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

Emmanuel Franck, Victor Michel-Dansac, Laurent Navoret

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

Emmanuel Franck[1], V. Michel-Dansac[1], and L. Navoret[1,2]

[1]Université de Strasbourg, CNRS, Inria, IRMA, F-67000 Strasbourg, France
[2]IRMA, Université de Strasbourg, CNRS UMR 7501, 7 rue René Descartes, 67084 Strasbourg, France

October 23, 2023

Abstract

This work concerns the enrichment of Discontinuous Galerkin (DG) bases, so that the resulting scheme provides a much better approximation of steady solutions to hyperbolic systems of balance laws. The basis enrichment leverages a prior – an approximation of the steady solution – which we propose to compute using a Physics-Informed Neural Network (PINN). To that end, after presenting the classical DG scheme, we show how to enrich its basis with a prior. Convergence results and error estimates follow, in which we prove that the basis with prior does not change the order of convergence, and that the error constant is improved. To construct the prior, we elect to use parametric PINNs, which we introduce, as well as the algorithms to construct a prior from PINNs. We finally perform several validation experiments on four different hyperbolic balance laws to highlight the properties of the scheme. Namely, we show that the DG scheme with prior is much more accurate on steady solutions than the DG scheme without prior, while retaining the same approximation quality on unsteady solutions.

1 Objectives and model

In the last decades, much work has been devoted to proposing numerical methods for hyperbolic systems with source terms, which correctly capture stationary solutions of the system, as well as perturbations of flows around these steady states. If the perturbation is smaller than the scheme error, traditional numerical schemes are not able to provide a good approximation of the perturbed steady solution. To address such an issue, a first possibility is to refine the mesh in space. However, for small perturbations, this would greatly increase the computational overhead. To avoid this, schemes specifically dedicated to capturing stationary solutions have been introduced. They are called *well-balanced schemes*.

There are two families of well-balanced (WB) schemes: exactly and approximately WB schemes. Exactly WB schemes give an exact representation of the equilibria. Such schemes are usually developed for subclasses of steady solutions, especially for complex balance laws, or multidimensional problems. For instance, first- and second-order accurate exactly WB schemes have been developed for the shallow water equations [4, 31, 34, 35] or the Euler equations with gravity [28, 49]. High-order exactly well-balanced schemes include [22, 37, 21, 36, 8, 9, 24] with finite volume methods or related approaches, or [53, 11] with discontinuous Galerkin methods. The second family, approximately WB schemes, consist in ensuring a better approximation of the equilibria compared to traditional numerical schemes. This better approximation can be under the form of a better order of accuracy [18, 20, 23] or a better error constant [1, 16]. Both families of WB schemes may incur significant additional computational cost compared to traditional schemes, due to the extensive modifications necessary to ensure the WB property, especially for complex systems and equilibria.

In this work, we focus on providing an approximately WB scheme for the following parametric partial differential equation (PDE):

$$\begin{cases} \partial_t u + \partial_x F_{\mu_1}(u) = S_{\mu_2}(u), \\ u(t=0,x) = u_0(x), \end{cases} \tag{1.1}$$

with μ_1 and μ_2 the parameters of the PDE. We set $\mu = \{\mu_1, \mu_2\}$, and we assume that $\mu \in \mathbb{P} \subset \mathbb{R}^m$. In (1.1), the unknown function is u; F_{μ_1} is celled the physical flux function, while S_{μ_2} is the source term. We assume that the equation is hyperbolic, that is to say that the Jacobian matrix of F_{μ_1} is diagonalizable with real eigenvalues. Our goal will be to construct an approximately well-balanced approach for the general steady state $\partial_x F_{\mu_1}(u) = S_{\mu_2}(u)$. The combination of learning and numerical methods (known as Scientific Machine Learning) has produced good results for hyperbolic PDEs. Examples include work on the design of limiters or shock detection [41, 7, 55], artificial viscosity [19, 42, 54, 10], or numerical schemes [5].

The approach proposed in this paper is also based on the hybridization of classical approaches and neural networks. We endeavor to improve the classical Discontinuous Galerkin (DG) method, which usually relies on a discontinuous approximation of the solution in a suitable polynomial basis. More information on the DG method can be found in [26, 39] for instance. A natural way of improving the traditional DG method to improve the accuracy on some family of solutions is to enrich the basis with a prior. This is for example the case of the Trefftz method [30, 6, 12, 27], or the non-polynomial bases studied in [56].

To perform this basis enrichment, we use a learning-based offline computation with a neural network to build a prior which approximates a parametrized family of equilibria. To that end, we use Physics-Informed Neural Networks (PINNs), see e.g. [40, 13], and parametric neural networks [48]. This prior is then introduced into the Discontinuous Galerkin basis, to increase the accuracy of the scheme around this family of equilibria. Note that the prior construction could be handled without the use of neural networks, but we will show that the neural network approach is more efficient. This framework could require significant offline calculation cost (depending on the problem), but will generate a very small additional cost in the online phase, i.e., when actually using the modified scheme. This method enhances the DG basis functions with a prior provided by a neural network. Similar techniques based on neural networks have already been successfully implemented for other applications. In [3], for elliptic problems, the authors use a network to provide a finite element basis that is dynamically enriched. In [46, 47], the authors show that random neural networks can be used as DG bases, and can be more accurate than classical ones for a sufficiently large number of basis function in each cell.

The paper is constructed as follows. First, we assume that we know a prior (an approximation) of a family of equilibria, and we introduce the modification of the DG basis. Theoretical results show that this modification does not change the order of accuracy of the method, but decreases the error constant close to steady solutions. Then, we introduce the learning methods that will enable us to build our prior for a family of equilibria, and finally we perform numerical experiments, in one and two space dimensions, on several linear and nonlinear systems of balance laws. A conclusion ends this paper.

2 Modified Discontinuous Galerkin scheme

This section is devoted to the presentation of the modified DG scheme. We start by quickly introducing the classical DG scheme in Section 2.1, and then move on to proposing the modification in Section 2.2. Theoretical convergence results related to this modification will be presented in Section 3. In this section and the following one, we write the scheme in the case of a scalar and one-dimensional PDE, but the method is easily extendable to systems and to higher dimensions.

2.1 Classical Discontinuous Galerkin scheme

The goal of this section is to present the classical DG scheme in order to discretize the PDE (1.1). To that end, we discretize the space domain $\Omega \subset \mathbb{R}^d$ in cells $\Omega_k = (x_{k-1/2}, x_{k+1/2})$ of size Δx_k, and of centers x_k.

The idea behind the classical DG scheme is to first compute the weak form of the considered PDE, and then to locally approximate the solution in each cell, by projecting it onto a finite-dimensional vector space V_h. We consider a space V_h of dimension $q+1$:

$$V_h = \text{Span}\left(\phi_{k,0}, \dots, \phi_{k,q}\right).$$

Note that the space V_h can be different for each cell k.

The first assumption of DG scheme is to approximate the solution u to the PDE, in each cell, with a value in V_h:

$$\forall k, \quad u\big|_{\Omega_k}(t,x) \simeq u_k(t,x).$$

Since $u_k \in V_h$, we can write

$$u_k(t,x) \coloneqq \sum_{j=0}^{q} u_{k,j}(t)\phi_{k,j}(x). \tag{2.1}$$

To obtain the DG scheme, we first write the weak form of the equation in each cell:

$$\int_{\Omega_k} \partial_t u(t,x)\phi(x)\,dx + \int_{\Omega_k} \partial_x F_{\mu_1}(u(t,x))\phi(x)\,dx = \int_{\Omega_k} S_{\mu_2}(u(t,x))\phi(x)\,dx. \tag{2.2}$$

with $\phi(x)$ a smooth test function. Performing an integration by parts, this form is equivalent to

$$\partial_t \left(\int_{\Omega_k} u\,\phi \right) - \int_{\Omega_k} F_{\mu_1}(u)\,\partial_x\phi + \left[F_{\mu_1}(u)\,\phi\right]_{x_{k-1/2}}^{x_{k+1/2}} = \int_{\Omega_k} S_{\mu_2}(u)\,\phi. \tag{2.3}$$

We now plug the DG representation (2.1) in the weak form (2.3), using $\phi_{k,i}$ as test function, for any $i \in \{0,\dots,q\}$.

1. We begin with the first term:

$$\int_{\Omega_k} u\,\phi_{k,i} = \sum_{j=0}^{q} \left(\int_{\Omega_k} u_{k,j}(t)\phi_{k,j}(x)\phi_{k,i}(x)\,dx \right) = \sum_{j=0}^{q} u_{k,j}(t) \left(\int_{\Omega_k} \phi_{k,j}(x)\phi_{k,i}(x)\,dx \right).$$

   To handle the integral in the expression above, we introduce the following quadrature formula, with weights $w_{k,p}$ and points $x_{k,p}$, valid for any smooth function ϕ:

$$\int_{\Omega_k} \phi(x)\,dx \simeq \sum_{p=1}^{N_q} w_{k,p}\,\phi(x_{k,p}).$$

   We assume that the first and last quadrature points coincide with the cell boundaries, i.e. $x_{k,1} = x_{k-1/2}$ and $x_{k,N_q} = x_{k+1/2}$. In practice, we use the well-known Gauss-Lobatto quadrature rule, see e.g. [2] for more information. Equipped with this quadrature formula, we introduce

$$M_{k,i,j} = \sum_{p=1}^{N_q} w_{k,p}\,\phi_{k,j}(x_{k,p})\,\phi_{k,i}(x_{k,p}) \simeq \int_{\Omega_k} \phi_{k,j}\phi_{k,i},$$

   so that the first term of (2.3) becomes

$$\int_{\Omega_k} u(t,x)\phi_{k,i}(x)\,dx \simeq \sum_{j=0}^{q} M_{k,i,j}u_{k,j}(t).$$

2. Using the same techniques, the second term is approximated in the following way:

$$\int_{\Omega_k} F_{\mu_1}(u)\,\partial_x\phi_{k,i} \simeq \sum_{p=1}^{N_q} \left[w_{k,p}F_{\mu_1}\left(\sum_{j=0}^{q} u_{k,j}(t)\phi_{k,j}(x_{k,p}) \right) \partial_x\phi_{k,i}(x_{k,p}) \right].$$

3. We note that the third term reduces to

$$\left[F_{\mu_1}(u)\,\phi_{k,i}\right]_{x_{k-1/2}}^{x_{k+1/2}} = F_{\mu_1}\left(u_k(t,x_{k+\frac{1}{2}})\right)\phi_{k,i}(x_{k+\frac{1}{2}}) - F_{\mu_1}\left(u_k(t,x_{k-\frac{1}{2}})\right)\phi_{k,i}(x_{k-\frac{1}{2}}),$$

   where the physical flux F_{μ_1} has to be approximated at the cell boundaries. To that end, like the well-known finite volumes method, the DG method requires the introduction of a consistent numerical flux

$$G_{\mu_1}(u_L,u_R) \quad \text{such that } G_{\mu_1}(u,u) = F_{\mu_1}(u).$$

   This numerical flux is then used to approximate the interface flux, as follows

$$F_{\mu_1}\left(u_k(t,x_{k+\frac{1}{2}})\right) \simeq G_{\mu_1}\left(u_k(t,x_{k+\frac{1}{2}}), u_{k+1}(t,x_{k+\frac{1}{2}})\right).$$

4. Finally, for the last term, we use a straightforward application of the quadrature rule:

$$\int_{\Omega_k} S_{\mu_2}(u)\,\phi \simeq \sum_{p=1}^{N_q} \left[w_{k,p} S_{\mu_2} \left(\sum_{j=0}^{q} u_{k,j}(t)\phi_{k,j}(x_{k,p}) \right) \phi_{k,i}(x_{k,p}) \right].$$

Gathering all these terms, we show that, in each cell, the DG scheme can be written as an ordinary differential equation, where the interface flux term couples the cell Ω_k with its neighbors:

$$\mathcal{M}_k\,\partial_t u_k(t) - \mathcal{F}_{\mu_1}(u_k) + \mathcal{G}_{\mu_1}(u_{k-1}, u_k, u_{k+1}) = \mathcal{S}_{\mu_2}(u_k).$$

Now that we have recalled the classical DG space discretization, we have all the tools we need to introduce a modification to this discretization that will enable us to provide an approximately WB scheme.

2.2 Enrichment of the modal DG basis

There are many vector spaces able to represent the solution in each cell. For instance, nodal DG schemes [26] use Lagrange polynomials or other polynomials based on nodes chosen within each cell. Legendre polynomials or Taylor expansions around the cell centers lead to *modal DG schemes*. In this work, we focus on the Taylor basis, given on each cell Ω_k by

$$V_h = \mathrm{Span}\left(\phi_{k,0}, \phi_{k,1}, \phi_{k,2}, \dots, \phi_{k,q}\right) = \mathrm{Span}\left(1, (x-x_k), \frac{1}{2}(x-x_k)^2, \dots, \frac{1}{q!}(x-x_k)^q\right). \tag{2.4}$$

In the remainder of this section, we assume that we have access to a prior on the equilibrium, denoted by $u_\theta(x,\mu)$. Obtaining such a prior is discussed in Section 4. For the moment, suffice it to say that u_θ provides an approximation of the steady solution for $x \in \Omega$ and for μ in some parameter space $\mathbb{P}$ to be defined.

Given the prior u_θ, we modify the local basis V_h to incorporate the prior: for that, we propose two possibilities.

- The *additive correction* V_h^+ consists in replacing the first element of V_h by the prior:

$$V_h^+ = \mathrm{Span}\left(\phi^+_{k,0}, \phi^+_{k,1}, \phi^+_{k,2}, \dots, \phi^+_{k,q}\right) = \mathrm{Span}\left(u_\theta(x,\mu), (x-x_k), \dots, \frac{1}{q!}(x-x_k)^q\right). \tag{2.5}$$

- The *multiplicative correction* V_h^* consists in multiplying each element of V_h by the prior:

$$V_h^* = \mathrm{Span}\left(\phi^*_{k,0}, \phi^*_{k,1}, \phi^*_{k,2}, \dots, \phi^*_{k,q}\right) = \mathrm{Span}\left(u_\theta(x,\mu), (x-x_k)\,u_\theta(x,\mu), \dots, \frac{1}{q!}(x-x_k)^q u_\theta(x,\mu)\right). \tag{2.6}$$

A first remark is that, if the prior is exactly equal to the steady solution, then it can be exactly represented by an element of V_h^+ or V_h^* (namely, the first one) in each cell, which is not the case for the classical space V_h. However, whether the prior is exact or not, the method will only be of interest if the projector onto the modified vector space is accurate (or even exact in the case of an exact prior). The second point to note is that, unlike conventional DG approaches, the bases are not polynomial. We must therefore ensure that this does not hinder the convergence of the DG method. In the next section, we follow Yuan and Shu's work [56] to study the convergence of the modified DG method, and provide error estimates.

3 Error estimates

In this section, we prove some convergence results on the modified DG scheme. We assume that our prior u_θ is p times continuously differentiable, i.e., that it has differentiability class $\mathcal{C}^p$, with $p \geqslant q+1$. This hypothesis is compatible with the construction of the prior from Section 4.

In [56], the authors study the convergence of the DG scheme for non-polynomial bases. They show that, if the non-polynomial basis can be represented in a specific way by a polynomial basis, then the convergence of the

local and global projection operators is not hampered. Using some stability results (given in [56] for the transport equation) together with these estimations, convergence can be recovered.

These theoretical results will be split in two parts. To begin with, in Section 3.1, by prove that the bases proposed in Section 2.2 fit into the hypotheses of [56], which ensures convergence. However, this study is insufficient to show that the better the prior, the more accurate the modified DG scheme. To that end, in Section 3.2, we derive the projector estimates in the case of V_h^*, in order to show the potential gains of the method.

3.1 Convergence in non-polynomial DG bases

In [56], the authors prove the following lemma.

Lemma 3.1. *Consider an approximation vector space V_h with local basis $(v_{k,0}, \dots, v_{k,q})$, which may depend on the cell Ω_k. If there exists constant real numbers $a_{j\ell}$ and b_j independent of the size of the cell Δx_k such that, in each cell Ω_k,*

$$\forall j \in \{0, \dots, q\}, \quad \left| v_{k,j}(x) - \sum_{\ell=0}^{q} a_{j\ell}(x - x_k)^\ell \right| \leq b_j (\Delta x_k)^{q+1}, \tag{3.1}$$

then for any function $u \in H^{q+1}(\Omega_k)$, there exists $v_h \in V_h$ and a constant real number C independent of Δx_k, such that

$$\|v_h - u\|_{L^\infty(\Omega_k)} \leq C \|u\|_{H^{q+1}(\Omega_k)} (\Delta x_k)^{q+\frac{1}{2}}.$$

Using this result, the authors show that the global projection error in the DG basis converges with an error in $(\Delta x_k)^{q+1}$ in the Sobolev norm H^{q+1}, and later prove the convergence of the whole scheme using a monotone flux for a scalar equation. In the remainder of this section, we prove that the two new bases proposed in Section 2.2 satisfy the assumptions of Lemma 3.1. Using these results together with the proofs of [56], we will obtain that both bases lead to a convergent scheme.

Proposition 3.2. *If the prior $u_\theta(x;\mu)$ has differentiability class $\mathcal{C}^{q+1}(\mathbb{R})$ with respect to x, then the approximation space V_h^+ satisfies the assumption (3.1).*

Proof. Since the prior is $\mathcal{C}^{q+1}(\mathbb{R})$, we can write its Taylor series expansion around the cell center x_k. Namely, there exists a constant $c \in [x_{k-1/2}, x_{k+1/2}]$ such that

$$u_\theta(x) = u_\theta(x_k) + (x - x_k) u'_\theta(x_k) + \cdots + \frac{1}{q!}(x - x_k)^q u^{(q)}(x_k) + \frac{(x - x_k)^{q+1}}{(q+1)!} u^{(q+1)}(c). \tag{3.2}$$

With that expansion, we can write our basis V_h^+ with respect to the classical modal basis V_h as follows:

$$\begin{pmatrix} u_\theta(x) \\ (x - x_k) \\ \vdots \\ (x - x_k)^q \end{pmatrix} = \underbrace{\begin{pmatrix} u_\theta(x_k) & u'_\theta(x_k) & \dots & \frac{1}{q!} u_\theta^{(q)}(x_k) \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}}_{A_+} \begin{pmatrix} 1 \\ (x - x_k) \\ \vdots \\ (x - x_k)^q \end{pmatrix} + (x - x_k)^{q+1} \underbrace{\begin{pmatrix} \frac{u^{(q+1)}(c)}{(q+1)!} \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{b_+}.$$

We remark that the matrix A_+ and the vector b_+ are independent of Δx_k. Hence, assumption (3.1) is verified, and Lemma 3.1 can be applied. □

Proposition 3.3. *If the prior $u_\theta(x;\mu)$ has differentiability class $\mathcal{C}^{q+1}(\mathbb{R})$ with respect to x, then the approximation space V_h^* satisfies the assumption (3.1).*

Proof. The proof follows the same lines as the proof of the previous proposition. Namely, (3.2) is still satisfied since the prior is $\mathcal{C}^{q+1}(\mathbb{R})$. Then, the basis V_h^* is written with respect to the classical modal basis V_h as follows:

$$\begin{pmatrix} u_\theta(x) \\ (x-x_k)\,u_\theta(x) \\ \vdots \\ (x-x_k)^q\,u_\theta(x) \end{pmatrix} = \underbrace{\begin{pmatrix} u_\theta(x_k) & u_\theta'(x_k) & \dots & \frac{u^{(q)}(x_k)}{q!} \\ 0 & u_\theta(x_k) & \dots & \frac{u_\theta^{(q-1)}(x_k)}{(q-1)!} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_\theta(x_k) \end{pmatrix}}_{A_*} \begin{pmatrix} 1 \\ (x-x_k) \\ \vdots \\ (x-x_k)^q \end{pmatrix} + (x-x_k)^{q+1} \underbrace{\begin{pmatrix} \frac{u_\theta^{q+1}(c)}{(q+1)!} \\ \frac{u_\theta^q(c)}{q!} \\ \vdots \\ 1 \end{pmatrix}}_{b_*}$$

Just like before, the matrix A_* and the vector b_* are independent of Δx_k. Hence, assumption (3.1) is verified, and Lemma 3.1 can be applied. □

These two propositions show that, if the prior is sufficiently smooth, we can apply the results of [56], which shows the convergence of the method. However, this approach does not give an estimation of the error with respect to the quality of the prior. Indeed, we expect the modified DG scheme to be more accurate when the prior is closer to the solution. Obtaining such an estimate is the objective of the following section.

