Approximately well-balanced Discontinuous Galerkin methods using bases enriched with Physics-Informed Neural Networks

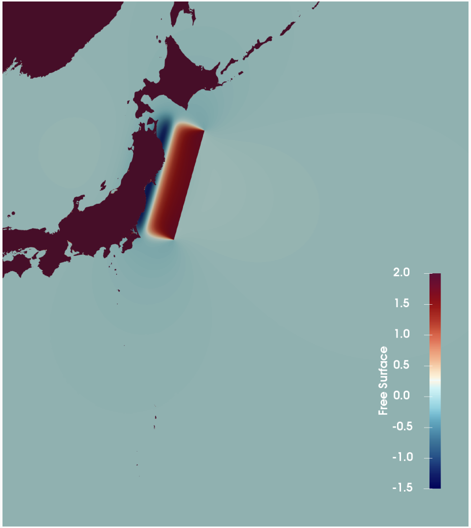
Emmanuel Franck*, Victor Michel-Dansac*, Laurent Navoret*

April 25, 2024
COMinDS Workshop, Delft

*MACARON project-team, Université de Strasbourg, CNRS, Inria, IRMA, France

Tsunami simulation: naive numerical method

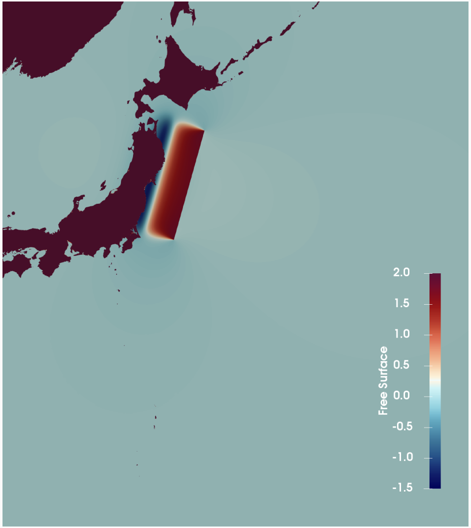
Tsunami initialization



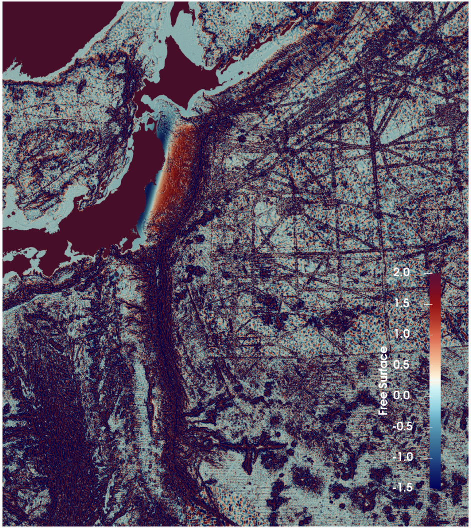
Simulation with a naive numerical method

