Well-balancing through Scientific Machine Learning

Emmanuel Franck*, Victor Michel-Dansac*, Laurent Navoret*

November 27, 2023

Journée d'intégration Inria à Strasbourg

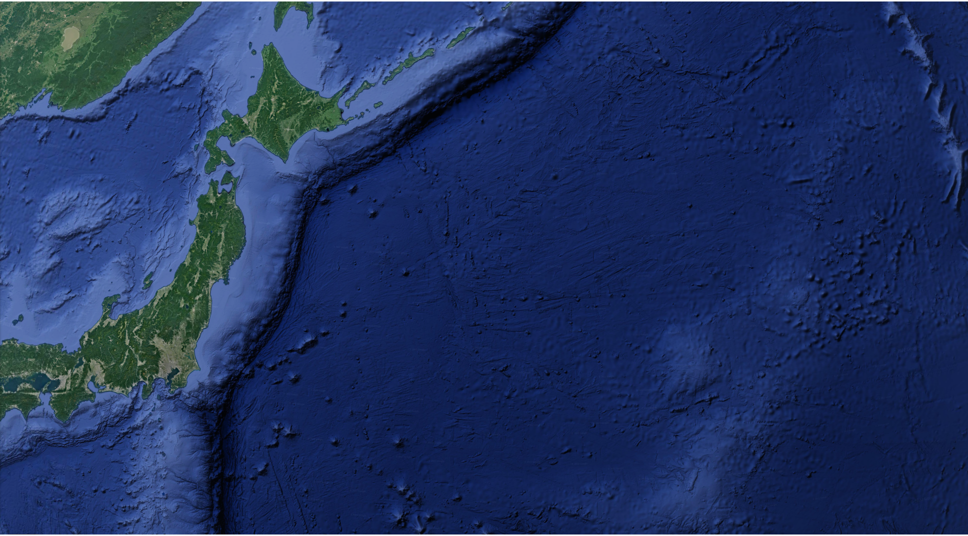
*TONUS (soon-to-be MACARON) project-team, Université de Strasbourg, CNRS, Inria, IRMA, France

Inria

IRMA
Institut de Recherche
Mathématique Avancée

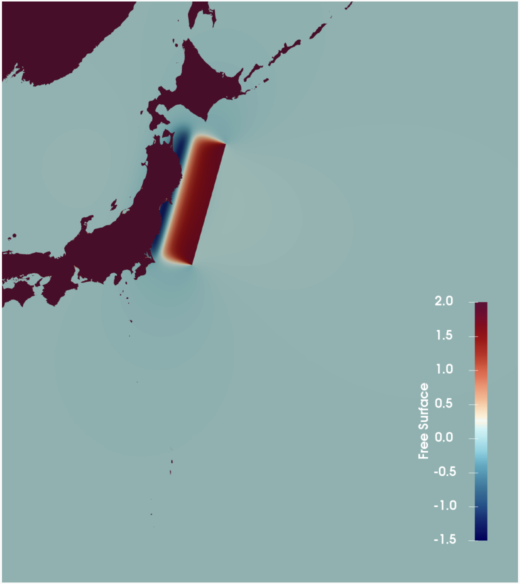
Numerical simulation of a tsunami

Context: 2011 Tōhoku tsunami



Ingredients required for a numerical simulation

Tsunami initialization

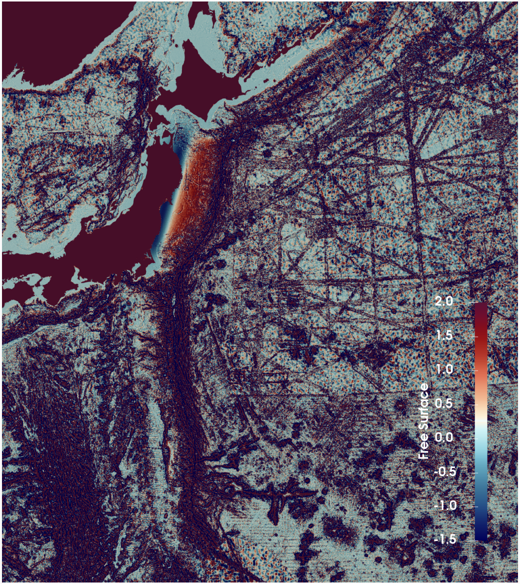


Numerical simulation of a tsunami

Starting the simulation with a naive numerical method

Numerical simulation of a tsunami

Starting the simulation with a naive numerical method



Numerical simulation of a tsunami: failure

... that did not work, the ocean at rest, far from the tsunami, starts spontaneously producing waves.