3.2 Estimate with prior dependency

The goal of this section is to refine the error estimates from Section 3.1 for a specific modified basis. We consider the case of V_h^*, since it is easier to write the projector onto the classical basis. This will enable us to quantify the gains that can be expected when using this new basis. The case of V_h^+ is more complicated, since the projector is harder to write. Nevertheless, we will show in the numerical experiments from Section 5 that both modified bases exhibit similar behavior.

Recall that the basis V_h^* is obtained by multiplying each element of V_h by the prior. Therefore, its basis functions are given by $\phi_{k,j}^* = \phi_{k,j} u_\theta$ for each cell Ω_k and for $j \in \{0, \dots, q\}$.

Lemma 3.4. *Assume that the prior u_θ satisfies*

$$u_\theta(x;\mu)^2 > m^2 > 0, \quad \forall x \in \Omega, \quad \forall \mu \in \mathbb{P}.$$

For a given cell Ω_k, for any function $u \in H^{q+1}(\Omega_k)$, the L^2 projector onto V_h^*, denoted by P_h and such that $P_h(u) \in V_h^*$, satisfies the inequality

$$\left\|u - P_h(u)\right\|_{L^\infty(\Omega_k)} \lesssim \left|\frac{u(\cdot)}{u_\theta(\cdot\,;\mu)}\right|_{H^{q+1}(\Omega_k)} (\Delta x_k)^{q+\frac{1}{2}} \left(1 + \frac{\left\|u_\theta(\cdot\,;\mu)^2\right\|_{L^\infty(\Omega_k)}}{m^2}\right) \left\|u_\theta(\cdot\,;\mu)\right\|_{L^\infty}.$$

Proof. The proof uses a strategy similar to [56]. We consider the cell Ω_k. For any smooth function f defined on Ω_k, we define the operator T by

$$T(f) = \left(\sum_{j=0}^{q} f^{(j)}(x_k) \frac{1}{j!}(x-x_k)^j\right)$$

and the operator T_θ by

$$T_\theta(f) = \left(\sum_{j=0}^{q} \left(\frac{f}{u_\theta}\right)^{(j)}(x_k;\mu) \frac{1}{j!}(x-x_k)^j\right) u_\theta(x;\mu).$$

For simplicity, we no longer explicitly write the dependence in μ in this proof. Let $u \in H^{q+1}(\Omega_k)$. Using T_θ, we write the following estimation:

$$\|u - P_h(u)\|_{L^\infty(\Omega_k)} \le \|u - T_\theta(u)\|_{L^\infty(\Omega_k)} + \|T_\theta(u) - P_h(u)\|_{L^\infty(\Omega_k)} =: N_1 + N_2. \tag{3.3}$$

To complete the proof, we need to estimate both terms N_1 and N_2.

We start with the estimation of N_1. We obtain, according to the relationship between T and T_θ,

$$N_1 = \|u - T_\theta(u)\|_{L^\infty(\Omega_k)} = \left\| \frac{u}{u_\theta} u_\theta - T\left(\frac{u}{u_\theta}\right) u_\theta \right\|_{L^\infty(\Omega_k)} \leq \left\| \frac{u}{u_\theta} - T\left(\frac{u}{u_\theta}\right) \right\|_{L^\infty(\Omega_k)} \|u_\theta\|_{L^\infty(\Omega_k)}. \tag{3.4}$$

We can now use an intermediate result from [56]: for all f smooth enough, the Taylor formula and the Cauchy-Schwartz inequality, followed by a direct computation, gives

$$\begin{aligned} \|f - T(f)\|_{L^\infty(\Omega_k)} &= \sup_{x\in\Omega_k} \left| \int_{x_k}^{x} f^{(q+1)}(\xi) \frac{(x-\xi)^q}{q!} d\xi \right| \\ &\leq \sup_{x\in\Omega_k} \left[\left(\int_{x_k}^{x} \left| f^{(q+1)}(\xi) \right|^2 d\xi \right)^{\frac{1}{2}} \left(\int_{x_k}^{x} \left| \frac{(x-\xi)^q}{q!} \right|^2 d\xi \right)^{\frac{1}{2}} \right], \\ &\lesssim |f|_{H^{q+1}(\Omega_k)} (\Delta x_k)^{q+\frac{1}{2}}. \end{aligned} \tag{3.5}$$

Going back to N_1 and plugging (3.5) into the estimate (3.4), we obtain

$$N_1 \lesssim \left| \frac{u}{u_\theta} \right|_{H^{q+1}(\Omega_k)} (\Delta x_k)^{q+\frac{1}{2}} \|u_\theta\|_{L^\infty(\Omega_k)} \tag{3.6}$$

Now, we proceed with estimating N_2, the second term of (3.3). The L^2 projector P_h onto V_h^* is defined by

$$P_h(u) = \sum_{j=0}^{q} \frac{\alpha_j}{(\Delta x_k)^j} \phi_j^*,$$

with $\alpha = (\alpha_j)_{j\in\{1,\dots,q\}} = (M^*)^{-1} b$, where

$$M^*_{j\ell} = \int_{\Omega_k} \frac{\phi_j^*(x)}{(\Delta x_k)^j} \frac{\phi_\ell^*(x)}{(\Delta x_k)^\ell} dx, \quad \text{and} \quad b_j = \int_{\Omega_k} u(x) \frac{\phi_j^*(x)}{(\Delta x_k)^j} dx. \tag{3.7}$$

We are now ready to start estimating N_2. Note that

$$\begin{aligned} N_2 = \|T_\theta(u) - P_h(u)\|_{L^\infty(\Omega_k)} &= \sup_{x\in\Omega_k} \left| \sum_{j=0}^{q} \left(\left(\frac{u}{u_\theta} \right)^{(j)} (x_k) - \frac{\alpha_j}{(\Delta x_k)^j} \right) \frac{1}{j!} (x - x_k)^j u_\theta(x) \right| \\ &\leq \sup_{x\in\Omega_k} \left| \sum_{j=0}^{q} \left((\Delta x_k)^j \left(\frac{u}{u_\theta} \right)^{(j)} (x_k) - \alpha_j \right) \frac{1}{j!} \frac{(x-x_k)^j}{(\Delta x_k)^j} \right| \|u_\theta\|_{L^\infty(\Omega_k)}. \end{aligned}$$

Using the Cauchy-Schwartz inequality on the sum, and bounding the resulting polynomial on the cell, we obtain the estimate

$$N_2 \lesssim \left[\sum_{j=0}^{q} \left((\Delta x_k)^j \left(\frac{u}{u_\theta} \right)^{(j)} (x_k) - \alpha_j \right)^2 \right]^{\frac{1}{2}} \|u_\theta\|_{L^\infty(\Omega_k)} = \|\delta - \alpha\|_2 \|u_\theta\|_{L^\infty(\Omega_k)} \tag{3.8}$$

where the vector $\delta = (\delta_j)_{j\in\{0,\dots,q\}}$ is defined by

$$\delta_j = (\Delta x_k)^j \left(\frac{u}{u_\theta} \right)^{(j)} (x_k).$$

Recalling the definition $\alpha = (M^*)^{-1} b$, we obtain

$$\|\delta - \alpha\|_2 = \left\| (M^*)^{-1} (M^* \delta - b) \right\|_2 \leq \left\| (M^*)^{-1} \right\|_2 \|M^* \delta - b\|_2. \tag{3.9}$$

We first take care of the term in $M^*\delta - b$. We have

$$\|M^*\delta - b\|_2^2 = \sum_{j=0}^{q} \left[\sum_{\ell=1}^{q} M^*_{j\ell}\delta_\ell - b_j \right]^2$$
$$= \sum_{j=0}^{q} \left[\sum_{\ell=0}^{q} \left(\int_{\Omega_k} \frac{\phi_j^*(x)}{(\Delta x_k)^j} \frac{\phi_\ell^*(x)}{(\Delta x_k)^\ell} dx \right) (\Delta x_k)^\ell \left(\frac{u}{u_\theta} \right)^{(\ell)} (x_k) - \int_{\Omega_k} u(x) \frac{\phi_j^*(x)}{(\Delta x_k)^j} dx \right]^2 =: \sum_{j=0}^{q} \Xi_j^2$$

We denote the summand by Ξ_j, and we use the definition of the basis to obtain

$$\begin{aligned}
\forall j \in \{0, \dots, q\}, \quad \Xi_j &\coloneqq \sum_{\ell=0}^{q} (\Delta x_k)^\ell \left(\frac{u}{u_\theta} \right)^{(\ell)} (x_k) \int_{\Omega_k} \frac{\phi_j^*(x)}{(\Delta x_k)^j} \frac{\phi_\ell^*(x)}{(\Delta x_k)^\ell} dx - \int_{\Omega_k} u(x) \frac{\phi_j^*(x)}{(\Delta x_k)^j} dx \\
&= \sum_{\ell=0}^{q} (\Delta x_k)^\ell \left(\frac{u}{u_\theta} \right)^{(\ell)} (x_k) \int_{\Omega_k} \frac{\phi_j(x)}{(\Delta x_k)^j} \frac{\phi_\ell(x)}{(\Delta x_k)^\ell} u_\theta^2(x) dx - \int_{\Omega_k} \frac{u(x)}{u_\theta(x)} \frac{\phi_j(x)}{(\Delta x_k)^j} u_\theta^2(x) dx \\
&= \int_{\Omega_k} \left(\sum_{\ell=0}^{q} \left(\frac{u}{u_\theta} \right)^{(\ell)} (x_k) \phi_\ell(x) - \frac{u(x)}{u_\theta(x)} \right) \frac{\phi_j(x)}{(\Delta x_k)^j} u_\theta^2(x) dx. \qquad (3.10)
\end{aligned}$$

Using a Taylor expansion, we obtain, for all $j \in \{1, \dots, q\}$,

$$\Xi_j = - \int_{\Omega_k} \left(\int_{x_k}^{x} \left(\frac{u}{u_\theta} \right)^{(q+1)} (\xi) \frac{(x-\xi)^q}{q!} d\xi \right) \frac{\phi_j(x)}{(\Delta x_k)^j} u_\theta^2(x) dx,$$

from which we get the following upper bound

$$\forall j \in \{1, \dots, q\}, \quad |\Xi_j| \leq \sup_{x \in \Omega_k} \left| \int_{x_k}^{x} \left(\frac{u}{u_\theta} \right)^{(q+1)} (\xi) \frac{(x-\xi)^q}{q!} d\xi \right| \left| \int_{\Omega_k} \frac{\phi_j(x)}{(\Delta x_k)^j} u_\theta^2(x) dx \right|.$$

Using the same ingredients as in the computation of (3.5) for the leftmost term and bounding the rightmost term by the L^∞ norm of the prior and by noting that the classical basis functions are bounded, we obtain the estimate

$$\forall j \in \{1, \dots, q\}, \quad |\Xi_j| \lesssim \left| \frac{u}{u_\theta} \right|_{H^{q+1}(\Omega_k)} (\Delta x_k)^{q+\frac{1}{2}} (\Delta x_k) \left\| u_\theta^2 \right\|_{L^\infty(\Omega_k)}.$$

Going back to what we had set out to prove, we get

$$\|M^*\delta - b\|_2 = \left(\sum_{j=0}^{q} |\Xi_j|^2 \right)^{\frac{1}{2}} \lesssim \left| \frac{u}{u_\theta} \right|_{H^{q+1}(\Omega_k)} (\Delta x_k)^{q+\frac{1}{2}} (\Delta x_k) \left\| u_\theta^2 \right\|_{L^\infty(\Omega_k)}. \qquad (3.11)$$

Plugging (3.11) into (3.9) and then into (3.8), we get

$$N_2 \lesssim \left\| (M^*)^{-1} \right\|_2 \left| \frac{u}{u_\theta} \right|_{H^{q+1}(\Omega_k)} (\Delta x_k)^{q+\frac{1}{2}} (\Delta x_k) \left\| u_\theta^2 \right\|_{L^\infty(\Omega_k)} \|u_\theta\|_{L^\infty(\Omega_k)}. \qquad (3.12)$$

Finally, we note that, for any $y \in \mathbb{R}^{q+1}$, given the expression (3.7) of M^*,

$$\begin{aligned}
(M^*y, M^*y) = \int_{\Omega_k} \left(\sum_{j=0}^{q} \frac{\phi_j^*(x)}{(\Delta x_k)^j} y_j \right)^2 dx &= \int_{\Omega_k} \left(\sum_{j=0}^{q} \frac{\phi_j(x)}{(\Delta x_k)^j} y_j \right)^2 u_\theta(x)^2 dx \\
&\geqslant m^2 \int_{\Omega_k} \left(\sum_{j=0}^{q} \frac{\phi_j(x)}{(\Delta x_k)^j} y_j \right)^2 dx = m^2 (My, My),
\end{aligned}$$

where M is the mass matrix associated with the classical basis functions

$$M_{j\ell} = \int_{\Omega_k} \frac{\phi_j(x)}{(\Delta x_k)^j} \frac{\phi_\ell(x)}{(\Delta x_k)^\ell} dx = \frac{\Delta x_k}{1+j+\ell} = \Delta x_k H_{j\ell},$$

where $H = (H_{j\ell})_{j\ell}$ is the Hilbert matrix. Then we deduce the following inequality

$$\|(M^*)^{-1}\|_2 \leqslant \frac{1}{m^2}\|M^{-1}\|_2 = \frac{1}{m^2}\frac{1}{\Delta x_k}\|H^{-1}\|_2. \tag{3.13}$$

Combining (3.12) and (3.13), we obtain

$$N_2 \lesssim \left|\frac{u}{u_\theta}\right|_{H^{q+1}(\Omega_k)} (\Delta x_k)^{q+\frac{1}{2}} \frac{\left\|u_\theta^2\right\|_{L^\infty(\Omega_k)}}{m^2} \|u_\theta\|_{L^\infty(\Omega_k)}. \tag{3.14}$$

We get, from (3.6) and (3.14), the expected result. □

The above proof relies on the smoothness of the prior. This may seem counter-intuitive in a hyperbolic context. However, since the prior will be obtained from a neural network in Section 4, this smoothness assumption becomes reasonable.

Lemma 3.5. *We make the same assumptions as in the previous lemma, and still consider the vector space V_h^*. For any function $u \in H^{q+1}(\Omega)$,*

$$\|u - P_h(u)\|_{L^2(\Omega)} \lesssim \left|\frac{u}{u_\theta}\right|_{H^{q+1}(\Omega)} (\Delta x_k)^{q+1} \|u_\theta\|_{L^\infty(\Omega)}.$$

Proof. We begin by stating the definition of the discrete L^2 norm: by assuming that

$$\Omega = \bigcup_{k=1}^N \Omega_k,$$

we obtain

$$\|u - P_h(u)\|_{L^2(\Omega)}^2 \leqslant \sum_{k=1}^N \Delta x_k \|u - P_h(u)\|_{L^\infty(\Omega_k)}^2.$$

Using the result from Lemma 3.4, we get

$$\begin{aligned}\|u - P_h(u)\|_{L^2(\Omega)}^2 &\lesssim \sum_{k=1}^N \Delta x_k \left(\left|\frac{u}{u_\theta}\right|_{H^{q+1}(\Omega_k)} (\Delta x_k)^{q+\frac{1}{2}} \left(1 + \frac{\left\|u_\theta^2\right\|_{L^\infty(\Omega_k)}}{m^2}\right) \|u_\theta\|_{L^\infty(\Omega_k)}\right)^2 \\ &\lesssim \sum_{k=1}^N \left|\frac{u}{u_\theta}\right|_{H^{q+1}(\Omega_k)}^2 (\Delta x_k)^{2q+2} \left(1 + \frac{\left\|u_\theta^2\right\|_{L^\infty(\Omega_k)}}{m^2}\right)^2 \|u_\theta\|_{L^\infty(\Omega_k)}^2.\end{aligned}$$

We assume that there exists δ_-, δ_+ and Δx such that, for all $k \in \{1, \dots, N\}$, $\delta_- \Delta x \leqslant \Delta x_k \leqslant \delta_+ \Delta x$. Then, since $\|u_\theta\|_{L^\infty(\Omega_k)} \leqslant \|u_\theta\|_{L^\infty(\Omega)}$, we obtain

$$\begin{aligned}\|u - P_h(u)\|_{L^2(\Omega)}^2 &\lesssim (\Delta x)^{2q+2} \sum_{k=1}^N \left|\frac{u}{u_\theta}\right|_{H^{q+1}(\Omega_k)}^2 \left(1 + \frac{\left\|u_\theta^2\right\|_{L^\infty(\Omega)}}{m^2}\right)^2 \|u_\theta\|_{L^\infty(\Omega_k)}^2 \\ &\lesssim (\Delta x)^{2q+2} \left(1 + \frac{\left\|u_\theta^2\right\|_{L^\infty(\Omega_k)}}{m^2}\right)^2 \|u_\theta\|_{L^\infty(\Omega)}^2 \sum_{k=1}^N \left|\frac{u}{u_\theta}\right|_{H^{q+1}(\Omega_k)}^2.\end{aligned}$$

The proof is concluded by recognizing the $H^{q+1}(\Omega)$ seminorm. □

This global error estimate Lemma 3.5 shows that projection error onto the basis V_h^* is bounded by

$$\left| \frac{u}{u_\theta} \right|_{H^{q+1}(\Omega)}.$$

This bound is equal to zero if the prior is exact, since it is nothing but the $(q+1)^{\text{th}}$ derivative of the constant function equal to one. This estimate also proves that the closer the prior is to the solution, the smaller the bound of the projection error. However, to obtain an even smaller bound, we need the prior and the solution to be close in the sense of the $H^{q+1}(\Omega)$ seminorm. This means that the prior must be constructed in such a way that it also gives a good approximation of the derivatives of the solution.

As a summary, we have shown that the L^2 projection error tends to zero when the prior tends to the solution. This result gives an idea of the expected gains in error ensured by using the modified basis V_h^*. The final convergence error depends on this projection error, as has been shown in [56]. The proof to obtain the final convergence result is the same as in [56].

For the additive basis V_h^+, such error estimates are harder to obtain, since the projection in the new basis is harder to write with respect to the traditional one. We expect an error bounded by a term in $\|u - u_\theta\|_{H^{q+1}(\Omega)}$, which would enable us to draw similar conclusions as for the multiplicative basis V_h^*. Namely, the error would also tend to zero when the prior tends to the solution, and the derivatives of the prior would need to be close to the derivatives of the solution. Proving this result is out of the scope of this paper, even though it should be ensured by the results of [56]. However, we will extensively study the behavior of the additive basis in Section 5.

4 Prior construction and algorithm

Equipped with the modified bases from Section 2.2 and with the theoretical results from Section 3, what is left to do is to propose a way to obtain a suitable prior u_θ.