Tsunami simulation: naive numerical method

Tsunami initialization



Simulation with a naive numerical method



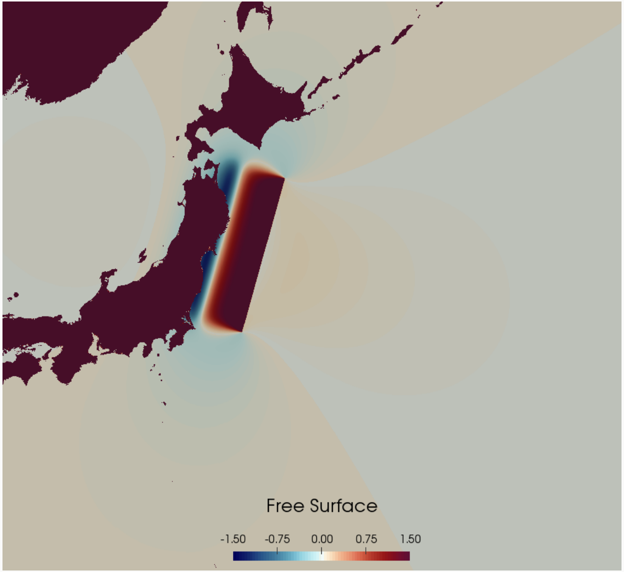
Tsunami simulation: failure

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

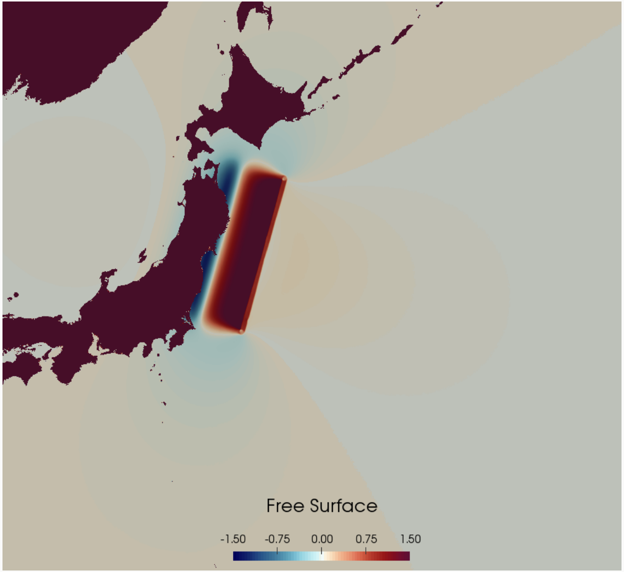
Indeed, the ocean at rest, far from the tsunami, started spontaneously producing waves.

This comes from the non-preservation of stationary solutions, hence the need to develop numerical methods that **preserve stationary solutions**: so-called **well-balanced** methods.

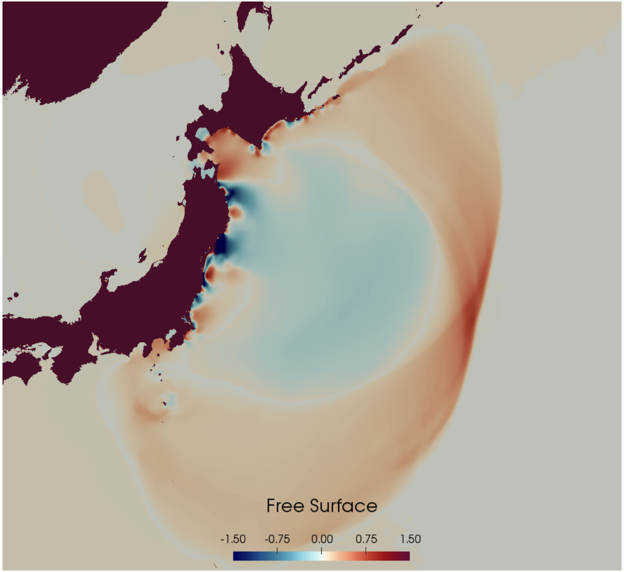
Tsunami simulation: well-balanced method



Tsunami simulation: well-balanced method



Tsunami simulation: well-balanced method



Objectives

The goal of this work is to provide a numerical method which:

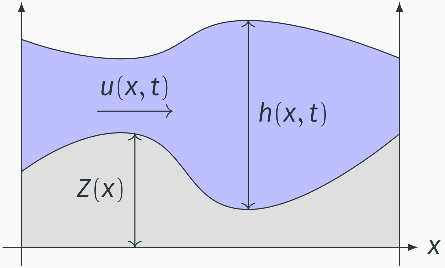
- is able to deal with generic systems of balance laws,
- can provide a very good approximation of families of steady solutions,
- is as accurate as classical methods on unsteady solutions,
- with provable convergence estimates.

To that end, we select the **Discontinuous Galerkin (DG)** framework.

The shallow water equations

The shallow water equations are governed by the following PDE:

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

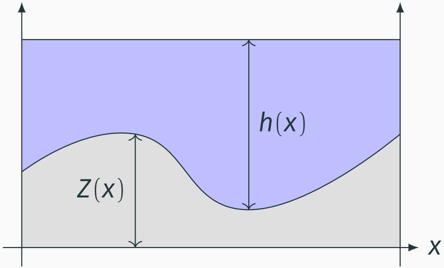


- $h(x,t)$: water depth
- $u(x,t)$: water velocity
- $q = hu$: water discharge
- $Z(x)$: known topography
- g: gravity constant

