\boldsymbol{\mu})=\nabla_{\mathbf{x}}\boldsymbol{\lambda}$$

Algorithm: Forward AD

Input: \mathbf{x} $\mathbf{s}_0 = \mathbf{x}, \ A_0 = I$ for i = 0:N-1 do $A_{i+1} = \frac{\partial \xi(\mathbf{s}_i)}{\partial \mathbf{s}_i} A_i$ $\mathbf{s}_{i+1} = \xi(\mathbf{s}_i)$

return $s_{0,...,N}$, $A_{0,...,N}$

To make it simple we ignore the input ${\bf u}$. Let's look at:

ook at:
$$T\left(\mathbf{x},oldsymbol{\mu}
ight)=oldsymbol{\mu}^{ op}\mathbf{f}\left(\mathbf{x}
ight) \quad (\in\mathbb{R})$$

We can compute the sensitivity of T using the adjoint mode, then:

$$\nabla_{\mathbf{x}} T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\lambda}$$

and

$$\nabla_{\mathbf{x}}^{2}T\left(\mathbf{x},\boldsymbol{\mu}\right)=\nabla_{\mathbf{x}}\boldsymbol{\lambda}$$

Algorithm: Forward AD

Input: x $s_0 = x$.

$$s_0 = x$$
, $A_0 = I$
for $i = 0$: N-1 do

$$A_{i+1} = \frac{\partial \boldsymbol{\xi}(\mathbf{s}_i)}{\partial \mathbf{s}_i} A_i$$
$$\mathbf{s}_{i+1} = \boldsymbol{\xi}(\mathbf{s}_i)$$

return $s_{0,...,N}$, $A_{0,...,N}$

Reminder (here $\zeta(\mathbf{x}) = \boldsymbol{\mu}^{\top}\mathbf{x}$):

Algorithm: Adjoint-mode AD

Input:
$$\mathbf{s}_{0,...,N}$$
, μ

$$\lambda = \nabla \zeta(\mathbf{s}_N) = \mu$$

for
$$i = N-1:0$$
 do $\lambda \leftarrow \nabla_{s_i} \boldsymbol{\xi}(s_i) \lambda$

return λ

To make it simple we ignore the input ${\bf u}$. Let's look at:

pook at:
$$T\left(\mathbf{x},oldsymbol{\mu}
ight) = oldsymbol{\mu}^{ op}\mathbf{f}\left(\mathbf{x}
ight) \quad (\in \mathbb{R})$$

We can compute the sensitivity of T using the adjoint mode, then:

$$\nabla_{\mathbf{x}} T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\lambda}$$

and

$$\nabla_{\mathbf{x}}^{2}T\left(\mathbf{x},\boldsymbol{\mu}\right)=\nabla_{\mathbf{x}}\boldsymbol{\lambda}$$

Algorithm: Forward AD

Input: x $\mathbf{s}_0 = \mathbf{x}, \ A_0 = I$ for i = 0:N-1 do $\begin{vmatrix} A_{i+1} = \frac{\partial \boldsymbol{\xi}(\mathbf{s}_i)}{\partial \mathbf{s}_i} A_i \\ \mathbf{s}_{i+1} = \boldsymbol{\xi}(\mathbf{s}_i) \end{vmatrix}$

return $s_{0,...,N}$, $A_{0,...,N}$

Reminder (here $\zeta(\mathbf{x}) = \boldsymbol{\mu}^{\top}\mathbf{x}$):

Algorithm: Adjoint-mode AD

 $\begin{array}{l} \text{Input: } \mathbf{s}_{0,...,N}, \ \boldsymbol{\mu} \\ \boldsymbol{\lambda} = \nabla \zeta \left(\mathbf{s}_{N} \right) = \boldsymbol{\mu} \\ \text{for } i = \textit{N-1:0} \ \text{do} \\ \quad \quad \boldsymbol{\lambda} \leftarrow \nabla_{\mathbf{s}_{i}} \boldsymbol{\xi} \left(\mathbf{s}_{i} \right) \boldsymbol{\lambda} \\ \text{return } \boldsymbol{\lambda} \end{array}$