Note that the approach described in Section 2 will be interesting if the prior u_θ is a good approximation of the steady solution to (1.1) for a wide range of parameters. In addition, according to Section 3.2, the derivatives of the prior must also be good approximations of the derivatives of the steady solution.

This means that we wish to capture large families of solutions, i.e. we want to be able to calculate an approximation for several parameters. For example, assuming that (1.1) depends on 4 physical parameters leads to $\mu \in \mathbb{R}^4$, and considering a problem in two space dimensions, leads to $x \in \mathbb{R}^2$. Therefore, we are looking for a prior $u_\theta(x;\mu)$, where $u_\theta \in \mathcal{C}^q(\mathbb{R}^2 \times \mathbb{R}^4, \mathbb{R})$. Approaching such a function using polynomials defined on a mesh would be a very difficult task, especially if the space or parameter domains have a complex geometry. Neural networks have demonstrated their ability to approximate functions in fairly high dimensions, notably thanks to their intrinsic regularity. PINNs are a mesh-free approach to solving PDEs using neural networks. Their properties make them good candidates for approaching solutions to high-dimensional problems.

To build our prior, we propose to solve the parametric steady problem with PINNs. To that end, we now briefly introduce this method in Section 4.1, and we show how to compute and store the prior. Then, our algorithm is summarized in Section 4.2.

4.1 Parametric PINNs

Note that the steady solutions to (1.1) are given by

$$\partial_x F_{\mu_1}(u) = S_{\mu_2}(u).$$

This is nothing but a parametric elliptic problem. Therefore, we introduce PINNs for the following generic boundary value problems (BVPs):

$$\begin{cases} \mathcal{D}(u, x; \mu) = f(x, \mu) & \text{for } x \in \Omega, \\ u(t, x) = g(x, \mu), & \text{for } x \in \partial\Omega, \end{cases} \tag{4.1}$$

where $\mathcal{D}$ is a differential operator involving the solution u and its space derivatives, and with μ some physical parameters. We recall that $\mu \in \mathbb{P} \subset \mathbb{R}^m$. PINNs use the fact that classical fully-connected neural networks are

smooth functions of their inputs, as long as their activation functions are also smooth, to approximate the solution to (4.1). Contrary to traditional numerical schemes such as the DG method, where the degrees of freedom encode some explicit modal or nodal values of the solutions, the degrees of freedom of PINNs representation are the weights θ of the neural network, and so do not explicitly represent the solution. Equipped with both of these remarks, the idea behind PINNs is to plug the network, which represents the solution to (4.1), into the equation. Then, the degrees of freedom (i.e. the weights θ of the network) are found by minimizing a loss function. Since the neural network is differentiable, the derivatives can be exactly computed. In our case, the PINN is thus a smooth neural network that takes as input the space variable x and the parameter vector μ, which we denote by $u_\theta(x;\mu)$.

Thanks to these definitions, solving the PDE can be rewritten as the following minimization problem:

$$\min_\theta \mathcal{J}(\theta), \quad \text{where } \mathcal{J}(\theta) = \mathcal{J}_r(\theta) + \mathcal{J}_b(\theta) + \mathcal{J}_{\text{data}}(\theta). \tag{4.2}$$

In (4.2), we have introduced three different terms: the residual loss function $\mathcal{J}_r$, the boundary loss function $\mathcal{J}_b$, and the data loss function $\mathcal{J}_{\text{data}}$. For parameters $\mu \in \mathbb{P}$, the residual loss function is defined by

$$\mathcal{J}_r(\theta) = \int_{\mathbb{P}} \int_\Omega \left\| \mathcal{D}(u_\theta, x; \mu) - f(x;\mu) \right\|_2^2 dx d\mu, \tag{4.3}$$

while the boundary loss function is given by

$$\mathcal{J}_b(\theta) = \int_{\mathbb{P}} \int_{\partial\Omega} \left\| u_\theta(x,\mu) - g(x,\mu) \right\|_2^2 dx d\mu. \tag{4.4}$$

Finally, to define the data loss function, we assume that we know the exact solution to (4.1) at some points x_i and for some parameters μ_i, and we set

$$\mathcal{J}_{\text{data}}(\theta) = \sum_i \left\| u_\theta(x_i,\mu_i) - u(x_i,\mu_i) \right\|_2^2.$$

In practice, the integrals in (4.3) and (4.4) are approximated using a Monte-Carlo method. This method relies on sampling a certain number of so-called "collocation points" in order to approximate the integrals. Then, the minimization problem on θ is solved using a gradient-type method, which corresponds to the learning phase.

The main advantage of PINNs is that they are mesh-free and less sensitive to dimension than classical methods. Indeed, neural networks easily deal with large input dimensions, and the Monte-Carlo method converges independently of the dimension. Consequently, PINNs are particularly well-suited to solving parametric PDEs such as (4.1). Thanks to that, we do not solve for a single equilibrium but rather for families of equilibria indexed by the parameters μ.

Traditional PINNs use this method to approximate both (4.3) and (4.4). However, for the boundary conditions, we elected to use another approach, which makes it possible to completely eliminate $\mathcal{J}_b$ from the minimization algorithm. The idea is to define the approximate solution through a boundary operator $\mathcal{B}$, which can for instance be a multiplication by a function which satisfies the boundary condition. We obtain

$$\widetilde{u}_\theta(x;\mu) = \mathcal{B}\big(u_\theta, x; \mu\big),$$

with u_θ the neural network and $\mathcal{B}$ a simple operator such as $\widetilde{u}_\theta$ exactly satisfies the boundary conditions. Using $\widetilde{u}_\theta$, the residual loss becomes, instead of (4.3):

$$\mathcal{J}_r(\theta) = \int_{\mathbb{P}} \int_\Omega \left\| \mathcal{D}(\widetilde{u}_\theta, x; \mu) - f(x;\mu) \right\|_2^2 dx d\mu. \tag{4.5}$$

Examples of such functions $\mathcal{B}$ are provided in Section 5.

With this approach, we have presented one method for offline construction of our prior for a family of equilibria. Note that it is possible to further enhance this prior with data from previous simulations, thanks to the loss function $\mathcal{J}_{\text{data}}$. Even though training PINNs may be harder than training traditional purely data-driven neural networks, they are much more efficient as priors. Indeed, the error estimates of Section 3.2 show that the error

depends on the q-th derivative of the ratio between the prior and the solution. Therefore, to obtain a small error, it is important for the prior to provide a good approximation of not only the steady solution, but also of its derivatives. Since the PINN loss (4.5) inherently contains derivatives of u_θ, the resulting trained PINN will be more efficient in this respect. Note that a purely data-driven network could also be interesting if the data contains information on the derivatives.

4.2 Algorithm

Now that we have discussed the strategy we use to obtain our prior, we give some details on the offline and online algorithms that we developed to construct the modified DG bases in practice. We start by describing the offline step, where the families of priors are computed. Then, we move on to an online algorithm, explaining how to construct the DG bases using the prior, and how to apply them to the actual DG time iterations.

Algorithm 1. Offline part: neural network training

Input: space domain Ω, parameter set $\mathbb{P}$, initial neural network $u_{\theta_0}(x;\mu)$, η learning rate, N number of collocation points, n_{epochs} number of training epochs
Output: trained neural network $u_\theta(x;\mu)$
1: initialize the weights: $\theta = \theta_0$
2: **for** $n \leq n_{\text{epochs}}$ **do**
3: sample N values of x in Ω and μ in $\mathbb{P}$
4: compute the loss function $\mathcal{J}(\theta)$
5: update θ using the gradient of $\mathcal{J}(\theta)$: $\theta = \theta - \eta \nabla_\theta J(\theta)$
6: **end for**

In practice, we do not use a classical gradient descent to update the weights, but rather the Adam algorithm. Moreover, sampling is done through a uniform law on the space and parameter domains. It would also be possible use non-uniform sampling like in [52] for instance, but we elected to use uniform sampling for the sake of simplicity. Note that Algorithm 1 does not contain solution data in its inputs. Indeed, almost all numerical experiments from Section 5 do not require data on the solution. This avoids the cost of data production, which would otherwise require sampling the exact solution if it is known, or using a numerical scheme otherwise.

Algorithm 2. Online part: using the neural network in the DG scheme

Input: prior u_θ, degree N_q of the Gauss-Lobatto quadrature rule, initial data u_0, space mesh Ω_h, parameters μ, n_t number of time steps
Output: numerical solution $u_k(t,x)$ on each cell Ω_k
1: use the mesh Ω_h to obtain all quadrature points $x_{k,p}$ in each cell Ω_k
2: evaluate the prior at each point $x_{k,p}$: we obtain $\tilde{u}_{k,p} \coloneqq u_\theta(x_{k,p};\mu)$
3: reconstruct $u_k(0,x)$ using $\tilde{u}_{k,p}$
4: **for** $n \leq n_t$ **do**
5: construct the mass matrix $\mathcal{M}$, the nonlinear flux $\mathcal{F}$, the interface flux $\mathcal{G}$ and the source term $\mathcal{S}$ using $\tilde{u}_{k,p}$ and the quadrature rule
6: update the solution u_k at the next time step, using u_k at the previous time step as well as the terms computed in the previous step
7: **end for**

In this second step, the additional computational overhead associated to our method, compared to the classical DG scheme, comes from two distinct sources. The first one is a preprocessing phase, where we evaluate the prior on the quadrature points (step 2 of Algorithm 2). Even though such networks have been made to be quickly evaluated on GPUs, this evaluation step remains fast on CPUs. The second source of computational cost is associated to the quadrature rule. Indeed, in some cases, we will require a quadrature rule with a higher degree than the traditional DG scheme. The classical approach is to use $N_q = q$ quadrature points for bases made of q polynomial functions, since the quadrature is exact for polynomials of degree q. However, in our case, our basis

is non-polynomial. Hence, to have a good approximation of the integral of the prior, we may need to increase the degree of the quadrature. In most cases, this increase is slight; for a few test cases, especially to approximate functions with large derivatives, we will need to use fine quadrature rules.

5 Applications and numerical results

This section is dedicated to a validation of the approach on several parametric hyperbolic systems of balance laws: the linear advection equation in Section 5.1, the 1D shallow water equations in Section 5.2, the Euler-Poisson system in Section 5.3, and the 2D shallow water equations in Section 5.4. In the first two cases, there exist some exact well-balanced schemes in the literature. However, for the Euler-Poisson system and the 2D shallow water equations, exact (or even approximate) WB schemes are either not available or very complicated to implement. Code replicating the experiments of Section 5.1 is freely available [33] on GitHub[1].

In this section, we denote by K the number of cells. We test both bases V_h^* and V_h^+ at first, showing that both display similar results. To cut down on the number of tables, we then only present the results for the additive basis V_h^+.

Moreover, the time step Δt is given by

$$\Delta t = C_{\mathrm{CFL}}\, C_{\mathrm{RK}}\, \frac{\min_{k\in\{1,\dots,K\}} \Delta x_k}{\lambda}, \tag{5.1}$$

where λ is the maximal wave speed of the system, C_{CFL} is a CFL (Courant-Friedrichs-Lewy) coefficient, and C_{RK} is the stability coefficient associated to the time discretization. All experiments are run using a strong stability-preserving Runge-Kutta (SSPRK) time discretization of the correct order. The time discretizations, with their associated stability coefficients C_{RK}, are collected in Table 1. To determine C_{CFL}, we run a study of the stability condition for the first experiment; this study is not repeated for the other experiments, since the new bases do not influence the stability condition.

number q of basis elements	0	1	2	3
time discretization	explicit Euler	SSPRK2 [45]	SSPRK3(5) [45]	SSPRK4(10) [29]
stability coefficient C_{RK}	1	1	2.65	3

Table 1: Stability coefficients C_{RK} of the high-order time discretizations used in the numerical experiments, with respect to the number q of basis elements.

5.1 Linear advection

We first consider the case of a linear advection equation with a source term, on the space domain $\Omega = (0, 1)$. The equation is given as follows:

$$\begin{cases} \partial_t u + \partial_x u = s(u;\mu), & \text{for } x \in \Omega \\ u(t=0,x) = u_{\mathrm{ini}}(x;\mu), \\ u(t, x=0) = u_0, \end{cases} \tag{5.2}$$

Here, the parameter vector μ is made of three elements:

$$\mu = \begin{pmatrix} \alpha \\ \beta \\ u_0 \end{pmatrix} \in \mathbb{P} \subset \mathbb{R}^3, \quad \alpha \in \mathbb{R}_+, \quad \beta \in \mathbb{R}_+, \quad u_0 \in \mathbb{R}_+^*.$$

The source term depends on μ as follows:

$$s(u;\mu) = \alpha u + \beta u^2,$$

[1] https://github.com/Victor-MichelDansac/DG-PINNs.git

and straightforward computations show that the associated steady solutions take the form

$$u_{\mathrm{eq}}(x;\mu) = \frac{\alpha u_0}{(\alpha+\beta u_0)e^{-\alpha x} - \beta u_0}. \tag{5.3}$$

To compute the time step, we take $\lambda = 1$ in (5.1), since the advection velocity in (5.2) is equal to 1. The first paragraph of this section shows how to choose C_{CFL} to complete the determination of the time step. Unless otherwise stated, we prescribe Dirichlet boundary conditions consisting in the steady solution.

To obtain a suitable prior u_θ, we train a PINN with parameters θ. To avoid cumbersome penalization of boundary conditions, we define $\widetilde{u}_\theta$ using a boundary operator $\mathcal{B}$, as follows:

$$\widetilde{u}_\theta(x;\mu) = \mathcal{B}(u_\theta, x;\mu) = u_0 + x u_\theta(x;\mu),$$

so that the boundary condition $\widetilde{u}_\theta(0;\mu) = u_0$ is automatically satisfied by $\widetilde{u}_\theta$. The parameter space $\mathbb{P}$ is chosen such that the steady solution is well-defined, and we take

$$\mathbb{P} = [0.5, 1] \times [0.5, 1] \times [0.1, 0.2]. \tag{5.4}$$

Thanks to the boundary operator $\mathcal{B}$, the loss function only concerns the ODE residue, and we set

$$\mathcal{J}(\theta) = \left\| \partial_x \widetilde{u}_\theta - \alpha \widetilde{u}_\theta - \beta \widetilde{u}_\theta^2 \right\|_2^2.$$

We use a neural network with 5 fully connected hidden layers, and around 1200 trainable parameters. Training takes about 4 minutes on a dual NVIDIA K80 GPU, until the loss is equal to about 10^{-6}. For this experiment, we increased the order of the quadrature compared to the baseline for the case with one basis function. Indeed, we take $n_Q = \max(q+2, 3)$, to ensure a sufficient precision when integrating the prior.

In this section, we compare four strategies: the basis V_h (2.4), the basis V_h^* with multiplicative prior (2.6), the basis V_h^+ with additive prior (2.5), and the basis $V_h^{\mathrm{ex},+}$ which uses the exact steady solution (5.3) as a prior. First, we study the stability condition in Section 5.1.1. Then, we tackle the approximation of a steady solution without perturbation in Section 5.1.2 and with perturbation in Section 5.1.3. Finally, the approximation of an unsteady solution is computed in Section 5.1.4.

5.1.1 Study of the stability condition

The very first experiment we run aims at making sure that the new bases do not alter the stability condition of the DG scheme. To that end, we slowly increase C_{CFL} until the time step Δt is too large for the scheme to be stable. For this experiment, the initial condition is made of the steady solution

$$u_{\mathrm{ini}}(x;\mu) = u_{\mathrm{eq}}(x;\mu), \tag{5.5}$$

and the final time is $T = 0.5$. Table 2 contains the optimal values of C_{CFL} (larger values leading to instabilities) obtained with the four bases and for $q \in \{0,1,2,3\}$. We observe that the new bases do not change the stability condition, except for $V_h^{\mathrm{ex},+}$ with $q = 1$, which is slightly more stable. This study will not be repeated for other experiments, since it would yield similar results. In practice, we take $C_{\mathrm{CFL}} = 0.1$ to ensure stability.

q	basis V_h	basis V_h^*	basis V_h^+	basis $V_h^{\mathrm{ex},+}$
0	1.250	1.250	1.250	1.250
1	0.399	0.399	0.399	0.416
2	0.209	0.209	0.209	0.209
3	0.185	0.185	0.185	0.185

Table 2: Maximal values of C_{CFL} obtained for the four bases and for a number of basis elements $q \in \{0,1,2,3\}$.

5.1.2 Steady solution

We now study the approximation of a steady solution, with and without perturbation. The goal of this section is to check whether the prior indeed makes it possible to decrease the error compared to the usual modal basis. For this experiment, the initial condition remains (5.5), and the final time is $T = 0.1$.

As a first step, the values of the parameters μ are set to the midpoints of the intervals making up the parameter space (5.4). The L^2 errors between the exact and approximate solutions are collected in Table 3.

In this case, we expect both V_h^* and V_h^+ to show similar behavior. Moreover, we expect the basis $V_h^{\mathrm{ex},+}$ to provide an exactly well-balanced scheme, up to machine precision. To that end, only for $V_h^{\mathrm{ex},+}$, we take $n_Q = \max(q+2, 5)$, to ensure that the quadrature of the exact prior is also exact, up to machine precision.

We observe that the bases with and without prior allow a convergence of the correct order, i.e. of the same order as the number of basis elements. Moreover, we observe a consistent gain for all mesh resolutions, for a given size of the modal basis, which is lower the larger the size of the basis. Bases V_h^* and V_h^+ seem to have comparable performance, with V_h^* being somewhat better for large values of q, and V_h^+ taking the lead for small values of q. Finally, we observe that the basis $V_h^{\mathrm{ex},+}$ is indeed able to provide a solution that is exact up to machine precision, thus validating the exact well-balanced property of the scheme using this basis.

As a second step, to refine this study, we now consider 10^3 parameters, randomly sampled from the parameter space (5.4). For $q \in \{0, 1, 2, 3\}$ and $K = 10$ discretization cells, we compute the minimum, average and maximum gains obtained with both bases V_h^* and V_h^+. These values are reported in Table 4. We observe, on average, a significant gain in all cases, with larger gains obtained for smaller values of q. Furthermore, the minimum gain is always greater than one. Like in the previous experiment, we observe that, even though both bases display similar behavior and very good results, V_h^+ behaves better than V_h^* for small values of q, and vise versa. Consequently, and to limit the number of tables in the remainder of this section, we perform all subsequent experiments with the basis V_h^+.

5.1.3 Perturbed steady solution

We now test the scheme on a perturbed steady solution, For this experiment, the initial condition is similar to (5.5), but with a perturbation. Indeed, we take

$$u_{\mathrm{ini}}(x;\mu) = \big(1 + \varepsilon \sin(2\pi x)\big) u_{\mathrm{eq}}(x;\mu),$$

where ε is taken nonzero or zero, to control the strength of the perturbation. The final time is $T = 2$, and we study the impact of the perturbation by taking $\varepsilon \in \{10^{-4}, 10^{-2}, 1\}$, and $K = 10$ discretization cells. The results are collected in Figure 1. We observe two different states: first, while the perturbation is being dissipated, the errors with the two bases are similar. Then, we note that the introduction of the prior has made it possible for the approximate solution to converge towards a final solution that is closer to the exact, unperturbed steady solution.

5.1.4 Unsteady solution

Next, we seek to confirm that our proposed basis does not deteriorate the approximation of unsteady solutions. To that end, we consider an unsteady solution of the homogeneous problem, i.e. a solution to (5.2) with $s(u;\mu) = 0$. We take the following initial condition:

$$u_0(x) = 0.1\left(1 + \exp\left(-100(x - 0.5)^2\right)\right),$$

so that $u(t,x) = u_0(x - t)$. The final time is set to $T = 1$, and periodic boundary conditions are prescribed.

