Model order reduction strategies for weakly dispersive waves

Davide Torlo, Mario Ricchiuto

Model order reduction strategies for weakly dispersive waves

Davide Torlo[a,*], Mario Ricchiuto[b]

[a] *Mathematics Area, mathLab, SISSA, via Bonomea 265, Trieste, I-34136, Italy*
[b] *INRIA, Univ. Bordeaux, CNRS, Bordeaux INP, IMB, UMR 5251, 200 Avenue de la Vieille Tour, Talence, 33405, France*

Abstract

We focus on the numerical modelling of water waves by means of depth averaged models. We consider in particular PDE systems which consist in a nonlinear hyperbolic model plus a linear dispersive perturbation involving an elliptic operator. We propose two strategies to construct reduced order models for these problems, with the main focus being the control of the overhead related to the inversion of the elliptic operators, as well as the robustness with respect to variations of the flow parameters. In a first approach, only a linear reduction strategies is applied only to the elliptic component, while the computations of the nonlinear fluxes are still performed explicitly. This hybrid approach, referred to as pdROM, is compared to a hyper-reduction strategy based on the empirical interpolation method to reduce also the nonlinear fluxes. We evaluate the two approaches on a variety of benchmarks involving a generalized variant of the BBM-KdV model with a variable bottom, and a one-dimensional enhanced weakly dispersive shallow water system. The results show the potential of both approaches in terms of cost reduction, with a clear advantage for the pdROM in terms of robustness, and for the EIMROM in terms of cost reduction.

Keywords: model order reduction, dispersive wave equations, BBM-KdV, Boussinesq, hyper-reduction
2010 MSC: 65M06, 76B15, 65M22

1. Introduction

Water waves equations can be modeled with various strategies [28]. Different models provide reasonable approximations in different contexts, according to which type of solution we are interested in. The most complete models consider Euler or Navier–Stokes equations and describe the motion of the water in each point in the 3 dimensional space. A first approximation level is given by averaging the water speed on the vertical direction, reducing by one the dimensions and allowing to display only the water surface as a function of the horizontal variables. The well–known shallow water equations, Boussinesq equations, Green-Naghdi equations, and other models fall in this category. In this work we are interested in studying dispersive models, which are able of reproducing dispersion phenomena. In contrast with other models, as shallow water equations, in these dispersive models shocks are avoided and instead the dispersion fragments the waves into different waves with different wavelength before they break. We will consider two dispersive models which approximate water waves at different levels: the Benjamin–Bona–Mahony (BBM) equation [5], in a more general form that it is linked to the Korteweg–De Vries (KdV) equation [11, 28], and an enhanced Boussinesq (EB) system of equations [30, 38].

The peculiarity of both models is the combination of hyperbolic systems and dispersive terms. Numerical solutions of such problems are often obtained using explicit solver for the hyperbolic part, while the dispersive terms are obtained first solving an elliptic problem, which requires greater computational costs, and then using this solution inside the hyperbolic system [38, 16]. In this work, we will try to reduce the computational costs of such problems, focusing on the compression of the elliptic operators, which are responsible of the largest part of the computational time of such methods. In case of parametric problems, where a fast response is necessary or we have a multi-query task, the reduction could lead to strong advantages for the computational costs, without degrading the quality of the solution.

*Corresponding author: davide.torlo@sissa.it

Hyperbolic problems are known for developing shocks and advection dominated solutions. Classically, they are badly reducible as the Kolmogorov n–width decays very slowly for such problems [33, 43, 8, 45, 31, 37, 35]. Nevertheless, for dispersive problems there is no shock formations and the wave traveling is often transformed into a oscillation of the whole domain. Hence, also the advection character of the solution is less pronounced. Hence, there is room to attempt a reduction with classical model order reduction (MOR) algorithms coming from the parabolic and elliptic community. In particular, we will use the proper orthogonal decomposition (POD) [25] to reduce the solution manifold and then we will apply a Galerkin projection to obtain a reduced problem. A further step will consist of interpolating the nonlinear fluxes with the empirical interpolation method (EIM) [4]. This will provide a second level of reduction that allows more reduction in the computational costs. Few works have already performed a model order reduction directly on shallow water equations, *inter alia* [41, 42, 39, 40]. Up to our knowledge, there are no other works trying to reduce the computational costs of dispersive wave equations by the means of model order reduction techniques.

