Supporting Information (S.I.) for Online Calibration of Deep Learning Sub-Models for Hybrid Numerical Modeling Systems

Table of contents

A Supplementary Note 1: Proofs

A.1 Proof of the proposition 1

Since the solver Ψ is of order p, and the explicit Euler scheme has order 1 it is trivial to prove proposition (1) from the Taylor expansion of Ψ. We include the proof here for completeness. We can write as as h approaches zero:

$$\begin{aligned}
\mathbf{u}_{t+h} &= \Psi(\mathbf{u}_t) \\
&= \mathbf{u}_t + \sum_{k=1}^{p} h^k \frac{1}{k!}(\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{u}_t))^{k-1} + O(h^{p+1}) \\
&= \underbrace{\mathbf{u}_t + h(\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{u}_t))}_{\Psi_E(\mathbf{u}_t)} + \sum_{k=2}^{p} h^k \frac{1}{k!}(\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{u}_t))^{k-1} + O(h^{p+1}) \\
&= \Psi_E(\mathbf{u}_t) + \underbrace{\sum_{k=2}^{p} h^k \frac{1}{k!}(\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{u}_t))^{k-1}}_{<Ch^2} + O(h^{p+1}) \\
&= \Psi_E(\mathbf{u}_t) + O(h^2)
\end{aligned} \tag{S.I. 1}$$

The second part of the proposition can be proven similarly by replacing $\mathbf{u}_t$ by $\Psi^{n-1}(\mathbf{u}_t)$.

A.2 Proof of theorem 1

Notice that for every n, the following holds:

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi^n(\mathbf{u}_t) &= \frac{\partial}{\partial\boldsymbol{\theta}}\Psi\circ\Psi^{n-1}(\mathbf{u}_t)\\ &= \underbrace{\frac{\partial\Psi(\Psi^{n-1}(\mathbf{u}_t))}{\partial\Psi^{n-1}(\mathbf{u}_t)}\frac{\partial\Psi^{n-1}(\mathbf{u}_t)}{\partial\boldsymbol{\theta}}}_{\text{Variation due to the initial condition}} + \underbrace{\frac{\partial\Psi(\Psi^{n-1}(\mathbf{u}_t))}{\partial\boldsymbol{\theta}}}_{\text{Variation of the gradient of the solver given the initial condition}}\end{aligned} \tag{S.I. 2}$$

We use (S.I. 2) to construct the following:

$$\frac{\partial}{\partial\boldsymbol{\theta}}\Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1}\Big(\prod_{i=1}^{i=n-j}\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)}\Big)\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{j-1}(\mathbf{u}_t)) + \frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n-1}(\mathbf{u}_t)) \tag{S.I. 3}$$

where all the derivatives with respect to θ are taken assuming that the initial condition is fixed. The proof of (S.I. 3) is conducted by recurrence, and is given in section A.6.

To prove theorem 1, we simply replace in (S.I. 3) the solver Ψ by its Euler approximation given in the proposition 1:

$$\frac{\partial}{\partial\boldsymbol{\theta}}\Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1}\Big(\prod_{i=1}^{i=n-j}\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)}\Big)\frac{\partial}{\partial\boldsymbol{\theta}}(\Psi_E(\Psi^{j-1}(\mathbf{u}_t)) + O(h^2)) + \frac{\partial}{\partial\boldsymbol{\theta}}(\Psi_E(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2)) \tag{S.I. 4}$$

if we develop the expression of the explicit Euler solver, we have:

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi^n(\mathbf{u}_t) &= \sum_{j=1}^{j=n-1}\Big(\prod_{i=1}^{i=n-j}\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)}\Big)\frac{\partial}{\partial\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t) + h(\mathbf{F}(\Psi^{j-1}(\mathbf{u}_t)) + \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))) + O(h^2))\\ &+ \frac{\partial}{\partial\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t) + h(\mathbf{F}(\Psi^{n-1}(\mathbf{u}_t)) + \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t))) + O(h^2))\\ &= \sum_{j=1}^{j=n-1}\Big(\prod_{i=1}^{i=n-j}\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)}\Big)h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t)) + \sum_{j=1}^{j=n-1}\Big(\prod_{i=1}^{i=n-j}\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)}\Big)O(h^2)\\ &+ h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2)\\ &= \sum_{j=1}^{j=n-1}\Big(\prod_{i=1}^{i=n-j}\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)}\Big)h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t)) + h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t)) + nO(h^2)\end{aligned} \tag{S.I. 5}$$

