Adjoint-based exact Hessian computation

Shin-ichi Ito[1,2]

Today's talk is based on BIT Numerical Mathematics (2021)

Joint work with

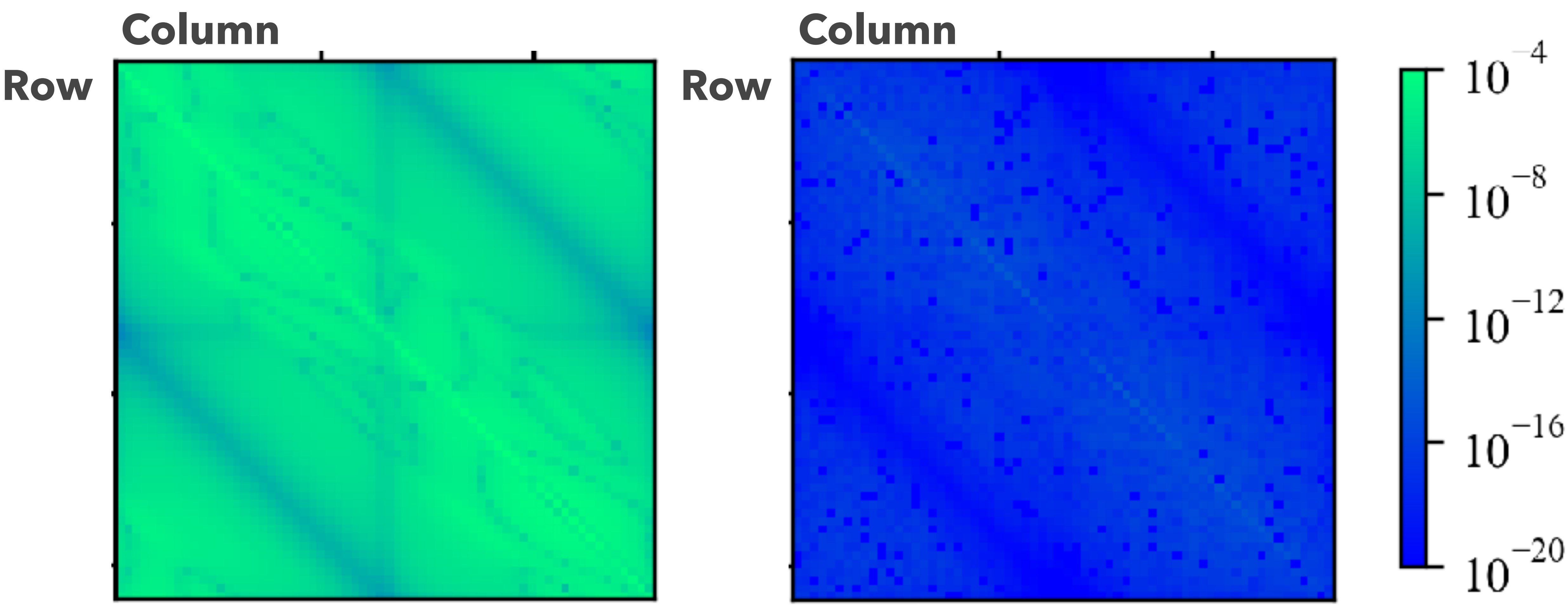
Takeru Matsuda[3] and **Yuto Miyatake**[4]

1. Earthquake Research Institute, The University of Tokyo
2. Graduate School of Information Science and Technology, The University of Tokyo
3. RIKEN Center for Brain Science
4. Cybermedia Center, Osaka University

One-sentence summary

We developed an adjoint-based method that enables exact Hessian computations up to a given floating point limit.

Error in a Hessian matrix



Naive method

Our method

Outline

Introduction

Adjoint-based gradient and Hessian computations

Sanz-Serna’s idea

and exact Hessian computation

Numerical verifications

Introduction

Adjoint-based gradient and Hessian computations

Adjoint method

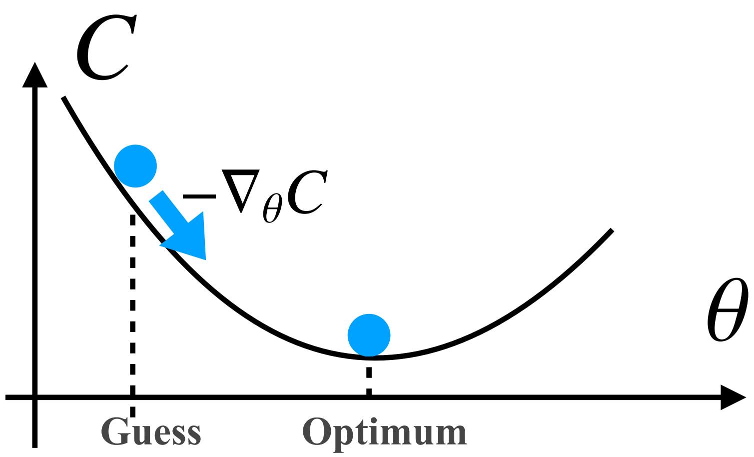
We consider an autonomous ordinary differential equation(ODE):

$$x_0 = \theta \rightarrow \boxed{\frac{d}{dt}x_t = f(x_t)} \rightarrow x_T(\theta)$$

Forward model

and a minimization problem of a cost function defined by the final state $t = T$

$$\min_{\theta} C\left(x_T(\theta)\right)$$



via an gradient method using a gradient $\nabla_\theta C$

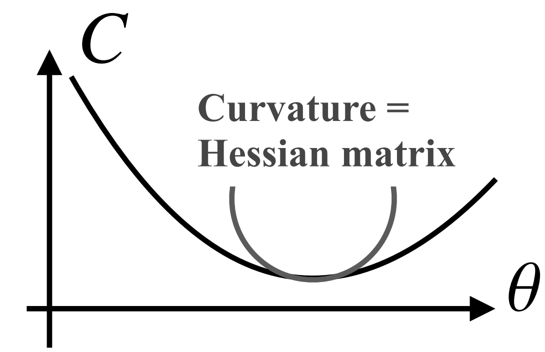
Adjoint method calculates $\nabla_\theta C$ exactly by solving

$$\boxed{\lambda_0 = \nabla_\theta C} \leftarrow \boxed{-\frac{d}{dt}\lambda_t = \left(\nabla_x f\right)^\top \lambda_t} \leftarrow \lambda_T = \nabla_{x_T} C$$

Adjoint model

Second-order adjoint method

Adjoint method is available for a first-order derivative computation, while second-order adjoint (SOA) method is available for a second-order derivative (Hessian matrix H_θ) computation.



$$\delta_0 = \gamma$$

$$\frac{d}{dt}\delta_t = (\nabla_x f)\,\delta_t$$

Tangent linear model

$$\delta_T$$

$$\xi_T = \mathsf{H}_{x_T}\delta_T$$

$$-\frac{d}{dt}\xi_t = (\nabla_x f)^\top \xi_t + \{(\nabla_x \nabla_x f)\,\delta_t\}^\top \lambda_t$$

SOA model

$$\xi_0 = \mathsf{H}_\theta \gamma$$

$$\mathsf{H}_\bullet = \nabla_\bullet \nabla_\bullet{}^\top C$$

*** These model are derived from the first-order perturbation of the forward and adjoint models.**

SOA provides an exact Hessian-vector product for arbitrary vectors.

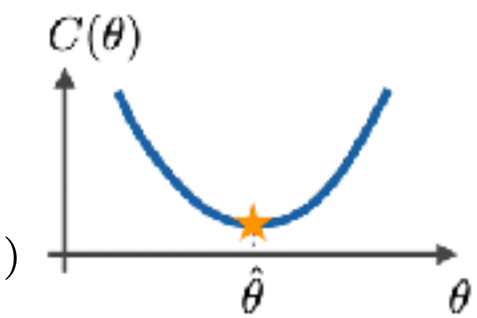
SOA is available for optimization methods, uncertainty quantification, ...