The paper is structured as follows. In Section 3 we introduce the BBM–KdV model, some energy properties and its classical discretization, i.e., the full order model (FOM), and we introduce few tests that we will study along the paper. In Section 3.5 we develop the reduction algorithms and the hyper-reduction steps. In Section 4 we test the presented algorithms on all the presented tests, showing the power and limits of these reduction techniques. In Section 5 we introduce an enhanced Boussinesq system of equations, its FOM discretization and some reference tests that we will use. We introduce then its ROM and its hyper-reduction algorithm. Then we apply the reduced algorithms for the presented tests in Section 6, where we compare the two reduced methods and the FOM one. In Section 7 we highlight the major steps of this work and we suggest possible developments and extensions.

2. Generalities: dispersive wave models and model reduction

We discuss the general form of the dispersive wave model used in this work, which also allows to explain the main underlying idea. Weakly dispersive wave models can be often seen as perturbations of some hyperbolic partial differential system [28], and can be written in general form

$$\partial_t u + \partial_x F(u) + S(u) - \mu^2 \mathcal{X}^t \left(\partial_t u + \partial_x G(u)\right) + \mu^2 \mathcal{D}^x u = 0 \tag{1}$$

where $u : \mathbb{R} \to \mathbb{R}^d$ are the unknowns of the system, $F : \mathbb{R}^d \to \mathbb{R}^d$ is a (nonlinear) flux which describes the hyperbolic operator, $S : \mathbb{R}^d \to \mathbb{R}^d$ is a source term which might include bathymetry effects, $\mathcal{X}^t : \mathbb{R}^d \to \mathbb{R}^d$ is a linear operator that contains second derivative terms, while $\mathcal{D}^x : \mathbb{R}^d \to \mathbb{R}^d$ is a linear operator defined by some dispersion terms with third order derivatives. The first three terms in the above equation define a hyperbolic balance law, as e.g. the shallow water equations with bathymetry, and μ^2 is a small parameter multiplying the weakly dispersive regularization [28]. Two specific examples of this general family are used in the paper: the BBM-KdV model [5, 11, 28] and an enhanced Boussinesq model [30, 38].

The main idea of this work is to exploit the smallness of μ^2 to defined a hybrid model, which enhances the hyperbolic equation with a reduced order approximation of the small terms. To this end, we follow [16, 9] and recast the system as

$$(1 - \mu^2 \mathcal{X}^t)(\partial_t u + \partial_x F(u) + S(u)) + \mu^2 \mathcal{X}^x u = 0, \tag{2}$$

having set

$$\mathcal{X}^x u := \mathcal{X}^t(\partial_x F(u) - \partial_x G(u) + S(u)) + \mathcal{D}^x u. \tag{3}$$

The models share the property of being composed of two parts: an elliptic (linear) operator and a hyperbolic (nonlinear) operator. The two operators are treated very differently in their discretization, where the hyperbolic part is often discretized in an explicit way, while the elliptic operator is discretized implicitly. Moreover, the combination of the two operators allows to obtain stable schemes with little or zero extra

numerical viscosity, which is not the case for pure hyperbolic problems. In particular, we can recast the system introducing an auxiliary variable $\Phi \in \mathbb{R}^d$:

$$\begin{cases} \left(1 - \mu^2 \mathcal{X}^t\right) \Phi + \mu^2 \mathcal{X}^x u = 0, \\ \partial_t u + \partial_x F(u) + S(u) = \Phi. \end{cases} \tag{4}$$

The operator $(1 - \mathcal{X}^t)$ is elliptic (and invertible), and by classical formal arguments we can show the smallness of Φ simply by noting that (see also [28, 27, 29])

$$\Phi = -\mu^2 \left(1 - \mu^2 \mathcal{X}^t\right)^{-1} \mathcal{X}^x u = \mathcal{O}(\mu^2).$$

Our main idea is to devise an approach where only the term Φ is reduced, possibly benefiting from the combination of the modelling error (controlled by μ^2) and model reduction error. The motivation for this is discussed in the next section.

