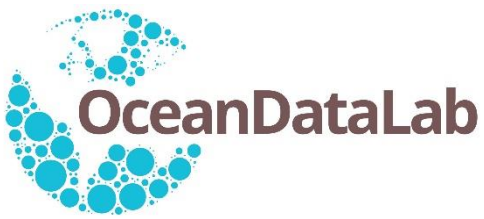
End-to-end Learning in Hybrid Modeling Systems: How to Deal with Backpropagation Through Numerical Solvers

Said Ouala[1], Bertrand Chapron[2], Fabrice Collard[3], Lucile Gaultier[3], and Ronan Fablet[1]

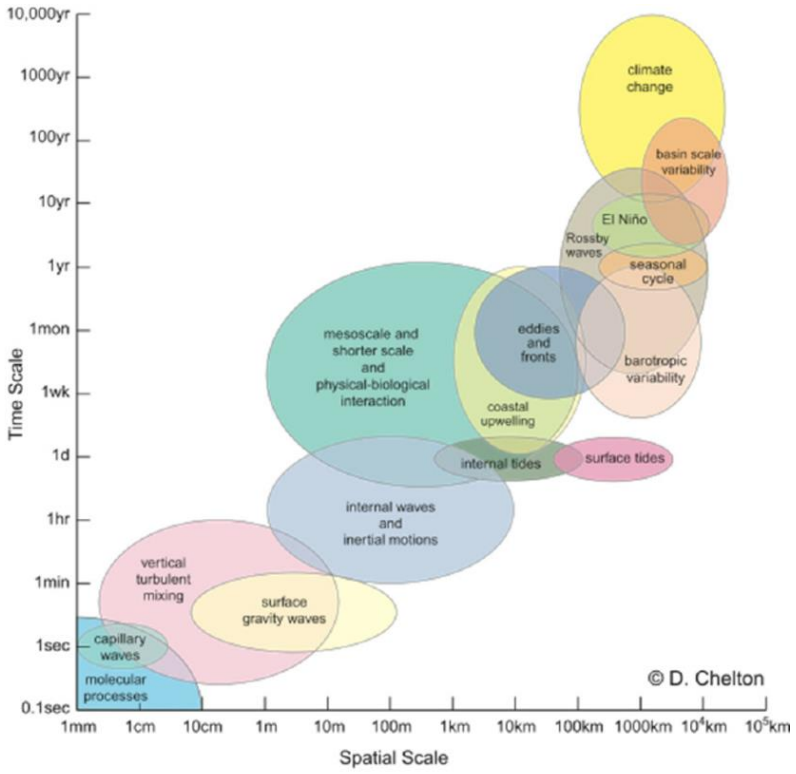
1) IMT Atlantique, Mathematical and Electrical Engineering department, 29280 Plouzané, France
2) Ifremer, LOPS, 29280 Plouzané, France
3) OceanDataLab, 29280 Locmaria-Plouzané, France



Online learning of hybrid models

Tuning geophysical models: Example on the ocean

Reality

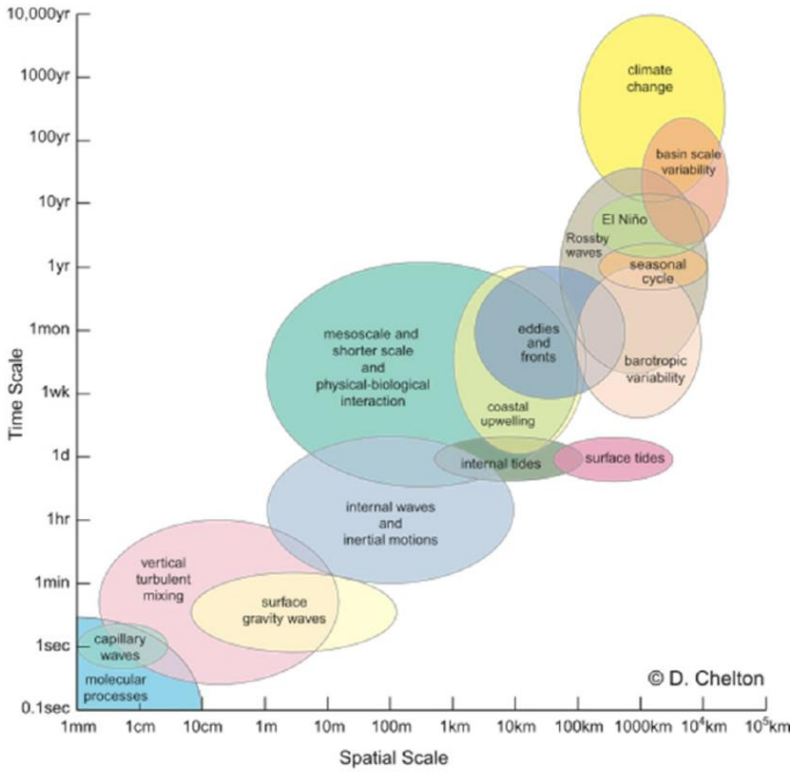


$$\begin{cases} \frac{\partial \mathbf{u}^\dagger{}_t}{\partial t} & = & f(\mathbf{u}^\dagger_t) \\ \mathbf{u}^\dagger_t & \in & \Omega \end{cases}$$

Online learning of hybrid models

Tuning geophysical models: Example on the ocean

Reality

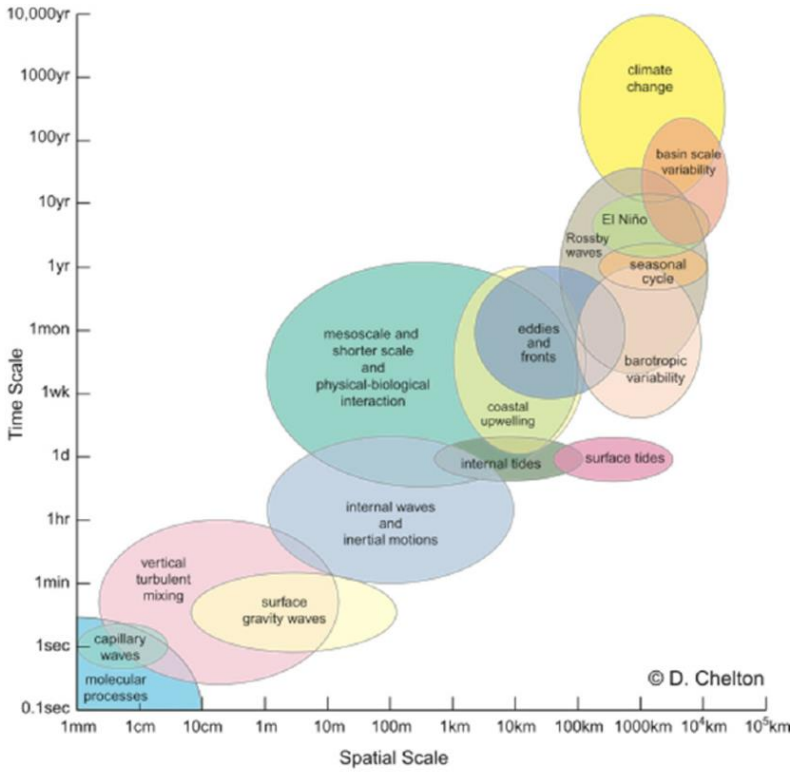


$$\left\{ \begin{array}{lcl} \frac{\partial \mathbf{u}^\dagger{}_t}{\partial t} & = & f(\mathbf{u}^\dagger_t) \\ \mathbf{u}^\dagger_t & \in & \Omega \end{array} \right.$$