We compute the approximate solution with the two bases, for several values of q. The results are collected in Table 5. We note that the basis with prior does not affect the approximate solution for $q \geq 1$, while the results are slightly worse with the prior for $q = 0$. To improve the results here, one could introduce a space-time basis in a space-time discontinuous Galerkin method; this will be the object of future work.

	basis V_h		basis V_h^*			basis V_h^+			basis $V_h^{\mathrm{ex},+}$
K	error	order	error	order	gain	error	order	gain	error
10	$1.75 \cdot 10^{-2}$	—	$1.45 \cdot 10^{-5}$	—	1200.02	$1.45 \cdot 10^{-5}$	—	1200.02	$1.66 \cdot 10^{-14}$
20	$8.75 \cdot 10^{-3}$	1.00	$7.61 \cdot 10^{-6}$	0.93	1149.11	$7.61 \cdot 10^{-6}$	0.93	1149.11	$2.78 \cdot 10^{-17}$
40	$4.38 \cdot 10^{-3}$	1.00	$3.92 \cdot 10^{-6}$	0.96	1118.29	$3.92 \cdot 10^{-6}$	0.96	1118.29	$5.89 \cdot 10^{-17}$
80	$2.19 \cdot 10^{-3}$	1.00	$2.00 \cdot 10^{-6}$	0.97	1098.77	$2.00 \cdot 10^{-6}$	0.97	1098.77	$2.78 \cdot 10^{-17}$
160	$1.10 \cdot 10^{-3}$	1.00	$1.01 \cdot 10^{-6}$	0.98	1085.96	$1.01 \cdot 10^{-6}$	0.98	1085.96	$2.19 \cdot 10^{-17}$

(a) Errors with a basis made of one element: $q = 0$.

	basis V_h		basis V_h^*			basis V_h^+			basis $V_h^{\mathrm{ex},+}$
K	error	order	error	order	gain	error	order	gain	error
10	$4.93 \cdot 10^{-4}$	—	$2.18 \cdot 10^{-6}$	—	226.00	$1.03 \cdot 10^{-6}$	—	479.78	$2.04 \cdot 10^{-14}$
20	$1.24 \cdot 10^{-4}$	2.00	$3.20 \cdot 10^{-7}$	2.77	386.66	$2.74 \cdot 10^{-7}$	1.91	450.59	$6.48 \cdot 10^{-16}$
40	$3.09 \cdot 10^{-5}$	2.00	$8.07 \cdot 10^{-8}$	1.99	382.88	$7.00 \cdot 10^{-8}$	1.97	441.39	$9.46 \cdot 10^{-16}$
80	$7.72 \cdot 10^{-6}$	2.00	$2.05 \cdot 10^{-8}$	1.98	376.52	$1.76 \cdot 10^{-8}$	1.99	438.98	$1.46 \cdot 10^{-15}$
160	$1.93 \cdot 10^{-6}$	2.00	$5.16 \cdot 10^{-9}$	1.99	374.49	$4.40 \cdot 10^{-9}$	2.00	438.34	$2.13 \cdot 10^{-15}$

(b) Errors with a basis made of two elements: $q = 1$.

	basis V_h		basis V_h^*			basis V_h^+			basis $V_h^{\mathrm{ex},+}$
K	error	order	error	order	gain	error	order	gain	error
10	$7.89 \cdot 10^{-6}$	—	$1.00 \cdot 10^{-7}$	—	78.58	$1.05 \cdot 10^{-7}$	—	74.90	$9.92 \cdot 10^{-13}$
20	$9.94 \cdot 10^{-7}$	2.99	$1.33 \cdot 10^{-8}$	2.91	74.60	$1.41 \cdot 10^{-8}$	2.90	70.65	$7.84 \cdot 10^{-15}$
40	$1.24 \cdot 10^{-7}$	3.00	$1.72 \cdot 10^{-9}$	2.95	72.13	$1.79 \cdot 10^{-9}$	2.97	69.19	$2.84 \cdot 10^{-15}$
80	$1.55 \cdot 10^{-8}$	3.00	$2.17 \cdot 10^{-10}$	2.99	71.43	$2.25 \cdot 10^{-10}$	2.99	68.81	$7.81 \cdot 10^{-15}$
160	$1.94 \cdot 10^{-9}$	3.00	$2.72 \cdot 10^{-11}$	3.00	71.25	$2.82 \cdot 10^{-11}$	3.00	68.72	$1.15 \cdot 10^{-14}$

(c) Errors with a basis made of three elements: $q = 2$.

	basis V_h		basis V_h^*			basis V_h^+			basis $V_h^{\mathrm{ex},+}$
K	error	order	error	order	gain	error	order	gain	error
10	$1.20 \cdot 10^{-7}$	—	$8.31 \cdot 10^{-9}$	—	14.40	$1.12 \cdot 10^{-8}$	—	10.67	$4.45 \cdot 10^{-11}$
20	$7.39 \cdot 10^{-9}$	4.02	$5.51 \cdot 10^{-10}$	3.91	13.40	$7.28 \cdot 10^{-10}$	3.95	10.15	$7.72 \cdot 10^{-13}$
40	$4.59 \cdot 10^{-10}$	4.01	$3.48 \cdot 10^{-11}$	3.99	13.19	$4.56 \cdot 10^{-11}$	4.00	10.06	$1.70 \cdot 10^{-14}$
80	$2.92 \cdot 10^{-11}$	3.98	$2.20 \cdot 10^{-12}$	3.99	13.27	$2.86 \cdot 10^{-12}$	3.99	10.18	$6.93 \cdot 10^{-15}$
160	$1.85 \cdot 10^{-12}$	3.98	$1.29 \cdot 10^{-13}$	4.10	14.38	$1.72 \cdot 10^{-13}$	4.06	10.76	$2.59 \cdot 10^{-14}$

(d) Errors with a basis made of four elements: $q = 3$.

Table 3: Advection equation: errors, orders of accuracy, and gain obtained when approximating a steady solution for bases without prior (basis V_h), with a PINN prior (bases V_h^* and V_h^+), and with an exact prior (basis $V_h^{\mathrm{ex},+}$).

	gains in basis V_h^*			gains in basis V_h^+		
q	minimum	average	maximum	minimum	average	maximum
0	63.46	735.08	4571.89	63.46	735.08	4571.89
1	32.22	149.38	450.74	26.01	190.08	830.20
2	6.20	54.16	118.45	5.92	45.47	313.07
3	1.55	19.54	108.10	1.56	13.69	184.17

Table 4: Advection equation: statistics of the gains obtained for the approximation of a steady solution in bases V_h^* and V_h^+ with respect to basis V_h.

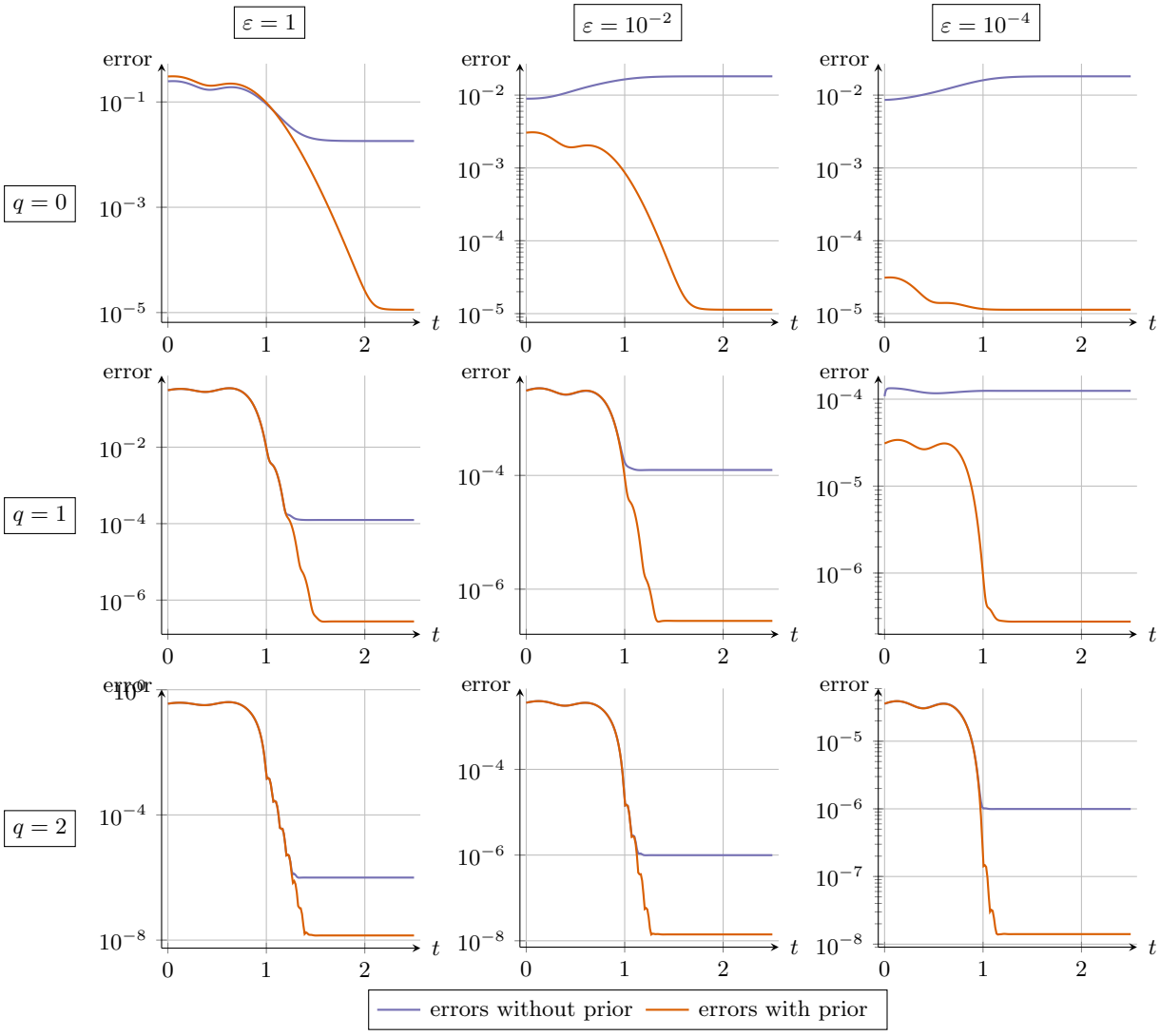

Figure 1: Advection equation: errors, with respect to time, for the approximation of a perturbed steady solution for bases with and without prior.

5.2 Shallow water equations

After studying a scalar linear advection equation in Section 5.1, we now turn to a nonlinear system of conservation laws. Namely, we tackle the shallow water equations

$$\begin{cases} \partial_t h + \partial_x Q = 0, \\ \partial_t Q + \partial_x \left(\dfrac{Q^2}{h} + \dfrac{1}{2} g h^2 \right) = -gh \partial_x Z(x; \alpha, \beta), \end{cases} \tag{5.6}$$

	basis V_h		basis V_h^+		
K	error	order	error	order	gain
10	$4.04 \cdot 10^{-2}$	—	$5.04 \cdot 10^{-2}$	—	0.80
20	$3.46 \cdot 10^{-2}$	0.22	$4.28 \cdot 10^{-2}$	0.24	0.81
40	$2.84 \cdot 10^{-2}$	0.28	$3.50 \cdot 10^{-2}$	0.29	0.81
80	$2.15 \cdot 10^{-2}$	0.40	$2.64 \cdot 10^{-2}$	0.40	0.81
160	$1.47 \cdot 10^{-2}$	0.55	$1.81 \cdot 10^{-2}$	0.55	0.81

(a) Errors with a basis made of one element: $q = 0$.

	basis V_h		basis V_h^+		
K	error	order	error	order	gain
10	$1.92 \cdot 10^{-2}$	—	$1.93 \cdot 10^{-2}$	—	1.00
20	$6.26 \cdot 10^{-3}$	1.62	$6.27 \cdot 10^{-3}$	1.62	1.00
40	$1.19 \cdot 10^{-3}$	2.39	$1.20 \cdot 10^{-3}$	2.39	1.00
80	$1.99 \cdot 10^{-4}$	2.59	$1.99 \cdot 10^{-4}$	2.59	1.00
160	$4.19 \cdot 10^{-5}$	2.24	$4.20 \cdot 10^{-5}$	2.24	1.00

(b) Errors with a basis made of two elements: $q = 1$.

	basis V_h		basis V_h^+		
K	error	order	error	order	gain
10	$5.15 \cdot 10^{-3}$	—	$5.15 \cdot 10^{-3}$	—	1.00
20	$4.56 \cdot 10^{-4}$	3.50	$4.56 \cdot 10^{-4}$	3.50	1.00
40	$4.55 \cdot 10^{-5}$	3.32	$4.55 \cdot 10^{-5}$	3.32	1.00
80	$5.42 \cdot 10^{-6}$	3.07	$5.42 \cdot 10^{-6}$	3.07	1.00
160	$6.75 \cdot 10^{-7}$	3.01	$6.75 \cdot 10^{-7}$	3.01	1.00

(c) Errors with a basis made of three elements: $q = 2$.

	basis V_h		basis V_h^+		
K	error	order	error	order	gain
10	$4.72 \cdot 10^{-4}$	—	$4.72 \cdot 10^{-4}$	—	1.00
20	$2.87 \cdot 10^{-5}$	4.04	$2.87 \cdot 10^{-5}$	4.04	1.00
40	$1.81 \cdot 10^{-6}$	3.99	$1.81 \cdot 10^{-6}$	3.99	1.00
80	$1.14 \cdot 10^{-7}$	3.98	$1.14 \cdot 10^{-7}$	3.98	1.00
160	$7.20 \cdot 10^{-9}$	3.99	$7.20 \cdot 10^{-9}$	3.99	1.00

(d) Errors with a basis made of four elements: $q = 3$.

Table 5: Advection equation: errors, orders of accuracy, and gain obtained when approximating an unsteady solution for bases with and without prior.

where $h > 0$ is the water height, Q the water discharge, $g = 9.81$ the gravity constant, and where the parameterized topography function is

$$Z(x; \alpha, \beta) = \beta\omega\left(\alpha\left(x - \frac{1}{2}\right)\right). \tag{5.7}$$

In (5.7), the function $\omega \in \{\omega_g, \omega_c\}$ is either a Gaussian bump function

$$\omega_g(x) = \frac{1}{4}e^{-50x^2} \tag{5.8}$$

or a compactly supported bump function, with parameter $\eth = 0.15$:

$$\omega_c(x) = \begin{cases} \exp\left(1 - \dfrac{1}{1 - \left(\dfrac{x}{\eth}\right)^2}\right) & \text{if } |x| < \eth, \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

Unless otherwise mentioned, the final physical time is $T = 0.05$, and the space domain is $\Omega = (0, 1)$. For each experiment, Dirichlet boundary conditions corresponding to the steady solution are prescribed.

The steady solutions are given by cancelling the time derivatives in (5.6), and we get the following characterization:

$$Q_{\text{eq}} = \text{constant} =: Q_0 \qquad \text{and} \qquad \left(1 - \frac{Q_0^2}{g h_{\text{eq}}(x; \mu)^3}\right) \partial_x h_{\text{eq}}(x; \mu) + \partial_x Z(x; \alpha, \beta) = 0. \tag{5.10}$$

To solve the nonlinear ODE on h, we impose $h = h_0$ at some point in space. Without loss of generality, we restrict the study to the case $Q_0 > 0$. This leads us to a family of steady solutions with four parameters, and thus a parameter vector μ made of four elements:

$$\mu = \begin{pmatrix} \alpha \\ \beta \\ h_0 \\ Q_0 \end{pmatrix} \in \mathbb{P} \subset \left(\mathbb{R}_+^*\right)^4.$$

Hence, we compute Δt in (5.1) by taking $\lambda = \frac{Q_0}{h_0} + \sqrt{g h_0}$.

Depending on the values of these parameters, the Froude number

$$\mathrm{Fr} = \sqrt{\frac{Q^2}{g h^3}}$$

controls the so-called flow regime for the steady solution. They can be in three distinct regimes: subcritical (Fr < 1 everywhere), supercritical (Fr > 1 everywhere) or transcritical (Fr $= 1$ somewhere in the domain). Each regime has its own parameter space for h_0 and Q_0, described later, but in all cases we take, unless otherwise stated,

$$0.5 \leq \alpha \leq 1.5 \quad ; \qquad 0.5 \leq \beta \leq 1.5. \tag{5.11}$$

To approximate the steady water height within this parameter space, we use a fully-connected PINN with about 4000 trainable parameters. Its result $\widetilde{h}_\theta$ is modified through a boundary function $\mathcal{B}$ that will be defined for each regime. The loss function is once again made only of the steady ODE, and we minimize

$$\mathcal{J}(\theta) = \left\| \left(1 - \frac{Q_0^2}{g \widetilde{h}_\theta(x;\mu)^3} \right) \partial_x \widetilde{h}_\theta(x;\mu) + \partial_x Z(x;\alpha,\beta) \right\|.$$

Training takes about 5 minutes on a dual NVIDIA K80 GPU, and lasts until the loss is about 10^{-4}, depending on the regime.

5.2.1 Subcritical flow

We start with a subcritical flow, where the parameter space for h_0 and Q_0 is:

$$2 \leq h_0 \leq 3 \quad ; \qquad 3 \leq Q_0 \leq 4. \tag{5.12}$$

To strongly enforce the boundary conditions, the prior $\widetilde{h}_\theta$ is obtained as follows from the result h_θ of the PINN:

$$\widetilde{h}_\theta(x;\mu) = \mathcal{B}(h_\theta, x; \mu) = h_0 + Z(x;\alpha,\beta)\, h_\theta(x;\mu). \tag{5.13}$$

To test the preservation of the steady solution, we set the initial water height to h_{eq}.

A goal of this section is to better understand the differences between the two topography functions: the Gaussian bump (5.8) and the compactly supported bump (5.9). It is well-known that compactly supported functions exhibit large derivatives close to the support, see for instance [44]. As a consequence, to get a good approximation of these derivatives when computing integrals involving the PINN, we take $n_Q = q + 6$ when $\omega = \omega_c$. Note that this choice is also motivated by the results in [44], where the authors had to take larger polynomial degrees to observe the correct orders of convergence. The Gaussian topography also suffers from the same drawback, but to a lesser extent, and we take $n_Q = q + 3$ when $\omega = \omega_g$ when integrating the result of the PINN.

For the compactly supported topography, the results are reported in Table 6; for the Gaussian topography, the results are reported in Table 7.

As a conclusion of this first test case, we observe that using a Gaussian topography compared to a compactly supported topography leads to a more stable order of accuracy, but with lower gains, except for small values of K where the compactly supported topography is not well-approximated. The most important point is that the Gaussian topography requires a lower order quadrature to converge. These results are in line with [44]. As a consequence, we use the Gaussian topography in the remainder of this section.