The shallow water equations: steady solutions

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

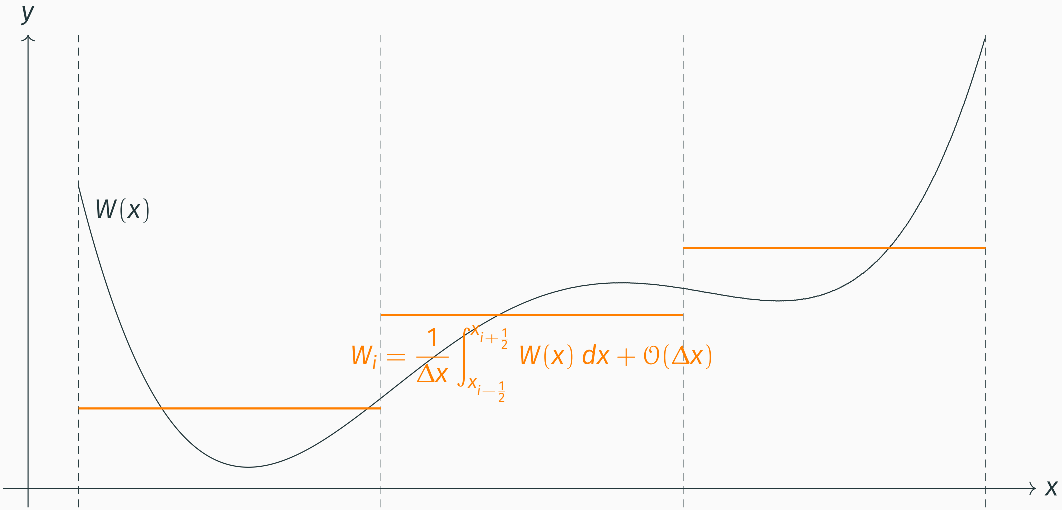
$$\begin{cases} \partial_x q = 0, \\ \partial_x \left(\dfrac{q^2}{h} + \dfrac{1}{2} g h^2 \right) = -g h \partial_x Z(x). \end{cases}$$



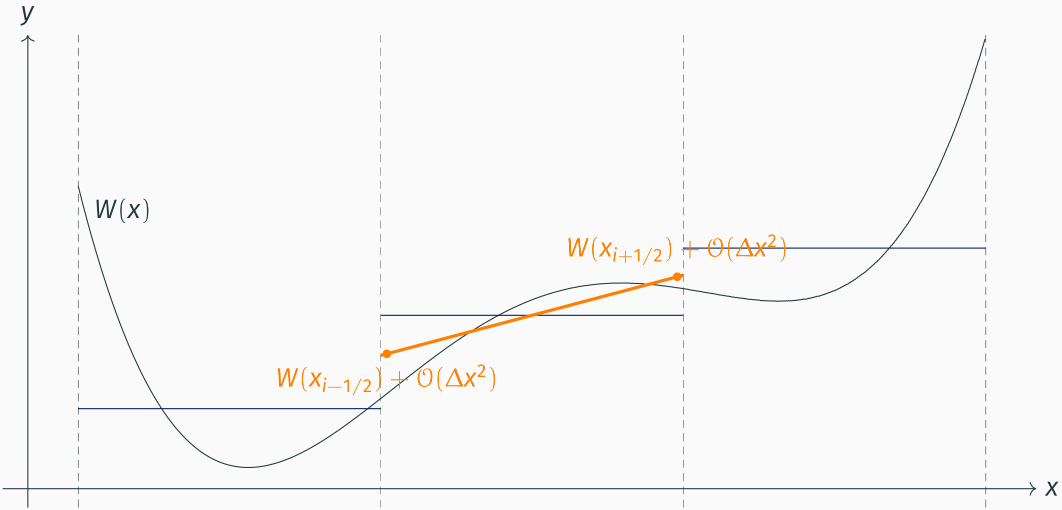
For the shallow water equations, if the velocity vanishes, we obtain the lake at rest steady solution:

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

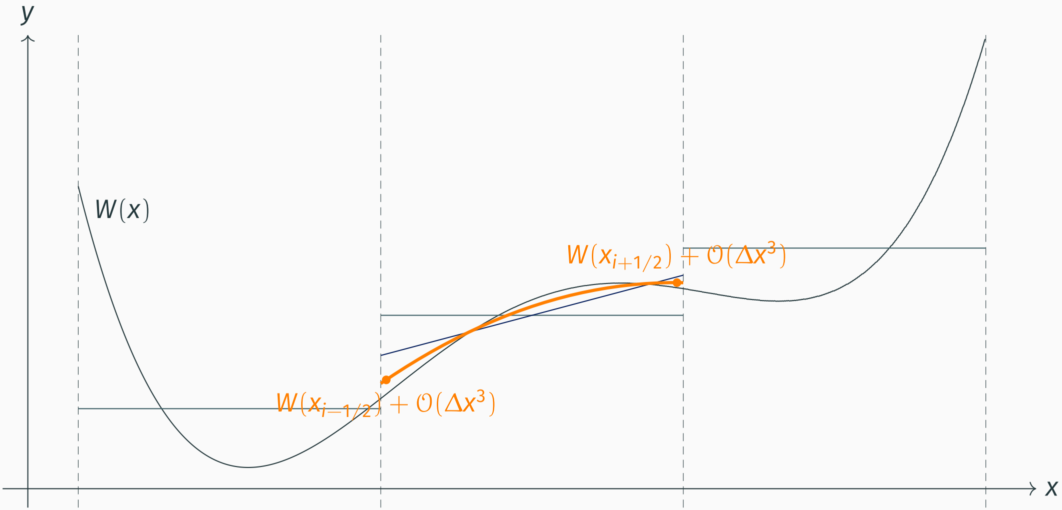
Finite volume method, visualized



Discontinuous Galerkin, visualized



Discontinuous Galerkin, visualized



Discontinuous Galerkin: an example

On the previous slide, the data 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 data 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 $\varphi_0, \dots, \varphi_N$ on each cell Ω_i and approximate the solution in this basis.

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

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

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, as soon as it is also a solution to the PDE.

Main idea

Recall that the DG scheme will be exact on every function that can be exactly represented in the DG basis, as soon as it is also a solution to the PDE.

Main idea

Enhance the DG basis by using the steady solution!

$\rightsquigarrow$ If the **steady solution or an approximation thereof is contained in the basis**, then:

- using the exact steady solution in the basis will make the scheme exactly well-balanced;
- using an approximation of the steady solution will make the scheme approximately well-balanced.

Enhanced DG bases

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

It can be the exact steady solution ($\overline{W} = W_{\text{eq}}$), or it can be an approximation ($\overline{W} \simeq W_{\text{eq}}$).

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

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

Enhanced DG bases

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

It can be the exact steady solution ($\overline{W} = W_{\text{eq}}$), or it can be an approximation ($\overline{W} \simeq W_{\text{eq}}$).

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

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

First possibility: multiply the whole basis by $\overline{W}$

$$\bar{V}_* = \{\overline{W}, x\,\overline{W}, x^2\,\overline{W}, \dots, x^N\,\overline{W}\}.$$

Enhanced DG bases

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

It can be the exact steady solution ($\overline{W} = W_{\text{eq}}$), or it can be an approximation ($\overline{W} \simeq W_{\text{eq}}$).

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

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

First possibility: multiply the whole basis by $\overline{W}$

$$\bar{V}_* = \{\overline{W}, x\,\overline{W}, x^2\,\overline{W}, \dots, x^N\,\overline{W}\}.$$

Second possibility: replace the first element with $\overline{W}$

$$\bar{V}_+ = \{\overline{W}, x, x^2, \dots, x^N\}.$$

Error estimates

We denote by:

- W_{ex} the exact solution,
- W_{DG} the approximate solution without prior,
- $\overline{W_{\text{DG}}}$ the approximate solution with prior $\overline{W}$ and basis $\overline{V}_*$.

For a DG scheme of order $q+1$, we obtain[1] the following error estimates:

$$\|W_{\text{ex}} - W_{\text{DG}}\| \lesssim \left|W_{\text{ex}}\right|_{H^{q+1}} \Delta x^{q+1},$$

$$\|W_{\text{ex}} - \overline{W_{\text{DG}}}\| \lesssim \left|\frac{W_{\text{ex}}}{\overline{W}}\right|_{H^{q+1}} \Delta x^{q+1} \|\overline{W}\|_{L^\infty}.$$

Conclusion of the error estimates: the prior $\overline{W}$ needs to provide a **good approximation of the derivatives** of the steady solution.