2.1. Time and space discretization of the model

As in [16, 9] for all models we use a splitting of the elliptic and hyperbolic operators, as in an IMEX time discretization. This leads to the semi discrete prototype

$$\begin{cases} \left(1 - \mathcal{X}^t\right) \Phi^{n+1} + \mathcal{X}^x u^n = 0, \\ \dfrac{u^{n+1} - u^n}{\Delta t} + \partial_x F(u^n) + S(u^n, u^{n+1}) = \Phi^{n+1}. \end{cases} \tag{5}$$

Here, we denote with n the time discretization index, and the μ^2 factor has been included in the operators $\mathcal{X}^t$ and $\mathcal{X}^x$ to lighten the expressions. As we can see, all the operations are vectorial except the solution of Φ^{n+1}, which requires the solution of a linear system. After a further spatial discretization the system can be written in the general form

$$\begin{cases} \left(\mathbb{M}^\Phi - \mathbb{X}^t\right) \Phi^{n+1} + \mathbb{X}^x u^n = 0, \\ \mathbb{M}^u \dfrac{u^{n+1} - u^n}{\Delta t} + \mathbb{F}(u^n) + \mathbb{S}^{ex}(u^n) + \mathbb{S}^{im} u^{n+1} = \Phi^{n+1}, \end{cases} \tag{6}$$

where, with an abuse of notation, we use the same symbols $u \in \mathbb{R}^{d \times N_h}$ and $\Phi \in \mathbb{R}^{N_h}$ to refer to the discretized variables, where N_h is the number of degrees of freedom for each variable. The discretization can be performed by finite differences (FD) or finite element methods (FEM). Specific choices used for our simulations are discussed later in the text. $\mathbb{M}^u$ and $\mathbb{M}^\Phi$ are mass matrices, $\mathbb{X}^t$ and $\mathbb{X}^x$ are the discretizations of $\mathcal{X}^t$ and $\mathcal{X}^x$ respectively. $\mathbb{S}^{ex}$ is the discretization of the explicit part of the source function S and $\mathbb{S}^{im}$ is the matrix that discretizes the linear implicit source terms and $\mathbb{F}(u)$ is a discretization of the (nonlinear) flux $\partial_x F(u)$, which might include some stabilization terms as well. In the following, we will denote (6) as full order model (FOM). The time discretization in (6) is done through an implicit–explicit (IMEX) scheme, where some of the terms are discretized with implicit Euler (Φ^{n+1}) and others with explicit Euler. In practice, we will will use a IMEX Runge–Kutta (RK) schemes with order matching the spatial discretization orders, but all the discussion can be easily derived from the first order discretization of (6). Hence, we will keep discussing it, and only for the specific problems we will go in detail with the IMEX RK discretization. To have stability of the hyperbolic operator, we need to impose some CFL conditions on the timestep of the type $\Delta t \rho(JF(u)) \leq \texttt{CFL} \Delta x$, where $\rho(JF(u))$ is the spectral radius of the Jacobian of F in u and `CFL` is a constant smaller than 1.

Typically, the most computationally intensive part is the solution of the linear systems, derived from the elliptic operator. While in one dimensional cases the linear systems are tridiagonal for simple discretizations and very efficient algorithms like Thomas algorithm can render very fast the solution of such systems, this is not the case in general. Even in this very favourable case, we find that the cost of

the solution of these linear systems is still the largest of the problems (between 60% and 90%).This is also due to the fact that the computation of the explicit flux and source terms can be efficiently done in parallel. The main goal of this work is to investigate hybrid methods in which we reduce the cost of the linear system, while keeping the evaluation of the fully nonlinear terms.