$\rightsquigarrow$ **The simulation is not usable!**

This comes from the non-preservation of stationary solutions:

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t)) = s(u(x,t))$$

Numerical simulation of a tsunami: failure

... that did not work, the ocean at rest, far from the tsunami, starts spontaneously producing waves.

$\rightsquigarrow$ **The simulation is not usable!**

This comes from the non-preservation of stationary solutions:

$$\frac{\partial}{\partial x} f(u(x,t)) = s(u(x,t)) \qquad \text{if } \frac{\partial}{\partial t} u(x,t) = 0 \;\text{ (stationary solution)}$$

Numerical simulation of a tsunami: failure

… that did not work, the ocean at rest, far from the tsunami, starts spontaneously producing waves.

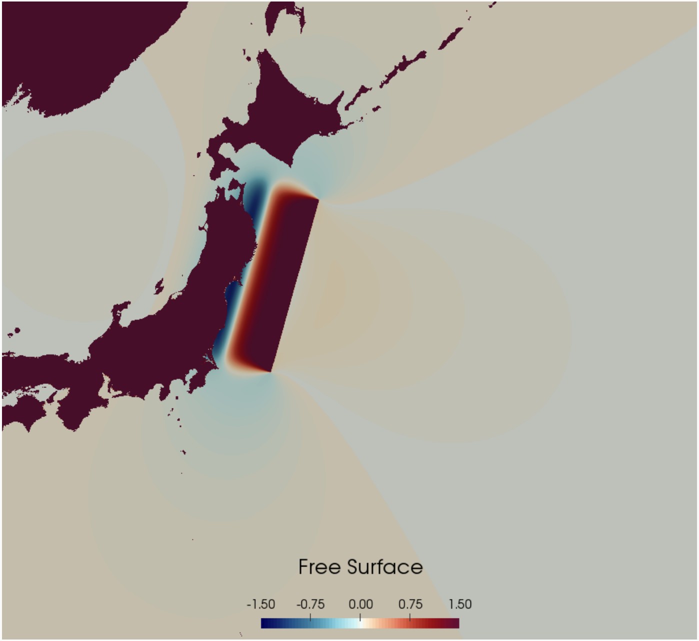
$\rightsquigarrow$ **The simulation is not usable!**

This comes from the non-preservation of stationary solutions:

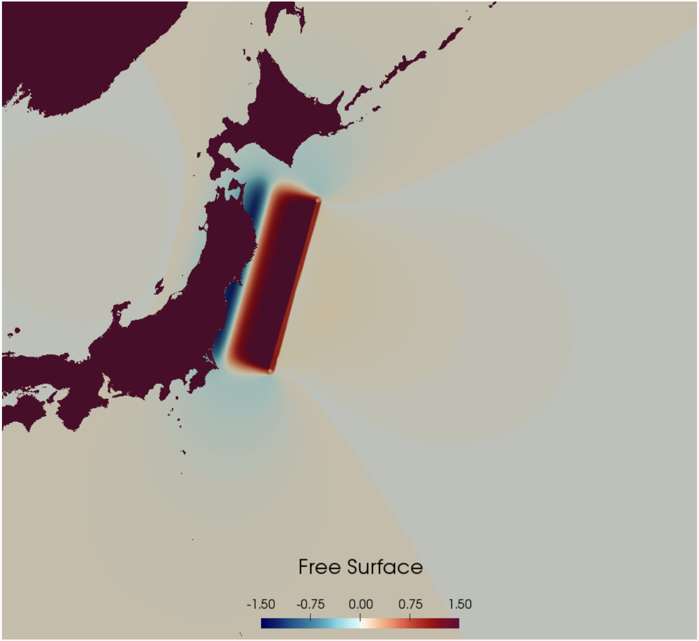
$$\frac{\partial}{\partial x} f(u(x,t)) = s(u(x,t)) \qquad \text{if } \frac{\partial}{\partial t} u(x,t) = 0 \text{ (stationary solution)}$$

Hence the need to develop numerical methods that **preserve stationary solutions**: so-called **well-balanced** methods.