Like in the previous section, we now consider 10^3 parameters in $\mathbb{P}$, and we compute the minimum, average and maximum gains for $q \in \{0, 1, 2\}$. To that end, we take $K = 20$ discretization cells. The results are reported in Table 8, where we observe that the average gains are substantial, whatever the value of q, and that the minimum gain is always greater than 1.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
20	$5.76 \cdot 10^{-2}$	—	$1.89 \cdot 10^{-1}$	—	$6.20 \cdot 10^{-4}$	—	92.82	$2.90 \cdot 10^{-3}$	—	65.38
40	$3.06 \cdot 10^{-2}$	0.91	$1.50 \cdot 10^{-1}$	0.34	$5.65 \cdot 10^{-5}$	3.46	541.59	$3.94 \cdot 10^{-4}$	2.88	380.39
80	$1.82 \cdot 10^{-2}$	0.75	$8.30 \cdot 10^{-2}$	0.85	$3.46 \cdot 10^{-5}$	0.71	525.20	$1.70 \cdot 10^{-4}$	1.21	488.25
160	$9.94 \cdot 10^{-3}$	0.87	$4.53 \cdot 10^{-2}$	0.87	$1.94 \cdot 10^{-5}$	0.84	511.96	$9.28 \cdot 10^{-5}$	0.87	488.31
320	$5.26 \cdot 10^{-3}$	0.92	$2.37 \cdot 10^{-2}$	0.93	$1.04 \cdot 10^{-5}$	0.91	507.63	$4.89 \cdot 10^{-5}$	0.92	484.02

(a) Error with a basis made of one element: $q = 0$.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
20	$2.13 \cdot 10^{-2}$	—	$6.69 \cdot 10^{-2}$	—	$1.05 \cdot 10^{-4}$	—	202.69	$3.96 \cdot 10^{-4}$	—	168.97
40	$3.90 \cdot 10^{-3}$	2.45	$1.37 \cdot 10^{-2}$	2.28	$1.93 \cdot 10^{-5}$	2.44	202.12	$8.14 \cdot 10^{-5}$	2.28	168.62
80	$8.35 \cdot 10^{-4}$	2.22	$2.91 \cdot 10^{-3}$	2.24	$1.59 \cdot 10^{-6}$	3.60	525.18	$7.27 \cdot 10^{-6}$	3.48	399.73
160	$2.04 \cdot 10^{-4}$	2.03	$6.72 \cdot 10^{-4}$	2.11	$3.67 \cdot 10^{-7}$	2.11	556.12	$1.55 \cdot 10^{-6}$	2.23	432.74
320	$5.13 \cdot 10^{-5}$	1.99	$1.65 \cdot 10^{-4}$	2.02	$9.06 \cdot 10^{-8}$	2.02	566.17	$3.62 \cdot 10^{-7}$	2.10	455.78

(b) Error with a basis made of two elements: $q = 1$.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
20	$6.08 \cdot 10^{-3}$	—	$1.89 \cdot 10^{-2}$	—	$1.44 \cdot 10^{-4}$	—	42.26	$6.14 \cdot 10^{-4}$	—	30.75
40	$7.98 \cdot 10^{-4}$	2.93	$2.57 \cdot 10^{-3}$	2.88	$2.52 \cdot 10^{-6}$	5.83	316.56	$7.71 \cdot 10^{-6}$	6.32	333.56
80	$1.05 \cdot 10^{-4}$	2.93	$3.93 \cdot 10^{-4}$	2.71	$2.24 \cdot 10^{-7}$	3.49	467.99	$8.54 \cdot 10^{-7}$	3.17	460.48
160	$1.71 \cdot 10^{-5}$	2.61	$7.02 \cdot 10^{-5}$	2.49	$4.09 \cdot 10^{-8}$	2.45	418.00	$1.76 \cdot 10^{-7}$	2.28	399.05
320	$2.22 \cdot 10^{-6}$	2.94	$1.01 \cdot 10^{-5}$	2.80	$6.02 \cdot 10^{-9}$	2.77	369.32	$2.91 \cdot 10^{-8}$	2.59	345.73

(c) Error with a basis made of three elements: $q = 2$.

Table 6: Shallow water system, compactly supported topography (5.9): errors, orders of accuracy, and gain obtained when approximating a subcritical steady solution for bases with and without prior.

5.2.2 Supercritical flow

We now turn to a supercritical flow. In this case, the remaining parameters h_0 and Q_0 are taken such that:

$$0.5 \leq h_0 \leq 0.75 \quad ; \qquad 4 \leq Q_0 \leq 5. \tag{5.14}$$

The boundary conditions are enforced using the same expression (5.13) as in the subcritical case. We check the approximate preservation of the steady solution by taking the initial water height equal to the steady solution.

The results are displayed in Table 9, and we note that the gains are in line with the subcritical case, from Table 7.

Furthermore, in Table 10, we display some statistics on the gains obtained by using the prior, in the same configuration as for the subcritical regime. We draw similar conclusions to the subcritical case.

5.2.3 Transcritical flow

The last steady experiment we study is the preservation of a transcritical steady solution. Such steady solutions are significantly harder to capture. Indeed, when Fr = 1, the steady ODE (5.10) yields $\partial_x Z = 0$, and therefore the derivative of the steady water height is not defined using only (5.10). This is a well-known issue when approximating transcritical steady solutions, see for instance [14, 24].

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
20	$4.07 \cdot 10^{-2}$	—	$1.65 \cdot 10^{-1}$	—	$9.27 \cdot 10^{-5}$	—	439.14	$3.87 \cdot 10^{-4}$	—	425.63
40	$2.30 \cdot 10^{-2}$	0.83	$1.04 \cdot 10^{-1}$	0.67	$5.85 \cdot 10^{-5}$	0.67	393.05	$2.65 \cdot 10^{-4}$	0.55	391.09
80	$1.26 \cdot 10^{-2}$	0.86	$5.86 \cdot 10^{-2}$	0.82	$3.28 \cdot 10^{-5}$	0.83	384.93	$1.60 \cdot 10^{-4}$	0.73	366.15
160	$6.74 \cdot 10^{-3}$	0.91	$3.13 \cdot 10^{-2}$	0.90	$1.74 \cdot 10^{-5}$	0.91	386.04	$8.72 \cdot 10^{-5}$	0.88	359.19
320	$3.50 \cdot 10^{-3}$	0.95	$1.62 \cdot 10^{-2}$	0.95	$9.27 \cdot 10^{-6}$	0.91	377.48	$4.56 \cdot 10^{-5}$	0.94	356.17

(a) Error with a basis made of one element: $q = 0$.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
20	$3.21 \cdot 10^{-3}$	—	$9.80 \cdot 10^{-3}$	—	$2.37 \cdot 10^{-5}$	—	135.38	$8.94 \cdot 10^{-5}$	—	109.61
40	$7.96 \cdot 10^{-4}$	2.01	$2.35 \cdot 10^{-3}$	2.06	$5.53 \cdot 10^{-6}$	2.10	143.75	$1.89 \cdot 10^{-5}$	2.24	124.54
80	$1.99 \cdot 10^{-4}$	2.00	$5.82 \cdot 10^{-4}$	2.01	$1.36 \cdot 10^{-6}$	2.02	145.47	$4.53 \cdot 10^{-6}$	2.06	128.58
160	$4.96 \cdot 10^{-5}$	2.00	$1.45 \cdot 10^{-4}$	2.00	$3.39 \cdot 10^{-7}$	2.01	146.20	$1.12 \cdot 10^{-6}$	2.02	129.69
320	$1.24 \cdot 10^{-5}$	2.00	$3.63 \cdot 10^{-5}$	2.00	$8.46 \cdot 10^{-8}$	2.00	146.56	$2.79 \cdot 10^{-7}$	2.00	129.97

(b) Error with a basis made of two elements: $q = 1$.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
20	$3.06 \cdot 10^{-4}$	—	$1.23 \cdot 10^{-3}$	—	$3.90 \cdot 10^{-6}$	—	78.49	$1.39 \cdot 10^{-5}$	—	88.29
40	$4.20 \cdot 10^{-5}$	2.86	$1.83 \cdot 10^{-4}$	2.75	$5.87 \cdot 10^{-7}$	2.73	71.56	$2.46 \cdot 10^{-6}$	2.50	74.27
80	$5.44 \cdot 10^{-6}$	2.95	$2.43 \cdot 10^{-5}$	2.91	$8.10 \cdot 10^{-8}$	2.86	67.24	$3.66 \cdot 10^{-7}$	2.75	66.26
160	$6.88 \cdot 10^{-7}$	2.98	$3.09 \cdot 10^{-6}$	2.98	$1.05 \cdot 10^{-8}$	2.95	65.54	$4.86 \cdot 10^{-8}$	2.91	63.52
320	$8.62 \cdot 10^{-8}$	3.00	$3.88 \cdot 10^{-7}$	2.99	$1.33 \cdot 10^{-9}$	2.99	65.05	$6.18 \cdot 10^{-9}$	2.98	62.76

(c) Error with a basis made of three elements: $q = 2$.

Table 7: Shallow water system, Gaussian topography (5.8): errors, orders of accuracy, and gain obtained when approximating a subcritical steady solution for bases with and without prior.

	minimum gain		average gain		maximum gain	
q	h	Q	h	Q	h	Q
0	21.28	17.40	309.84	269.59	1562.20	1628.39
1	7.47	5.47	161.16	129.90	845.97	729.03
2	4.37	5.02	96.54	102.36	707.41	704.55

Table 8: Shallow water system, Gaussian topography (5.8): statistics of the gains obtained for the approximation of a subcritical steady solution in basis V_h^+ with respect to basis V_h.

However, in our case, we provide a simple PINN by defining the water height such that the Froude number is equal to 1 at the top of the topography bump, i.e. at $x = 1/2$, where $\partial_x Z = 0$. When $\mathrm{Fr} = 1$, the water height becomes equal to $h_c(\mu) = Q_0^{2/3} g^{-1/3}$, and we fix this value for the prior evaluated at $x = 1/2$. This eliminates h_0 as a degree of freedom, and we choose $2 \leq Q_0 \leq 3$. Moreover, we take $0.75 \leq \alpha \leq 1.25$ for this regime.

Then, to ensure a correct treatment of the boundary conditions and to obtain the correct value of $\widetilde{h}_\theta$ at the

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
20	$1.25 \cdot 10^{-2}$	—	$4.49 \cdot 10^{-2}$	—	$2.20 \cdot 10^{-5}$	—	566.64	$7.25 \cdot 10^{-5}$	—	619.66
40	$8.37 \cdot 10^{-3}$	0.58	$3.21 \cdot 10^{-2}$	0.48	$1.54 \cdot 10^{-5}$	0.51	542.09	$5.17 \cdot 10^{-5}$	0.49	621.10
80	$5.03 \cdot 10^{-3}$	0.74	$2.02 \cdot 10^{-2}$	0.67	$9.46 \cdot 10^{-6}$	0.71	531.46	$3.20 \cdot 10^{-5}$	0.69	630.14
160	$2.80 \cdot 10^{-3}$	0.84	$1.16 \cdot 10^{-2}$	0.80	$5.18 \cdot 10^{-6}$	0.87	540.85	$1.81 \cdot 10^{-5}$	0.82	638.36
320	$1.49 \cdot 10^{-3}$	0.91	$6.23 \cdot 10^{-3}$	0.89	$2.81 \cdot 10^{-6}$	0.88	529.25	$1.04 \cdot 10^{-5}$	0.80	599.95

(a) Error with a basis made of one element: $q = 0$.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
20	$5.32 \cdot 10^{-4}$	—	$1.96 \cdot 10^{-3}$	—	$4.96 \cdot 10^{-6}$	—	107.22	$1.57 \cdot 10^{-5}$	—	124.92
40	$1.16 \cdot 10^{-4}$	2.19	$4.50 \cdot 10^{-4}$	2.12	$8.11 \cdot 10^{-7}$	2.61	143.39	$3.21 \cdot 10^{-6}$	2.29	140.36
80	$2.80 \cdot 10^{-5}$	2.05	$1.11 \cdot 10^{-4}$	2.03	$1.67 \cdot 10^{-7}$	2.28	167.57	$7.30 \cdot 10^{-7}$	2.14	151.60
160	$6.93 \cdot 10^{-6}$	2.02	$2.76 \cdot 10^{-5}$	2.01	$3.97 \cdot 10^{-8}$	2.07	174.56	$1.78 \cdot 10^{-7}$	2.04	154.79
320	$1.73 \cdot 10^{-6}$	2.00	$6.88 \cdot 10^{-6}$	2.00	$9.82 \cdot 10^{-9}$	2.02	175.99	$4.42 \cdot 10^{-8}$	2.01	155.59

(b) Error with a basis made of two elements: $q = 1$.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
20	$8.33 \cdot 10^{-5}$	—	$2.50 \cdot 10^{-4}$	—	$6.98 \cdot 10^{-7}$	—	119.47	$2.59 \cdot 10^{-6}$	—	96.70
40	$1.33 \cdot 10^{-5}$	2.64	$3.85 \cdot 10^{-5}$	2.70	$1.45 \cdot 10^{-7}$	2.27	92.12	$4.45 \cdot 10^{-7}$	2.54	86.58
80	$1.83 \cdot 10^{-6}$	2.87	$5.21 \cdot 10^{-6}$	2.89	$2.47 \cdot 10^{-8}$	2.55	73.83	$7.19 \cdot 10^{-8}$	2.63	72.48
160	$2.34 \cdot 10^{-7}$	2.96	$6.67 \cdot 10^{-7}$	2.97	$3.50 \cdot 10^{-9}$	2.82	67.04	$1.00 \cdot 10^{-8}$	2.84	66.59
320	$2.95 \cdot 10^{-8}$	2.99	$8.39 \cdot 10^{-8}$	2.99	$4.54 \cdot 10^{-10}$	2.95	65.00	$1.30 \cdot 10^{-9}$	2.95	64.74

(c) Error with a basis made of three elements: $q = 2$.

Table 9: Shallow water system, Gaussian topography (5.8): errors, orders of accuracy, and gain obtained when approximating a supercritical steady solution for bases with and without prior.

	minimum gain		average gain		maximum gain	
q	h	Q	h	Q	h	Q
0	19.83	23.50	309.13	314.36	1789.56	1923.34
1	5.36	5.54	111.41	120.11	354.89	376.47
2	7.29	7.18	123.58	104.49	468.92	381.27

Table 10: Shallow water system, Gaussian topography (5.8): statistics of the gains obtained for the approximation of a supercritical steady solution in basis V_h^+ with respect to basis V_h.

top of the bump, we take

$$\widetilde{h}_\theta(x;\mu) = h_R(\mu) + \left(1 - \tanh\left(15\left(x - \frac{1}{2}\right)\right)\right)\frac{h_L(\mu) - h_R(\mu)}{2}\, h_\theta(x;\mu).$$

In this expression, $h_L(\mu)$ and $h_R(\mu)$ are the left and right boundary conditions. Since we consider a smooth

steady solution, relations (5.10) lead to

$$E(h, x; \mu) := \frac{Q_0^2}{2h^2} + g\big(h + Z(x; \alpha, \beta)\big) = \text{constant}.$$

Since $Z(0.5; \alpha, \beta) = \beta/4$, we obtain that $h_L(\mu) > h_R(\mu)$ are the two solutions of the following equation, with unknown h:

$$\frac{Q_0^2}{2h^2} + gh = E\left(h_c(\mu), \frac{1}{2}; \mu\right).$$

Table 11 contains the errors, order of convergence and the gains. We observe that the gains are lower than in the other two cases, but that was to be expected since the transcritical solution comes from a singular ODE, and it is harder for the PINN to approximate its solutions.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
40	$4.81 \cdot 10^{-2}$	—	$4.29 \cdot 10^{-2}$	—	$1.79 \cdot 10^{-4}$	—	268.84	$2.10 \cdot 10^{-4}$	—	204.54
80	$2.58 \cdot 10^{-2}$	0.90	$2.55 \cdot 10^{-2}$	0.75	$1.37 \cdot 10^{-4}$	0.39	189.00	$1.53 \cdot 10^{-4}$	0.45	165.82
160	$1.34 \cdot 10^{-2}$	0.94	$1.40 \cdot 10^{-2}$	0.86	$9.50 \cdot 10^{-5}$	0.52	141.21	$1.00 \cdot 10^{-4}$	0.61	139.28
320	$6.84 \cdot 10^{-3}$	0.97	$7.35 \cdot 10^{-3}$	0.93	$6.00 \cdot 10^{-5}$	0.66	114.00	$6.03 \cdot 10^{-5}$	0.74	121.99
640	$3.46 \cdot 10^{-3}$	0.98	$3.77 \cdot 10^{-3}$	0.96	$3.56 \cdot 10^{-5}$	0.75	97.20	$3.40 \cdot 10^{-5}$	0.82	110.81

(a) Error with a basis made of one element: $q = 0$.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
40	$6.69 \cdot 10^{-4}$	—	$6.15 \cdot 10^{-4}$	—	$1.85 \cdot 10^{-5}$	—	36.18	$1.16 \cdot 10^{-5}$	—	52.90
80	$1.67 \cdot 10^{-4}$	2.00	$1.53 \cdot 10^{-4}$	2.01	$3.11 \cdot 10^{-6}$	2.57	53.69	$2.11 \cdot 10^{-6}$	2.46	72.26
160	$4.17 \cdot 10^{-5}$	2.00	$3.81 \cdot 10^{-5}$	2.00	$6.77 \cdot 10^{-7}$	2.20	61.66	$4.36 \cdot 10^{-7}$	2.28	87.42
320	$1.04 \cdot 10^{-5}$	2.00	$9.53 \cdot 10^{-6}$	2.00	$1.65 \cdot 10^{-7}$	2.04	63.28	$1.05 \cdot 10^{-7}$	2.06	91.04
640	$2.61 \cdot 10^{-6}$	2.00	$2.38 \cdot 10^{-6}$	2.00	$4.10 \cdot 10^{-8}$	2.01	63.67	$2.59 \cdot 10^{-8}$	2.01	91.90

(b) Error with a basis made of two elements: $q = 1$.

	h, basis V_h		Q, basis V_h		h, basis V_h^+			Q, basis V_h^+		
K	error	order	error	order	error	order	gain	error	order	gain
40	$9.76 \cdot 10^{-5}$	—	$7.21 \cdot 10^{-5}$	—	$2.95 \cdot 10^{-6}$	—	33.10	$1.81 \cdot 10^{-6}$	—	39.88
80	$1.91 \cdot 10^{-5}$	2.35	$1.19 \cdot 10^{-5}$	2.60	$6.89 \cdot 10^{-7}$	2.10	27.74	$3.86 \cdot 10^{-7}$	2.23	30.84
160	$3.25 \cdot 10^{-6}$	2.56	$1.80 \cdot 10^{-6}$	2.72	$1.66 \cdot 10^{-7}$	2.05	19.54	$8.68 \cdot 10^{-8}$	2.15	20.76
320	$5.01 \cdot 10^{-7}$	2.70	$2.51 \cdot 10^{-7}$	2.84	$3.82 \cdot 10^{-8}$	2.12	13.10	$1.78 \cdot 10^{-8}$	2.29	14.16
640	$7.42 \cdot 10^{-8}$	2.76	$3.34 \cdot 10^{-8}$	2.91	$7.74 \cdot 10^{-9}$	2.30	9.58	$3.13 \cdot 10^{-9}$	2.50	10.66

(c) Error with a basis made of three elements: $q = 2$.

Table 11: Shallow water system, Gaussian topography (5.8): errors, orders of accuracy, and gain obtained when approximating a transcritical steady solution for bases with and without prior.

Finally, we report in Table 12 the minimum, average and maximum gains obtained by using the basis V_h^+ instead of the basis V_h. We draw the same conclusions as in the other two regimes, even though the gains are, on average, lower. This was to be expected, since the transcritical regime is harder to capture than the subcritical and supercritical ones, and therefore that the prior is of lower quality. Nevertheless, the gains remain substantial for all values of q.

	minimum gain		average gain		maximum gain	
q	h	Q	h	Q	h	Q
0	35.82	26.19	254.53	177.02	928.03	668.73
1	5.51	4.73	30.83	38.69	134.83	142.11
2	4.55	6.16	16.49	24.29	96.95	109.94

Table 12: Shallow water system, Gaussian topography (5.8): statistics of the gains obtained for the approximation of a transcritical steady solution in basis V_h^+ with respect to basis V_h.