Application

- **Fast optimization**
  - **Newton method** $\theta \leftarrow \theta + y$ **where** $\mathrm{H}_\theta y = -\nabla_\theta C$ y **: Descent vector**
  - **Nonlinear conjugate gradient method**

$$
\begin{cases}
z \leftarrow \nabla_\theta C - \dfrac{z^\top \mathrm{H}_\theta \nabla_\theta C}{z^\top \mathrm{H}_\theta z} z \\
\theta \leftarrow \theta + \eta z
\end{cases}
$$

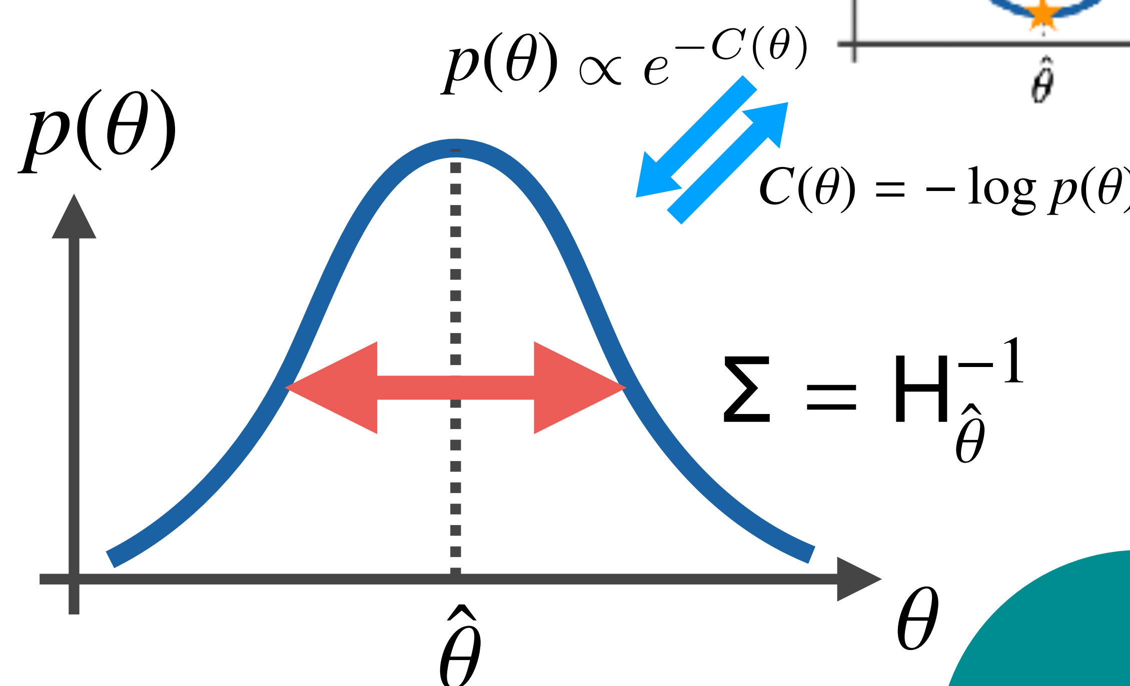
z **: Conjugate vector**
η **: Learning rate**



- **Uncertainty quantification (UQ)**

Solve $\mathrm{H}_{\hat{\theta}} \Sigma = \mathrm{I}$ **via Krylov subspace methods**

Ito et al., (2016)



Question

Adjoint and SOA methods give exact gradient $\nabla_\theta C$ and Hessian-vec. prod. $\mathrm{H}_\theta \gamma$

if all of the models are solved analytically.

In practice, all of the models are solved numerically,

using time integral schemes, e.g., Runge–Kutta methods.

→ inexact due to discretization errors, machine errors

Forward model

$$\frac{d}{dt}x_t = f(x_t)$$

Tangent linear (TL) model

$$\frac{d}{dt}\delta_t = (\nabla_x f)\,\delta_t$$

?

Adjoint model

$$-\frac{d}{dt}\lambda_t = (\nabla_x f)^\top \lambda_t$$

?

SOA model

$$-\frac{d}{dt}\xi_t = (\nabla_x f)^\top \xi_t + \{(\nabla_x \nabla_x f)\,\delta_t\}^\top \lambda_t$$

?

For a given forward scheme, are there "optimal schemes" for other models?

Question

Adjoint and SOA methods give **exact** gradient $\nabla_\theta C$ and Hessian-vec. prod. $\mathsf{H}_\theta \gamma$ if all of the models are solved **analytically**.

In practice, all of the models are solved **numerically**, using time integral schemes, e.g., Runge–Kutta methods.

→ **inexact** due to discretization errors, machine errors

Forward model

$$\frac{d}{dt}x_t = f(x_t)$$

Adjoint model

$$-\frac{d}{dt}\lambda_t = (\nabla_x f)^\top \lambda_t$$

Sanz-Serna (2016)

Tangent linear (TL) model

$$\frac{d}{dt}\delta_t = (\nabla_x f)\,\delta_t$$

?

SOA model

$$-\frac{d}{dt}\xi_t = (\nabla_x f)^\top \xi_t + \{(\nabla_x \nabla_x f)\,\delta_t\}^\top \lambda_t$$

?

For a given forward scheme, are there "optimal schemes" for other models?

→ Yes. There exists **an optimal scheme for the adjoint model** that can avoid any discretization errors.

Question

Adjoint and SOA methods give exact gradient $\nabla_\theta C$ and Hessian-vec. prod. $\mathrm{H}_\theta \gamma$ if all of the models are solved analytically.

In practice, all of the models are solved numerically, using time integral schemes, e.g., Runge–Kutta methods.

→ inexact due to discretization errors, machine errors

Forward model

$$\frac{d}{dt} x_t = f(x_t)$$

Adjoint model

$$-\frac{d}{dt} \lambda_t = (\nabla_x f)^\top \lambda_t$$

Sanz-Serna (2016)

Tangent linear (TL) model

$$\frac{d}{dt} \delta_t = (\nabla_x f)\, \delta_t$$

SOA model

$$-\frac{d}{dt} \xi_t = (\nabla_x f)^\top \xi_t + \{(\nabla_x \nabla_x f)\, \delta_t\}^\top \lambda_t$$

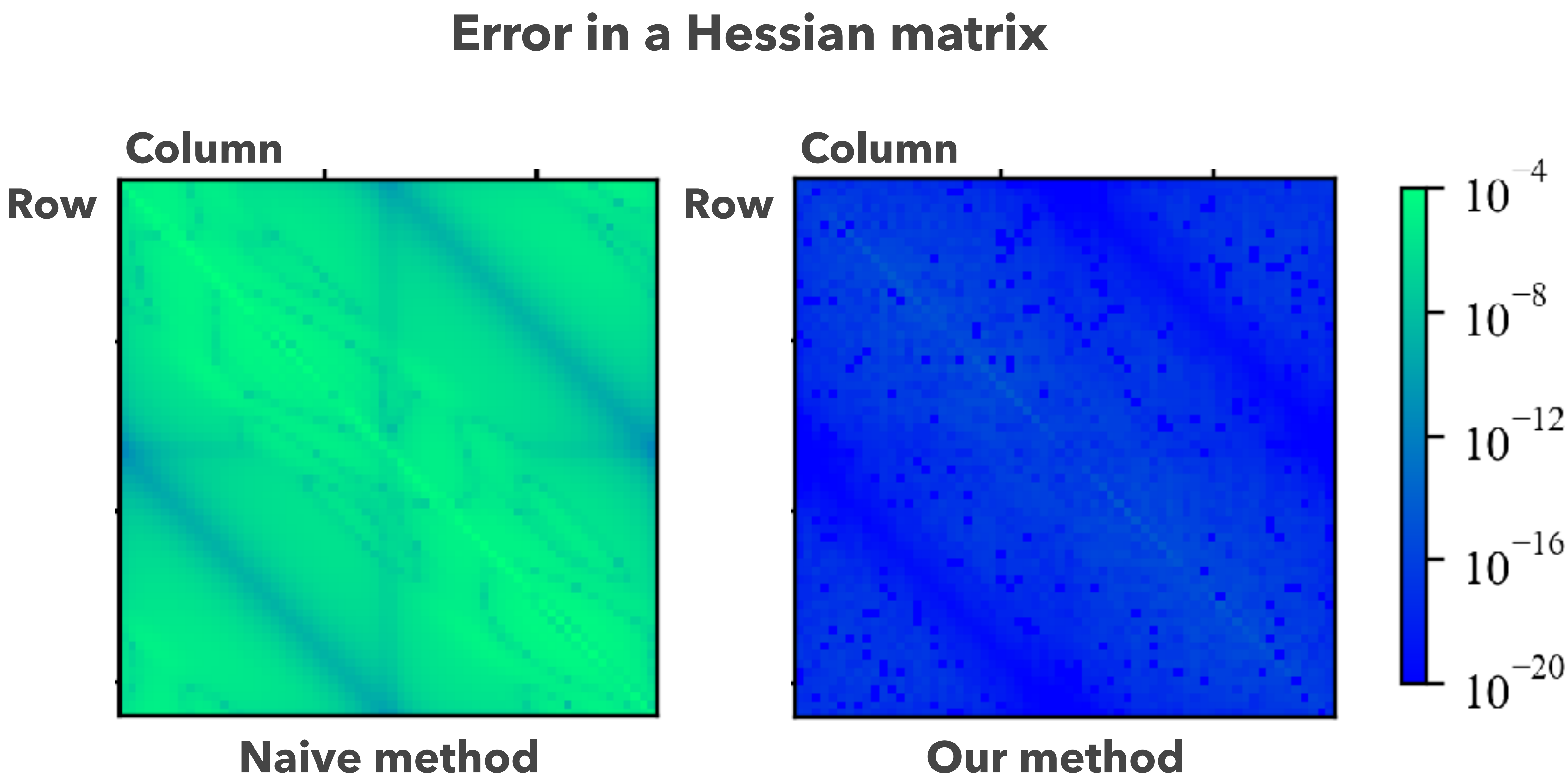
Ours

For a given forward scheme, are there "optimal schemes" for other models?