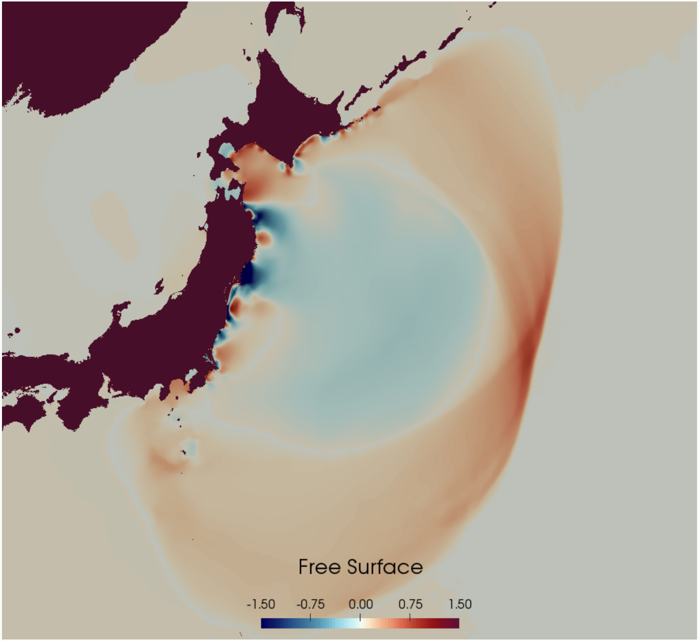
Numerical simulation of a tsunami: well-balanced method



Numerical simulation of a tsunami: well-balanced method



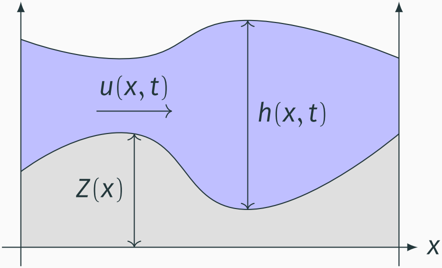
Numerical simulation of a tsunami: well-balanced method



The shallow water equations

The shallow water equations are governed by the following PDE:

$$\begin{cases} \partial_t h + \partial_x q = 0, \\ \partial_t q + \partial_x \left(\dfrac{q^2}{h} + \dfrac{1}{2} g h^2 \right) = -g h \partial_x Z(x). \end{cases}$$

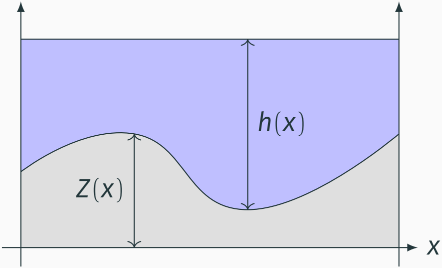


- $h(x, t)$: water height
- $u(x, t)$: water velocity
- $q = hu$: water discharge
- $Z(x)$: known topography
- g: gravity constant

The shallow water equations: steady solutions

The steady solutions of the shallow water equations are governed by the following ODEs:

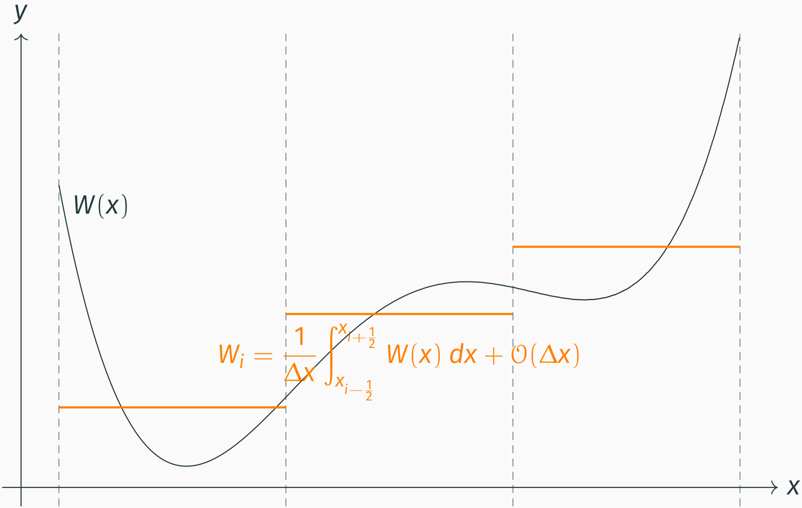
$$\begin{cases} \partial_x q = 0, \\ \partial_x \left(\dfrac{q^2}{h} + \dfrac{1}{2} g h^2 \right) = -g h \partial_x Z(x). \end{cases}$$