5.2.4 Perturbation of a steady flow

This last experiment related to the shallow water equations concerns a perturbed steady flow. We only perform this study on the subcritical flow, but the other regimes behave the same. We take $\varepsilon \in \{5 \cdot 10^{-k}\}_{k\in\{1,2,3\}}$ and 20 space cells, and set the initial water height to $h(0,x;\mu) = \big(1+\varepsilon\sin(2\pi x)\big)h_{\text{eq}}(x;\mu)$.

The errors on h with respect to time are displayed in Figure 2, until the final physical time $T=1$. Like in Section 5.1, with the prior, the error decreases to a much lower level than without the prior. This good behavior was expected since the prior makes it possible for the enhanced DG scheme to achieve higher accuracy on steady solutions.

5.3 Euler-Poisson equations in spherical geometry

We now consider the Euler-Poisson equations in spherical geometry. This system is used in astrophysics, for instance, where it serves to model stars held together by gravitation, see e.g. [15, 17, 32]. They are given by

$$\begin{cases} \partial_t \rho + \partial_r Q = -\dfrac{2}{r}Q, \\ \partial_t Q + \partial_r\left(\dfrac{Q^2}{\rho}+p\right) = -\dfrac{2}{r}\dfrac{Q^2}{\rho} - \rho\partial_r\phi, \\ \partial_t E + \partial_r\left(\dfrac{Q}{\rho}(E+p)\right) = -\dfrac{2}{r}\dfrac{Q}{\rho}(E+p) - Q\partial_r\phi, \\ \dfrac{1}{r^2}\partial_{rr}(r^2\phi) = 4\pi G\rho, \end{cases} \tag{5.15}$$

where G is a gravity constant, fixed to $G=1$ in our applications, and where we take p as a function of ρ, Q and E through a pressure law to be specified. In (5.15), ρ is the density, Q is the momentum, E is the energy, and ϕ is the gravitational potential. Unless otherwise mentioned, the boundary conditions are of Dirichlet type, with the value of the steady solution prescribed of the boundaries.

The space domain is $r \in (0,1)$. The apparent singularity at $r=0$ is resolved by imposing suitable boundary conditions, namely $\rho(0)=1$ and $\partial_r\rho(0)$ given according to the pressure law. Indeed, the assumption that there is no gravity at $r=0$ leads to $\partial_r p(0)=0$, which makes it possible to determine $\partial_r\rho(0)$. For more information on the boundary conditions and on the DG discretization of (5.15), the reader is referred to [57].

The steady solutions at rest are given by

$$\begin{cases} Q = 0, \\ \partial_r p + \rho\partial_r\phi = 0, \\ \partial_{rr}(r^2\phi) = 4\pi r^2 G\rho. \end{cases}$$

For the steady solutions, we shall distinguish two cases for the pressure law: a polytropic pressure law, and a temperature-dependent pressure law.

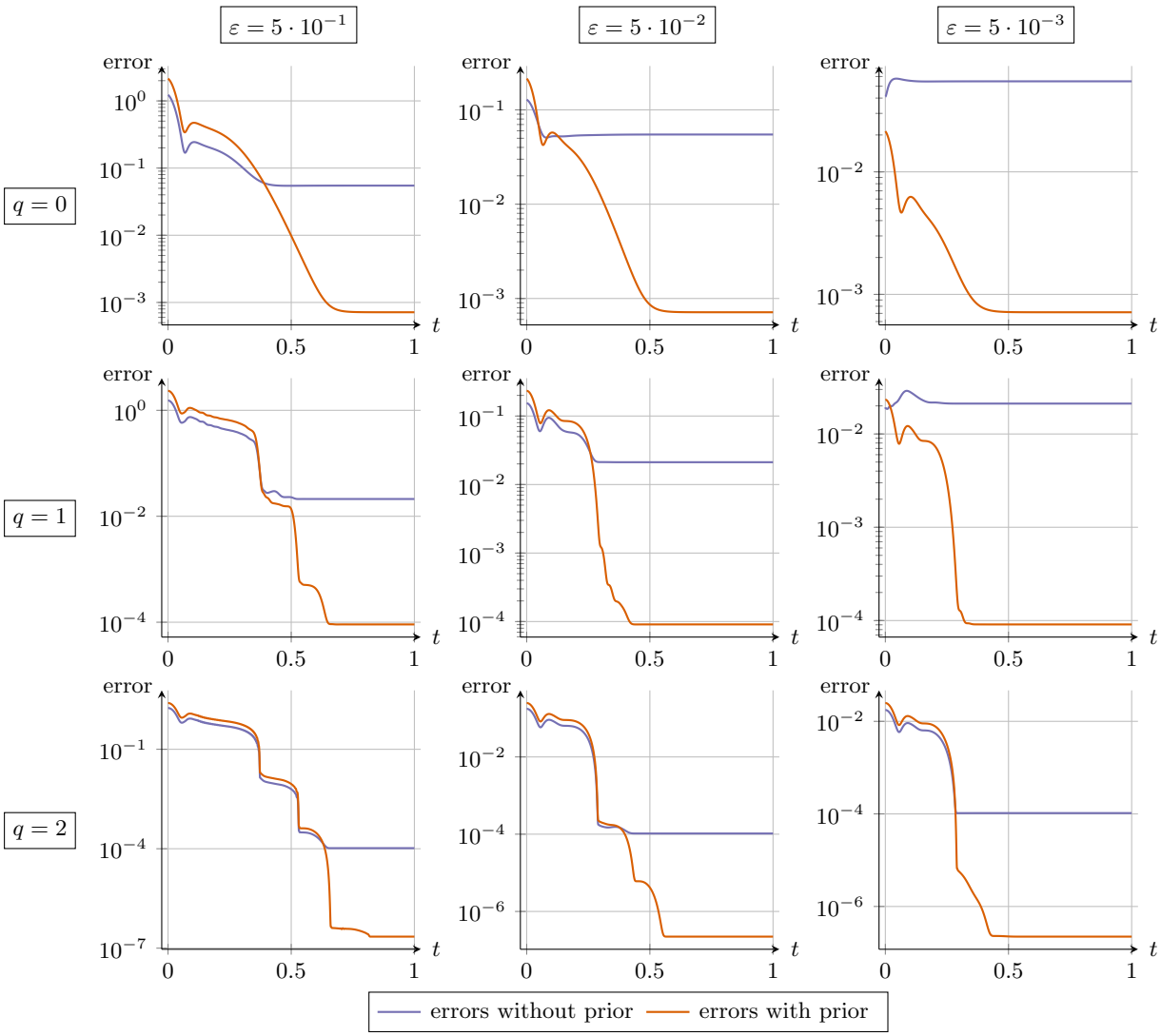

Figure 2: Shallow water equations, compactly supported topography: errors, with respect to time, for the approximation of a perturbed subcritical steady solution for bases with and without prior.

5.3.1 Polytropic pressure law

In this case, we introduce two parameters κ and γ, so the parameter vector μ is composed of two elements:

$$\mu = \begin{pmatrix} \kappa \\ \gamma \end{pmatrix} \in \mathbb{P} \subset \mathbb{R}^2, \quad \kappa \in \mathbb{R}_+, \quad \gamma \in (1, +\infty).$$

Equipped with this parameter vector, we define the polytropic pressure law

$$p(\rho; \mu) = \kappa \rho^\gamma,$$

and the steady solutions are then given as solutions to the following nonlinear second-order ordinary differential equation:

$$\frac{d}{dr}\left(r^2 \kappa \gamma \rho^{\gamma-2} \frac{d\rho}{dr}\right) = 4\pi r^2 G \rho.$$

In general, this ODE does not have analytic solutions. However, it turns out that, for specific values of γ, there exists an analytic solution to this ODE. For instance, with $\gamma = 2$, we obtain

$$\rho(r) = \frac{\sin(\alpha r)}{\alpha r}, \qquad \text{with} \quad \alpha = \sqrt{\frac{2\pi G}{\kappa}}.$$

Regarding the boundary conditions, the condition $\partial_r p(0) = 0$ leads to $\partial_r \rho(0) = 0$ for this pressure law. In this case, we take $\lambda = 1 + \sqrt{\gamma}$ in (5.1) to compute the time step Δt.

To obtain a prior ρ_θ, as usual, we train a PINN with about 1400 trainable parameters on 7 fully connected layers. The boundary conditions are taken into account by setting

$$\widetilde{\rho}_\theta(r;\mu) = 1 + r^2 \rho_\theta(r;\mu),$$

where ρ_θ is the result of the PINN. The PINN is trained on the parameter space

$$\mathbb{P} = [2,5] \times [1.5, 3.5], \tag{5.16}$$

with only the physics-based loss function corresponding to the steady solution:

$$\mathcal{L}_\theta = \left\| \frac{d}{dr}\left(r^2 \kappa \gamma \widetilde{\rho}_\theta^{\gamma-2} \frac{d\widetilde{\rho}_\theta}{dr} \right) - 4\pi r^2 G \widetilde{\rho}_\theta \right\|.$$

In addition, the prior for q is set to $Q_\theta = 1$ since we wish to approximate a constant momentum. Finally, the prior for E is set to $E_\theta = p(\widetilde{\rho}_\theta;\mu)/(\gamma - 1)$. Training takes about 5 minutes on a dual NVIDIA K80 GPU, until the loss is equal to about $5 \cdot 10^{-5}$. In the DG discretization, the degree of the quadrature formula is the usual $n_Q = q + 2$: there is no need to further increase the order of the quadrature rule in this case.

We first collect, in Table 13, the results of the approximation in both bases (with and without prior), for $\kappa = 2$ and $\gamma = 2.5$, and until the final time $T = 0.01$. As usual, the observed gain is larger for smaller number of basis elements. We observe a slight superconvergence on the momentum Q when using the prior with $q = 0$. For these values of κ and γ, gains on the density are not very large for $q = 2$, but this is compensated by larger gains on the energy.

To extend this study, we compute the statistics over the whole parameter space (5.16) by uniformly sampling 10^3 values and taking 10 cells in the mesh. The results are reported in Table 14. Just like before, the average gain is substantial, while the minimum rarely falls below 1. Moreover, note that the gains recorded in Table 13 correspond to a rather bad set of parameters compared to the average.

5.3.2 Temperature-dependent pressure law

In this case, we take a given smooth temperature function $T(r,\mu)$ parameterized by μ, where the parameter vector μ is composed of two elements:

$$\mu = \begin{pmatrix} \kappa \\ \alpha \end{pmatrix} \in \mathbb{P} \subset \mathbb{R}^2, \quad \kappa \in \mathbb{R}_+, \quad \alpha \in \mathbb{R}_+.$$

This allows us to define the parameterized temperature function $T(r;\alpha) = e^{-\alpha r}$, and so we get the following temperature-based pressure law:

$$p(\rho;\mu) = \kappa \rho T.$$

For this pressure law, the steady solutions are given by the following nonlinear second-order ODE:

$$\frac{d}{dr}\left(r^2 \kappa \frac{T}{\rho} \frac{d\rho}{dr} \right) + \frac{d}{dr}\left(r^2 \kappa \frac{dT}{dr} \right) = 4\pi r^2 G \rho,$$

and the boundary condition $\partial_r p(0) = 0$ leads to $\partial_r \rho(0) = \alpha$. For this pressure law, we also take $\lambda = 1 + \sqrt{\gamma}$ in (5.1) to compute Δt.

	ρ, basis V_h		Q, basis V_h		E, basis V_h		ρ, basis V_h^+			Q, basis V_h^+			E, basis V_h^+		
K	error	order	error	order	error	order	error	order	gain	error	order	gain	error	order	gain
10	$3.37 \cdot 10^{-2}$	—	$2.60 \cdot 10^{-3}$	—	$7.55 \cdot 10^{-2}$	—	$1.08 \cdot 10^{-4}$	—	312.50	$8.94 \cdot 10^{-4}$	—	2.91	$3.43 \cdot 10^{-4}$	—	219.99
20	$1.69 \cdot 10^{-2}$	1.00	$1.51 \cdot 10^{-3}$	0.79	$3.78 \cdot 10^{-2}$	1.00	$6.49 \cdot 10^{-5}$	0.73	259.80	$3.34 \cdot 10^{-4}$	1.42	4.51	$2.10 \cdot 10^{-4}$	0.71	180.21
40	$8.44 \cdot 10^{-3}$	1.00	$8.27 \cdot 10^{-4}$	0.87	$1.89 \cdot 10^{-2}$	1.00	$3.41 \cdot 10^{-5}$	0.93	247.75	$1.06 \cdot 10^{-4}$	1.65	7.78	$1.11 \cdot 10^{-4}$	0.93	171.17
80	$4.22 \cdot 10^{-3}$	1.00	$4.60 \cdot 10^{-4}$	0.85	$9.46 \cdot 10^{-3}$	1.00	$1.72 \cdot 10^{-5}$	0.99	246.26	$3.24 \cdot 10^{-5}$	1.71	14.20	$5.57 \cdot 10^{-5}$	0.99	169.99
160	$2.11 \cdot 10^{-3}$	1.00	$2.59 \cdot 10^{-4}$	0.83	$4.73 \cdot 10^{-3}$	1.00	$8.17 \cdot 10^{-6}$	1.07	258.59	$9.15 \cdot 10^{-6}$	1.82	28.25	$2.64 \cdot 10^{-5}$	1.07	178.96

(a) Errors with a basis made of one element: $q = 0$.

	ρ, basis V_h		Q, basis V_h		E, basis V_h		ρ, basis V_h^+			Q, basis V_h^+			E, basis V_h^+		
K	error	order	error	order	error	order	error	order	gain	error	order	gain	error	order	gain
10	$1.03 \cdot 10^{-3}$	—	$1.30 \cdot 10^{-3}$	—	$2.23 \cdot 10^{-3}$	—	$1.06 \cdot 10^{-5}$	—	96.73	$1.11 \cdot 10^{-5}$	—	117.58	$3.29 \cdot 10^{-5}$	—	67.74
20	$2.57 \cdot 10^{-4}$	2.00	$4.17 \cdot 10^{-4}$	1.64	$5.61 \cdot 10^{-4}$	1.99	$2.74 \cdot 10^{-6}$	1.95	93.71	$4.33 \cdot 10^{-6}$	1.36	96.42	$8.44 \cdot 10^{-6}$	1.96	66.43
40	$6.43 \cdot 10^{-5}$	2.00	$1.11 \cdot 10^{-4}$	1.91	$1.41 \cdot 10^{-4}$	2.00	$6.96 \cdot 10^{-7}$	1.98	92.37	$1.25 \cdot 10^{-6}$	1.79	88.87	$2.14 \cdot 10^{-6}$	1.98	65.80
80	$1.61 \cdot 10^{-5}$	2.00	$2.80 \cdot 10^{-5}$	1.99	$3.51 \cdot 10^{-5}$	2.00	$1.74 \cdot 10^{-7}$	2.00	92.17	$3.22 \cdot 10^{-7}$	1.96	86.81	$5.34 \cdot 10^{-7}$	2.00	65.70
160	$4.01 \cdot 10^{-6}$	2.00	$6.99 \cdot 10^{-6}$	2.00	$8.74 \cdot 10^{-6}$	2.01	$4.33 \cdot 10^{-8}$	2.01	92.65	$8.10 \cdot 10^{-8}$	1.99	86.38	$1.32 \cdot 10^{-7}$	2.01	65.99

(b) Errors with a basis made of two elements: $q = 1$.

	ρ, basis V_h		Q, basis V_h		E, basis V_h		ρ, basis V_h^+			Q, basis V_h^+			E, basis V_h^+		
K	error	order	error	order	error	order	error	order	gain	error	order	gain	error	order	gain
10	$1.38 \cdot 10^{-6}$	—	$1.74 \cdot 10^{-6}$	—	$4.38 \cdot 10^{-5}$	—	$8.88 \cdot 10^{-7}$	—	1.55	$9.49 \cdot 10^{-7}$	—	1.83	$2.74 \cdot 10^{-6}$	—	15.97
20	$1.85 \cdot 10^{-7}$	2.90	$4.05 \cdot 10^{-7}$	2.10	$5.74 \cdot 10^{-6}$	2.93	$1.25 \cdot 10^{-7}$	2.83	1.48	$2.58 \cdot 10^{-7}$	1.88	1.57	$3.58 \cdot 10^{-7}$	2.94	16.01
40	$2.89 \cdot 10^{-8}$	2.68	$5.84 \cdot 10^{-8}$	2.79	$7.40 \cdot 10^{-7}$	2.95	$1.79 \cdot 10^{-8}$	2.81	1.62	$3.31 \cdot 10^{-8}$	2.96	1.76	$4.80 \cdot 10^{-8}$	2.90	15.43
80	$3.52 \cdot 10^{-9}$	3.04	$6.95 \cdot 10^{-9}$	3.07	$9.26 \cdot 10^{-8}$	3.00	$2.24 \cdot 10^{-9}$	2.99	1.57	$4.25 \cdot 10^{-9}$	2.96	1.63	$5.96 \cdot 10^{-9}$	3.01	15.55
160	$4.45 \cdot 10^{-10}$	2.98	$8.87 \cdot 10^{-10}$	2.97	$1.16 \cdot 10^{-8}$	3.00	$2.83 \cdot 10^{-10}$	2.98	1.57	$5.50 \cdot 10^{-10}$	2.95	1.61	$7.47 \cdot 10^{-10}$	2.99	15.50

(c) Errors with a basis made of three elements: $q = 2$.

Table 13: Euler-Poisson system, polytropic pressure law: errors, orders of accuracy, and gain obtained when approximating a steady solution for bases with and without prior.

	minimum gain			average gain			maximum gain		
q	ρ	Q	E	ρ	Q	E	ρ	Q	E
0	19.14	2.33	17.04	233.48	3.73	197.28	510.42	4.48	371.87
1	7.61	8.28	6.98	158.25	188.92	130.57	1095.68	1291.90	1024.59
2	0.14	0.22	2.99	12.11	16.55	23.73	89.47	109.93	169.28

Table 14: Euler-Poisson system, polytropic pressure law: statistics of the gains obtained for the approximation of a steady solution in basis V_h^+ with respect to basis V_h.

The prior ρ_θ is obtained *via* a PINN with the same characteristics as in the polytropic case, and whose result is still denoted by ρ_θ. To impose the boundary conditions, this time, we set

$$\widetilde{\rho}_\theta(r;\mu) = 1 + \alpha r + r^2 \rho_\theta(r;\mu).$$

The parameter space is

$$\mathbb{P} = [2,5] \times [0.5, 1.5], \tag{5.17}$$

and the PINN is trained using only the physics-based loss function

$$\mathcal{L}_\theta = \left\| \frac{d}{dr}\left(r^2 \kappa \frac{T}{\widetilde{\rho}_\theta} \frac{d\widetilde{\rho}_\theta}{dr} \right) + \frac{d}{dr}\left(r^2 \kappa \frac{dT}{dr} \right) = 4\pi r^2 G \widetilde{\rho}_\theta \right\|.$$

Training takes about 5 minutes on a dual NVIDIA K80 GPU, until the loss is equal to about $5 \cdot 10^{-4}$. The priors Q_θ and E_θ are then defined in the same way as in the polytropic case. In this case, we also take $n_Q = q+2$.

As is becoming usual, we first report, in Table 15, the results of the approximation in both bases (with and without prior). The final time is set to $T = 0.01$, and we take $\kappa = 3.5$ and $\alpha = 0.5$. As usual, using the prior provides significant gains, especially for low values of q. Compared to the polytropic case, gains are consistently better for the large values of q.