[1] Rigorous error estimates are written in terms of the error in the projection onto both bases.

Obtaining a prior

For very simple systems, one can use the exact steady solution as a prior.

However, in many cases, even for some simple and well-known systems, one cannot compute the exact steady solution. Therefore, **an approximation is required**.

How to obtain such an approximation?

Obtaining a prior

For very simple systems, one can use the exact steady solution as a prior.

However, in many cases, even for some simple and well-known systems, one cannot compute the exact steady solution. Therefore, **an approximation is required**.

How to obtain such an approximation?

1. **First possibility**: use a traditional numerical approximation, obtained by classical ODE solvers (e.g. Runge-Kutta schemes).
2. **Second possibility**: use a Physics-Informed Neural Network (PINN), a specifically-trained neural network.

Next step: Present the PINNs, which will be preferred since they are mesh-less and able to approximate solutions to parametric PDEs.

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**.

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 $W_\theta(x)$.

Hence, the PINN W_θ will approximate the solution to the PDE

$$\mathcal{D}(W, x) = 0,$$

with $\mathcal{D}$ a differential operator.

Ommitting boundary conditions, the problem becomes

$$\text{find } W \text{ such that } \mathcal{D}(W, x) = 0 \text{ for all } x \in \Omega \subset \mathbb{R}^d.$$

Based on this observation, the PINN W_θ should approximately satisfy the above PDE, and the problem becomes:

$$\text{find } \theta_{\text{opt}} \text{ such that } \mathcal{D}(W_{\theta_{\text{opt}}}, x) \simeq 0 \text{ for all } x \in \Omega \subset \mathbb{R}^d.$$

PINNs: loss function

Ommitting boundary conditions, the problem becomes

$$\text{find } W \text{ such that } \mathcal{D}(W, x) = 0 \text{ for all } x \in \Omega \subset \mathbb{R}^d.$$

Based on this observation, the PINN W_θ should approximately satisfy the above PDE, and the problem becomes:

$$\text{find } \theta_{\text{opt}} \text{ such that } \mathcal{D}(W_{\theta_{\text{opt}}}, x) \simeq 0 \text{ for all } x \in \Omega \subset \mathbb{R}^d.$$

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

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

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

Parametric PINNs

A parametric PDE is nothing but the following problem:

$$\text{find } W \text{ such that } \mathcal{D}(W, x; \boldsymbol{\mu}) = 0 \text{ for all } x \in \Omega \text{ and } \boldsymbol{\mu} \in \mathbb{P} \subset \mathbb{R}^m.$$

The parametric PINN $W_\theta(x; \boldsymbol{\mu})$ should approximately satisfy the above PDE, and the problem becomes:

$$\text{find } \theta_{\text{opt}} \text{ such that } \mathcal{D}(W_{\theta_{\text{opt}}}, x; \boldsymbol{\mu}) \simeq 0 \text{ for all } x \in \Omega \text{ and } \boldsymbol{\mu} \in \mathbb{P} \subset \mathbb{R}^m.$$

Parametric PINNs

A parametric PDE is nothing but the following problem:

$$\text{find } W \text{ such that } \mathcal{D}(W, x; \mu) = 0 \text{ for all } x \in \Omega \text{ and } \mu \in \mathbb{P} \subset \mathbb{R}^m.$$

The parametric PINN $W_\theta(x; \mu)$ should approximately satisfy the above PDE, and the problem becomes:

$$\text{find } \theta_{\text{opt}} \text{ such that } \mathcal{D}(W_{\theta_{\text{opt}}}, x; \mu) \simeq 0 \text{ for all } x \in \Omega \text{ and } \mu \in \mathbb{P} \subset \mathbb{R}^m.$$

The minimization problem then becomes

$$\theta_{\text{opt}} = \underset{\theta}{\operatorname{argmin}} \int_{\mathbb{P}} \int_{\Omega} \|\mathcal{D}(W_\theta, x; \mu)\|_2^2 \, dx d\mu.$$

Setup: the advection equation

We run experiments on the **advection equation with source term**, with a given initial condition $W_0 : \mathbb{R} \to \mathbb{R}$:

$$\begin{cases} \partial_t W + c\partial_x W = aW + bW^2 & \text{for } x \in (0,1),\ t \in (0,T), \\ W(0,x) = W_0(x) & \text{for } x \in (0,1), \\ W(t,0) = u_0 & \text{for } t \in (0,T). \end{cases}$$