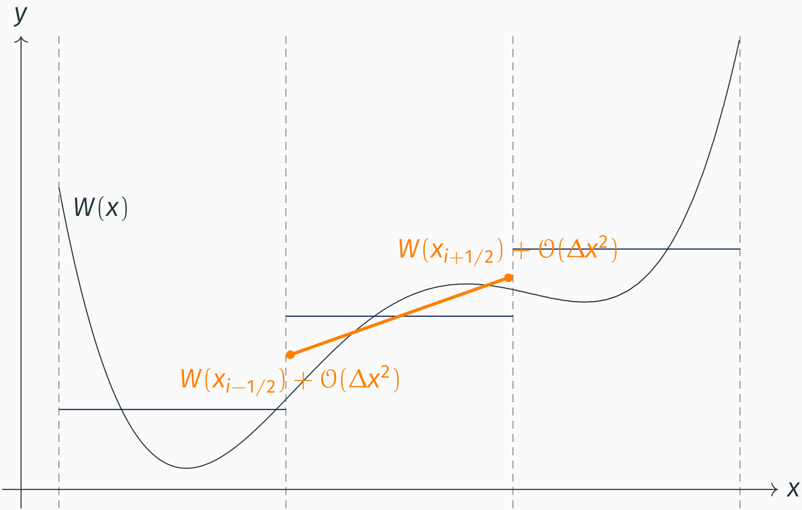
For the shallow water equations, if the velocity vanishes, we obtain the lake at rest steady solution:

$$h + Z = \text{cst}.$$

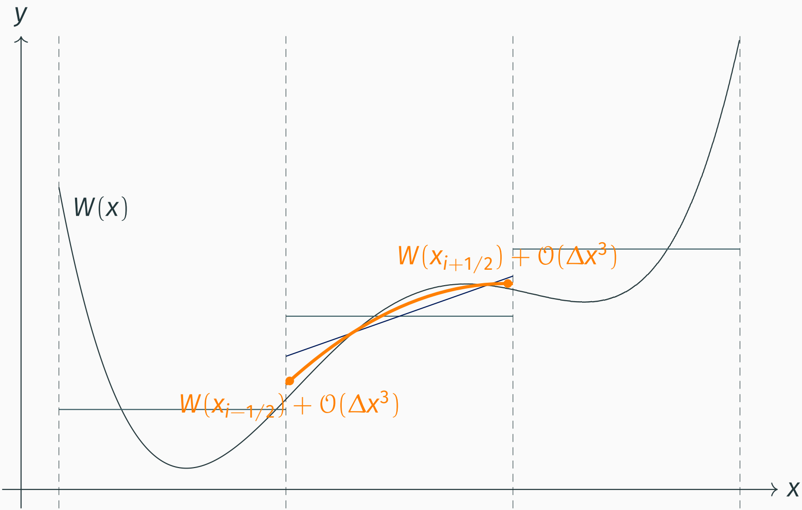
Finite volume method, visualized



Discontinuous Galerkin, visualized



Discontinuous Galerkin, visualized



Discontinuous Galerkin: an example

On the previous slide, the unknown function W is represented by

- a polynomial of degree 2 in each cell (Galerkin approximation),
- which is Discontinuous at interfaces between cells.

Discontinuous Galerkin: an example

On the previous slide, the unknown function W is represented by

- a polynomial of degree 2 in each cell (Galerkin approximation),
- which is Discontinuous at interfaces between cells.

Therefore, in each cell Ω_i, W is approximated by

$$W\big|_{\Omega_i} \simeq W_i^{\mathrm{DG}} := \alpha_0 + \alpha_1 x + \alpha_2 x^2 = \sum_{j=0}^{2} \alpha_j x^j,$$

where the polynomial coefficients α_0, α_1 and α_2 are determined to ensure fitness between the continuous data and its polynomial approximation.