→ Yes. There exists a set of optimal schemes for all of these models that can avoid any discretization errors.

Importance of exact Hessian computation

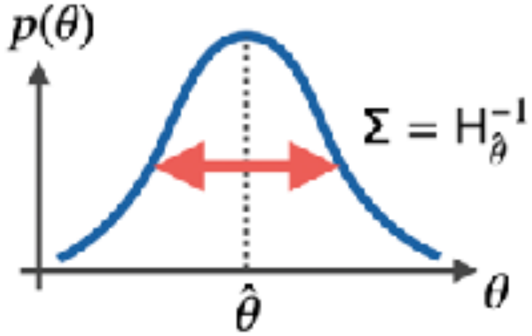
＊ Numerical details will be explained later.

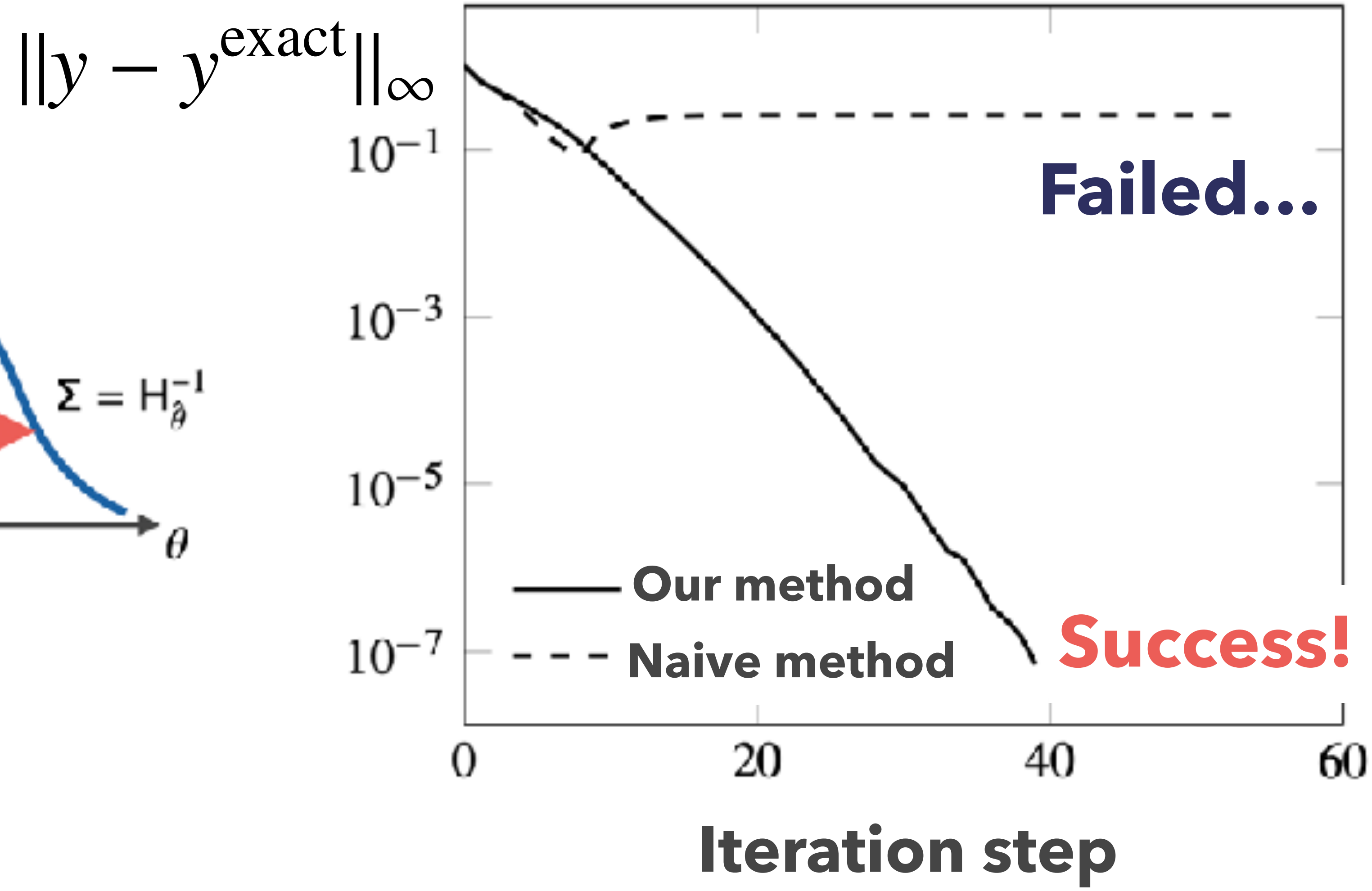


Our method allows exact Hessian computation up to a given floating point limit.

Importance of exact Hessian computation

Convergence behavior when solving $\mathrm{H}y = b$





When using an incorrect scheme,

- Optimization may not converge to correct solution.
- Uncertainty quantification may fail.

Sanz-Serna's idea

and exact Hessian computation

Invariant

Why can the analytical solution of the adjoint model provide the exact gradient ?

A product of TL and adjoint variables is time-invariant

$$\frac{d}{dt}\left(\delta_t^\top \lambda_t\right) = \left(\frac{d}{dt}\delta_t\right)^\top \lambda_t + \delta_t^\top \frac{d}{dt}\lambda_t = 0$$

$$\boxed{\delta_t^\top \lambda_t = \delta_0^\top \lambda_0}$$

TL model

$$\frac{d}{dt}\delta_t = (\nabla_x f)\,\delta_t$$

Adjoint model

$$-\frac{d}{dt}\lambda_t = (\nabla_x f)^\top \lambda_t$$

If we choose $\lambda_t = \nabla_{x_t} C(x_t(\theta))$ **, we obtain**

$$\delta_t^\top \nabla_{x_t} C(x_t) = ((\nabla_\theta x_t)\,\delta_0)^\top \nabla_{x_t} C(x_t)$$

$$= \delta_0^\top \left[(\nabla_\theta x_t)^\top \nabla_{x_t} C(x_t)\right] = \delta_0^\top \nabla_\theta C(x_t)$$

$$\because\ x_t(\theta + \epsilon\delta_0) = x_t(\theta) + \epsilon\delta_t + O(\epsilon^2)$$

$$\Rightarrow\ \delta_t = (\nabla_\theta x_t)\,\delta_0$$

$$\Rightarrow\ \lambda_0 = \nabla_\theta C(x_t(\theta))$$

Runge–Kutta family

Forward model

$$\frac{d}{dt}x_t = f(x_t)$$

Discretize

$$x_n := x_{t_n}$$
$$x_{n+1} := x_{t_n+h}$$

s-stage Runge–Kutta (RK) method

$$x_{n+1} = x_n + h\sum_{i=1}^{s} b_i k_{n,i},$$
$$k_{n,i} = f(X_{n,i}) \quad (i = 1,\dots,s),$$
$$X_{n,i} = x_n + h\sum_{j=1}^{s} a_{ij} k_{n,j} \quad (i = 1,\dots,s).$$

h **: stepsize**
a_{ij}, b_i **: weights**

* (…) : order

- **Classical 4-stage RK (4)**

$$x_{n+1} = x_n + \frac{h}{6}(k_{n,1} + 2k_{n,2} + 2k_{n,3} + k_{n,4}),$$
$$k_{n,i} = f(X_{n,i}) \quad (i = 1,\dots,4),$$
$$X_{n,1} = x_n \qquad X_{n,3} = x_n + \frac{h}{2}k_{n,2}$$
$$X_{n,2} = x_n + \frac{h}{2}k_{n,1} \qquad X_{n,4} = x_n + hk_{n,3}$$