The **steady solution** W_{eq} satisfies the BVP

$$\begin{cases} c\partial_x W_{\text{eq}} - aW_{\text{eq}} - bW_{\text{eq}}^2 = 0 & \text{for } x \in (0,1), \\ W_{\text{eq}}(0) = u_0, \end{cases}$$

whose unique solution is, with parameters $\boldsymbol{\mu} = \{a, b, c, u_0\} \in \mathbb{P} \subset \mathbb{R}^4$:

$$W_{\text{eq}}(x; \boldsymbol{\mu}) = \frac{au_0}{(a + bu_0)e^{-\frac{ax}{c}} - bu_0}.$$

PINNs as a DG prior: steady solution

We use the DG scheme to solve the advection equation with the **steady solution as initial condition**. We expect the DG scheme with prior:

- to provide a **better approximation of the steady solution** than the classical DG scheme (approximate well-balanced property),
- while converging with the **same order of accuracy**.

We report below some statistics on the gains with 1000 random sets of parameters in $\mathbb{P}$, for a DG scheme of order $q+1$.

q	minimum gain	average gain	maximum gain
0	63.46	735.08	4571.89
1	32.22	149.38	450.74
2	6.20	54.16	118.45
3	1.55	19.54	108.10

PINNs as a DG prior: computation time

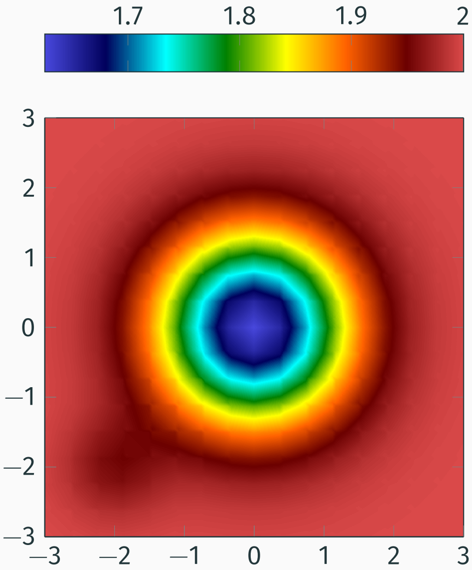
Finally, we compare the computation time in bases V and $\bar{V}_+$. We expect the prior to:

- **increase** the computation time of the **DG mass matrices**,
- **have no effect on the computation time of the main loop**.

The table below shows the **CPU time increase factor** when using the prior, for several values of the number n of space cells. We observe that the **increase in computation time due to the prior is negligible**.

q	factor, $n = 10$	factor, $n = 40$	factor, $n = 160$
0	1.26	1.07	1.01
1	1.15	1.01	1.00
2	1.04	1.03	1.01
3	1.07	1.00	1.01

Perturbation of a shallow water steady solution



PINN trained on a parametric steady solution, driven by the topography

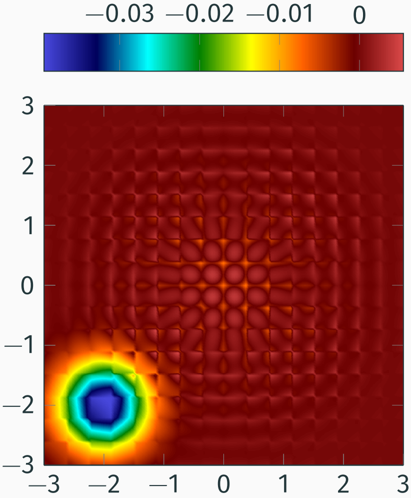
$$Z(x;\mu) = \Gamma \exp\big(\alpha(r_0^2 - \|x\|^2)\big),$$

with physical parameters

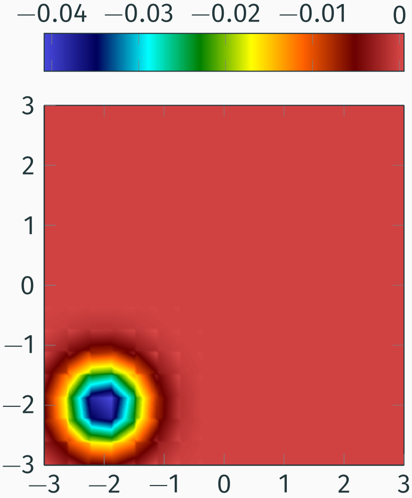
$$\mu \in \mathbb{P} \iff \begin{cases} \alpha \in [0.25, 0.75], \\ \Gamma \in [0.1, 0.4], \\ r_0 \in [0.5, 1.25]. \end{cases}$$

Left plot: initial condition, made of a perturbed steady solution.