2.2. Reduction of the general model

In this section, we describe how to perform some reduction techniques on the operators presented in the FOM. As described above, there are two types of operations in the previous discretized model. The first ones are the vector based operations constituted of the fluxes, sources and diffusion terms, of derivatives and of sums of all terms and (sparse) matrix vector multiplications. The second one are the matrix based operations which include essentially the solution of linear systems. Even if in some particular cases efficient *ad hoc* algorithms can be used (e.g. Thomas algorithm for the tridiagonal matrices), we have in mind more challenging applications, e.g. two–dimensional problems or high order methods, where such algorithms cannot be used. In those cases, we need to recast to sparse linear solver methods, which in general allow at best to obtain costs of the order of $\mathcal{O}(N_h \log(N_h))$ at each time step. These systems require greater computational costs to be solved, with respect to the vector part, and are the ones that mostly weigh on the FOM algorithm.

Our objective here is to perform some model reduction to gain control on the computational costs. To this end, we introduce different approximations of the model, via reduced basis expansions [21]. We want to find an appropriate set of N_{RB} basis of vectors in $\mathbb{R}^{N_h}$, with $N_{RB} \ll N_h$, allowing to express the approximate solutions as

$$u^n \approx V\hat{u}^n, \qquad \hat{u}^n \in \mathbb{R}^{N_{RB}} \tag{7}$$

with the columns of $V \in \mathbb{R}^{(d\times N_h)\times N_{RB}}$ containing the basis vectors, and $\hat{u}^n$ the vector of the reduced basis coefficients.

A classical way of obtaining a reduced basis method is by projecting onto the reduced space. We introduce a test matrix $W \in \mathbb{R}^{(d\times N_h)\times N_{RB}}$ and then we project the whole equation (27), i.e.,

$$W^T\left(\mathbb{M}^\Phi - \mathbb{X}^t\right)V\hat{\Phi}^{n+1} + W^T\mathbb{X}^x V\hat{u}^n = 0, \tag{8a}$$

$$W^T\mathbb{M}^u V\frac{\hat{u}^{n+1}-\hat{u}^n}{\Delta t} + W^T\mathbb{F}(V\hat{u}^n) + W^T\mathbb{S}^{ex}(V\hat{u}^n) + W^T\mathbb{S}^{im}V\hat{u}^{n+1} = W^TV\hat{\Phi}^{n+1}. \tag{8b}$$

This open the question on which matrix W we should consider. If we choose $W = V$ we obtain a Galerkin projection, while different choices leads to Petrov–Galerkin projections. There are several choices that one can take to obtain W, for example controlling the energy of the system [19]. We will comment in each model what we use as test matrix W.

It should be noted that with the reduction (8), if $N_{RB} \ll N_h$, one can get rid of the costs of the solution of the large systems of equations. If we define, for every matrix $\mathbb{A} \in \mathbb{R}^{N_h\times N_h}$, its reduced version as

$$\hat{\mathbb{A}} := W^T\mathbb{A}V \in \mathbb{R}^{N_{RB}\times N_{RB}}, \tag{9}$$

we can reduce the computational costs of (8), obtaining

$$\left(\hat{\mathbb{M}}^\Phi - \hat{\mathbb{X}}^t\right)\hat{\Phi}^{n+1} + \hat{\mathbb{X}}^x\hat{u}^n = 0, \tag{10a}$$

$$\hat{\mathbb{M}}^u\frac{\hat{u}^{n+1}-\hat{u}^n}{\Delta t} + W^T\mathbb{F}(V\hat{u}^n) + W^T\mathbb{S}^{ex}(V\hat{u}^n) + \hat{\mathbb{S}}^{im}\hat{u}^{n+1} = \hat{\mathbb{I}}\hat{\Phi}^{n+1}, \tag{10b}$$

where $\mathbb{I} \in \mathbb{R}^{N_h\times N_h}$ is the identity matrix. In this model, the costs of the linear systems are independent of N_h, while still some computations (the computations of fluxes and source terms) are still dependent of N_h. Nevertheless, their implementation can be easily parallelized and their computational costs do not impact as much as the solver of the full linear system. Still, the computational cost reduction that one can obtain in this context is limited by these terms.