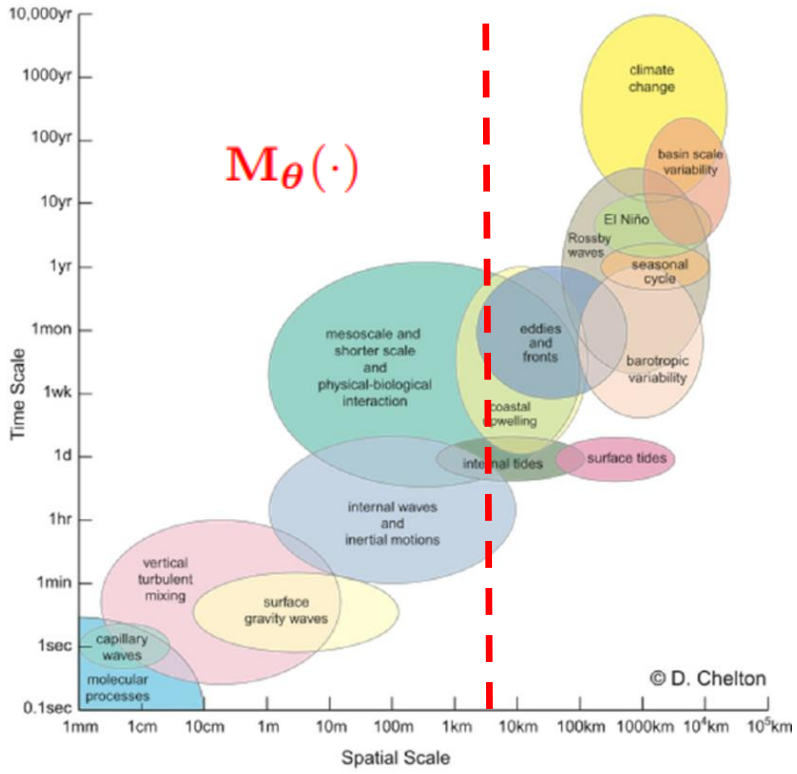
Online learning of hybrid models

Tuning geophysical models: Example on the ocean

Reality



Computer

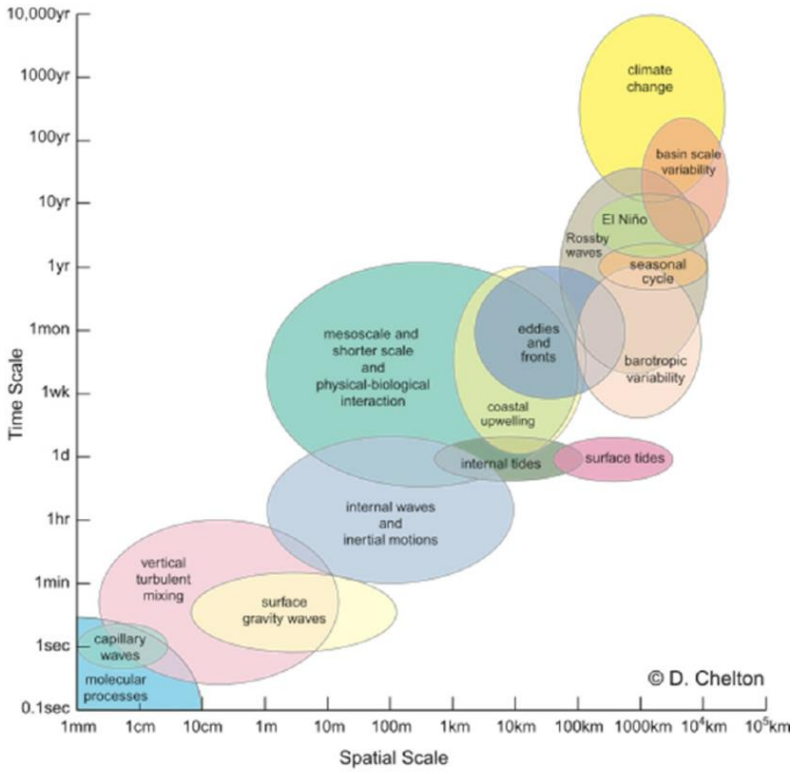


$$\begin{cases} \frac{\partial \mathbf{u}^\dagger{}_t}{\partial t} & = & f(\mathbf{u}^\dagger_t) \\ \mathbf{u}^\dagger_t & \in & \Omega \end{cases}$$

Online learning of hybrid models

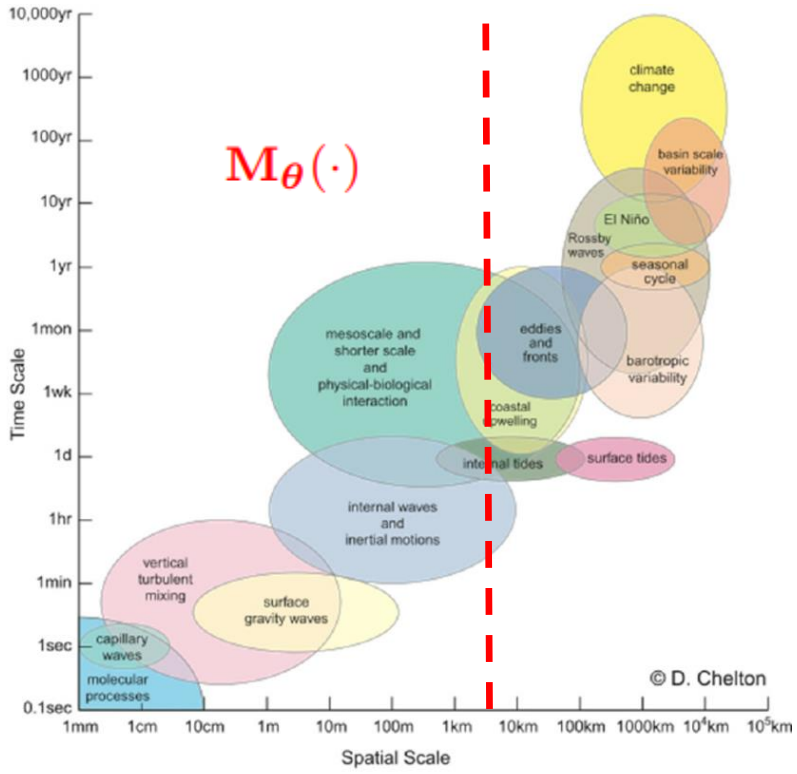
Tuning geophysical models: Example on the ocean