	ρ, basis V_h		Q, basis V_h		E, basis V_h		ρ, basis V_h^+			Q, basis V_h^+			E, basis V_h^+		
K	error	order	error	order	error	order	error	order	gain	error	order	gain	error	order	gain
10	$3.91 \cdot 10^{-2}$	—	$2.80 \cdot 10^{-3}$	—	$1.56 \cdot 10^{-1}$	—	$2.65 \cdot 10^{-4}$	—	147.33	$1.18 \cdot 10^{-3}$	—	2.37	$8.41 \cdot 10^{-4}$	—	186.02
20	$1.96 \cdot 10^{-2}$	1.00	$1.66 \cdot 10^{-3}$	0.76	$7.83 \cdot 10^{-2}$	1.00	$1.39 \cdot 10^{-4}$	0.93	140.95	$4.18 \cdot 10^{-4}$	1.50	3.97	$4.96 \cdot 10^{-4}$	0.76	157.73
40	$9.81 \cdot 10^{-3}$	1.00	$9.02 \cdot 10^{-4}$	0.88	$3.92 \cdot 10^{-2}$	1.00	$7.04 \cdot 10^{-5}$	0.98	139.37	$1.23 \cdot 10^{-4}$	1.77	7.35	$2.58 \cdot 10^{-4}$	0.94	151.77
80	$4.91 \cdot 10^{-3}$	1.00	$5.30 \cdot 10^{-4}$	0.77	$1.96 \cdot 10^{-2}$	1.00	$3.61 \cdot 10^{-5}$	0.96	135.81	$3.85 \cdot 10^{-5}$	1.67	13.75	$1.41 \cdot 10^{-4}$	0.87	138.76
160	$2.46 \cdot 10^{-3}$	1.00	$2.94 \cdot 10^{-4}$	0.85	$9.80 \cdot 10^{-3}$	1.00	$1.80 \cdot 10^{-5}$	1.00	136.36	$1.09 \cdot 10^{-5}$	1.82	26.86	$6.98 \cdot 10^{-5}$	1.02	140.52

(a) Errors with a basis made of one element: $q = 0$.

	ρ, basis V_h		Q, basis V_h		E, basis V_h		ρ, basis V_h^+			Q, basis V_h^+			E, basis V_h^+		
K	error	order	error	order	error	order	error	order	gain	error	order	gain	error	order	gain
10	$1.89 \cdot 10^{-3}$	—	$2.19 \cdot 10^{-3}$	—	$4.61 \cdot 10^{-3}$	—	$2.85 \cdot 10^{-5}$	—	66.41	$2.67 \cdot 10^{-5}$	—	81.99	$8.80 \cdot 10^{-5}$	—	52.42
20	$4.74 \cdot 10^{-4}$	2.00	$7.95 \cdot 10^{-4}$	1.46	$1.16 \cdot 10^{-3}$	1.99	$7.46 \cdot 10^{-6}$	1.93	63.53	$1.16 \cdot 10^{-5}$	1.21	68.68	$2.39 \cdot 10^{-5}$	1.88	48.51
40	$1.19 \cdot 10^{-4}$	2.00	$2.31 \cdot 10^{-4}$	1.79	$2.90 \cdot 10^{-4}$	2.00	$1.92 \cdot 10^{-6}$	1.96	61.82	$3.68 \cdot 10^{-6}$	1.65	62.63	$6.28 \cdot 10^{-6}$	1.93	46.22
80	$2.96 \cdot 10^{-5}$	2.00	$6.04 \cdot 10^{-5}$	1.93	$7.24 \cdot 10^{-5}$	2.00	$4.83 \cdot 10^{-7}$	1.99	61.27	$1.00 \cdot 10^{-6}$	1.88	60.25	$1.58 \cdot 10^{-6}$	2.00	45.96
160	$7.40 \cdot 10^{-6}$	2.00	$1.51 \cdot 10^{-5}$	2.00	$1.80 \cdot 10^{-5}$	2.01	$1.20 \cdot 10^{-7}$	2.00	61.43	$2.54 \cdot 10^{-7}$	1.98	59.61	$3.80 \cdot 10^{-7}$	2.05	47.45

(b) Errors with a basis made of two elements: $q = 1$.

	ρ, basis V_h		Q, basis V_h		E, basis V_h		ρ, basis V_h^+			Q, basis V_h^+			E, basis V_h^+		
K	error	order	error	order	error	order	error	order	gain	error	order	gain	error	order	gain
10	$3.83 \cdot 10^{-5}$	—	$4.49 \cdot 10^{-5}$	—	$1.27 \cdot 10^{-4}$	—	$2.77 \cdot 10^{-6}$	—	13.83	$3.75 \cdot 10^{-6}$	—	11.98	$8.95 \cdot 10^{-6}$	—	14.20
20	$5.71 \cdot 10^{-6}$	2.75	$8.25 \cdot 10^{-6}$	2.44	$2.67 \cdot 10^{-5}$	2.25	$4.88 \cdot 10^{-7}$	2.50	11.70	$7.62 \cdot 10^{-7}$	2.30	10.82	$2.03 \cdot 10^{-6}$	2.14	13.14
40	$7.37 \cdot 10^{-7}$	2.95	$8.72 \cdot 10^{-7}$	3.24	$3.66 \cdot 10^{-6}$	2.87	$7.19 \cdot 10^{-8}$	2.76	10.25	$9.64 \cdot 10^{-8}$	2.98	9.05	$3.07 \cdot 10^{-7}$	2.73	11.93
80	$8.88 \cdot 10^{-8}$	3.05	$1.09 \cdot 10^{-7}$	3.00	$4.48 \cdot 10^{-7}$	3.03	$8.89 \cdot 10^{-9}$	3.02	9.99	$1.14 \cdot 10^{-8}$	3.08	9.55	$3.85 \cdot 10^{-8}$	2.99	11.64
160	$1.11 \cdot 10^{-8}$	3.00	$1.36 \cdot 10^{-8}$	3.01	$5.61 \cdot 10^{-8}$	3.00	$1.14 \cdot 10^{-9}$	2.96	9.74	$1.47 \cdot 10^{-9}$	2.96	9.23	$4.96 \cdot 10^{-9}$	2.96	11.31

(c) Errors with a basis made of three elements: $q = 2$.

Table 15: Euler-Poisson system, temperature-based pressure law: errors, orders of accuracy, and gain obtained when approximating a steady solution for bases with and without prior.

To understand gains on the whole parameter space (5.17), we uniformly sample 10^3 values of κ and α and take a mesh made of 10 cells. We compute the minimum, average and maximum gains. These values are reported in Table 16. For this pressure law, the minimum gain is always larger than 1, and we obtain consistently large average gains, even for $q = 2$.

5.3.3 Spherical blast wave

The goal of this last test case is to show that our prior does not negatively affect the capability of the scheme to capture discontinuous solutions. Let us emphasize that numerical viscosity is not an object of this study, and

	minimum gain			average gain			maximum gain		
q	ρ	Q	E	ρ	Q	E	ρ	Q	E
0	13.30	1.05	16.24	151.96	1.88	150.63	600.13	2.91	473.83
1	6.30	7.53	5.40	72.63	77.20	51.09	321.20	302.58	257.19
2	3.35	3.45	2.20	18.96	22.58	13.56	55.47	63.45	47.83

Table 16: Euler-Poisson system, temperature-based pressure law: statistics of the gains obtained for the approximation of a steady solution in basis V_h^+ with respect to basis V_h.

therefore that we have not used any regularization procedure. Consequently, results will show some oscillations.

This experiment is nothing but a Riemann problem in spherical geometry, inspired by the experiments in [50]. As such, the initial condition is piecewise constant on the space domain $r \in (0, 0.4)$, as follows:

$$\rho(0,x) = \begin{cases} 2 & \text{if } r < 0.2, \\ 1 & \text{otherwise;} \end{cases} \qquad Q(0,x) = 0; \qquad p(0,x) = \begin{cases} 2 & \text{if } r < 0.2, \\ 1 & \text{otherwise.} \end{cases}$$

For this experiment, the pressure law is the standard ideal gas law

$$p = (\gamma - 1)\left(E - \frac{1}{2}\frac{Q^2}{\rho}\right),$$

and we take the gas constant γ equal to 1.4. The experiment is run until the final time $T = 0.1$, and with Neumann boundary conditions. We take 25 discretization cells, and we use a basis made of 3 elements. Moreover, the source term is deactivated: we set $\phi = 0$, and we merely consider the Euler equations in spherical geometry, without gravity effects.

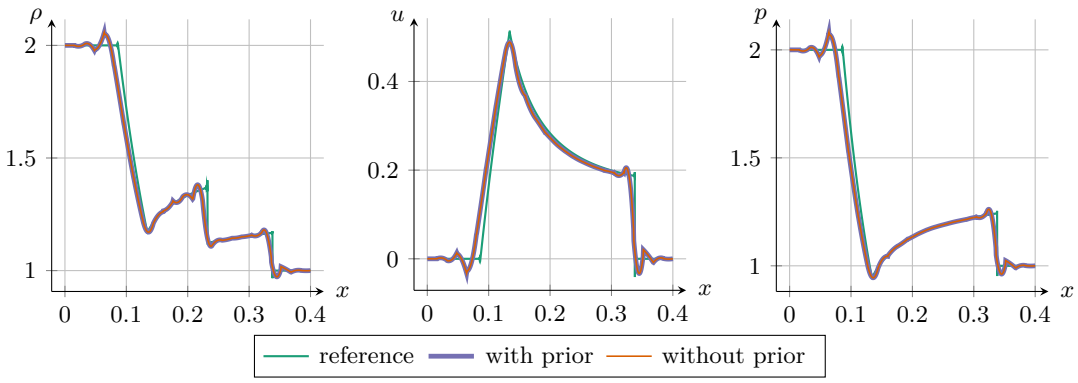

Figure 3: Euler equations, temperature-based pressure law: statistics for the approximation of a steady solution for bases with and without prior.

The results are depicted in Figure 3, where we compare the two bases (with and without prior, blue and orange lines respectively) to a reference solution (green line). We observe very good agreement with the reference solution, even though oscillations are present, as expected. We also note that the graphs for the solutions with and without prior are superimposed, which means that the quality of the approximation of this discontinuous solution has not been degraded by the introduction of the prior in the basis.

5.4 Shallow water equations in two space dimensions

The last system considered in this series of experiments is the two-dimensional shallow water system. It is given by

$$\begin{cases} \partial_t h + \boldsymbol{\nabla} \cdot \boldsymbol{Q} = 0, \\ \partial_t \boldsymbol{Q} + \boldsymbol{\nabla} \cdot \left(\dfrac{\boldsymbol{Q} \otimes \boldsymbol{Q}}{h} + \dfrac{1}{2} g h^2 \mathrm{Id} \right) = -gh \boldsymbol{\nabla} Z(\boldsymbol{x}; \mu), \end{cases}$$

where g is the gravity constant, Id is the 2×2 identity matrix, h is the water height, $\boldsymbol{Q}$ is the water discharge, and Z is the topography. For this system, Δt is computed by setting $\lambda = 2 + \sqrt{\gamma}$ in (5.1).

The space variable $\boldsymbol{x} = (x_1, x_2)$ belongs to the space domain $\Omega = [-3, 3]^2$, and we introduce three parameters:

$$\mu = \begin{pmatrix} \alpha \\ \Gamma \\ r_0 \end{pmatrix} \in \mathbb{P} \subset \mathbb{R}^3, \quad \alpha \in \mathbb{R}_+^*, \quad \Gamma \in \mathbb{R}_+^*, \quad r_0 \in \mathbb{R}_+^*.$$

This enables us to define the topography as the following Gaussian bump function, with $r = \|\boldsymbol{x}\|$:

$$Z(\boldsymbol{x}; \mu) = \Gamma \exp\left(\alpha(r_0^2 - r^2)\right),$$

see for instance [43] for a similar test case. On this topography, we consider the following steady solution:

$$\begin{cases} h_{\mathrm{eq}}(\boldsymbol{x}; \mu) = 2 - Z(\boldsymbol{x}; \mu) - \dfrac{\Gamma}{8\alpha g} Z(\boldsymbol{x}; \mu)^4, \\ \boldsymbol{Q}_{\mathrm{eq}}(\boldsymbol{x}; \mu) = -\boldsymbol{x}^\perp \, h_{\mathrm{eq}}(\boldsymbol{x}; \mu) \, u_{\mathrm{eq}}(\boldsymbol{x}; \mu), \end{cases} \tag{5.18}$$

with $\boldsymbol{x}^\perp = (-x_2, x_1)$ and where

$$u_{\mathrm{eq}}(\boldsymbol{x}; \mu) = \alpha Z(\boldsymbol{x}; \mu)^2.$$

To obtain a relevant prior, we approximate h_{eq} and u_{eq}, using a different PINN for each of the two functions. The results of the PINN are denoted by h_θ and u_θ, and we define the priors $\widetilde{h}_\theta$ and $\widetilde{u}_\theta$ as follows, to include the boundary conditions:

$$\widetilde{h}_\theta(\boldsymbol{x}; \mu) = 2 - Z(\boldsymbol{x}; \mu) \, h_\theta(\boldsymbol{x}; \mu)^2 \quad \text{and} \quad \widetilde{u}_\theta(\boldsymbol{x}; \mu) = Z(\boldsymbol{x}; \mu) \, u_\theta(\boldsymbol{x}; \mu).$$

Another possibility would be to strongly impose the divergence-free constraint, by learning a potential and taking the prior $\boldsymbol{Q}_\theta(\boldsymbol{x}; \mu)$ as the curl of this potential. However, we elected not to do so, since the current strategy was able to train faster. The parameter space is

$$\mathbb{P} = [0.25, 0.75] \times [0.1, 0.4] \times [0.5, 1.25].$$

The loss function is made in equal parts of the now usual PDE loss, and of the minimization with respect to data. Data is regenerated at each epoch, and helps to avoid falling in a local minimum corresponding to a lake at rest, where $\widetilde{h}_\theta + Z = \text{constant}$ and $u_\theta = 0$. Each PINN has about 2500 parameters, and training takes about 10 minutes on an NVIDIA V100 GPU, until the loss functions reaches about 4×10^{-7}. This prior is integrated with a quadrature formula of degree $n_Q = q + 3$: we needed to increase the usual quadrature degree by 1 to obtain the best possible approximation.

5.4.1 Approximation of a steady solution

We take the steady solution (5.18) as the initial condition to test the approximate well-balanced property. The experiments are run until the final physical time $T = 0.01$. We prescribe Dirichlet boundary conditons consisting in the value of the steady solution.

First, we take the parameters as the center of the parameter cube $\mathbb{P}$. The results are collected in Table 17, and we note that, as expected, the presence of the prior makes it possible to reach much lower errors, especially for the water height h.

In addition, we provide some statistics over the whole parameter space $\mathbb{P}$, computed on a mesh with 25×25 cells, in Table 18. We not that, on average, the gains are substantial. However, note that the minimum gains may be smaller than 1, which denotes a loss of precision due to the prior. This happens in around 0.75% of cases, so we obtain an improvement in an overwhelming majority of cases.

	h, basis V_h		Q_1, basis V_h		Q_2, basis V_h		h, basis V_h^+			Q_1, basis V_h^+			Q_2, basis V_h^+		
K	error	order	error	order	error	order	error	order	gain	error	order	gain	error	order	gain
20	$1.94 \cdot 10^{-1}$	—	$4.31 \cdot 10^{-2}$	—	$4.31 \cdot 10^{-2}$	—	$1.31 \cdot 10^{-4}$	—	1477.51	$6.24 \cdot 10^{-3}$	—	6.91	$6.24 \cdot 10^{-3}$	—	6.91
40	$9.75 \cdot 10^{-2}$	0.99	$2.19 \cdot 10^{-2}$	0.98	$2.19 \cdot 10^{-2}$	0.98	$6.37 \cdot 10^{-5}$	1.04	1531.52	$2.84 \cdot 10^{-3}$	1.14	7.69	$2.84 \cdot 10^{-3}$	1.14	7.69
80	$4.88 \cdot 10^{-2}$	1.00	$1.09 \cdot 10^{-2}$	1.00	$1.09 \cdot 10^{-2}$	1.00	$3.17 \cdot 10^{-5}$	1.01	1540.17	$1.43 \cdot 10^{-3}$	0.99	7.63	$1.43 \cdot 10^{-3}$	0.99	7.63
160	$2.44 \cdot 10^{-2}$	1.00	$5.48 \cdot 10^{-3}$	1.00	$5.48 \cdot 10^{-3}$	1.00	$1.59 \cdot 10^{-5}$	1.00	1539.94	$7.21 \cdot 10^{-4}$	0.99	7.60	$7.21 \cdot 10^{-4}$	0.99	7.60
320	$1.22 \cdot 10^{-2}$	1.00	$2.74 \cdot 10^{-3}$	1.00	$2.74 \cdot 10^{-3}$	1.00	$7.93 \cdot 10^{-6}$	1.00	1539.59	$3.61 \cdot 10^{-4}$	1.00	7.58	$3.61 \cdot 10^{-4}$	1.00	7.58

(a) Errors with a basis made of one element: $q = 0$.

	h, basis V_h		Q_1, basis V_h		Q_2, basis V_h		h, basis V_h^+			Q_1, basis V_h^+			Q_2, basis V_h^+		
K	error	order	error	order	error	order	error	order	gain	error	order	gain	error	order	gain
20	$2.17 \cdot 10^{-2}$	—	$2.58 \cdot 10^{-2}$	—	$2.58 \cdot 10^{-2}$	—	$8.51 \cdot 10^{-5}$	—	254.60	$1.42 \cdot 10^{-3}$	—	18.21	$1.42 \cdot 10^{-3}$	—	18.21
40	$5.46 \cdot 10^{-3}$	1.99	$8.88 \cdot 10^{-3}$	1.54	$8.88 \cdot 10^{-3}$	1.54	$3.23 \cdot 10^{-5}$	1.40	169.11	$3.70 \cdot 10^{-4}$	1.94	23.99	$3.70 \cdot 10^{-4}$	1.94	23.99
80	$1.37 \cdot 10^{-3}$	2.00	$2.50 \cdot 10^{-3}$	1.83	$2.50 \cdot 10^{-3}$	1.83	$9.43 \cdot 10^{-6}$	1.78	145.10	$9.35 \cdot 10^{-5}$	1.98	26.74	$9.35 \cdot 10^{-5}$	1.98	26.74
160	$3.42 \cdot 10^{-4}$	2.00	$6.46 \cdot 10^{-4}$	1.95	$6.46 \cdot 10^{-4}$	1.95	$2.47 \cdot 10^{-6}$	1.94	138.89	$2.35 \cdot 10^{-5}$	2.00	27.54	$2.35 \cdot 10^{-5}$	2.00	27.54
320	$8.56 \cdot 10^{-5}$	2.00	$1.62 \cdot 10^{-4}$	2.00	$1.62 \cdot 10^{-4}$	2.00	$6.19 \cdot 10^{-7}$	1.99	138.25	$5.87 \cdot 10^{-6}$	2.00	27.55	$5.87 \cdot 10^{-6}$	2.00	27.55

(b) Errors with a basis made of two elements: $q = 1$.

	h, basis V_h		Q_1, basis V_h		Q_2, basis V_h		h, basis V_h^+			Q_1, basis V_h^+			Q_2, basis V_h^+		
K	error	order	error	order	error	order	error	order	gain	error	order	gain	error	order	gain
20	$1.61 \cdot 10^{-3}$	—	$3.03 \cdot 10^{-3}$	—	$3.03 \cdot 10^{-3}$	—	$1.63 \cdot 10^{-5}$	—	98.79	$2.95 \cdot 10^{-4}$	—	10.27	$2.95 \cdot 10^{-4}$	—	10.27
40	$2.18 \cdot 10^{-4}$	2.89	$4.83 \cdot 10^{-4}$	2.65	$4.83 \cdot 10^{-4}$	2.65	$2.55 \cdot 10^{-6}$	2.68	85.60	$4.03 \cdot 10^{-5}$	2.87	11.97	$4.03 \cdot 10^{-5}$	2.87	11.97
80	$2.85 \cdot 10^{-5}$	2.94	$5.77 \cdot 10^{-5}$	3.06	$5.77 \cdot 10^{-5}$	3.06	$3.12 \cdot 10^{-7}$	3.03	91.29	$5.11 \cdot 10^{-6}$	2.98	11.30	$5.11 \cdot 10^{-6}$	2.98	11.30
160	$3.47 \cdot 10^{-6}$	3.04	$6.86 \cdot 10^{-6}$	3.07	$6.86 \cdot 10^{-6}$	3.07	$3.69 \cdot 10^{-8}$	3.08	94.23	$6.33 \cdot 10^{-7}$	3.01	10.84	$6.33 \cdot 10^{-7}$	3.01	10.84
320	$4.35 \cdot 10^{-7}$	3.00	$8.56 \cdot 10^{-7}$	3.00	$8.56 \cdot 10^{-7}$	3.00	$4.66 \cdot 10^{-9}$	2.98	93.43	$7.85 \cdot 10^{-8}$	3.01	10.91	$7.85 \cdot 10^{-8}$	3.01	10.91

(c) Errors with a basis made of three elements: $q = 2$.