2.2.1. Choice of the reduced space

The underlying technique used in this work is proper orthogonal decomposition (POD) [25, 21] or principal component analysis (PCA). It consists in a singular value decomposition (SVD) of the snapshot matrix (a simple juxtaposition of the FOM solutions for different parameters and timesteps) of which we retain the most energetic *eigenvectors*, ordering the singular values. The retained N_{RB} eigenvectors form the basis $V \in \mathbb{R}^{N_h \times N_{RB}}$ for the reduced order model (ROM).

The number of retained modes/eigenvectors can be chosen either directly, or according to a tolerance on the discarded energy, i.e.,

$$N_{RB} := \arg\max_{N} \left\{ \frac{\sum_{j=1}^{N} \sigma_i}{\sum_{j=1}^{N_{\max}} \sigma_i} \leq 1 - tol_{POD} \right\}, \tag{11}$$

where $N_{\max}$ is the minimum between N_h and the number of snapshots considered.

The POD (and the linear ROMs) are based on the superposition ansatz, for which the solutions can be reasonably represented by a linear combination of few modes $u(x) \approx \sum_{j=1}^{N_{RB}} v_j(x)\hat{u}_j$. This is not always true, in particular for hyperbolic problems. More sophisticated algorithms exist to obtain nonlinear ROMs [8, 45, 33, 43, 37, 31], but these will not be considered in this work.

2.2.2. Hyper-reduction of nonlinear operators

To further alleviate the computational cost one can introduce some linearization of the nonlinear operator. There are several algorithms for this. The approach selected in this work is the so-called empirical interpolation method (EIM) [4]. The EIM is a very simple approximate-then-project approach. It is perhaps not the most advanced technique to handle nonlinearities, however it will serve as a reference in terms of cost. Using it with different tolerance will allow to have a fair comparison. The hyper-reduced model will be referred to as EIMROM in the following. We want to mention that several other techniques have been developed in recent years and more stable algorithms are available [48, 36, 12, 10, 15]. Still, the focus of this work is not on the hyper-reduction, hence we will stick to a simple EIM algorithm.

Given a nonlinear flux, the goal of the EIM procedure is to approximate it through an interpolation into few *magic points* $\{z_j\}_{j=1}^{N_{EIM}}$, and some basis functions $\{\psi_j\}_{j=1}^{N_{EIM}} \subset \mathbb{R}^{N_h}$. We denote by $Z \in \mathbb{R}^{N_{EIM}}$ the vector of the magic points, and by $\Psi_{ij} := \psi_j(x_i) \in \mathbb{R}^{N_h \times N_{EIM}}$ the matrix of the reduced basis functions evaluated in the magic points, i.e., the slice of V in the magic points. Given a nonlinear flux $\varphi\{\boldsymbol{u}\} : (\mathbb{R}^{N_h})^d \to \mathbb{R}^{N_h}$, we approximate it as

$$\varphi^{EIM}\{\boldsymbol{u}\}(x) := \sum_{j=1}^{N_{EIM}} \psi_j(x)\varphi\{\boldsymbol{u}\}(z_j), \tag{12}$$