The leading order term in $\sum_{j=1}^{j=n-1}(\prod_{i=1}^{i=n-j}\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)})O(h^2)$ is $O(h^2)$. Furthermore, and since n is fixed, $nO(h^2)$ is simply equal to $O(h^2)$. Reporting this in equation (S.I. 5) completes the proof as follows:

$$\frac{\partial}{\partial\boldsymbol{\theta}}\Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1}\Big(\prod_{i=1}^{i=n-j}\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)}\Big)h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t)) + h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2) \tag{S.I. 6}$$

A.3 Proof of the corollary 1.1

We prove corollary (1.1), we start from (S.I. 5) and replace n by $\frac{t_f-t_0}{h}$. This makes the convergence linear in h:

$$\frac{\partial}{\partial\boldsymbol{\theta}}\Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1}\Big(\prod_{i=1}^{i=n-j}\frac{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial\Psi^{n-i}(\mathbf{u}_t)}\Big)h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t)) + h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t)) + O(h) \tag{S.I. 7}$$

A.4 Proof of the Corollary 1.2

In order to prove corollary 1.2, we write the Jacobian of the taylor expansion of the solver Ψ and we keep the zero order term.

$$\begin{aligned}\frac{\partial}{\partial\Psi(\mathbf{u}_t)}\Psi(\Psi(\mathbf{u}_t)) &= \frac{\partial}{\partial\Psi(\mathbf{u}_t)}(\Psi(\mathbf{u}_t)+\sum_{k=1}^{p}h^k\frac{1}{k!}(\mathbf{F}(\Psi(\mathbf{u}_t))+\mathbf{M}_{\boldsymbol{\theta}}(\Psi(\mathbf{u}_t)))^{k-1}+O(h^{p+1}))\\ &= \mathbf{I}+\frac{\partial}{\partial\Psi(\mathbf{u}_t)}\sum_{k=1}^{p}h^k\frac{1}{k!}(\mathbf{F}(\Psi(\mathbf{u}_t))+\mathbf{M}_{\boldsymbol{\theta}}(\Psi(\mathbf{u}_t)))^{k-1}+O(h^{p+1})\\ &= \mathbf{I}+O(h)\end{aligned} \tag{S.I. 8}$$

If we replace the Jacobian in (17) by the approximation in (S.I. 8):

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi^n(\mathbf{u}_t) &= \sum_{j=1}^{j=n-1}(\prod_{i=1}^{i=n-j}\mathbf{I}+O(h))h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))+h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t))+O(h^2)\\ &= \sum_{j=1}^{j=n-1}(\mathbf{I}+O(h))h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))+h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t))+O(h^2)\\ &= \sum_{j=1}^{j=n-1}h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))+h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t))+\sum_{j=1}^{j=n-1}O(h)h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))+O(h^2)\\ &= \sum_{j=1}^{j=n-1}h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))+h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t))+nO(h^2)\\ &= \sum_{j=1}^{j=n}h\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))+O(h^2)\end{aligned} \tag{S.I. 9}$$

The second part of the corollary, *i.e.* assuming that $n=\frac{t_f-t_0}{h}$ can be proven similarly.

A.5 Proof of the Corollary 1.3

$$\begin{aligned}\frac{\partial}{\partial\Psi(\mathbf{u}_t)}\Psi(\Psi(\mathbf{u}_t)) &= \frac{\partial}{\partial\Psi(\mathbf{u}_t)}(\Psi(\mathbf{u}_t)+\sum_{k=1}^{p}h^k\frac{1}{k!}(\mathbf{F}(\Psi(\mathbf{u}_t))+\mathbf{M}_{\boldsymbol{\theta}}(\Psi(\mathbf{u}_t)))^{k-1}+O(h^{p+1}))\\ &= \frac{\partial}{\partial\Psi(\mathbf{u}_t)}(\Psi(\mathbf{u}_t)+\sum_{k=1}^{p}h^k\frac{1}{k!}(\mathbf{F}(\Psi(\mathbf{u}_t)))^{k-1}+\sum_{k=1}^{p}h^k\frac{1}{k!}(\mathbf{M}_{\boldsymbol{\theta}}(\Psi(\mathbf{u}_t)))^{k-1}+O(h^{p+1}))\\ &= \underbrace{\frac{\partial}{\partial\Psi(\mathbf{u}_t)}(\Psi(\mathbf{u}_t)+\sum_{k=1}^{p}h^k\frac{1}{k!}(\mathbf{F}(\Psi(\mathbf{u}_t)))^{k-1}+O(h^{p+1}))}_{\frac{\partial}{\partial\Psi(\mathbf{u}_t)}\Psi_o(\Psi(\mathbf{u}_t))}+\frac{\partial}{\partial\Psi(\mathbf{u}_t)}\sum_{k=1}^{p}h^k\frac{1}{k!}(\mathbf{M}_{\boldsymbol{\theta}}(\Psi(\mathbf{u}_t)))^{k-1}\\ &= \frac{\partial}{\partial\Psi(\mathbf{u}_t)}\Psi_o(\Psi(\mathbf{u}_t))+\frac{\partial}{\partial\Psi(\mathbf{u}_t)}\sum_{k=1}^{p}h^k\frac{1}{k!}(\mathbf{M}_{\boldsymbol{\theta}}(\Psi(\mathbf{u}_t)))^{k-1}\\ &= TLM(\Psi(\mathbf{u}_t))+\frac{\partial}{\partial\Psi(\mathbf{u}_t)}\sum_{k=1}^{p}h^k\frac{1}{k!}(\mathbf{M}_{\boldsymbol{\theta}}(\Psi(\mathbf{u}_t)))^{k-1}\end{aligned} \tag{S.I. 10}$$