Reality



$$\left\{ \begin{array}{lcl} \frac{\partial \mathbf{u}^\dagger{}_t}{\partial t} & = & f(\mathbf{u}^\dagger_t) \\ \mathbf{u}^\dagger_t & \in & \Omega \end{array} \right.$$

Computer

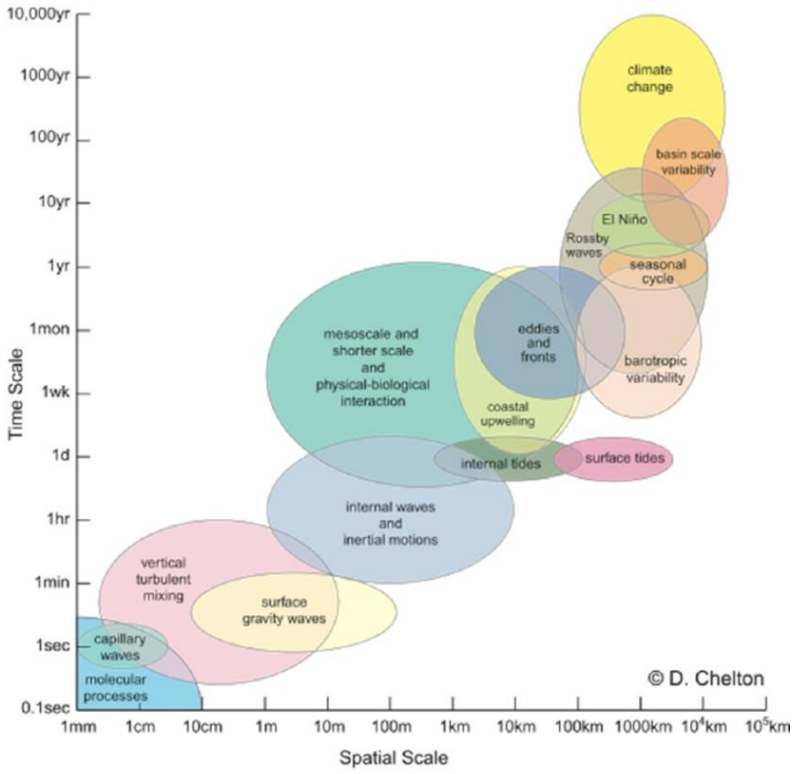


$$\left\{ \begin{array}{lcl} \frac{\partial \mathbf{u}_t}{\partial t} & = & \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\cdot) \\ \tau(\mathbf{u}^\dagger_t) & = & \mathbf{u}_t \in \bar{\Omega} \end{array} \right.$$

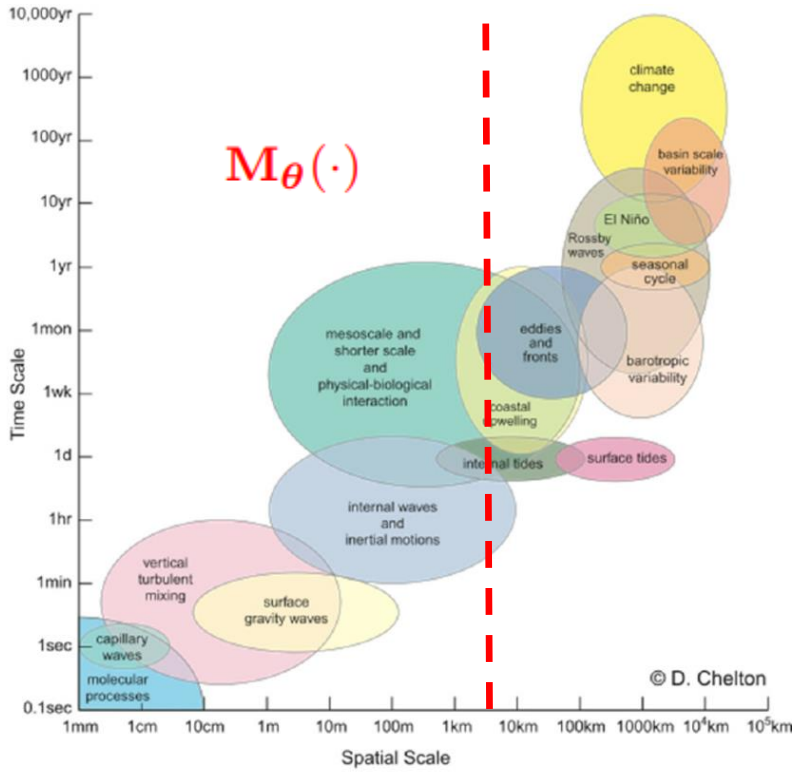
Online learning of hybrid models

Tuning geophysical models: Example on the ocean

Reality



Computer



$$\begin{cases} \frac{\partial \mathbf{u}^\dagger{}_t}{\partial t} & = & f(\mathbf{u}^\dagger_t) \\ \mathbf{u}^\dagger_t & \in & \Omega \end{cases}$$

$$\begin{cases} \frac{\partial \mathbf{u}_t}{\partial t} & = & \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\cdot) \\ \mathbf{u}_t & = & \mathbf{u}_t \in \bar{\Omega} \end{cases}$$

Online learning with hybrid models ?

Online learning of hybrid models

Training hybrid models

Definition of a hybrid model:

$$\frac{\partial \mathbf{u}_t}{\partial t} = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\cdot)$$

Online learning of hybrid models

Training hybrid models

Definition of a hybrid model:

$$\frac{\partial \mathbf{u}_t}{\partial t} = \overbrace{\mathbf{F}(\mathbf{u}_t)}^{\text{Physical core}} + \mathbf{M}_{\boldsymbol{\theta}}(\cdot)$$

Online learning of hybrid models

Training hybrid models

Definition of a hybrid model:

$$\frac{\partial \mathbf{u}_t}{\partial t} = \underbrace{\mathbf{F}(\mathbf{u}_t)}_{\text{Physical core}} + \underbrace{\mathbf{M}_{\boldsymbol{\theta}}(\cdot)}_{\text{Sub-model}}$$

Online learning of hybrid models

Training hybrid models

Definition of a hybrid model:

Physical core | Sub-model

$$\frac{\partial \mathbf{u}_t}{\partial t} = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\cdot)$$