- **Explicit Euler (1)**

$$x_{n+1} = x_n + hk_{n,1},$$
$$k_{n,1} = f(X_{n,1}),$$
$$X_{n,1} = x_n$$

- **Heun (2)**

$$x_{n+1} = x_n + \frac{h}{2}(k_{n,1} + k_{n,2}),$$
$$k_{n,i} = f(X_{n,i}) \quad (i = 1,\dots,2),$$
$$X_{n,1} = x_n$$
$$X_{n,2} = x_n + hk_{n,1}$$

- **Adaptive stepsize RK: Bogacki–Shampine(2-3), Runge-Kutta-Fehlberg(4-5), Dormand-Prince(4-5), …**

Sanz-Serna's idea

Solve forward models via a s-stage Runge–Kutta (RK) method

$$\frac{d}{dt}x_t = f(x_t) \quad \Rightarrow \quad \begin{aligned} x_{n+1} &= x_n + h\sum_{i=1}^{s} b_i k_{n,i}, \\ k_{n,i} &= f(X_{n,i}) \quad (i = 1, \ldots, s), \\ X_{n,i} &= x_n + h\sum_{j=1}^{s} a_{ij} k_{n,j} \quad (i = 1, \ldots, s). \end{aligned}$$

$$\frac{d}{dt}\delta_t = (\nabla_x f)\,\delta_t \quad \Rightarrow \quad \begin{aligned} \delta_{n+1} &= \delta_n + h\sum_{i=1}^{s} b_i d_{n,i}, \\ d_{n,i} &= \nabla_x f(X_{n,i}) D_{n,i} \quad (i = 1, \ldots, s), \\ D_{n,i} &= \delta_n + h\sum_{j=1}^{s} a_{ij} d_{n,j} \quad (i = 1, \ldots, s). \end{aligned}$$

and solve adjoint model via another s-stage RK method

$$-\frac{d}{dt}\lambda_t = (\nabla_x f)^\top \lambda_t \quad \Rightarrow \quad \begin{aligned} \lambda_{n+1} &= \lambda_n + h\sum_{i=1}^{s} B_i l_{n,i}, \\ l_{n,i} &= -\nabla_x f(X_{n,i})^\top \Lambda_{n,i} \quad (i = 1, \ldots, s), \\ \Lambda_{n,i} &= \lambda_n + h\sum_{j=1}^{s} A_{ij} l_{n,j} \quad (i = 1, \ldots, s). \end{aligned}$$

Sanz-Serna (2016) provided a relation between (a, b) and (A, B)

so that $\boxed{\delta_{n+1}^\top \lambda_{n+1} = \delta_n^\top \lambda_n}$ **exactly.**

Sanz-Serna's idea

$$\boxed{\delta_{n+1}^\top \lambda_{n+1} = \delta_n^\top \lambda_n} \quad \Longleftrightarrow \quad \boxed{\begin{aligned} & b_i = B_i \quad (i = 1, \ldots, s)\,, \\ & b_i A_{ij} + B_j a_{ji} = B_i b_j \quad (i, j = 1, \ldots, s)\,. \end{aligned}}$$

$$\delta_{n+1} = \delta_n + h \sum_{i=1}^{s} b_i d_{n,i},$$

$$d_{n,i} = \nabla_x f(X_{n,i}) D_{n,i} \quad (i = 1, \ldots, s)\,,$$

$$D_{n,i} = \delta_n + h \sum_{j=1}^{s} a_{ij} d_{n,j} \quad (i = 1, \ldots, s)\,.$$

$$\lambda_{n+1} = \lambda_n + h \sum_{i=1}^{s} B_i l_{n,i},$$

$$l_{n,i} = -\nabla_x f(X_{n,i})^\top \Lambda_{n,i} \quad (i = 1, \ldots, s)\,,$$

$$\Lambda_{n,i} = \lambda_n + h \sum_{j=1}^{s} A_{ij} l_{n,j} \quad (i = 1, \ldots, s)\,.$$

Proof

$$S(\delta_t, \lambda_t) = \delta_t^\top \lambda_t$$

$$\begin{aligned}
& S(\delta_{n+1}, \lambda_{n+1}) - S(\delta_n, \lambda_n) \\
&\quad = h \sum_{i=1}^{s} B_i S(\delta_n, l_{n,i}) + h \sum_{i=1}^{s} b_i S(d_{n,i}, \lambda_n) + h^2 \sum_{i,j=1}^{s} B_i b_j S(d_{n,i}, l_{n,j}) \\
&\quad = h \sum_{i=1}^{s} B_i S(D_{n,i}, l_{n,i}) - h^2 \sum_{i,j=1}^{s} B_i a_{ij} S(d_{n,j}, l_{n,i}) \\
&\qquad + h \sum_{i=1}^{s} b_i S(d_{n,i}, \Lambda_{n,i}) - h^2 \sum_{i,j=1}^{s} b_i A_{ij} S(d_{n,i}, l_{n,j}) + h^2 \sum_{i,j=1}^{s} B_i b_j S(d_{n,i}, l_{n,j}) \\
&\quad = h \sum_{i=1}^{s} \underbrace{(b_i - B_i)}_{=0} S\left(D_{n,i}, \nabla_x f(X_{n,i})^\top \Lambda_{n,i}\right) + h^2 \sum_{i,j=1}^{s} \underbrace{\left(B_i b_j - b_i A_{ij} - B_j a_{ji}\right)}_{=0} S(d_{n,i}, l_{n,j})
\end{aligned}$$

Exact gradient computation

Forward models solved by a s-stage Runge–Kutta (RK) method

$$\frac{d}{dt}x_t = f(x_t) \quad \Rightarrow \quad \begin{aligned} x_{n+1} &= x_n + h\sum_{i=1}^{s} b_i k_{n,i}, \\ k_{n,i} &= f(X_{n,i}) \quad (i = 1, \ldots, s), \\ X_{n,i} &= x_n + h\sum_{j=1}^{s} a_{ij} k_{n,j} \quad (i = 1, \ldots, s). \end{aligned}$$

$$\frac{d}{dt}\delta_t = (\nabla_x f)\,\delta_t \quad \Rightarrow \quad \begin{aligned} \delta_{n+1} &= \delta_n + h\sum_{i=1}^{s} b_i d_{n,i}, \\ d_{n,i} &= \nabla_x f(X_{n,i}) D_{n,i} \quad (i = 1, \ldots, s), \\ D_{n,i} &= \delta_n + h\sum_{j=1}^{s} a_{ij} d_{n,j} \quad (i = 1, \ldots, s). \end{aligned}$$