Any polynomial of degree two can be exactly represented this way.

Discontinuous Galerkin: polynomial basis

More generally, we define a polynomial basis $\varphi_0, \dots, \varphi_N$ on each cell Ω_i and approximate the solution in this basis.

A usual example is the following so-called **modal basis**:

$$\forall j \in \{0, \dots, N\}, \quad \varphi_j(x) = x^j.$$

Discontinuous Galerkin: polynomial basis

More generally, we define a polynomial basis $\varphi_0, \ldots, \varphi_N$ on each cell Ω_i and approximate the solution in this basis.

A usual example is the following so-called **modal basis**:

$$\forall j \in \{0, \ldots, N\}, \quad \varphi_j(x) = x^j.$$

Main takeaway: The DG scheme is exact on every function that can be exactly represented in the basis!

Main idea

Recall that the DG scheme will be exact on every function that can be exactly represented in the DG basis.

Main idea

Recall that the DG scheme will be exact on every function that can be exactly represented in the DG basis.

Main idea

Enhance the DG basis by using the steady solution!

$\rightsquigarrow$ If **the basis is enhanced with an approximation of the steady solution**, then the enhanced DG scheme will provide a better approximation of the steady solution than the classical version.

Enhanced DG bases

Assume that you know a **prior** $\overline{W}$ on the steady solution.

The goal is now to **enhance the modal basis** V using $\overline{W}$:

$$V = \{1, x, x^2, \dots, x^N\}.$$

Enhanced DG bases

Assume that you know a **prior $\overline{W}$** on the steady solution.

The goal is now to **enhance the modal basis** V using $\overline{W}$:

$$V = \{1, x, x^2, \ldots, x^N\}.$$

A possibility is to replace the first element with $\overline{W}$

$$\bar{V} = \{\overline{W}, x, x^2, \ldots, x^N\}.$$

We can prove that the prior $\overline{W}$ needs to provide a **good approximation of the derivatives** of the steady solution (in addition to the steady solution itself).

$\rightsquigarrow$ A Physics-Informed Neural Network (**PINN**) is the ideal candidate!

Steady solutions as boundary value problems

As seen in the previous section, we seek an approximation of a steady solution using a PINN.

A **steady solution** is nothing but the **solution to a boundary value problem (BVP)**:

$$\begin{cases} \mathcal{D}(W, x) = 0 & \text{for } x \in \Omega, \\ W(x) = g(x) & \text{for } x \in \partial\Omega, \end{cases}$$

where $\mathcal{D}$ is a differential operator containing derivatives of W.

Remark: Neural networks are smooth functions of the inputs (provided smooth activation functions are used!).

Since their derivatives are easily computable by automatic differentiation, they are therefore **natural objects to approximate solutions to PDEs or ODEs**.

PINNs

Remark: Neural networks are smooth functions of the inputs (provided smooth activation functions are used!).

Since their derivatives are easily computable by automatic differentiation, they are therefore **natural objects to approximate solutions to PDEs or ODEs**.

Definition: PINN

A PINN is a neural network with input x and trainable weights θ, approximating the solution to a PDE or ODE, and denoted by $W_\theta(x)$.

PINNs: using the ODE residual

Recall that the PINN W_θ approximates the solution to the BVP

$$\begin{cases} \mathcal{D}(W, x) = 0 & \text{for } x \in \Omega, \\ W(x) = g(x) & \text{for } x \in \partial\Omega. \end{cases}$$

Based on this observation, we know that the PINN W_θ should approximately satisfy the above BVP:

$$\begin{cases} \mathcal{D}(W_\theta, x) \simeq 0 & \text{for } x \in \Omega, \\ W_\theta(x) \simeq g(x) & \text{for } x \in \partial\Omega. \end{cases}$$

PINNs: using the ODE residual

Recall that the PINN W_θ approximates the solution to the BVP

$$\begin{cases} \mathcal{D}(W, x) = 0 & \text{for } x \in \Omega, \\ W(x) = g(x) & \text{for } x \in \partial\Omega. \end{cases}$$

Based on this observation, we know that the PINN W_θ should approximately satisfy the above BVP:

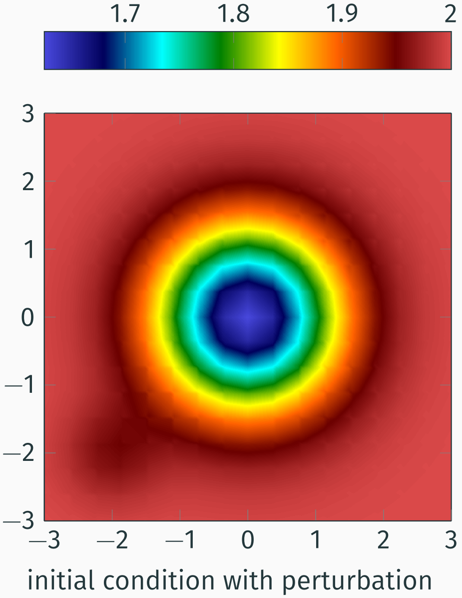
$$\begin{cases} \mathcal{D}(W_\theta, x) \simeq 0 & \text{for } x \in \Omega, \\ W_\theta(x) \simeq g(x) & \text{for } x \in \partial\Omega. \end{cases}$$

The idea behind PINNs training is to find the optimal weights θ_{opt} by **minimizing a loss function built from the ODE residual**:

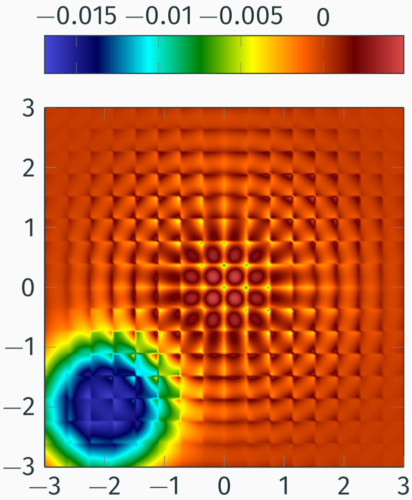
$$\theta_{\text{opt}} = \underset{\theta}{\operatorname{argmin}} \left(\int_\Omega \|\mathcal{D}(W_\theta, x)\|_2^2 \, dx + \int_{\partial\Omega} \|W_\theta(x) - g(x)\|_2^2 \, dx. \right)$$

The Monte-Carlo method is used for the integrals, which makes the whole approach **mesh-less** and able to deal with **parametric BVPs**.

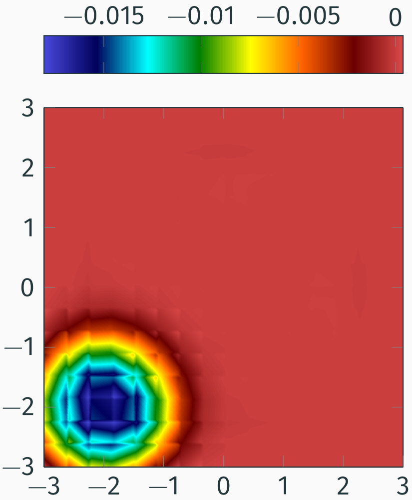
Perturbation of a shallow water steady solution