To make it simple we ignore the input ${\bf u}$. Let's look at:

$$T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\mu}^{\top} \mathbf{f}(\mathbf{x}) \quad (\in \mathbb{R})$$

We can compute the sensitivity of T using the adjoint mode, then:

$$\nabla_{\mathbf{x}} T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\lambda}$$

and

$$\nabla_{\mathbf{x}}^{2}T\left(\mathbf{x},\boldsymbol{\mu}\right)=\nabla_{\mathbf{x}}\boldsymbol{\lambda}$$

Algorithm: Forward Over Adjoints

Input:
$$\mathbf{s}_{0,\dots,N}$$
, $A_{0,\dots,N-1}$, μ
 $\lambda_N = \mu$, $H_N = 0$
for $i = N-1:0$ do
$$\begin{vmatrix}
H_i & & \\ \nabla_{\mathbf{s}_i}^2 \left(\lambda_{i+1}^{\top} \xi\left(\mathbf{s}_i\right)\right) A_i + \nabla_{\mathbf{s}_i} \xi\left(\mathbf{s}_i\right) H_{i+1} \\
\lambda_i & = \nabla_{\mathbf{s}_i} \xi\left(\mathbf{s}_i\right) \lambda_{i+1}
\end{vmatrix}$$

return
$$\nabla_{\mathbf{x}}^2 T = H_0$$

Algorithm: Forward AD

$$\begin{split} & \overline{\text{Input: x}} \\ & \mathbf{s}_0 = \mathbf{x}, \ A_0 = I \\ & \mathbf{for} \ i = 0 \text{:} N \text{-} 1 \ \mathbf{do} \\ & A_{i+1} = \frac{\partial \boldsymbol{\xi}(\mathbf{s}_i)}{\partial \mathbf{s}_i} A_i \\ & \mathbf{s}_{i+1} = \boldsymbol{\xi}\left(\mathbf{s}_i\right) \end{split}$$

return $\mathbf{s}_{0,...,N}$, $A_{0,...,N}$

Reminder (here $\zeta(\mathbf{x}) = \boldsymbol{\mu}^{\top} \mathbf{x}$):

Algorithm: Adjoint-mode AD

 $\begin{array}{l} \text{Input: } \mathbf{s}_{0,...,N}, \ \boldsymbol{\mu} \\ \boldsymbol{\lambda} = \nabla \zeta \left(\mathbf{s}_{N} \right) = \boldsymbol{\mu} \\ \text{for } i = \textit{N-1:0} \ \mathbf{do} \\ \quad \quad \boldsymbol{\lambda} \leftarrow \nabla_{\mathbf{s}_{i}} \boldsymbol{\xi} \left(\mathbf{s}_{i} \right) \boldsymbol{\lambda} \\ \text{return } \boldsymbol{\lambda} \end{array}$

To make it simple we ignore the input ${\bf u}$. Let's look at:

$$T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\mu}^{\top} \mathbf{f}(\mathbf{x}) \quad (\in \mathbb{R})$$

We can compute the sensitivity of T using the adjoint mode, then:

$$\nabla_{\mathbf{x}} T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\lambda}$$

and

$$\nabla_{\mathbf{x}}^{2}T\left(\mathbf{x},\boldsymbol{\mu}\right)=\nabla_{\mathbf{x}}\boldsymbol{\lambda}$$

Algorithm: Forward Over Adjoints (F.O.A.)

Input:
$$\mathbf{s}_{0,...,N}$$
, $A_{0,...,N-1}$ and μ

$$\lambda = \mu$$
, $H = 0$
for $i = N-1:0$ do
$$H = \nabla_{\mathbf{s}_{i}}^{2} (\lambda^{\top} \boldsymbol{\xi}(\mathbf{s}_{i})) A_{i} + \nabla_{\mathbf{s}_{i}} \boldsymbol{\xi}(\mathbf{s}_{i}) H$$

$$\lambda = \nabla_{\mathbf{s}_{i}} \boldsymbol{\xi}(\mathbf{s}_{i}) \lambda$$

return
$$\nabla_{\mathbf{x}}^2 T = H$$

Algorithm: Forward AD

 $\begin{aligned} & \overline{\textbf{Input: x}} \\ & \mathbf{s_0} = \mathbf{x}, \ A_0 = I \\ & \mathbf{for} \ i = 0 \text{:} N \text{-} 1 \ \mathbf{do} \\ & A_{i+1} = \frac{\partial \boldsymbol{\xi}(\mathbf{s_i})}{\partial \mathbf{s_i}} A_i \\ & \mathbf{s}_{i+1} = \boldsymbol{\xi}\left(\mathbf{s_i}\right) \end{aligned}$

return $s_{0,...,N}$, $A_{0,...,N}$

Reminder (here $\zeta(\mathbf{x}) = \boldsymbol{\mu}^{\top}\mathbf{x}$):