and adjoint model solved by another s-stage RK method

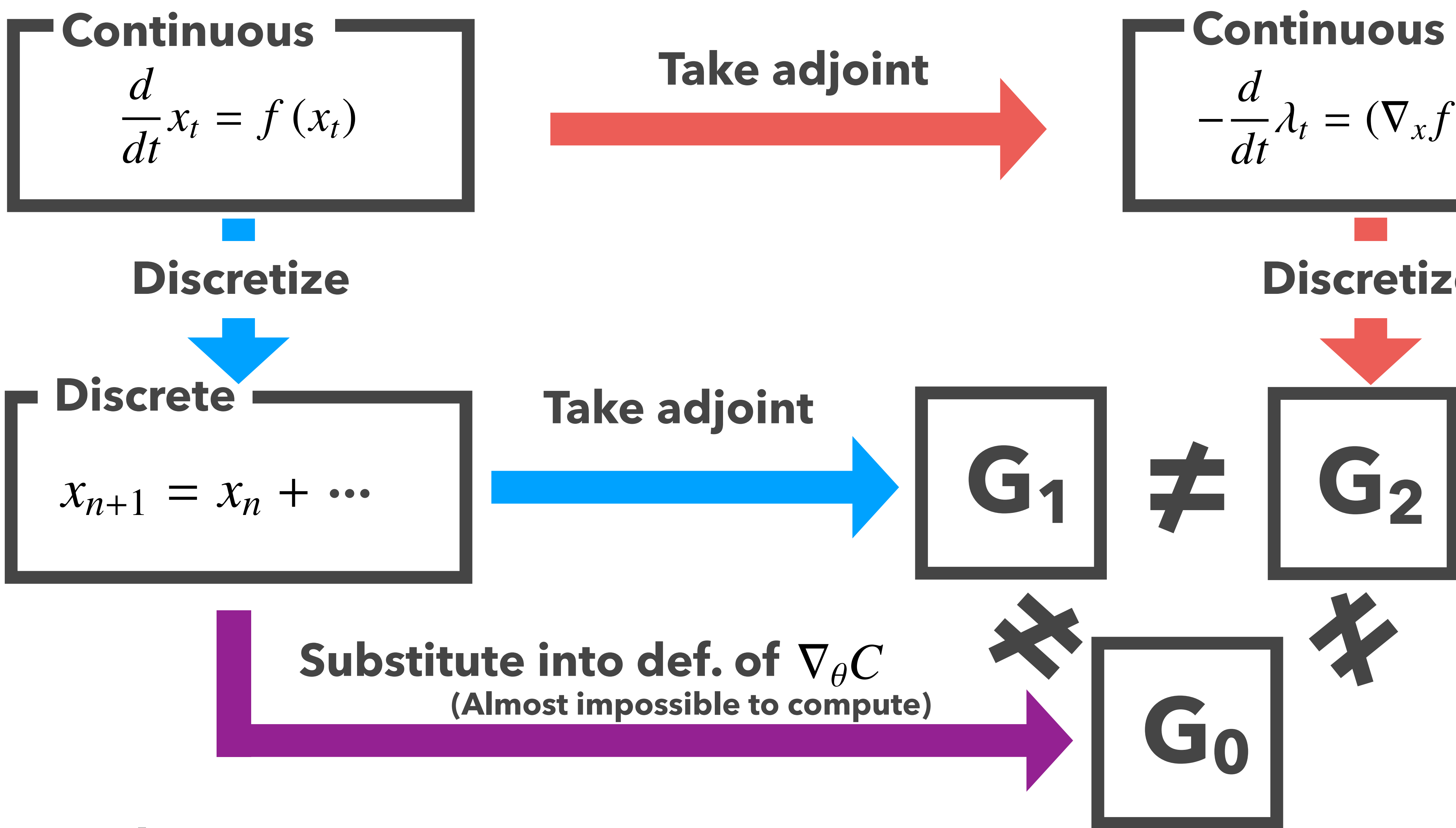
$$-\frac{d}{dt}\lambda_t = (\nabla_x f)^\top \lambda_t \quad \Rightarrow \quad \begin{aligned} \lambda_{n+1} &= \lambda_n + h\sum_{i=1}^{s} B_i l_{n,i}, \\ l_{n,i} &= -\nabla_x f(X_{n,i})^\top \Lambda_{n,i} \quad (i = 1, \ldots, s), \\ \Lambda_{n,i} &= \lambda_n + h\sum_{j=1}^{s} A_{ij} l_{n,j} \quad (i = 1, \ldots, s). \end{aligned}$$

where

$$\boxed{\begin{aligned} b_i &= B_i \quad (i = 1, \ldots, s), \\ b_i A_{ij} + B_j a_{ji} &= B_i b_j \quad (i, j = 1, \ldots, s). \end{aligned}}$$

provide exact gradient as $\lambda_0 = \nabla_\theta C$

Adjoint consistency in time



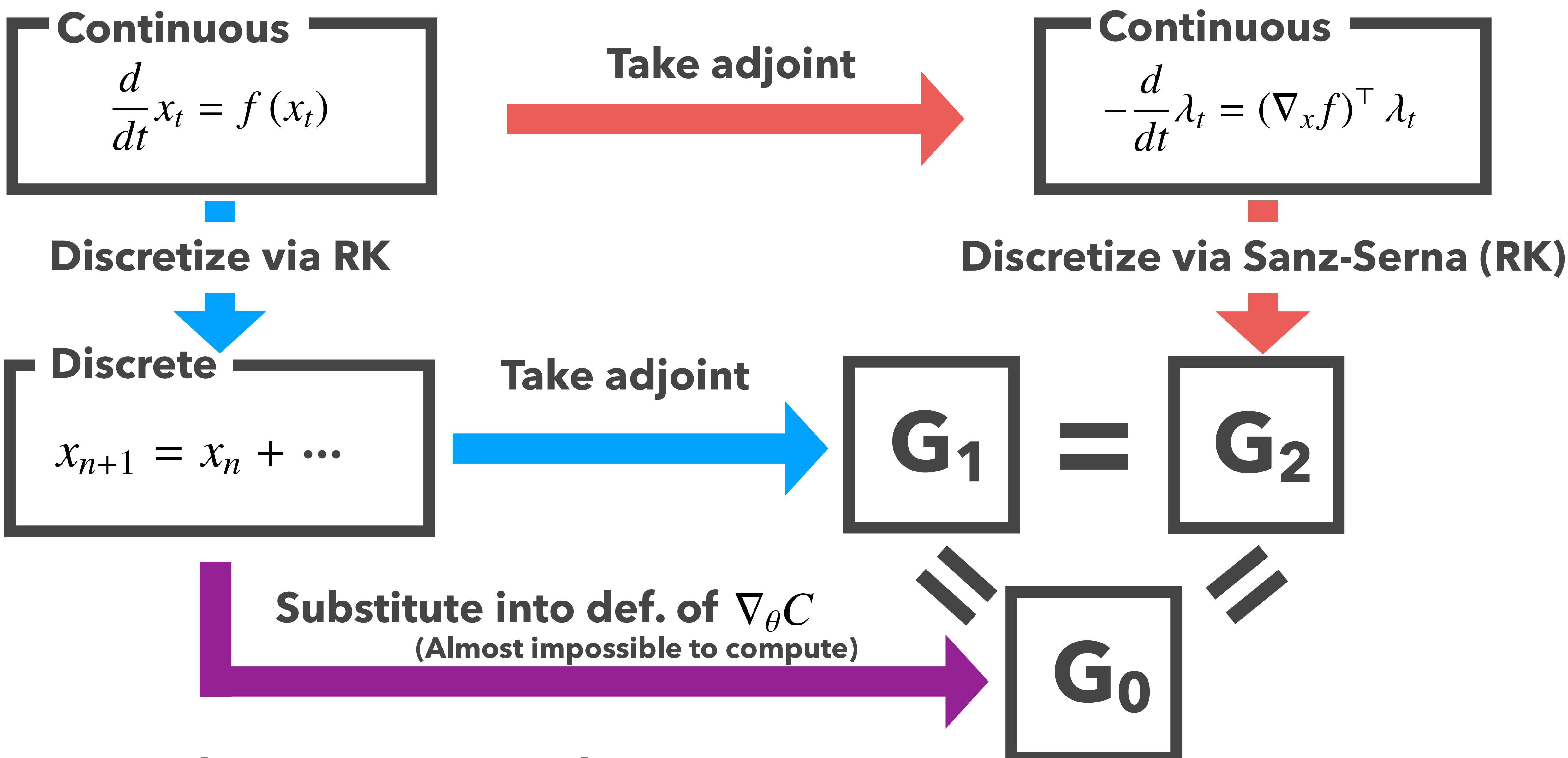
In general,

G_1 and G_2 are different.

Neither G_1 nor G_2 gives an exact gradient (G_0)

because $\delta_{n+1}^\top \lambda_{n+1} \neq \delta_n^\top \lambda_n$ in both cases.

Adjoint consistency in time



Sanz-Serna's scheme guarantees that

G_1 and G_2 are identical

They give an exact gradient (G_0)

since $\delta_{n+1}^\top \lambda_{n+1} = \delta_n^\top \lambda_n$

Open problems(?)

Other types of schemes

Consistency in space, i.e., PDEs

(e.g., advection eq.)

Towards exact Hessian computation

Exact gradient can be computed by using Sanz-Serna's scheme.

Exact Hessian ???

Forward model

$$\frac{d}{dt}x_t = f(x_t)$$

Adjoint model

$$-\frac{d}{dt}\lambda_t = (\nabla_x f)^\top \lambda_t$$

Sanz-Serna (2016)

Tangent linear (TL) model

$$\frac{d}{dt}\delta_t = (\nabla_x f)\,\delta_t$$

SOA model

$$-\frac{d}{dt}\xi_t = (\nabla_x f)^\top \xi_t + \{(\nabla_x \nabla_x f)\,\delta_t\}^\top \lambda_t$$

?