A.6 Proof of the equation (S.I. 3)

The proof of (S.I. 3) is conducted by recurrence. For $n=1$ we have:

$$\frac{\partial}{\partial\boldsymbol{\theta}}\Psi^1(\mathbf{u}_t)=\frac{\partial\Psi(\Psi^0(\mathbf{u}_t))}{\partial\boldsymbol{\theta}} \tag{S.I. 11}$$

which is by inspection of (S.I. 2) true.

Assume that (S.I. 3) is true for all n, for $n+1$ we have:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}}\Psi^{n+1}(\mathbf{u}_t) &= \sum_{j=1}^{j=n}\Big(\prod_{i=1}^{i=n+1-j}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\Big)\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{j-1}(\mathbf{u}_t)) + \frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n}(\mathbf{u}_t)) \\
&= \prod_{i=1}^{i=n}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{0}(\mathbf{u}_t)) \\
&+ \prod_{i=1}^{i=n-1}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{1}(\mathbf{u}_t)) \\
&\vdots \\
&+ \prod_{i=1}^{i=2}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n-2}(\mathbf{u}_t)) \\
&+ \prod_{i=1}^{i=1}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n-1}(\mathbf{u}_t)) \\
&+ \frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n}(\mathbf{u}_t))
\end{aligned} \tag{S.I. 12}$$

if we take $\frac{\partial\Psi(\Psi^{n}(\mathbf{u}_t))}{\partial\Psi^{n}(\mathbf{u}_t)}$ as a common factor we have:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}}\Psi^{n+1}(\mathbf{u}_t) &= \frac{\partial\Psi(\Psi^{n}(\mathbf{u}_t))}{\partial\Psi^{n}(\mathbf{u}_t)}\prod_{i=2}^{i=n}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{0}(\mathbf{u}_t)) \\
&+ \frac{\partial\Psi(\Psi^{n}(\mathbf{u}_t))}{\partial\Psi^{n}(\mathbf{u}_t)}\prod_{i=2}^{i=n-1}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{1}(\mathbf{u}_t)) \\
&\vdots \\
&+ \frac{\partial\Psi(\Psi^{n}(\mathbf{u}_t))}{\partial\Psi^{n}(\mathbf{u}_t)}\prod_{i=2}^{i=2}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n-2}(\mathbf{u}_t)) \\
&+ \frac{\partial\Psi(\Psi^{n}(\mathbf{u}_t))}{\partial\Psi^{n}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n-1}(\mathbf{u}_t)) \\
&+ \frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n}(\mathbf{u}_t))
\end{aligned} \tag{S.I. 13}$$

Factorization of $\frac{\partial\Psi(\Psi^{n}(\mathbf{u}_t))}{\partial\Psi^{n}(\mathbf{u}_t)}$:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}}\Psi^{n+1}(\mathbf{u}_t) = \frac{\partial\Psi(\Psi^{n}(\mathbf{u}_t))}{\partial\Psi^{n}(\mathbf{u}_t)}\Big(&\prod_{i=2}^{i=n}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{0}(\mathbf{u}_t)) \\
&+ \prod_{i=2}^{i=n-1}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{1}(\mathbf{u}_t)) \\
&\vdots \\
&+ \prod_{i=2}^{i=2}\frac{\partial\Psi(\Psi^{n+1-i}(\mathbf{u}_t))}{\partial\Psi^{n+1-i}(\mathbf{u}_t)}\frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n-2}(\mathbf{u}_t)) \\
&+ \frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n-1}(\mathbf{u}_t))\Big) \\
&+ \frac{\partial}{\partial\boldsymbol{\theta}}\Psi(\Psi^{n}(\mathbf{u}_t))
\end{aligned} \tag{S.I. 14}$$