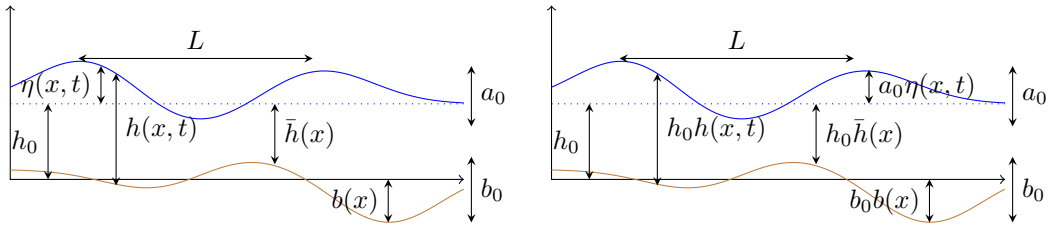
which, after the projection, reads

$$\hat{\varphi}^{EIM}(\boldsymbol{u}) = W^T\varphi^{EIM}(\boldsymbol{u}) = \sum_{j=1}^{N_{EIM}} W^T\psi_j(x)\varphi\{\boldsymbol{u}\}(z_j) = W^T\Psi\varphi\{\boldsymbol{u}\}(Z). \tag{13}$$

The reduction lies in the computation of the flux $\varphi\{\boldsymbol{u}\}$ only in few magic points $\{z_j\}_{j=1}^{N_{EIM}}$, while we can pre-compute and store the matrix $W^T\Psi \in \mathbb{R}^{N_{RB} \times N_{EIM}}$ in the offline phase.

The choice of the points and of the number of the EIM basis functions is crucial in order to maintain a good accuracy of the whole algorithm. Typically, they are chosen in a greedy fashion that minimizes the ∞-norm of the worst approximated fluxes. All the details can be found in [4, 13]. In practice we will stop such algorithm when the error gets lower than a certain tolerance set proportionally with the error of the RB approximation.

Remark 1 (Implementation details)**.** *To assemble and solve the systems at each stage we will use a Python implementation based on* `numpy` *[20] and on* `Numba` *[26]. These packages allow to perform a multithread computation of the vector assembly (all the nonlinear terms), while the solution of the sparse*



(a) Dimensional variables (b) Dimensionless variables

Figure 1: Water waves: notation

FOM systems will be performed with Thomas algorithm, since the matrices are tridiagonal. This leads to have the largest cost in the solution of the system $\mathcal{O}(N_h)$, while the nonlinear operators can be assembled at a lower cost, depending on the architecture of the machine. We will compare also the sparse solver of `scipy` *to emulate the case of a general sparse solver without the a priori knowledge of the tridiagonality. In that version of the algorithm, the whole computational time, except the fluxes, will be affected by a non precompiled code. For ROM systems, which are not sparse, we will use the* `solve` *function of the linear algebra package of* `numpy`*. The computations all along this work will be performed on an 8 cores Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz.*

3. BBM-KdV model

As a first exercise, we study the BBM-KdV equations, which provide a scalar one way approximation of the water wave equations [5, 11, 28]. It is useful to write the scalar partial differential equation in terms of the dimensionless parameters

$$\mu = \frac{h_0}{L}, \quad \epsilon = \frac{a_0}{h_0}, \quad \beta = \frac{b_0}{h_0},$$

with h_0 a characteristic water depth, L the wave length, a_0 the wave amplitude, and with b_0 a characteristic bathymetry amplitude, see Figure 1 for both dimensional and non-dimensional notations. These parameters are classically used to define the different propagation regimes. More precisely, the dispersive character of the waves is measured by μ, while ε is a measure of their nonlinearity.

For $\varepsilon = \mathcal{O}(\mu^2)$, the BBM–KdV equation provides a 1-way $\mathcal{O}(\mu^4 + \epsilon\mu^2)$ approximation of the water equations. Its classical form for constant bathymetry reads

$$\left(1 - \alpha_p \mu^2 \partial_{xx}\right) \partial_t \eta + \partial_x \left(\eta + \frac{3}{2}\varepsilon \frac{\eta^2}{2}\right) + \mu^2 p \partial_{xxx} \eta = 0, \qquad p \leq \frac{1}{6}, \tag{14}$$

having set $\alpha_p = 1/6 - p$. For different values of the parameter p we obtain different approximations. In particular, the KdV equation is obtained for $p = \frac{1}{6}$, while for $p = 0$ the equation reduces to the BBM model.