Table 17: Shallow water equations in two space dimensions: errors, orders of accuracy, and gain obtained when approximating a steady solution for bases with and without prior.

	minimum gain			average gain			maximum gain		
q	ρ	Q_1	Q_2	ρ	Q_1	Q_2	ρ	Q_1	Q_2
0	48.21	0.39	0.39	1131.55	6.18	6.18	1592.52	11.24	11.24
1	0.96	0.16	0.16	186.82	21.00	21.00	422.68	49.05	49.05
2	0.06	0.02	0.02	82.45	8.76	8.76	206.29	22.43	22.43

Table 18: Shallow water equations in two space dimensions: statistics of the gains obtained for the approximation of a steady solution in basis V_h^+ with respect to basis V_h.

5.4.2 Perturbed steady solution

We now compare the new basis to the classical one when the initial condition is a perturbed steady solution. To that end, the initial water height is set to

$$h(0, \boldsymbol{x}; \mu) = h_{\text{eq}}(\boldsymbol{x}; \mu) - 0.02 \exp\left(-2((x_1 + 2)^2 + (x_2 + 2)^2)\right),$$

thus creating a bump-shaped perturbation whose center is located at $(-2, -2)$. For simplicity, we still use the value of the steady solution as Dirichlet boundary conditions. Moreover, we set the parameters μ to be the center of the cube $\mathbb{P}$, and we take $q = 1$ with 16^2 discretization cells.

The pointwise difference between h and h_{eq} is displayed in Figure 4. We observe that the prior-enriched basis V_h^+ (right panels) is able to capture the perturbation much better than the classical basis V_h (left panels). Indeed, the background, underlying steady solution has been smeared by basis V_h, while is is preserved with much greater resolution by basis V_h^+.

6 Conclusion

In this work, we proposed a Discontinuous Galerkin scheme whose basis has been enriched by neural networks to ensure an approximate well-balance property for a generic PDE and a generic equilibrium. The offline phase of the algorithm consists in learning a family of equilibria using parametric PINNs. Then, during the online phase, the trained network is used to enrich the DG basis and to approximate the solution to the PDE.

The results show significant gains in accuracy compared with the conventional DG method, particularly for low-dimensional approximation spaces. To obtain the same accuracy, we can significantly reduce the number of cells and use larger time steps. The method has been validated on a wide range of PDEs and equilibria, showing that it is a general-purpose approach. Furthermore, it makes it possible to handle complicated equilibria, on complex geometries, which are rarely treated by conventional WB schemes, especially in two space dimensions. The cost of training the network is low, as is the cost of inference. The main additional cost of the method comes from the quadrature rule, whose order has to be increased to ensure a good approximation of the integral of the prior. In most cases, this increase in order is not very important, and the gain between our approach and the classical ones remains significant.

There are several possible ways of extending our approach. From an application point of view, we wish to deal with more difficult equilibria, such as equilibria for the magnetohydrodynamics in tokamaks. From a methodological point of view, we would like to improve the determination of the prior by replacing parametric PINNs with physics-informed neural operators [51, 25] in order to widen the family of equilibria that can be considered. The other approach is to extend the method with time-dependent priors, in order to increase the accuracy of the scheme around families of unsteady solutions. To that end, we wish to move on to space-time DG methods, see e.g. [38].

References

[1] M. H. Abbasi, S. Naderi Lordejani, C. Berg, L. Iapichino, W. H. A. Schilders, and N. van de Wouw. An approximate well-balanced upgrade of Godunov-type schemes for the isothermal Euler equations and the drift flux model with laminar friction and gravitation. *Int. J. Numer. Meth. Fluids*, 2020.

[2] M. Abramowitz and I. A. Stegun, editors. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover Publications, Inc., New York, 1992. Reprint of the 1972 edition.

[3] M. Ainsworth and J. Dong. Galerkin Neural Networks: A Framework for Approximating Variational Equations with Error Control. *SIAM J. Sci. Comput.*, 43(4):A2474–A2501, 2021.

[4] E. Audusse, F. Bouchut, M.-O. Bristeau, R. Klein, and B. Perthame. A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows. *SIAM J. Sci. Comput.*, 25(6):2050–2065, 2004.

[5] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proc. Natl. Acad. Sci.*, 116(31):15344–15349, 2019.

[6] H. Barucq, H. Calandra, J. Diaz, and E. Shishenina. Space–time Trefftz-DG approximation for elasto-acoustics. *Appl. Anal.*, 99(5):747–760, 2018.

[7] A. D. Beck, J. Zeifang, A. Schwarz, and D. G. Flad. A neural network based shock detection and localization approach for discontinuous Galerkin methods. *J. Comput. Phys.*, 423:109824, 2020.

[8] J. P. Berberich, R. Käppeli, P. Chandrashekar, and C. Klingenberg. High Order Discretely Well-Balanced Methods for Arbitrary Hydrostatic Atmospheres. *Commun. Comput. Phys.*, 30(3):666–708, jun 2021.

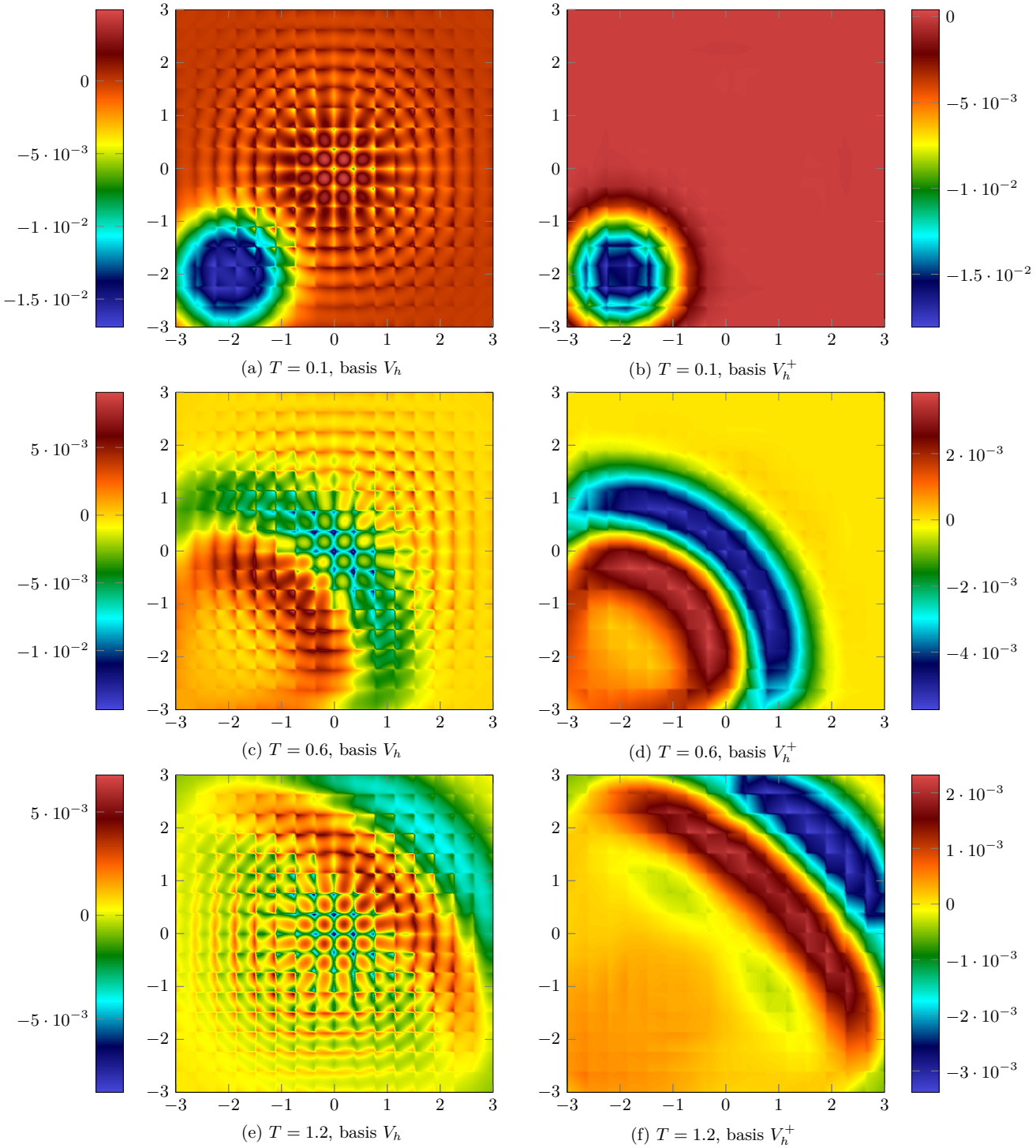

Figure 4: 2D shallow water equations: representation of the error between the perturbed solution and the underlying steady solution. Left panels: classical basis V_h; right panels: prior-enriched basis V_h^+. From top to bottom, several final times are displayed ($T = 0.1$, $T = 0.6$, $T = 1.2$).

[9] C. Berthon, S. Bulteau, F. Foucher, M. M'Baye, and V. Michel-Dansac. A Very Easy High-Order

Well-Balanced Reconstruction for Hyperbolic Systems with Source Terms. *SIAM J. Sci. Comput.*, 44(4):A2506–A2535, 2022.

[10] L. Bois, E. Franck, L. Navoret, and V. Vigon. An optimal control deep learning method to design artificial viscosities for Discontinuous Galerkin schemes. *arXiv preprint arXiv:2309.11795*, 2023.

[11] J. Britton and Y. Xing. High Order Still-Water and Moving-Water Equilibria Preserving Discontinuous Galerkin Methods for the Ripa Model. *J. Sci. Comput.*, 82(2), 2020.

[12] C. Buet, B. Despres, and G. Morel. Trefftz discontinuous Galerkin basis functions for a class of Friedrichs systems coming from linear transport. *Adv. Comput. Math.*, 46(3), 2020.

[13] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: a review. *Acta Mech. Sin.*, 37(12):1727–1738, 2021.

[14] O. Castro-Orgaz and H. Chanson. Minimum Specific Energy and Transcritical Flow in Unsteady Open-Channel Flow. *J. Irrig. Drain. E. - ASCE*, 142(1):04015030, 2016.

[15] S. Chandrasekhar. *An introduction to the study of stellar structure.* Dover Publications, Mineola, 1967.

[16] M. Ciallella, D. Torlo, and M. Ricchiuto. Arbitrary High Order WENO Finite Volume Scheme with Flux Globalization for Moving Equilibria Preservation. *J. Sci. Comput.*, 96(2), 2023.

[17] P. E. Crittenden and S. Balachandar. The impact of the form of the Euler equations for radial flow in cylindrical and spherical coordinates on numerical conservation and accuracy. *Shock Waves*, 28(4):653–682, 2018.

[18] V. Desveaux, M. Zenk, C. Berthon, and C. Klingenberg. A well-balanced scheme to capture non-explicit steady states in the Euler equations with gravity. *Internat. J. Numer. Methods Fluids*, 81(2):104–127, 2016.

[19] N. Discacciati, J. S. Hesthaven, and D. Ray. Controlling oscillations in high-order Discontinuous Galerkin schemes using artificial viscosity tuned by neural networks. *J. Comput. Phys.*, 409:109304, 2020.

[20] E. Franck and L. S. Mendoza. Finite Volume Scheme with Local High Order Discretization of the Hydrostatic Equilibrium for the Euler Equations with External Forces. *J. Sci. Comput.*, 69(1):314–354, 2016.

[21] E. Gaburro, M. J. Castro, and M. Dumbser. Well-balanced Arbitrary-Lagrangian-Eulerian finite volume schemes on moving nonconforming meshes for the Euler equations of gas dynamics with gravity. *Mon. Not. R. Astron Soc.*, 477(2):2251–2275, 2018.

[22] J. M. Gallardo, C. Parés, and M. Castro. On a well-balanced high-order finite volume scheme for shallow water equations with topography and dry areas. *J. Comput. Phys.*, 227(1):574–601, 2007.

[23] I. Gómez-Bueno, M. J. Castro, and C. Parés. High-order well-balanced methods for systems of balance laws: a control-based approach. *Appl. Math. Comput.*, 394:125820, 2021.

[24] I. Gómez-Bueno, M. J. Castro Díaz, C. Parés, and G. Russo. Collocation Methods for High-Order Well-Balanced Methods for Systems of Balance Laws. *Mathematics*, 9(15):1799, 2021.

[25] S. Goswami, A. Bora, Y. Yu, and G. E. Karniadakis. Physics-Informed Deep Neural Operator Networks. In *Computational Methods in Engineering & the Sciences*, pages 219–254. Springer International Publishing, 2023.

[26] J. S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods.* Springer New York, 2008.

[27] L.-M. Imbert-Gérard, A. Moiola, and P. Stocker. A space–time quasi-Trefftz DG method for the wave equation with piecewise-smooth coefficients. *Math. Comp.*, 92(341):1211–1249, 2022.

[28] R. Käppeli and S. Mishra. Well-balanced schemes for the Euler equations with gravitation. *J. Comput. Phys.*, 259:199–219, 2014.

[29] D. I. Ketcheson. Highly Efficient Strong Stability-Preserving Runge–Kutta Methods with Low-Storage Implementations. *SIAM J. Sci. Comput.*, 30(4):2113–2136, jan 2008.

[30] F. Kretzschmar, S. M. Schnepp, I. Tsukerman, and T. Weiland. Discontinuous Galerkin methods with Trefftz approximations. *J. Comput. Appl. Math.*, 270:211–222, 2014.

[31] A. Kurganov and G. Petrova. A second-order well-balanced positivity preserving central-upwind scheme for the Saint-Venant system. *Commun. Math. Sci.*, 5(1):133–160, 2007.

[32] R. Käppeli. Well-balanced methods for computational astrophysics. *Living Reviews in Computational Astrophysics*, 8(1), oct 2022.

[33] V. Michel-Dansac. Victor-MichelDansac/DG-PINNs, GitHub repository, 2023.

[34] V. Michel-Dansac, C. Berthon, S. Clain, and F. Foucher. A well-balanced scheme for the shallow-water equations with topography. *Comput. Math. Appl.*, 72(3):568–593, 2016.

[35] V. Michel-Dansac, C. Berthon, S. Clain, and F. Foucher. A well-balanced scheme for the shallow-water equations with topography or Manning friction. *J. Comput. Phys.*, 335:115–154, 2017.

[36] V. Michel-Dansac, C. Berthon, S. Clain, and F. Foucher. A two-dimensional high-order well-balanced scheme for the shallow water equations with topography and Manning friction. *Comput. & Fluids*, 230:105152, 2021.

[37] S. Noelle, Y. Xing, and C.-W. Shu. High-order well-balanced finite volume WENO schemes for shallow water equation with moving water. *J. Comput. Phys.*, 226(1):29–58, 2007.

[38] S. Petersen, C. Farhat, and R. Tezaur. A space-time discontinuous Galerkin method for the solution of the wave equation in the time domain. *Internat. J. Numer. Methods Engrg.*, 78(3):275–295, 2008.

[39] D. A. Di Pietro and A. Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*, volume 69 of *Mathématiques et Applications*. Springer Berlin Heidelberg, 2012.

[40] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

[41] D. Ray and J. S. Hesthaven. An artificial neural network as a troubled-cell indicator. *J. Comput. Phys.*, 367:166–191, 2018.

[42] L. Schwander, D. Ray, and J. S. Hesthaven. Controlling oscillations in spectral methods by local artificial viscosity governed by neural networks. *J. Comput. Phys.*, 431:110144, 2021.

[43] C.-W. Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In *Advanced numerical approximation of nonlinear hyperbolic equations (Cetraro, 1997)*, volume 1697 of *Lecture Notes in Math.*, pages 325–432. Springer, Berlin, 1998.

[44] S. C. Spiegel, H. T. Huynh, and J. R. DeBonis. A Survey of the Isentropic Euler Vortex Problem using High-Order Methods. In *22nd AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, 2015.

[45] R. J. Spiteri and S. J. Ruuth. A New Class of Optimal High-Order Strong-Stability-Preserving Time Discretization Methods. *SIAM J. Numer. Anal.*, 40(2):469–491, 2002.

[46] J. Sun, S. Dong, and F. Wang. Local Randomized Neural Networks with Discontinuous Galerkin Methods for Partial Differential Equations. *arXiv preprint arXiv:2206.05577*, 2022.

[47] J. Sun and F. Wang. Local Randomized Neural Networks with Discontinuous Galerkin Methods for Diffusive-Viscous Wave Equation. *arXiv preprint arXiv:2305.16060*, 2023.

[48] L. Sun, H. Gao, S. Pan, and J.-X. Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comput. Methods Appl. Mech. Engrg.*, 361:112732, 2020.

[49] A. Thomann, G. Puppo, and C. Klingenberg. An all speed second order well-balanced IMEX relaxation scheme for the Euler equations with gravity. *J. Comput. Phys.*, 420:109723, 2020.

[50] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics. A practical introduction*. Springer-Verlag, Berlin, third edition, 2009.

[51] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. *Sci. Adv.*, 7(40), 2021.

[52] C. Wu, M. Zhu, Q. Tan, Y. Kartha, and L. Lu. A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Comput. Methods Appl. Mech. Engrg.*, 403:115671, 2023.

[53] Y. Xing, X. Zhang, and C.-W. Shu. Positivity-preserving high order well-balanced discontinuous Galerkin methods for the shallow water equations. *Adv. Water Resour.*, 33(12):1476–1493, 2010.

[54] J. Yu and J. S. Hesthaven. A data-driven shock capturing approach for discontinuous Galekin methods. *Comput. & Fluids*, 245:105592, 2022.

[55] X. Yu and C.-W. Shu. Multi-layer Perceptron Estimator for the Total Variation Bounded Constant in Limiters for Discontinuous Galerkin Methods. *La Matematica*, 1(1):53–84, 2021.

[56] L. Yuan and C.-W. Shu. Discontinuous Galerkin method based on non-polynomial approximation spaces. *J. Comput. Phys.*, 218(1):295–323, 2006.

[57] W. Zhang, Y. Xing, and E. Endeve. Energy conserving and well-balanced discontinuous Galerkin methods for the Euler-Poisson equations in spherical symmetry. *Mon. Not. R. Astron. Soc.*, 514(1):370–389, 2022.