Perturbation of a shallow water steady solution

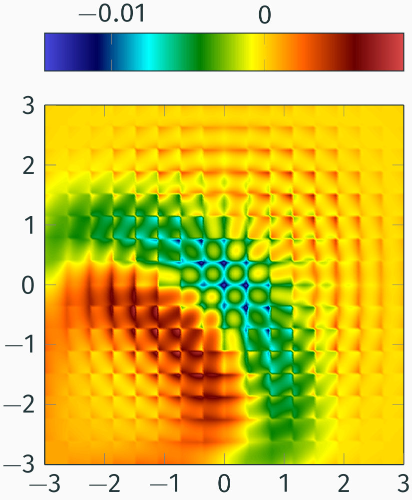


(a) $T = 0.1$, classical basis

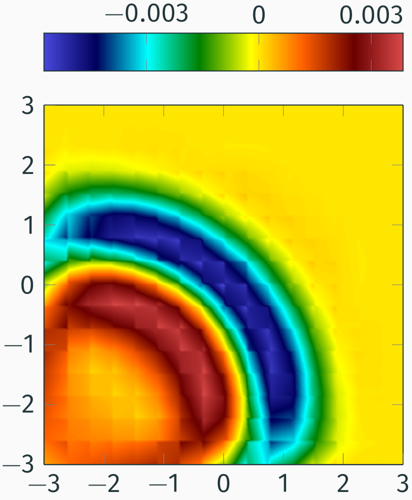


(b) $T = 0.1$, enhanced basis

Perturbation of a shallow water steady solution

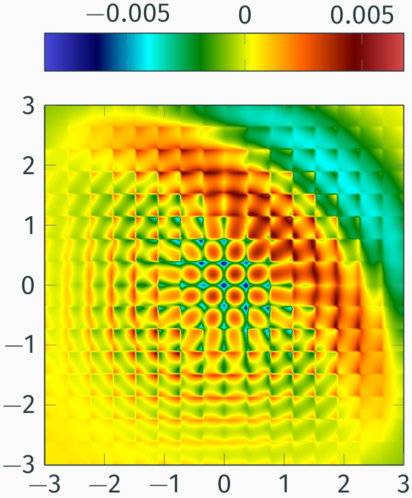


(c) $T = 0.6$, classical basis

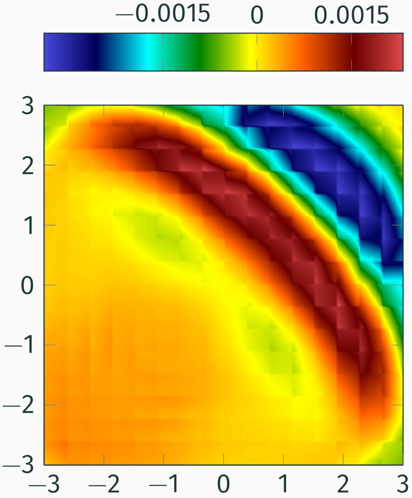


(d) $T = 0.6$, enhanced basis

Perturbation of a shallow water steady solution



(e) $T = 1.2$, classical basis

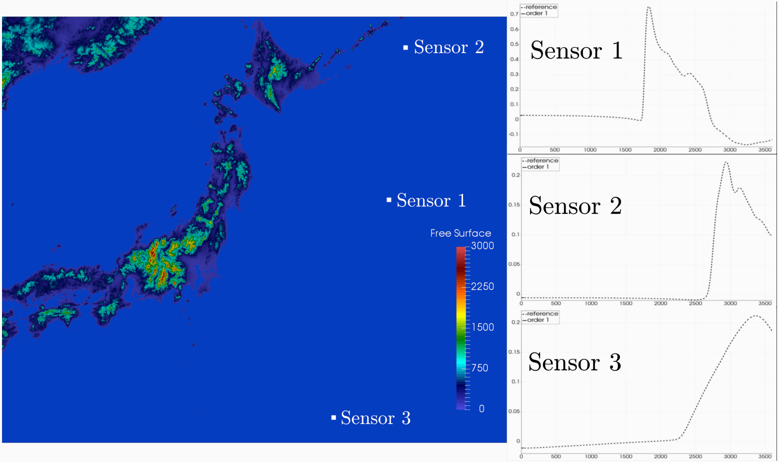


(f) $T = 1.2$, enhanced basis

Thank you for your attention!

Ingredients required for a numerical simulation

Fourth step: Verification of the numerical results

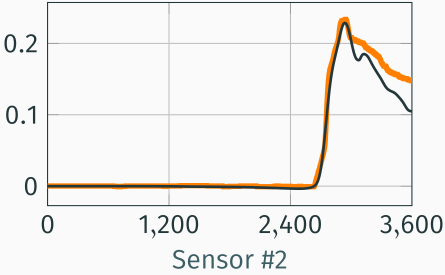


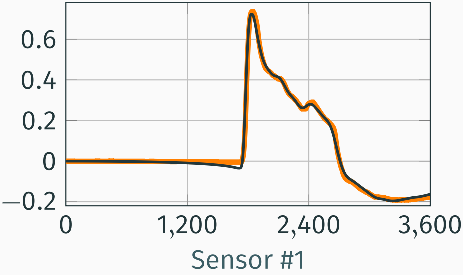
Simulation of the 2011 Japan tsunami

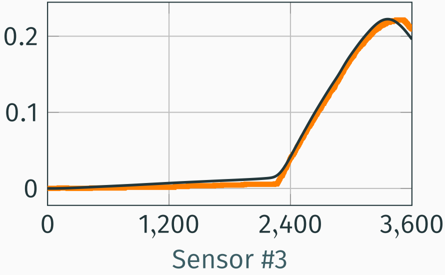
Water depth at sensors:

- #1: 5700 m;
- #2: 6100 m;
- #3: 4400 m.