Physical core

$$\partial_t u = \delta u, \quad \delta u = fv - \frac{1}{\cos\phi}\partial_x p_{hyd} - \frac{1}{\cos\phi}\partial_x p_s + \frac{uv}{a}\tan\phi - \boldsymbol{\nabla}\cdot\boldsymbol{u}u + \partial_z A_v \partial_z u$$

$$\partial_t v = \delta v, \quad \delta v = -fu - \partial_y p_{hyd} - \partial_y p_s - \frac{u^2}{a}\tan\phi + \partial_z A_v \partial_z v - \boldsymbol{\nabla}\cdot\boldsymbol{u}v$$

$$0 = \frac{1}{\cos\phi}\left(\partial_x u + \partial_y (v\cos\phi)\right) + \partial_z w$$

$$0 = p_{hyd} - \int_z^0 dz g\rho/\rho_0$$

$$\partial_t S = \delta S, \quad \delta S = -\boldsymbol{\nabla}\cdot\boldsymbol{u}S + D_S, \quad \boldsymbol{u}\cdot S = \frac{1}{\cos\phi}\left(\partial_x(uS) + \partial_y(vS\cos\phi)\right) + \partial_z(wS)$$

$$\partial_t T = \delta T, \quad \delta T = -\boldsymbol{\nabla}\cdot\boldsymbol{u}T + D_T, \quad \boldsymbol{u}\cdot T = \frac{1}{\cos\phi}\left(\partial_x(uT) + \partial_y(vT\cos\phi)\right) + \partial_z(wT)$$

$$0 = -\left(\partial_x \frac{1}{\cos\phi} h\partial_x p_s + \partial_y h\cos\phi\partial_y p_s\right) + \left(\partial_x \int_{-h}^0 dz\delta u + \partial_y \cos\phi \int_{-h}^0 dz\delta v\right)$$

$$\rho = f(p, T, S)$$

Online learning of hybrid models

Training hybrid models

Definition of a hybrid model:

Physical core | Sub-model

$$\frac{\partial \mathbf{u}_t}{\partial t} = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\cdot)$$

Physical core

$$\partial_t u = \delta u, \quad \delta u = fv - \frac{1}{\cos\phi}\partial_x p_{hyd} - \frac{1}{\cos\phi}\partial_x p_s + \frac{uv}{a}\tan\phi - \boldsymbol{\nabla}\cdot\boldsymbol{u}u + \partial_z A_v \partial_z u$$

$$\partial_t v = \delta v, \quad \delta v = -fu - \partial_y p_{hyd} - \partial_y p_s - \frac{u^2}{a}\tan\phi + \partial_z A_v \partial_z v - \boldsymbol{\nabla}\cdot\boldsymbol{u}v$$

$$0 = \frac{1}{\cos\phi}\left(\partial_x u + \partial_y(v\cos\phi)\right) + \partial_z w$$

$$0 = p_{hyd} - \int_z^0 dz g\rho/\rho_0$$

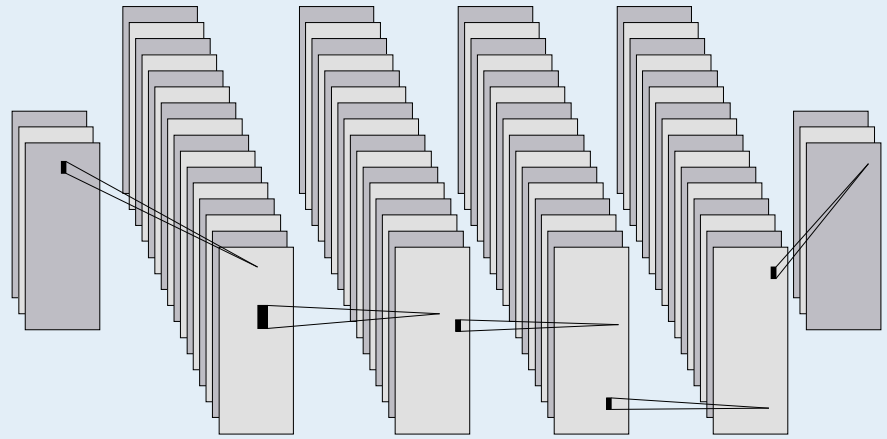
$$\partial_t S = \delta S, \quad \delta S = -\boldsymbol{\nabla}\cdot\boldsymbol{u}S + D_S, \quad \boldsymbol{u}\cdot S = \frac{1}{\cos\phi}\left(\partial_x(uS) + \partial_y(vS\cos\phi)\right) + \partial_z(wS)$$

$$\partial_t T = \delta T, \quad \delta T = -\boldsymbol{\nabla}\cdot\boldsymbol{u}T + D_T, \quad \boldsymbol{u}\cdot T = \frac{1}{\cos\phi}\left(\partial_x(uT) + \partial_y(vT\cos\phi)\right) + \partial_z(wT)$$

$$0 = -\left(\partial_x \frac{1}{\cos\phi} h\partial_x p_s + \partial_y h\cos\phi\partial_y p_s\right) + \left(\partial_x \int_{-h}^0 dz\delta u + \partial_y \cos\phi \int_{-h}^0 dz\delta v\right)$$

$$\rho = f(p, T, S)$$

Deep learning based sub-model



Online learning of hybrid models

Training hybrid models

Definition of a hybrid model:

Physical core

Sub-model

$$\frac{\partial \mathbf{u}_t}{\partial t} = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\cdot)$$

The model is solved using some appropriate numerical solver:

$$\Psi^n(\mathbf{u}_t) = \mathbf{u}_{t+nh} \approx \mathbf{u}_t + \int_{t_0}^{t0+nh} (\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{u}_t))dt$$

Online learning of hybrid models

Training hybrid models

Definition of a hybrid model:

Physical core | Sub-model

$$\frac{\partial \mathbf{u}_t}{\partial t} = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\cdot)$$

The model is solved using some appropriate numerical solver:

$$\Psi^n(\mathbf{u}_t) = \mathbf{u}_{t+nh} \approx \mathbf{u}_t + \int_{t_0}^{t0+nh} (\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{u}_t))dt$$

- How to calibrate the parameters of the model **online** given some observations ?