Algorithm: Adjoint-mode AD

$$\begin{array}{l} \text{Input: } \mathbf{s}_{0,...,N}, \ \boldsymbol{\mu} \\ \boldsymbol{\lambda} = \nabla \zeta \left(\mathbf{s}_{N} \right) = \boldsymbol{\mu} \\ \text{for } i = \textit{N-1:0} \ \mathbf{do} \\ \quad \boldsymbol{\lambda} \leftarrow \nabla_{\mathbf{s}_{i}} \boldsymbol{\xi} \left(\mathbf{s}_{i} \right) \boldsymbol{\lambda} \\ \text{return } \boldsymbol{\lambda} \end{array}$$

To make it simple we ignore the input u. Let's look at:

$$T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\mu}^{\top} \mathbf{f}(\mathbf{x}) \quad (\in \mathbb{R})$$

We can compute the sensitivity of T using the adjoint mode, then:

$$\nabla_{\mathbf{x}} T(\mathbf{x}, \boldsymbol{\mu}) = \boldsymbol{\lambda}$$

and

$$abla_{\mathbf{x}}^{2}T\left(\mathbf{x},\boldsymbol{\mu}\right)=\nabla_{\mathbf{x}}\boldsymbol{\lambda}$$

Algorithm: Forward Over Adjoints (F.O.A.)

Input:
$$\mathbf{s}_{0,...,N},\, A_{0,...,N-1}$$
 and $\boldsymbol{\mu}$ $\boldsymbol{\lambda}=\boldsymbol{\mu},\, H=0$

for
$$i = N-1:0$$
 do
$$\begin{vmatrix}
H = \nabla_{\mathbf{s}_i}^2 (\boldsymbol{\lambda}^{\top} \boldsymbol{\xi}(\mathbf{s}_i)) A_i + \nabla_{\mathbf{s}_i} \boldsymbol{\xi}(\mathbf{s}_i) H \\
\boldsymbol{\lambda} = \nabla_{\mathbf{s}_i} \boldsymbol{\xi}(\mathbf{s}_i) \boldsymbol{\lambda}
\end{vmatrix}$$

$$oldsymbol{\lambda} =
abla_{\mathrm{s}_i} oldsymbol{\xi}\left(\mathrm{s}_i
ight) oldsymbol{\lambda}$$

return
$$\nabla_{\mathbf{x}}^2 T = H$$

- Forward AD pass, store $s_{0,...,N}$, $A_{0,...,N}$
- F.O.A pass, return $\nabla_{\mathbf{x}}^2 T(\mathbf{x}, \lambda)$
- Lower computational complexity possible with reformulation (CDC 2013) !!

Algorithm: Forward AD

Input: x

 $s_0 = x$, $A_0 = I$ for i = 0:N-1 do

 $A_{i+1} = \frac{\partial \boldsymbol{\xi}(\mathbf{s}_i)}{\partial \mathbf{s}_i} A_i$ $\mathbf{s}_{i+1} = \boldsymbol{\xi}(\mathbf{s}_i)$

return s_0, \dots, N , A_0, \dots, N

Reminder (here $\zeta(\mathbf{x}) = \boldsymbol{\mu}^{\top} \mathbf{x}$):

Algorithm: Adjoint-mode AD

Input: $s_{0,\ldots,N}$, μ $\lambda = \nabla \zeta(\mathbf{s}_N) = \boldsymbol{\mu}$

for i = N-1:0 do $\lambda \leftarrow \nabla_{\mathbf{s}_i} \boldsymbol{\xi}\left(\mathbf{s}_i\right) \lambda$

return λ