Perturbation of a shallow water steady solution

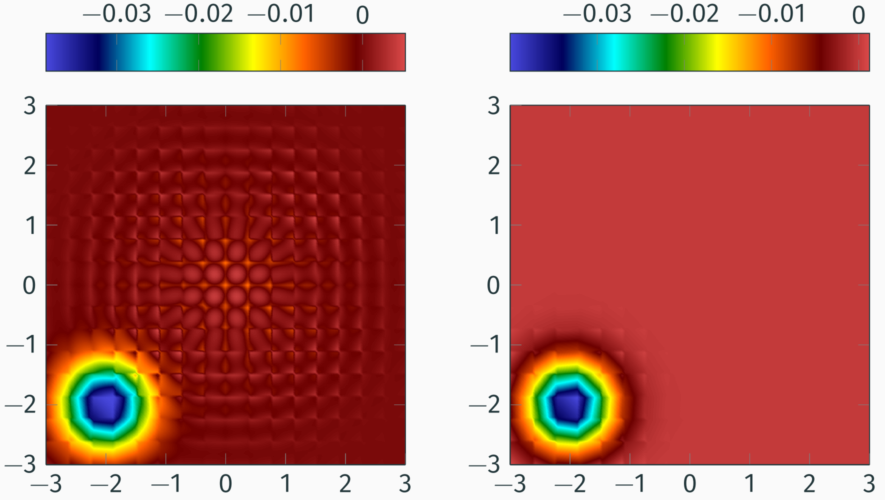


(a) classical basis



(b) enhanced basis

Perturbation of a shallow water steady solution



(a) classical basis (b) enhanced basis

Conclusion and perspectives

We have obtained:

- an exactly or approximately well-balanced DG scheme,
- displaying large gains on parameterized families of steady solutions,
- available for arbitrary balance laws.

Perspectives include:

- using a space-time DG method and time-dependent priors,
- replacing PINNs with neural operators for added flexibility,
- coding the method in the SciMBA framework.

Related preprint: E. Franck, V. Michel-Dansac and L. Navoret.
"Approximately WB DG methods using bases enriched with PINNs."
git repository: https://github.com/Victor-MichelDansac/DG-PINNs

Thank you for your attention!

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.

Advection equation: loss function

Thanks to the boundary ansatz and the ODE loss, the final loss function **does not need any data**, and there is **no competition between loss functions**: we get

$$\mathcal{J}(\theta) = \int_{\mathbb{P}} \int_{\Omega} \left\| c \partial_x \widetilde{W_\theta} - a \widetilde{W_\theta} - b \widetilde{W_\theta}^2 \right\|_2^2 dx d\mu,$$

with the ansatz

$$\widetilde{W_\theta} = u_0 + x\, W_\theta,$$

with W_θ the result of the neural network.

In practice, we take $c = 1$ and make sure the steady solution is well-defined, by taking

$$\mathbb{P} = \left\{ (a, b, u_0) \in (0.5, 1) \times (0.5, 1) \times (0.1, 0.2) \right\}.$$

Hence, the neural network is a function $W_\theta \in \mathcal{C}^\infty(\mathbb{R} \times \mathbb{R}^3, \mathbb{R})$.

PINNs as a DG prior: unsteady solution

We use the DG scheme to solve an unsteady advection problem, without a source term. We expect the DG scheme with prior:

- to provide a **similar approximation of the solution** than the classical DG scheme,
- while converging with the **same order of accuracy**.

The table below shows the gains made by using the prior, for several values of the number n of space cells.

q	gain, $n = 10$	gain, $n = 40$	gain, $n = 160$
0	0.80	0.81	0.81
1	1.00	1.00	1.00
2	1.00	1.00	1.00
3	1.00	1.00	1.00