Which is conveniently written as:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\theta}}\Psi^{n+1}(\mathbf{u}_t) = \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)}\Big(&\prod_{i=1}^{i=n-1} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \frac{\partial}{\partial \boldsymbol{\theta}}\Psi(\Psi^0(\mathbf{u}_t)) \\ &+ \prod_{i=1}^{i=n-2} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \frac{\partial}{\partial \boldsymbol{\theta}}\Psi(\Psi^1(\mathbf{u}_t)) \\ &\vdots \\ &+ \prod_{i=1}^{i=1} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \frac{\partial}{\partial \boldsymbol{\theta}}\Psi(\Psi^{n-2}(\mathbf{u}_t)) \\ &+ \frac{\partial}{\partial \boldsymbol{\theta}}\Psi(\Psi^{n-1}(\mathbf{u}_t))\Big) \\ &+ \frac{\partial}{\partial \boldsymbol{\theta}}\Psi(\Psi^n(\mathbf{u}_t))\end{aligned} \tag{S.I. 15}$$

Equation (S.I. 15) can be written in a more compact form as:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\theta}}\Psi^{n+1}(\mathbf{u}_t) &= \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \underbrace{\Big(\sum_{j=1}^{j=n-1} \Big(\prod_{i=1}^{i=n-j} \frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)} \Big) \frac{\partial}{\partial \boldsymbol{\theta}}\Psi(\Psi^{j-1}(\mathbf{u}_t)) + \frac{\partial}{\partial \boldsymbol{\theta}}\Psi(\Psi^{n-1}(\mathbf{u}_t))\Big)}_{=\frac{\partial}{\partial \boldsymbol{\theta}}\Psi^n(\mathbf{u}_t)} \\ &+ \frac{\partial}{\partial \boldsymbol{\theta}}\Psi(\Psi^n(\mathbf{u}_t)) \\ &= \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \Psi^n(\mathbf{u}_t)} \frac{\partial \Psi^n(\mathbf{u}_t)}{\partial \boldsymbol{\theta}} + \frac{\partial \Psi(\Psi^n(\mathbf{u}_t))}{\partial \boldsymbol{\theta}}\end{aligned} \tag{S.I. 16}$$

which is by inspection of (S.I. 2) true. This completes the proof of the formula (S.I. 3).

B Supplementary Results 1: Additional experiment on the two scale Lorenz 96

The two scale L96 system describes a coupled system of equations Lorenz [1995] with S slow variables, $u_t^\dagger = [u_{t,1}^\dagger, u_{t,2}^\dagger, \cdots, u_{t,S}^\dagger]^T$ each of which is coupled to B fast variables $(y_{t,1,s}, y_{t,2,s}, \cdots, y_{t,B,s})$:

$$\begin{aligned}\dot{u}_{t,s}^\dagger &= -u^\dagger{}_{t,s-1}\left(u^\dagger{}_{t,s-2} - u^\dagger{}_{t,s+1}\right) - u^\dagger{}_{t,s} + A + R_{t,s} \\ \dot{y}_{t,b,s} &= -c\gamma y_{t,b+1,s}\left(y_{t,b+2,s} - y_{t,b-1,s}\right) - c y_{t,b,s} + \frac{dc}{\gamma} u^\dagger{}_{t,s}\end{aligned} \tag{S.I. 17}$$

where $R_{t,s} = -\left(\frac{dc}{\gamma}\right)\sum_{b=1}^{B} y_{t,b,s}$

In this experiment, We assume that the physical core $\mathbf{F}$ represents the equations that govern the slow variables and we use a sub-model $\mathbf{M}_{\boldsymbol{\theta}}$ to mimic the impact of the fast variables $y_{t,b,s}$:

$$\dot{\mathbf{u}}_t = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{u}_t) \tag{S.I. 18}$$

where $\mathbf{u}_t = [u_{t,1}, u_{t,1}, \cdots, u_{t,S}]^T \in \mathbb{R}^S$ and $\mathbf{M}_{\boldsymbol{\theta}}$ is a fully connected neural network with parameters θ. The physical core $\mathbf{F} = [F_1, F_2, \cdots, F_S]^T$ is given by:

$$\dot{u}_{t,s} = F_s = -u^\dagger{}_{t,s-1}\left(u^\dagger{}_{t,s-2} - u^\dagger{}_{t,s+1}\right) - u^\dagger{}_{t,s} + A \tag{S.I. 19}$$