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}; \text{ where } \mathcal{L} = Q(\mathbf{y}_{t+nh}, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \boldsymbol{\theta})$$

Online learning of hybrid models

Training hybrid models

Definition of a hybrid model:

Physical core | Sub-model

$$\frac{\partial \mathbf{u}_t}{\partial t} = \mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\cdot)$$

The model is solved using some appropriate numerical solver:

$$\Psi^n(\mathbf{u}_t) = \mathbf{u}_{t+nh} \approx \mathbf{u}_t + \int_{t_0}^{t0+nh} (\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{u}_t))dt$$

- How to calibrate the parameters of the model **online** given some observations ?

$$\hat{\theta} = \arg\min_{\theta} \mathcal{L}; \text{ where } \mathcal{L} = Q(\mathbf{y}_{t+nh}, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \boldsymbol{\theta})$$

Online learning of hybrid models

Training hybrid models, end-to-end learning

Online learning:

- Need to do back-propagation through the solver

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} Q(\mathbf{y}_{t+nh}, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \boldsymbol{\theta})$$

Online learning of hybrid models

Training hybrid models, end-to-end learning

Online learning:

- Need to do back-propagation through the solver

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} Q(\mathbf{y}_{t+nh}, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \boldsymbol{\theta})$$

$$= \underbrace{\frac{\partial Q(\cdot, \cdot, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}}_{\text{Gradient of the regularization}} + \underbrace{\frac{\partial Q(\cdot, \mathbf{g}(\Psi^n(\mathbf{u}_t)), \cdot)}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \Psi}}_{\text{Gradient of the online cost w.r.t. } \Psi} \underbrace{\frac{\partial \Psi^n(\mathbf{u}_t)}{\partial \boldsymbol{\theta}}}_{\text{Gradient of the solver}}$$

Online learning of hybrid models
Euler Gradient Approximation

- Let us consider an explicit Euler solver Ψ_{E}, a single step integration using Ψ_{E} can be written as:

$$\mathbf{u}_{t+h} = \Psi_E(\mathbf{u}_t)$$

where

$$\Psi_E(\mathbf{u}_t) = \mathbf{u}_t + h(\mathbf{F}(\mathbf{u}_t) + \mathbf{M}_{\boldsymbol{\theta}}(\mathbf{u}_t))$$

- Assuming that the solver Ψ has order p ≥ 1, we can write for any initial condition:

$$\begin{aligned}\mathbf{u}_{t+h} &= \Psi(\mathbf{u}_t) \\ &= \Psi_E(\mathbf{u}_t) + O(h^2)\end{aligned}$$

Online learning of hybrid models
Euler Gradient Approximation

- By using this approximation, we can show that the gradient of the solver can be decomposed as (for a fixed n):

$$\frac{\partial}{\partial \boldsymbol{\theta}} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1} \Big(\prod_{i=1}^{i=n-j} \underbrace{\frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)}}_{\textit{Jacobian of the flow}} \Big) h \underbrace{\frac{\partial}{\partial \boldsymbol{\theta}} \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))}_{\textit{Gradient of the sub-model}} + h \frac{\partial}{\partial \boldsymbol{\theta}} \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2)$$

- For a fixed n, the gradients converge to the true ones quadratically in h;
- If we approximate the Jacobian (we can use a static/ensemble approximation, a TLM if any), we can compute the gradients only using the gradient of the sub-model;

Online learning of hybrid models
Euler Gradient Approximation

- By using this approximation, we can show that the gradient of the solver can be decomposed as (for a fixed n):

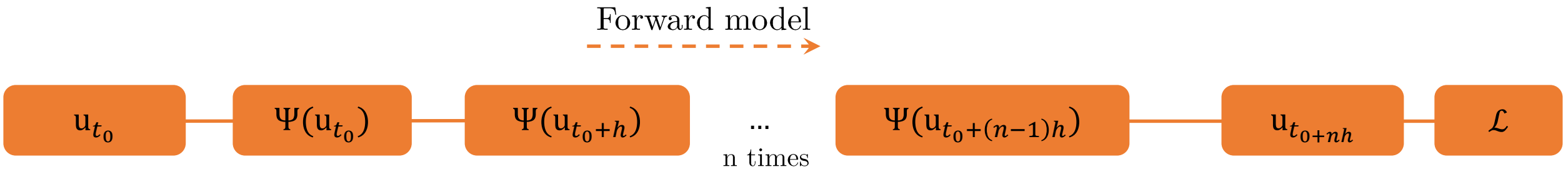
$$\frac{\partial}{\partial \boldsymbol{\theta}} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n-1} \Big(\prod_{i=1}^{i=n-j} \underbrace{\frac{\partial \Psi(\Psi^{n-i}(\mathbf{u}_t))}{\partial \Psi^{n-i}(\mathbf{u}_t)}}_{\textit{Jacobian of the flow}} \Big) h \underbrace{\frac{\partial}{\partial \boldsymbol{\theta}} \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t))}_{\textit{Gradient of the sub-model}} + h \frac{\partial}{\partial \boldsymbol{\theta}} \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{n-1}(\mathbf{u}_t)) + O(h^2)$$

- For a fixed n, the gradients converge to the true ones quadratically in h;
- If we approximate the Jacobian (we can use a static/ensemble approximation, a TLM if any), we can compute the gradients only using the gradient of the sub-model;

$$\frac{\partial}{\partial \boldsymbol{\theta}} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n} h \frac{\partial}{\partial \boldsymbol{\theta}} \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t)) + O(h^2)$$