For the purpose of its numerical approximation, we manipulate the model equation following [16], and recast it as

$$\left(1 - \alpha_p \mu^2 \partial_{xx}\right) \left(\partial_t \eta + \partial_x F(\eta)\right) + \frac{\mu^2}{6} \partial_{xxx} \eta = -\varepsilon \mu^2 \alpha_p \partial_{xxx} \left(\frac{3}{2} \frac{\eta^2}{2}\right), \tag{15}$$

having introduced the short notation for nonlinear flux

$$F(\eta) := \left(\eta + \frac{3}{2}\varepsilon \frac{\eta^2}{2}\right). \tag{16}$$

Within the same approximation hypotheses, the BBM-KdV equation is equivalent to

$$\left(1 - \alpha_p \mu^2 \partial_{xx}\right) \left(\partial_t \eta + \partial_x F(\eta)\right) + \frac{\mu^2}{6} \partial_{xxx} \eta = 0. \tag{17}$$

which we refer to as the modified BBM-KdV equation. Note that both for (15) and (17), the limit $\mu \to 0$ corresponds to the hyperbolic conservation law

$$\partial_t \eta + \partial_x F(\eta) = 0. \tag{18}$$

Equation (17) can be recast into the form of (2) by setting $u := \eta$, $\mathcal{X}^x := \frac{\mu^2}{6}\partial_{xxx}$, $\mathcal{X}^t := \alpha_p \mu^2 \partial_{xx}$ and $S(u) := 0$.

Remark 2 (Dimensional equations)**.** *One can easily recover the dimensional form of the equation, and more importantly of the variables involved, from the scaling:*

$$\tilde{x} = Lx, \quad \tilde{t} = \frac{L}{c_0}t, \quad \tilde{\eta} = \epsilon h_0 \eta, \quad \tilde{b} = \beta h_0 b, \quad \tilde{\bar{h}} = h_0 \bar{h}, \quad \tilde{h} = h_0 h \tag{19}$$

with $c_0 = \sqrt{g h_0}$.

3.1. Energy conservation

A property of (14) which we will use in the following is the existence of a certain number of additional derived conservation laws (see also [1**?**]). In particular, (14) is endowed with an energy-energy flux pair also verifying a conservation law. This can be shown quite classically by premultiplying by η and manipulating the resulting expression. Setting $\Xi(u) = \eta^2/2 + \varepsilon\eta^3/2$, this leads to

$$\begin{aligned} &\partial_t \mathcal{E} + \partial_x \mathcal{F} = 0, \\ &\mathcal{E}(\eta) := \frac{\eta^2}{2} + \mu^2 \alpha_p \frac{(\partial_x \eta)^2}{2}, \\ &\mathcal{F}(\eta) := \Xi(\eta) - \mu^2 \alpha_p \eta \partial_{xt}\eta + \mu^2 p \eta \partial_{xx}\eta - \mu^2 p \frac{(\partial_x \eta)^2}{2}. \end{aligned} \tag{20}$$

From this relation, we know that the energy $\mathcal{E}(\eta)$ is conserved for (14). For the simplified model (17), this balance is only valid within the $\mathcal{O}(\epsilon\mu^2, \mu^4)$ asymptotic error of the model, namely

$$\partial_t \mathcal{E} + \partial_x \mathcal{F} = \mathcal{O}(\epsilon\mu^2, \mu^4). \tag{21}$$

Other balance laws may be derived in a similar spirit, see e.g. [1]. In both cases, it is convenient to consider the scalar product associated to $\mathcal{E}$

$$\langle \eta, \rho \rangle_{\mathcal{E}} := \frac{1}{2}\int_\Omega \eta\rho + \mu^2 \alpha_p \partial_x \eta \partial_x \rho \tag{22}$$

which is essentially a modified H^1 norm scalar product.