The goal of this experiment is to evaluate the proposed Euler approximation of the online learning of the sub-model $\mathbf{M}_{\boldsymbol{\theta}}$ (based on a static approximation of the Jacobian of the flow) on a multiscale dynamical systems, for varying time steps of the Euler approximation. We set the number of slow variables $S = 8$ and the number of fast variables $B = 5$. Regarding the values of the parameters of the equation, we use the following configuration: $A = 8$, $d = 1$, $\gamma = 10$, and $c = 10$. This simulation configuration yields chaotic dynamics, where the statistical properties can not be solely explained by the slow model (S.I. 19) (this can be visualized in S.I. Figure 1, where the PDF of the physical core is

given with respect to the one of the multiscale system). We assume here that the number of time steps n is fixed, and we run a series of two experiments for which the time step of the Euler approximation of the gradients decreases from $h = 0.1$ to $h = 0.05$.

We compare the proposed Euler approximation to both offline and online calibration techniques. The online calibration is carried using an exact gradient and also with an emulator. The emulator is trained sequentially to the sub-model $\mathbf{M}_{\boldsymbol{\theta}}$ as discussed in Results. In the online learning experiments, the training datasets correspond to time series of the full Lorenz 96 system $\{(u^{\dagger}_{t_k+jh_i}, u^{\dagger}_{t_k})|$ with $k = 1 \dots N$ and $j = 1 \dots n\}$ and in the offline learning experiment, the training data corresponds to $\{(u^{\dagger}_{t_k}, \mathbf{R}_{t_k} = [R_{t_k,1}, R_{t_k,2}, \cdots, R_{t_k,S})]^T|$ with $k = 1 \dots N\}$. The number of simulation steps is set to $n = 10$ and the size of the dataset N is equal to 20000.

Overall, the following models are tested:

- **Online calibration with exact gradient**: In this calibration scheme, we implement the solver of (S.I. 18) in a differentiable language and we optimize the parameters of the sub-model with the exact gradient of the online objective function.
- **Online calibration with a static approximation**: In this experiment, we evaluate the performance of the proposed online learning scheme in which the gradient of the online loss function is approximated based on the Static-EGA (19). In this experiment, the solver of (S.I. 18) is not assumed to be differentiable.
- **Online calibration with an emulator**: We also evaluate and compare the proposed approximation schemes to online learning with emulators as presented in the related works subsection. We recall that in this experiment, physical core $\mathbf{F}$ in (S.I. 18) is replaced (in the training phase) by a neural network that is trained sequentially with $\mathbf{M}_{\boldsymbol{\theta}}$. This network is a linear quadratic model, similar to the one discussed in Fablet et al. [2018].
- **Offline calibration**: We also compare the proposed static approximation to a simple offline learning strategy. In this experiment, the parameters of the sub-model are optimized to minimize the following offline objective function:

$$\mathcal{Q} = \frac{1}{N} \sum_{k} \left\| \mathbf{R}_{t_k} - \mathbf{M}_{\boldsymbol{\theta}}(u^{\dagger}_{t_k}) \right\|^2 \tag{S.I. 20}$$