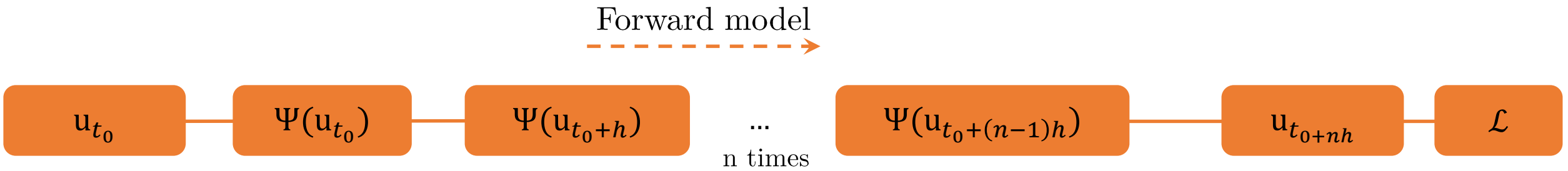
Online learning of hybrid models

Euler Gradient Approximation, in practice



Online learning of hybrid models

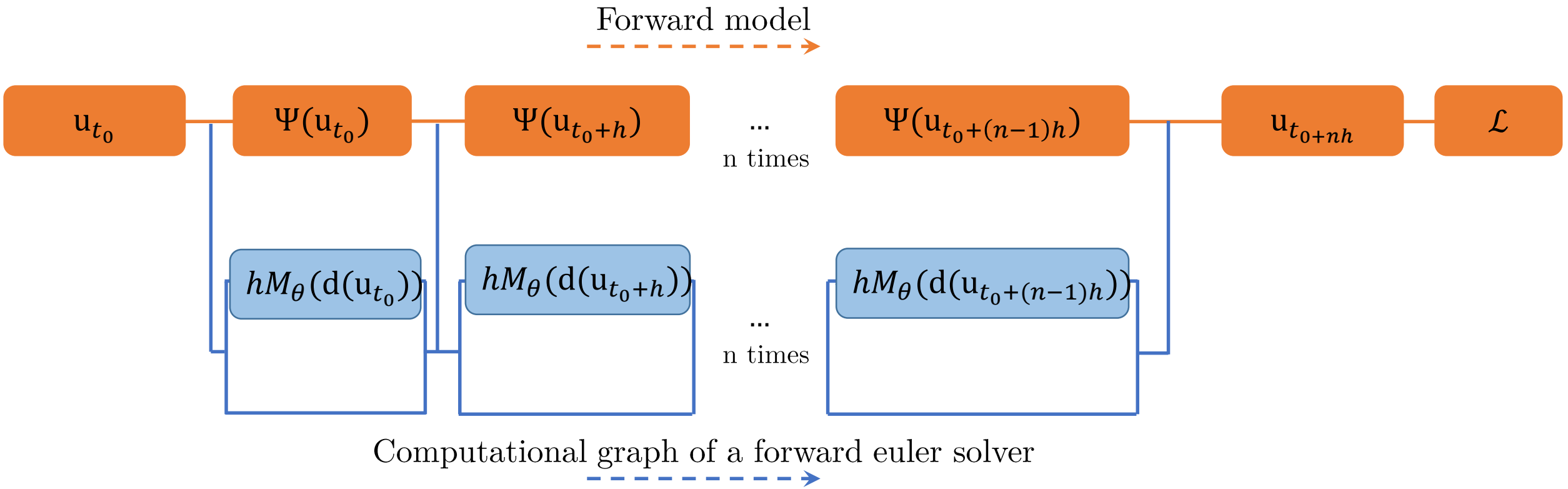
Euler Gradient Approximation, in practice