Plots of the time variation
of the water height (in meters).
data in black, simulation in orange







PINNs: advantages and drawbacks

Once trained, PINNs with Monte-Carlo integration are able to

- quickly provide an approximation to the steady solution,
- in a mesh-less fashion,
- independently of the dimension.

PINNs: advantages and drawbacks

Once trained, PINNs with Monte-Carlo integration are able to

- quickly provide an approximation to the steady solution,
- in a mesh-less fashion,
- independently of the dimension.

However, PINNs

- have trouble generalizing to $x \notin \Omega$;
- are **not competitive with classical numerical methods for computational fluid dynamics**: to reach a given error (if possible), training takes longer than using a classical numerical method.

PINNs: advantages and drawbacks

Once trained, PINNs with Monte-Carlo integration are able to

- quickly provide an approximation to the steady solution,
- in a mesh-less fashion,
- independently of the dimension.

However, PINNs

- have trouble generalizing to $x \notin \Omega$;
- are **not competitive with classical numerical methods for computational fluid dynamics**: to reach a given error (if possible), training takes longer than using a classical numerical method.

The most interesting use of PINNs, in our case, is to deal with **parametric ODEs and PDEs**, where dimension-insensitivity is paramount.

Parametric PINNs: approximation using the ODE residual

The parametric PINN $W_\theta(x;\boldsymbol{\mu})$, with parameters $\boldsymbol{\mu} \in \mathbb{P} \subset \mathbb{R}^m$ approximates the solution to the parametric BVP

$$\begin{cases} \mathcal{D}(W, x; \boldsymbol{\mu}) = 0 & \text{for } x \in \Omega, \boldsymbol{\mu} \in \mathbb{P}, \\ W(x) = g(x; \boldsymbol{\mu}) & \text{for } x \in \partial\Omega, \boldsymbol{\mu} \in \mathbb{P}. \end{cases}$$

Based on this observation, we know that the PINN W_θ should approximately satisfy the above BVP:

$$\begin{cases} \mathcal{D}(W_\theta, x; \boldsymbol{\mu}) \simeq 0 & \text{for } x \in \Omega, \boldsymbol{\mu} \in \mathbb{P}, \\ W_\theta(x; \boldsymbol{\mu}) \simeq g(x; \boldsymbol{\mu}) & \text{for } x \in \partial\Omega, \boldsymbol{\mu} \in \mathbb{P}. \end{cases}$$

Parametric PINNs: approximation using the ODE residual

The parametric PINN $W_\theta(x;\mu)$, with parameters $\mu \in \mathbb{P} \subset \mathbb{R}^m$ approximates the solution to the parametric BVP

$$\begin{cases} \mathcal{D}(W, x; \mu) = 0 & \text{for } x \in \Omega, \mu \in \mathbb{P}, \\ W(x) = g(x; \mu) & \text{for } x \in \partial\Omega, \mu \in \mathbb{P}. \end{cases}$$

Based on this observation, we know that the PINN W_θ should approximately satisfy the above BVP:

$$\begin{cases} \mathcal{D}(W_\theta, x; \mu) \simeq 0 & \text{for } x \in \Omega, \mu \in \mathbb{P}, \\ W_\theta(x; \mu) \simeq g(x; \mu) & \text{for } x \in \partial\Omega, \mu \in \mathbb{P}. \end{cases}$$

The loss function then becomes

$$\mathcal{J}_{\text{ODE}}(\theta) = \underbrace{\int_{\mathbb{P}} \int_{\Omega} \|\mathcal{D}(W_\theta, x; \mu)\|_2^2 \, dx d\mu}_{\mathcal{J}_\Omega(\theta)} + \underbrace{\int_{\mathbb{P}} \int_{\partial\Omega} \|W_\theta(x; \mu) - g(x; \mu)\|_2^2 \, dx d\mu}_{\mathcal{J}_{\text{boundary}}(\theta)}.$$