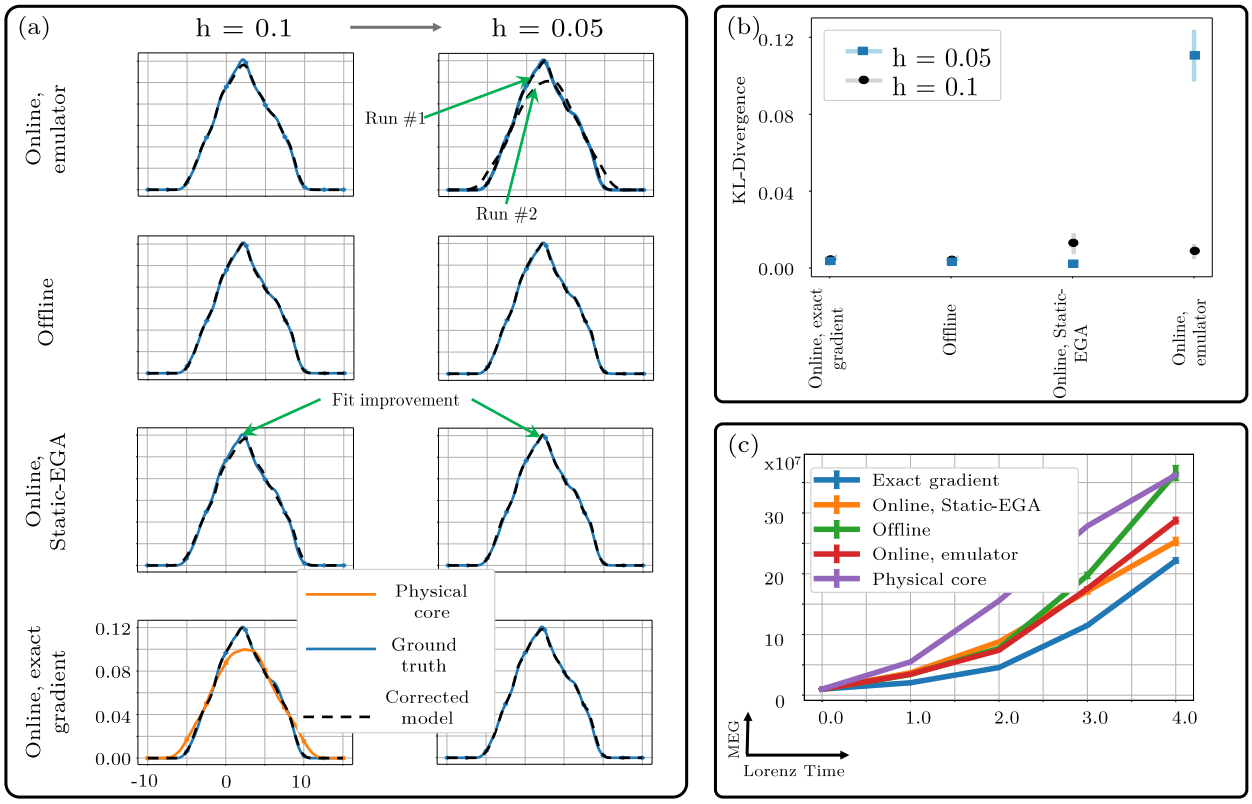
Besides the offline calibration scheme, the online cost function used in this experiment is the mean squared error of the numerical integration of (S.I. 18) with respect to the true two-scale Lorenz 96 sequence. All the tested experiments share the same parameterization of the sub-model $\mathbf{M}_{\boldsymbol{\theta}}$, which is a fully connected neural network with 6 hidden layers, each with 100 neurons and a hyperbolic tangent activation.

B.1 Performance of the learnt sub-model

We plot the performance of both online and offline optimization approaches in the Lorenz 96 case study in S.I. Figure 1. We evaluate the approaches with respect to the true Lorenz 96 simulation and also with respect to a simulation issued from the physical core. Overall, we notice that both offline and online schemes noticeably outperform the physical core which highlights the relevance of using such corrections. Regarding the short-term prediction performance, panel (c) of S.I. Figure 1 shows that all models are able to provide better predictions than the physical core. in this experiment, the online learning with true gradient is able to provide the best prediction performance. When evaluating the qualitative properties of the simulation of the models highlighted for instance in panels (a) and (b) in S.I. Figure 1, We also found that the proposed approximate gradient for the online learning scheme provides a very nice correction that is on the same level as online optimization with the true gradient. The proposed approximation also improves when we reduce the time step of the Euler approximation of the gradient from $h = 0.1$ to $h = 0.05$. This experiment also reveals that online learning with an emulator can be challenging. Specifically, the qualitative and quantitative comparison of the PDF of the models trained online with an emulator in S.I. Figure 1 shows that the sub-model calibration can be highly sensitive and shows that two different model initializations can lead to distinct sub-models.

C Supplementary Results 2: Additional experiment on the Lorenz 63 model

We also analyse in S.I. Figure 2 the memory and time complexities of the EGA compared to standard backpropagation through a DOPRI8 solver. The EGA achieves a significant reduction in both execution time and memory usage while maintaining error levels below 38%.



S.I. Figure 1: ***Simulation example of the tested models in the Lorenz 96 esperiment***. We compare a multiscale simulation of the true system (S.I. 17) to the physical core in (30) (without the neural network sub-model) and to corrected models where the correction term $\mathbf{M}_{\boldsymbol{\theta}}$ is calibrated both online and offline. We compare the PDF of the first state of the tested models with respect to the one computed from a simulation of the true system given by (S.I. 17) both qualitatively in (a) and quantitatively in (b). (c) Mean error growth of the tested models. We plot both the mean and standard deviation that was computed based on an ensemble of 20 trajectories issued from 5 different learning realizations. The error bars are scaled by 1/20.