$$\frac{\partial \mathcal{L}}{\partial \theta} = ?$$

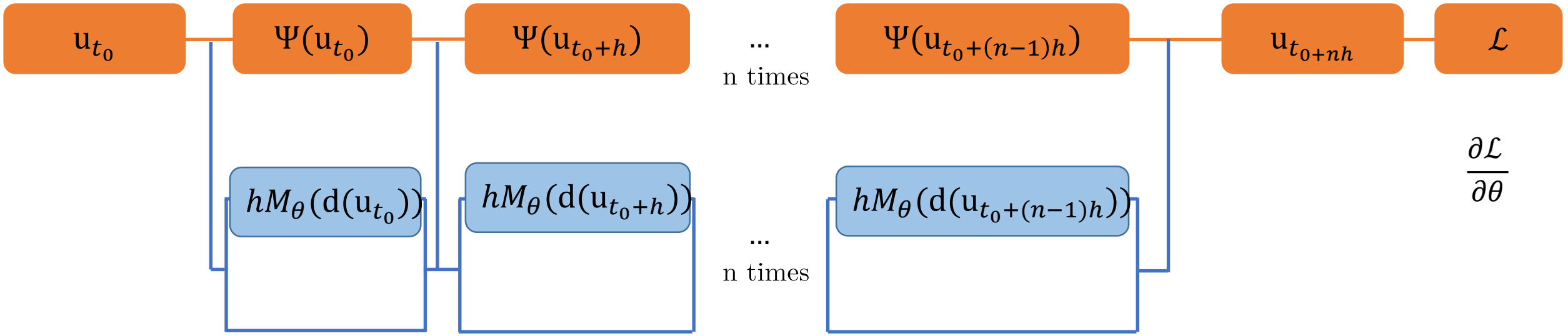
Online learning of hybrid models

Euler Gradient Approximation



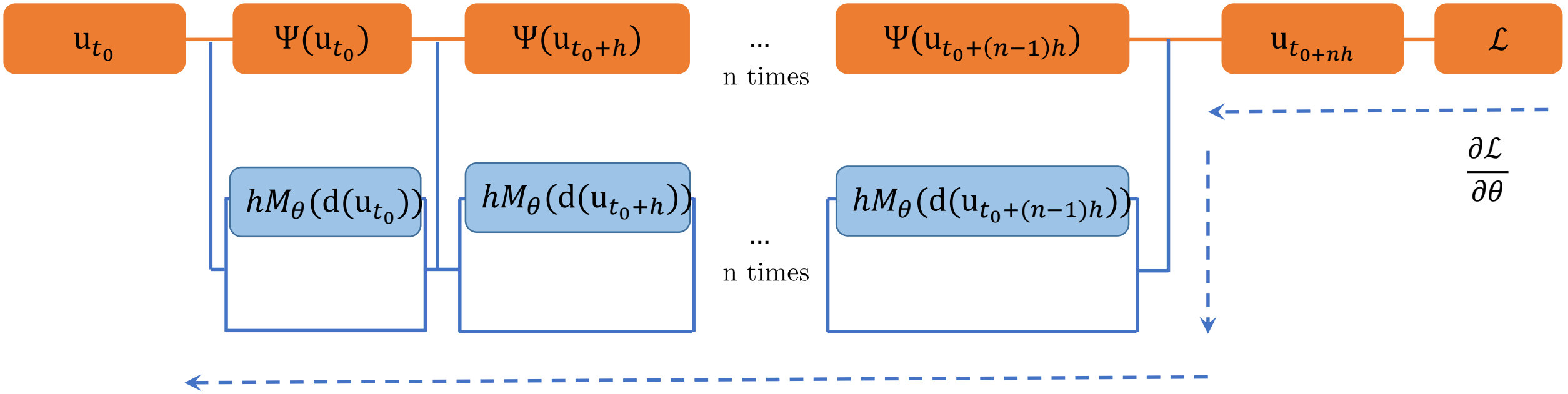
Online learning of hybrid models

Euler Gradient Approximation



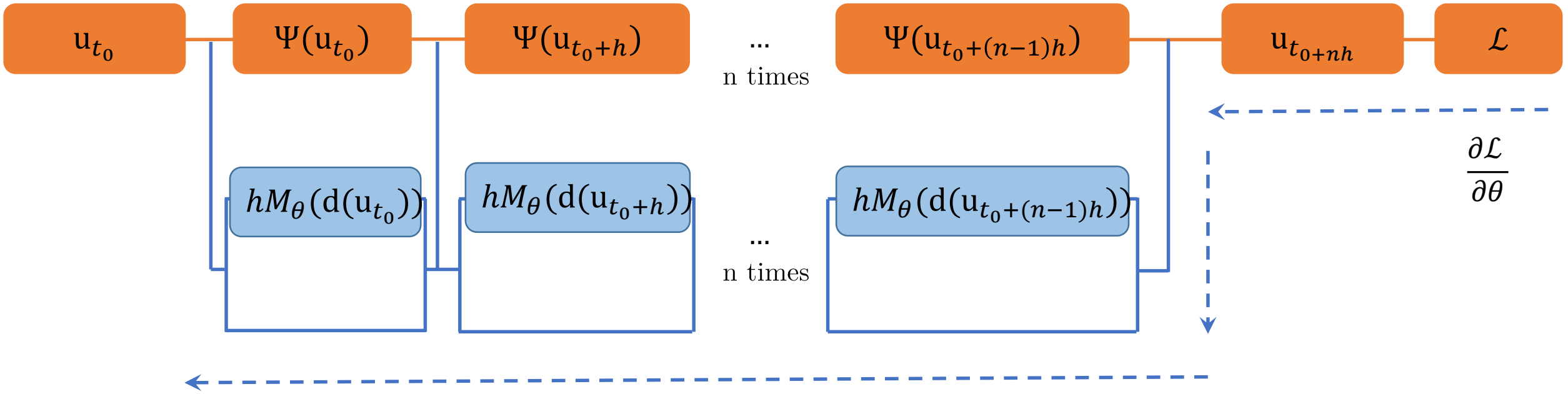
Online learning of hybrid models

Euler Gradient Approximation



Online learning of hybrid models

Euler Gradient Approximation



$$\frac{\partial}{\partial \boldsymbol{\theta}} \Psi^n(\mathbf{u}_t) = \sum_{j=1}^{j=n} h \frac{\partial}{\partial \boldsymbol{\theta}} \mathbf{M}_{\boldsymbol{\theta}}(\Psi^{j-1}(\mathbf{u}_t)) + O(h^2)$$

QG turbulence

- The dimensionless governing equations in the vorticity (ω) and stream function (ψ) formulation in a doubly periodic square domain with length L = 2π are:

$$\frac{\partial \omega_t}{\partial t} + \mathcal{A}(\omega_t, \psi_t) = \frac{1}{\mathrm{Re}} \nabla^2 \omega_t - f - r\omega_t$$
$$\nabla^2 \psi_t = -\omega_t$$

- where $\mathcal{A}(\omega_t, \psi_t)$r represents the nonlinear advection term:

$$\mathcal{A}(\omega_t, \psi_t) = \frac{\partial \psi_t}{\partial y}\frac{\partial \omega_t}{\partial x} - \frac{\partial \psi_t}{\partial x}\frac{\partial \omega_t}{\partial y}$$

- and f represents a deterministic forcing:

$$f(x, y) = k_f \left[\cos\left(k_f x\right) + \cos\left(k_f y\right)\right]$$

QG turbulence, LES

- The dimensionless governing equations in the vorticity (ω) and stream function (ψ) formulation in a doubly periodic square domain with length $\mathrm{L} = 2\pi$ are:

$$\frac{\partial \omega_t}{\partial t} + \mathcal{A}(\omega_t, \psi_t) = \frac{1}{\mathrm{Re}}\nabla^2\omega_t - f - r\omega_t \qquad \xrightarrow{\overline{(\cdot)}} \qquad \frac{\partial \bar{\omega}_t}{\partial t} + \mathcal{A}(\bar{\omega}_t, \bar{\psi}_t) = \frac{1}{\mathrm{Re}}\nabla^2\bar{\omega}_t - \bar{f} - r\bar{\omega}_t + \underbrace{\mathcal{A}(\bar{\omega}_t, \bar{\psi}_t) - \overline{\mathcal{A}(\omega_t, \psi_t)}}_{\Pi_t \approx \mathbf{M}_{\boldsymbol{\theta}}}$$

$$\nabla^2\psi_t = -\omega_t \qquad\qquad \nabla^2\bar{\psi}_t = -\bar{\omega}_t$$

- Model the subgrid-scale term $\Pi_t \approx \mathbf{M}_{\boldsymbol{\theta}}$

QG turbulence

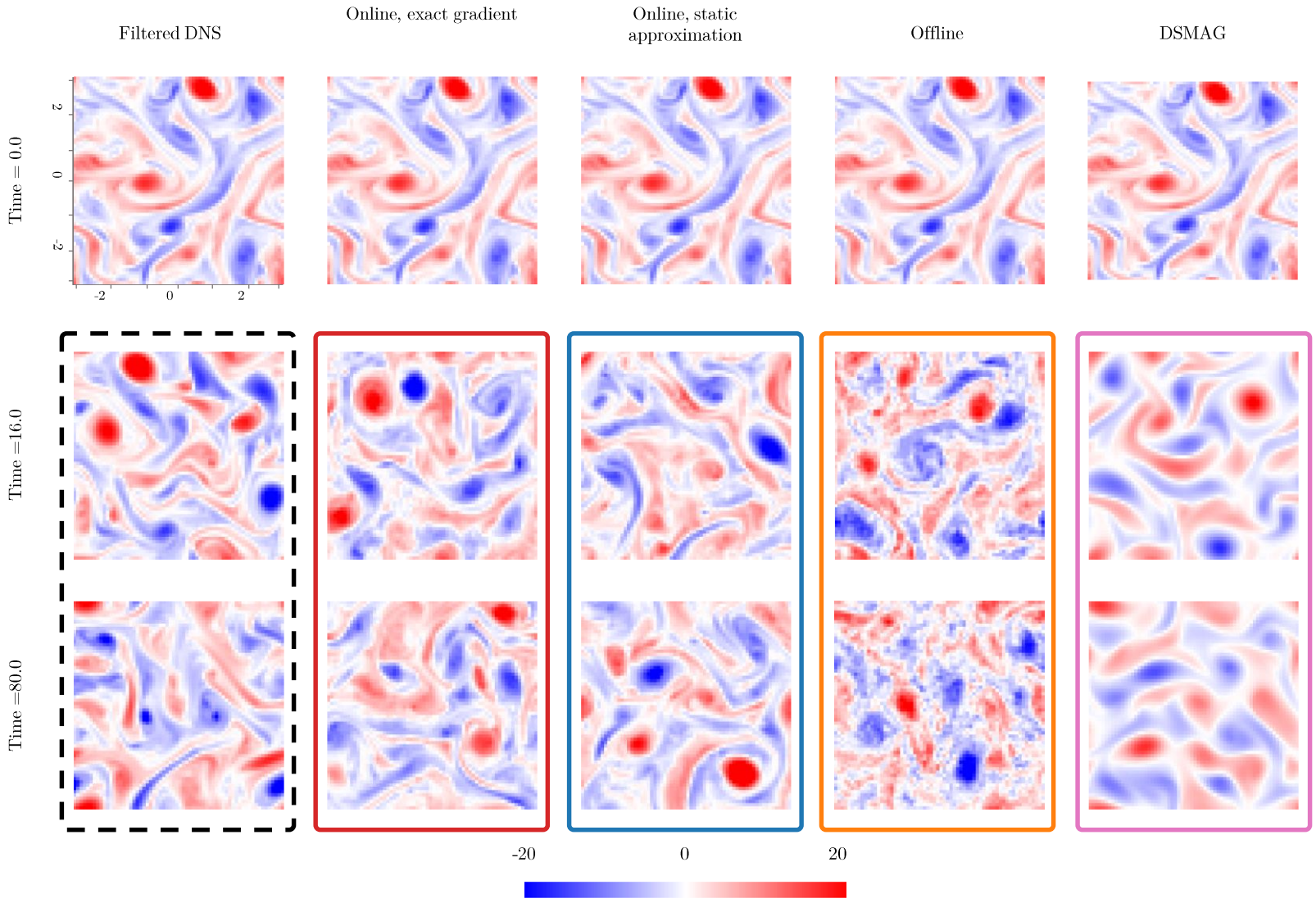
Flow configuration:

- High resolution grid : 1024×1024.
- Low resolution : 64×64.
- Re : 20000, r $= 0.1$, kf $= 4$;

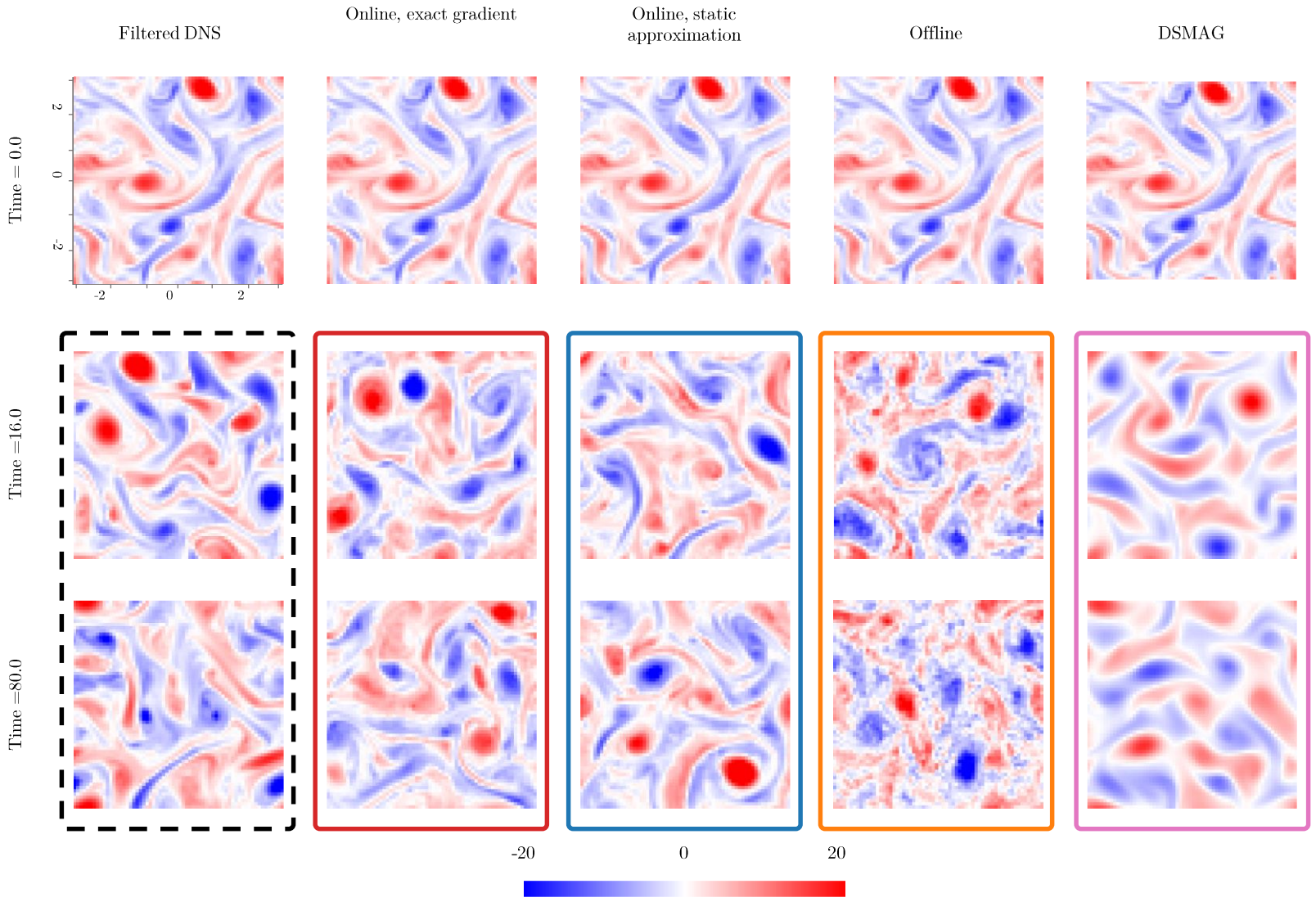
Tested models:

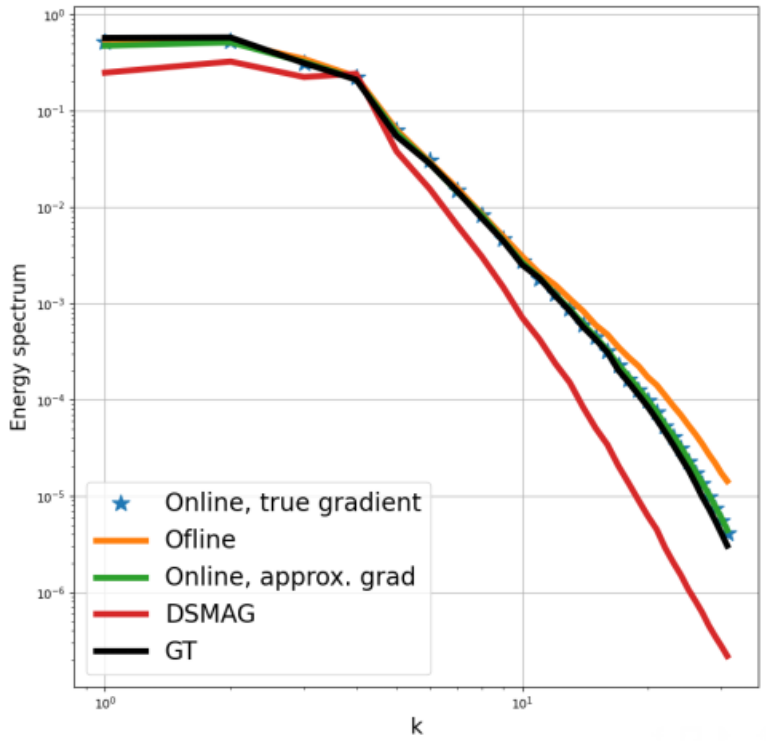
- Online learning with exact gradient;
- Online learning with approximate gradient;
- Offline learning;
- Dynamic Smagorinsky (DSMAG)

QG turbulence



QG turbulence





QG turbulence

- We proposed a simple gradient estimator for learning online hybrid models;
- Proposed methodology does not rely on a differentiable physical model, and can (in theory) be applied on non-differentiable CFD/GFD codes;
- Can use better Jacobian approximation and can be extended to definition of non additive correction terms;
- Currently working on applications on large scale models