Towards exact Hessian computation

Our idea: Equivalence between SOA model and adjoint model

$$\frac{d}{dt}x_t = f(x_t) \qquad \frac{d}{dt}\delta_t = (\nabla_x f)\,\delta_t$$

$$-\frac{d}{dt}\lambda_t = (\nabla_x f)^\top \lambda_t \qquad -\frac{d}{dt}\xi_t = (\nabla_x f)^\top \xi_t + \{(\nabla_x \nabla_x f)\,\delta_t\}^\top \lambda_t$$

Define augmented vectors: $q_t = \begin{bmatrix} x_t \\ \delta_t \end{bmatrix}$ **and** $p_t = \begin{bmatrix} \lambda_t \\ \xi_t \end{bmatrix}$

$$\boxed{\begin{aligned} \frac{d}{dt}q_t &= \begin{bmatrix} f(x_t) \\ (\nabla_x f)\,\delta_t \end{bmatrix} = G(q_t) \\ -\frac{d}{dt}p_t &= \begin{bmatrix} (\nabla_x f)^\top \lambda_t \\ (\nabla_x f)^\top \xi_t + \{(\nabla_x \nabla_x f)\,\delta_t\}^\top \lambda_t \end{bmatrix} = \left(\nabla_q G\right)^\top p_t \end{aligned}}$$

Set of adjoint and SOA models constitutes a large adjoint system

→ **Many techniques for adjoint method can be used for SOA method.**
e.g., Sanz-Serna’s idea, adjoint code generator

Exact gradient and Hessian computations

The augmented models solved by Sanz-Serna (2016), i.e.,

the augmented forward model solved by a s-stage RK method

$$\frac{d}{dt}q_t = G(q_t) \quad \Rightarrow \quad \begin{aligned} q_{n+1} &= q_n + h\sum_{i=1}^{s} b_i k_{n,i}, \\ k_{n,i} &= G(Q_{n,i}) \quad (i = 1, \ldots, s), \\ Q_{n,i} &= q_n + h\sum_{j=1}^{s} a_{ij} k_{n,j} \quad (i = 1, \ldots, s). \end{aligned}$$

and the augmented adjoint model solved by another s-stage RK method

$$-\frac{d}{dt}p_t = \left(\nabla_q G\right)^\top p_t \quad \Rightarrow \quad \begin{aligned} p_{n+1} &= p_n + h\sum_{i=1}^{s} B_i l_{n,i}, \\ l_{n,i} &= -\nabla_q G(Q_{n,i})^\top P_{n,i} \quad (i = 1, \ldots, s), \\ P_{n,i} &= p_n + h\sum_{j=1}^{s} A_{ij} l_{n,j} \quad (i = 1, \ldots, s). \end{aligned}$$

where

$$\boxed{\begin{aligned} &b_i = B_i \quad (i = 1, \ldots, s), \\ &b_i A_{ij} + B_j a_{ji} = B_i b_j \quad (i, j = 1, \ldots, s). \end{aligned}}$$

provide exact gradient and Hessian computations.

Note that our method itself is not limited to RK.

Numerical verifications

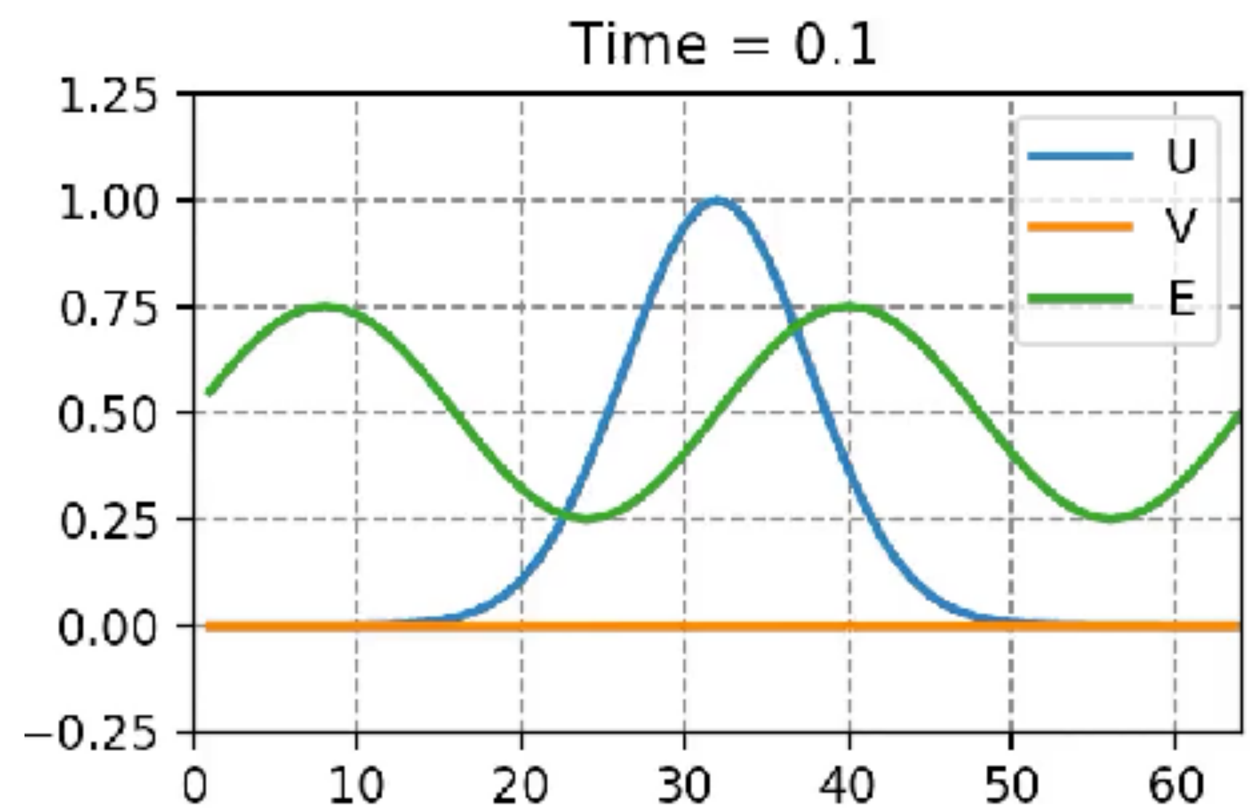
- Inhomogeneous wave equation

Inhomogeneous wave equation

Validate our method through numerical test of data assimilation based on a one-dimensional inhomogeneous wave equation

$$\frac{\partial^2 U(z,t)}{\partial t^2} = \frac{\partial}{\partial z}\left[E(z)\frac{\partial}{\partial z}U(z,t)\right]$$

(discretized in space by finite volume method)



Time integral scheme:

Heun method (2-stage RK)

Assume initial states of U and $\dot{U}$ are known but E is not.

Cost function:

$$C(E) = \sum_{t_n \in T^{\mathrm{obs}}} \|U_n(E) - U_n^{\mathrm{obs}}\|_2^2,$$

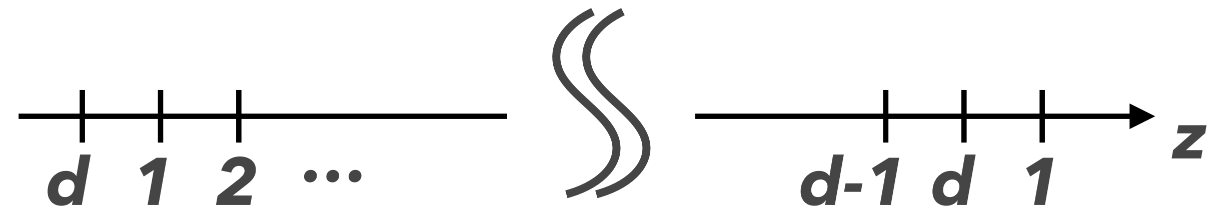
where

$T^{\mathrm{obs}} = \{0, 0.2, 0.4, \ldots, 1.8, 2.0\}$

U^{obs} : U **in the above movie**

Discretization in space

$$\frac{\partial^2 U(z,t)}{\partial t^2} = \frac{\partial}{\partial z}\left[E(z)\frac{\partial}{\partial z}U(z,t)\right]$$



Discretization via finite volume method (staggered lattice)

$$\frac{\mathrm{d}}{\mathrm{d}t}U_i = V_i \quad (i = 1, \ldots, d)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}V_i = \frac{1}{\Delta z^2}\begin{cases} E_{\frac{3}{2}}\left(U_2 - U_1\right) - E_{d+\frac{1}{2}}\left(U_1 - U_d\right) & (i = 1) \\ E_{i+\frac{1}{2}}\left(U_{i+1} - U_i\right) - E_{i-\frac{1}{2}}\left(U_i - U_{i-1}\right) & (i = 2, \ldots, d-1) \\ E_{d+\frac{1}{2}}\left(U_1 - U_d\right) - E_{d-\frac{1}{2}}\left(U_d - U_{d-1}\right) & (i = d) \end{cases}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}E_{i+\frac{1}{2}} = 0 \quad (i = 1, \ldots, d),$$

Time integral schemes

Augmented forward model

$$\frac{d}{dt}q_t = G(q_t)$$

Heun method

$$q_{n+1} = q_n + \frac{h}{2}\left(k_{n,1} + k_{n,2}\right),$$
$$k_{n,1} = G(Q_{n,1}),$$
$$k_{n,2} = G(Q_{n,2}),$$
$$Q_{n,1} = q_n,$$
$$Q_{n,2} = q_n + hk_{n,1}.$$