D Supplementary Methods 1: Algorithms of the proposed online learning methodology

D.1 Gradient evaluation using composable function transforms

A direct evaluation of (27) can be based on composable function transforms of modern languages such as JAX **?** or PYTORCH (based on the functorch tool). These tools allow to evaluate vector valued gradients (not only vector Jacobian products), and it can be adapted to the computation of (27). Algorithm 1 highlights how this can be achieved.

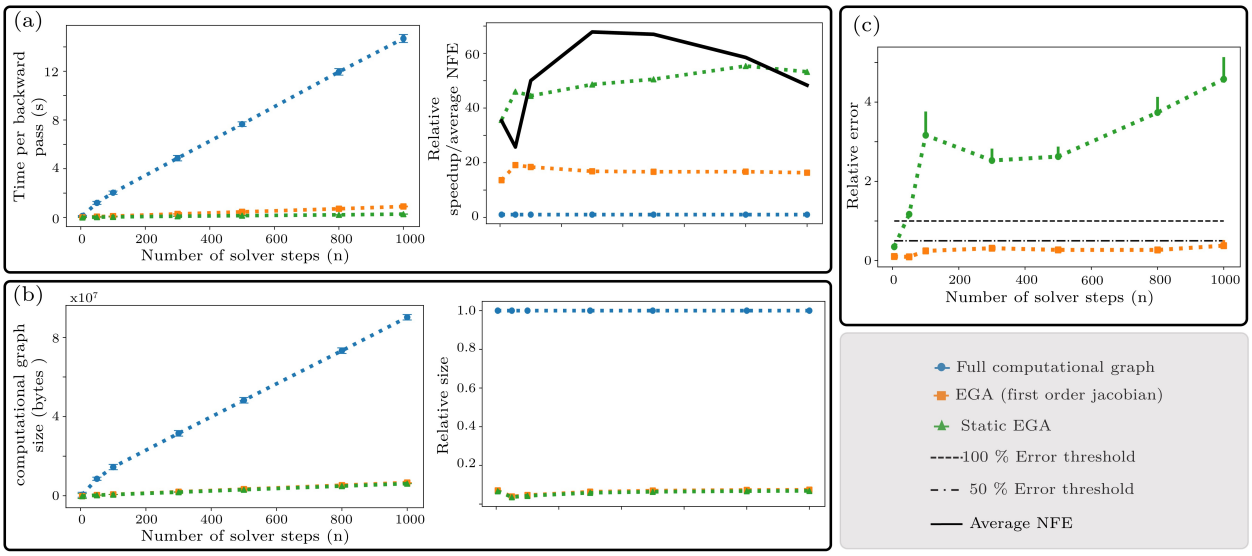
D.2 Modification of the backward call

The gradient of the online cost function can be computed by a modification of the backward call of modern automatic differentiation languages. The idea is to construct a ResNet-like computational graph using the sub-model $\mathbf{M}_{\boldsymbol{\theta}}$. We modify the gradient of the output of each ResNet block using a hook to include the information of the Jacobian of the non-differentiable solver. And the gradient of each block will correspond to $\partial \mathbf{M}_{\boldsymbol{\theta}}(\cdot)/\partial \boldsymbol{\theta}$. We provide an implementation of this technique, inspired by the syntax of PyTorch, in Algorithm 2.

S.I. References

E.N. Lorenz. *Predictability: a problem partly solved*. PhD thesis, Shinfield Park, Reading, 1995 1995.

R. Fablet, S. Ouala, and C. Herzet. Bilinear residual neural network for the identification and forecasting of geophysical dynamics. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 1477–1481, Sep. 2018. doi:10.23919/EUSIPCO.2018.8553492.



S.I. Figure 2: ***Analysis of the computational complexity of the EGA with respect to backpropagation through the DOPRI8 numerical solver***. (a) Wall time per backward pass and speedup relative to the standard backpropagation through the numerical solver. (b) Size of the computational graph and relative size with respect to standard backpropagation. (c) Relative error of the EGA with respect to backpropagation through the numerical solver (size of the error bars of the figures in panel c was divided by 50). This benchmark was executed on a single NVIDIA GeForce GTX 1080 Ti GPU.

Algorithm 1 Gradient computation based on composable function transforms

Input:
Ψ: Non-differentiable solver of the hybrid system (5)
$\mathbf{M}_{\boldsymbol{\theta}}$: Deep learning based sub-model
$\mathbf{u}_t$: Initial condition
n, h: Number of simulation steps and time step
$\mathcal{J}_{online}$: Online cost function
l : Approximation scheme for the Jacobian of the flow
return
▷ *Iterate through the solver* Ψ
for $j \leftarrow 1$ **to** n **do**
$\mathbf{u}_{t+jh} = \Psi^j(\mathbf{u}_t)$
end for
▷ *Precompute the Jacobians* $\mathbf{J}_{j,l}$
for $j \leftarrow 1$ **to** $n-1$ **do**
$\mathbf{J}_{j,l} = \prod_{i=1}^{i=n-j} {}^{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}/_{\partial \Psi^{n-i}(\mathbf{u}_t)}$
end for
▷ *Compute the vector valued gradients of the sub-model*
for $j \leftarrow 1$ **to** n **do**
${}^{\partial \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))}/_{\partial \boldsymbol{\theta}}$ ▷ *This can be done, for example, using functorch*
end for
▷ *Compute the Euler approximation of the gradient of the solver*
$\mathbf{A}_{l,p} = \sum_{j=1}^{j=n-1} \mathbf{J}_{j,l} h {}^{\partial}/_{\partial \boldsymbol{\theta}} \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t)) + h {}^{\partial}/_{\partial \boldsymbol{\theta}} \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t)))$
▷ *Compute the remaining gradients and evaluate the gradient of the online cost*
$\mathbf{v} = {}^{\partial Q(\cdot,\cdot,\theta)}/_{\partial \boldsymbol{\theta}}$
$\mathbf{w} = {}^{\partial Q(\cdot,\mathbf{g}(\Psi^n(\mathbf{u}_t)),\cdot)}/_{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \Psi}$
${}^{\partial \mathcal{J}_{online}}/_{\partial \boldsymbol{\theta}} = \mathbf{v} + \mathbf{w}\mathbf{A}_{l,p}(\mathbf{u}_t)$

Algorithm 2 Gradient computation based on backward call modification

Input:
Ψ: Non-differentiable solver of the hybrid system (5)
$\mathrm{M}_{\boldsymbol{\theta}}$: Deep learning based sub-model
$\mathbf{u}_t$: Initial condition
n, h: Number of simulation steps and time step
$\mathcal{J}_{online}$: Online cost function
l : Approximation scheme for the Jacobian of the flow
return
▷ *Backward hook function*
def HOOK($\mathbf{z}_{t+jh}, j$)
▷ grad$(\cdot)$ *refers to a modification of the gradient*
grad$(\mathbf{z}_{t+jh})$ = $h \cdot$ grad$(\mathbf{z}_{t+nh}) \cdot \mathbf{J}_{j,l}$ ▷ *notice that* $\mathbf{w} = grad(\mathbf{z}_{t+nh})$
end def
▷ *Iterate through the solver* Ψ
for $j \leftarrow 1$ **to** n **do**
$\mathbf{u}_{t+jh} = \Psi^j(\mathbf{u}_t)$
end for
▷ *Precompute the Jacobians* $\mathbf{J}_{j,l}$
for $j \leftarrow 1$ **to** $n-1$ **do**
$\mathbf{J}_{j,l} = \prod_{i=1}^{i=n-j} {}^{\partial\Psi(\Psi^{n-i}(\mathbf{u}_t))}\!/_{\partial\Psi^{n-i}(\mathbf{u}_t)}$
end for
▷ *Generate a ResNet like computational graph*
$\mathbf{z}_t = \mathbf{u}_t$ ▷ *Initialize the ResNet state*
for $j \leftarrow 1$ **to** n **do**
$\mathbf{z}_{t+jh} = \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{z}_{t+(j-1)h})$
data$(\mathbf{z}_{t+jh}) = \mathbf{u}_{t+jh}$ ▷ data$(\cdot)$ *refers to a modification of the value*
if $j \neq n$ **then**
▷ *Modify the gradient of the ResNet State*
HOOK$(\mathbf{z}_{t+jh}, j)$
end if
end for
▷ *Compute the online objective function*
$Q(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \boldsymbol{\theta}) = Q(\cdot, \mathbf{g}(\mathbf{z}(\mathbf{u}_{t+nh})), \boldsymbol{\theta})$ ▷ *The simulated states now have a computational graph*
Backward$(Q(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \boldsymbol{\theta}))$ ▷ *Run a backward call*