Augmented adjoint model

$$-\frac{d}{dt}p_t = \left(\nabla_q G\right)^\top p_t$$

Naively

Our method

$$p_n = p_{n+1} + \frac{h}{2}\left(\bar{l}_{n,1} + \bar{l}_{n,2}\right),$$
$$\bar{l}_{n,2} = \nabla_q G(Q_{n,2})^\top P_{n,2},$$
$$\bar{l}_{n,1} = \nabla_q G(q_n)^\top P_{n,1},$$
$$P_{n,2} = p_{n+1},$$
$$P_{n,1} = p_{n+1} + h\bar{l}_{n,2}.$$

Heun method

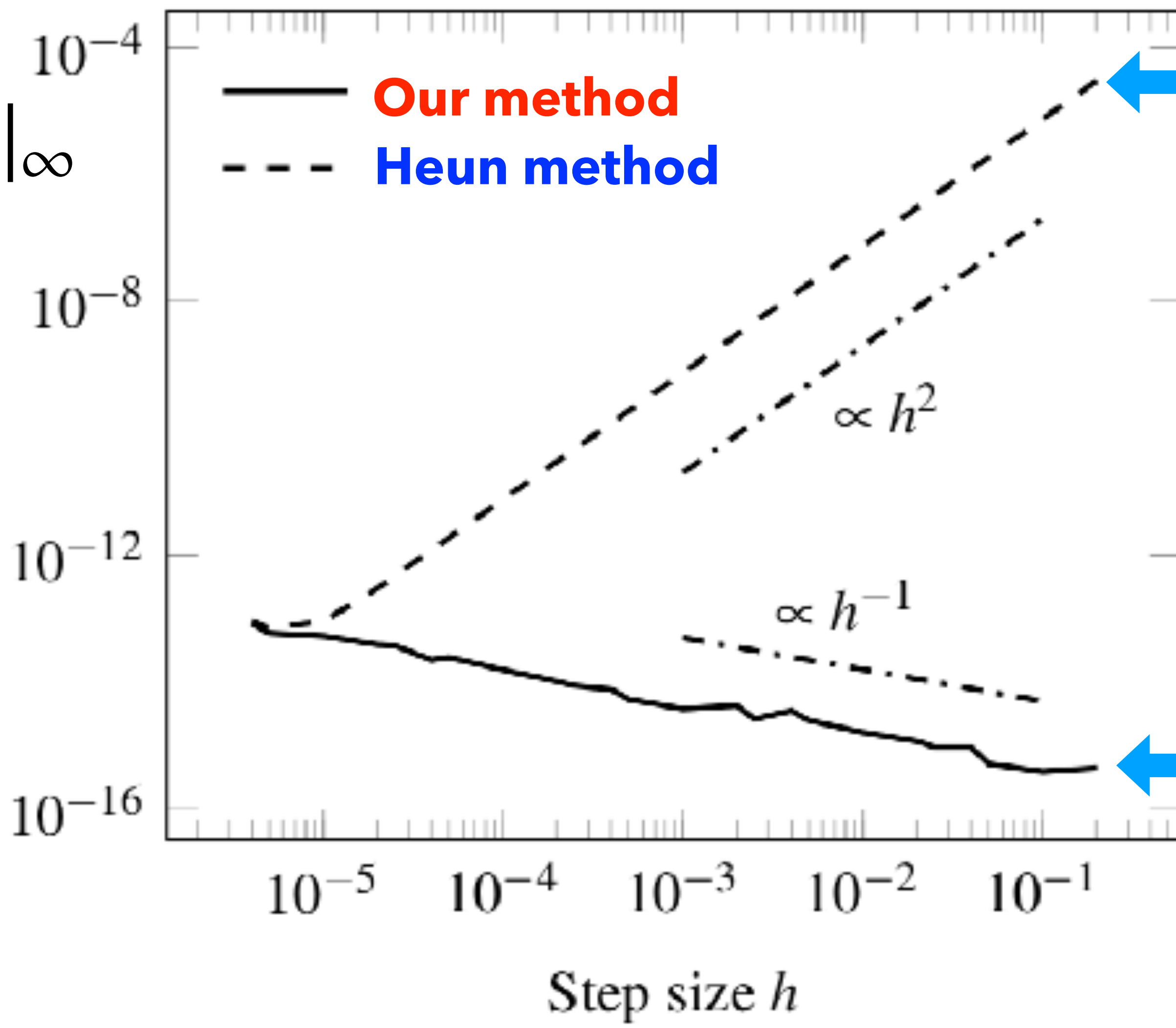
$$p_n = p_{n+1} + \frac{h}{2}\left(\bar{l}_{n,1} + \bar{l}_{n,2}\right),$$
$$\bar{l}_{n,2} = \nabla_q G(q_{n+1})^\top P_{n,2},$$
$$\bar{l}_{n,1} = \nabla_q G(q_n)^\top P_{n,1},$$
$$P_{n,2} = p_{n+1},$$
$$P_{n,1} = p_{n+1} + h\bar{l}_{n,2}.$$

Symmetry of Hessian matrix

$\|\mathrm{H}_\theta - \mathrm{H}_\theta{}^\top\|_\infty$: **Degree of asymmetry, which is zero if H_θ is symmetric.**

$\|\mathrm{H}_{\hat\theta} - \mathrm{H}_{\hat\theta}{}^\top\|_\infty$

$\hat\theta$: **Optimum sol.**



Discretization errors resulting from Heun method

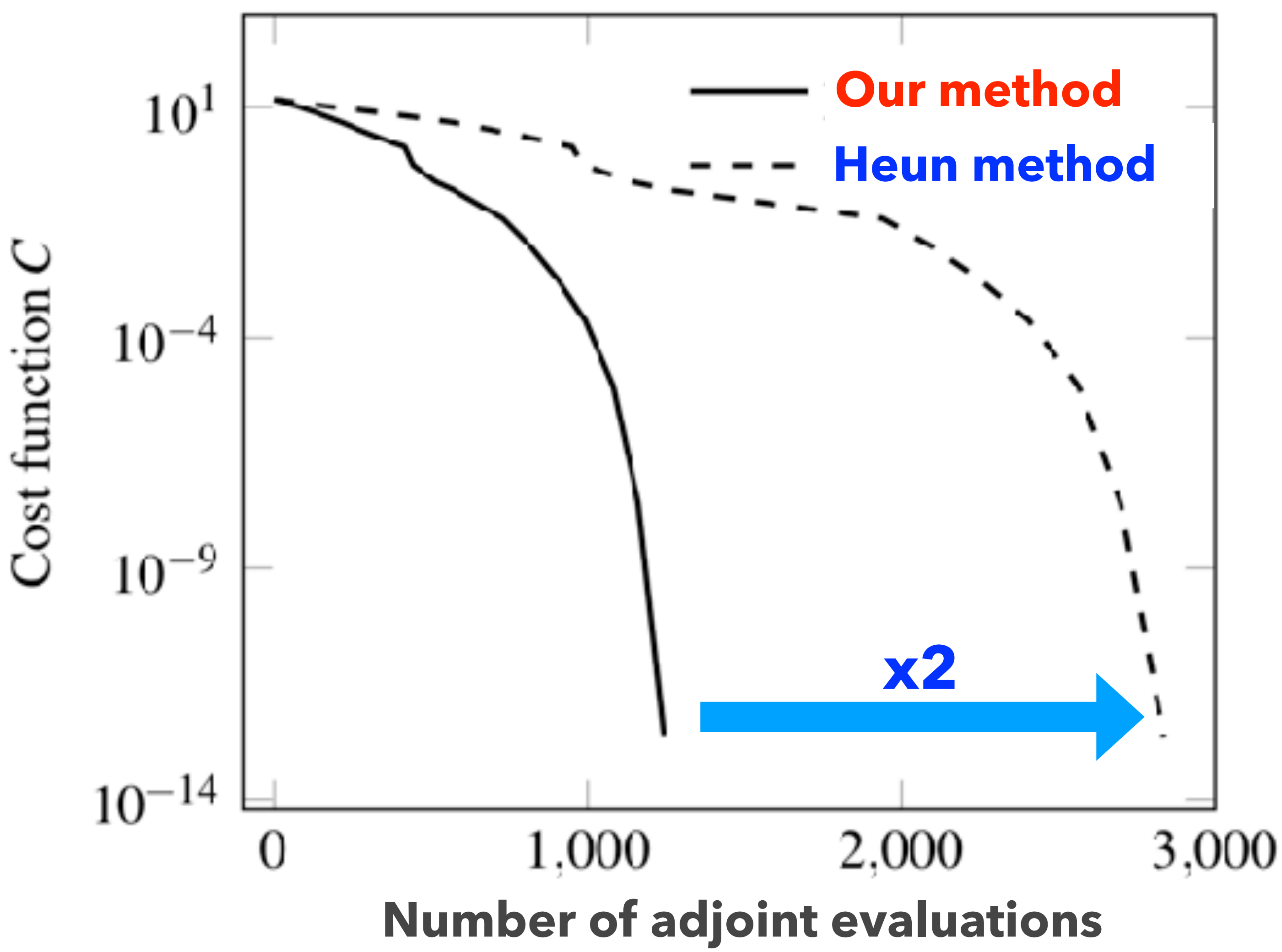
Only accumulation of machine errors (e.g., round-off error)

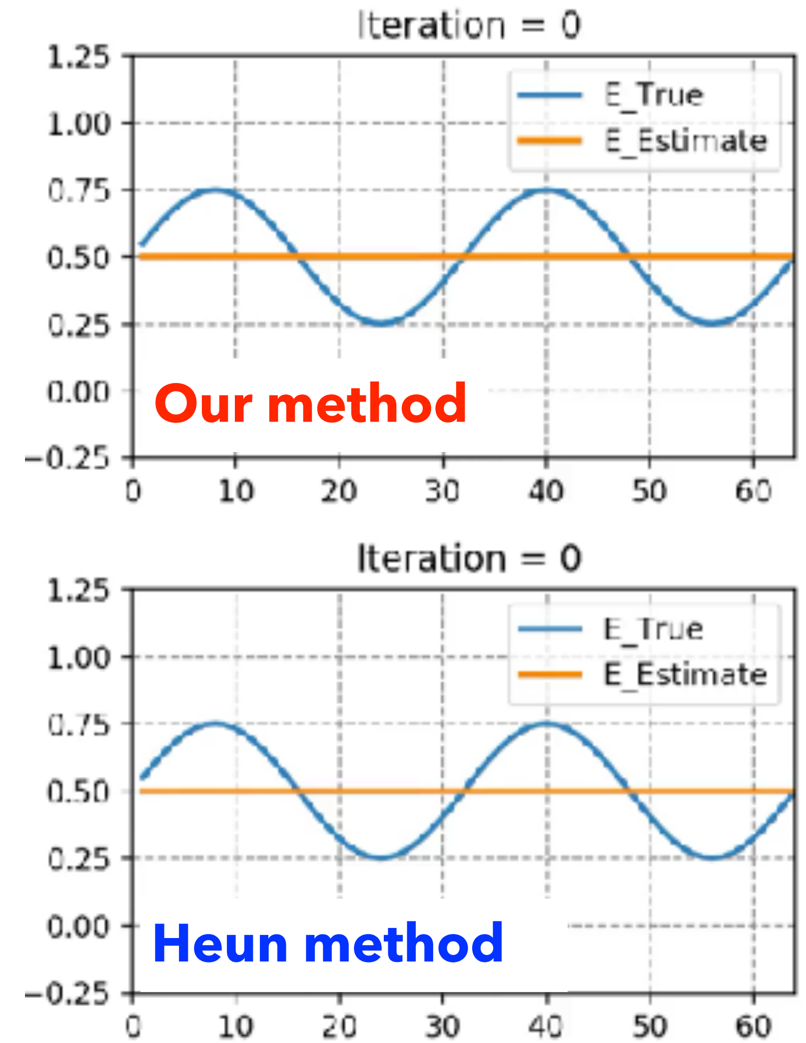
Our method reproduces the symmetry of Hessian without any discretization errors even when using a large step size.

Convergence to optimum solution

Optimizer : Regularized Newton method

$$\theta \leftarrow \theta + y \quad \text{where} \quad (\mathsf{H}_\theta + \sigma \mathsf{I})\, y = -\nabla_\theta C$$

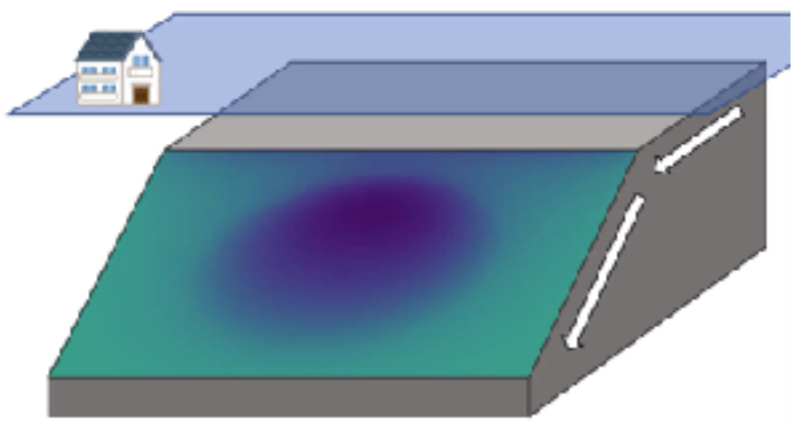




Our method contributes to rapid convergence.

Large-scale uncertainty quantification

Slow-slip motion model



* Damped EOM [Hirahara *et al.*(2019)]

$$\frac{d\tau}{dt} = \mathscr{G} \bullet (V_{plate} - V) + \mathrm{K}(V_{plate} - V_{lock}) - \frac{G}{2c}\frac{dV}{dt}$$

* Rate-and-state dependent friction law

$$\tau = \tau_0 + A \log V + B \log \theta$$

* Aging law

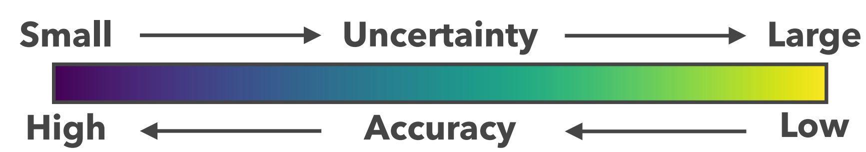
$$\frac{d\theta}{dt} = 1 - \frac{V\theta}{L}$$

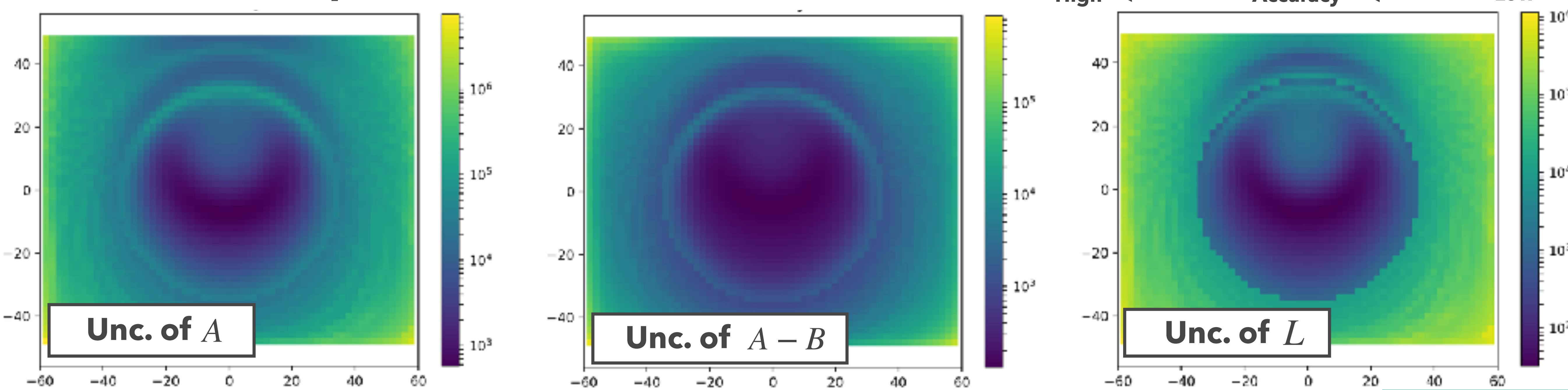
A, B, and L are spatially dependent

Ito et al., in prep.

Time integrator: Runge-Kutta-Fehlberg method

UQ for frictional parameter fields





Conclusions

* We proposed a method to compute an exact Hessian-vector product.

* The fact that the second-order adjoint system can be reformulated to a part of a large adjoint system is the key point to obtain the exact Hessian-vector product based on the Sanz-Serna scheme.

* Our method is capable of being applied to various types of ODEs.
    * Seismic imaging, aerodynamic design, structural materials, ...
    * Neural ODE

* Thank you for your attention

End