3.2. Including the bathymetric effects

Generalizations of the KdV equation and of other dispersive one-wave models are discussed in some detail in [22, 14, 23]. For our purposes, these generalizations are useful as they allow to broaden the spectrum of benchmarks. We use here the $\mathcal{O}(\varepsilon\mu^2, \mu^2\beta, \mu^4)$ approximation obtained from the variable depth model of [22], still referred to here as to the modified BBM-KdV model, and reading

$$(1 - \alpha\partial_{xx})(\partial_t \eta + (\gamma + \delta\eta)\partial_x \eta) + \omega \partial_{xxx}\eta + \nu\eta = 0, \tag{23}$$

where the coefficient depend now on the bathymetry profile $b(x)$ as (in dimensionless form)

$$\begin{aligned} c(x) = \sqrt{1 - \beta b(x)}, \quad \alpha(x) =& \alpha_p \mu^2, \quad \omega(x) = \frac{1}{6}\mu^2 c(x)^5, \\ \gamma(x) = c(x), \quad \delta(x) =& \frac{3}{2}\varepsilon, \quad \nu(x) = \frac{3}{2}c'(x). \end{aligned} \tag{24}$$

The equation can be rewritten up to an $\mathcal{O}(\beta\mu^2)$ as

$$(1 - \alpha\partial_{xx})(\partial_t \eta + (\gamma + \delta\eta)\partial_x \eta + \nu\eta) + \omega \partial_{xxx}\eta = 0, \tag{25}$$

to better match (2). We can indeed define $S(u) = \nu\eta$ to the previous formulation.

As before, a conservation law for the energy (21) can only be shown within an $\mathcal{O}(\beta, \epsilon\mu^2, \mu^4)$.

3.3. Full order model discretization

We discretize (23) using a finite difference (FD) approach. The resulting discrete model will be denoted by full order model (FOM). Classically, we consider a discretization of the spatial domain $\Omega = \cup_{i=1}^{N_h}[x_i, x_{i+1}]$ into N_h equi-spaced intervals, and define $\Delta x := x_{i+1} - x_i$. The temporal domain $[0, T]$ is subdivided into K time steps $[0, T] = \cup_{k=1}^{K}[t^{k-1}, t^k]$,. We denote by u_i^k the numerical approximation of a quantity u in a point x_i at time t^k, and, with an abuse of notation, we use the same symbol for the continuous quantity $u : \mathbb{R}^+ \times \mathbb{R} \to \mathbb{R}$, and for the discrete one: $u \in \mathbb{R}^{N_h \times K}$. For convenience, we use the following notation for the classical centered finite difference operators $\mathbb{D}$, $\mathbb{D}_2$ and $\mathbb{D}_3 : \mathbb{R}^{N_h} \to \mathbb{R}^{N_h}$:

$$(\mathbb{D}u)_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x}, \quad (\mathbb{D}_2 u)_i = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}, \quad (\mathbb{D}_3 u)_i = \frac{u_{i+2} - 2u_{i+1} + 2u_{i-1} - u_{i-2}}{2\Delta x^3}. \tag{26}$$

We start by defining the IMEX first order Euler discrete approximation

$$\begin{cases} (\mathbb{I} - \alpha\mathbb{D}_2)\Phi^{n+1} + \omega\mathbb{D}_3\eta^n = 0, \\ \frac{\eta^{n+1} - \eta^n}{\Delta t} + (\gamma + \delta\eta^n)\mathbb{D}\eta^n + \nu\eta^n + \mathcal{J}(\eta^n, \lambda) = \Phi^{n+1}. \end{cases} \tag{27}$$

Here, the vector-by-vector multiplication is meant component-wise in $\eta^n\mathbb{D}\eta^n$. The term $\mathcal{J}(\eta^n, \lambda)$ is a high order numerical diffusion term, which depends also on the spectral radius λ, where $\lambda_i = \gamma + \delta|\eta_i|$. In this work we have chosen to use to this purpose an operator originated from the continuous interior penalty (CIP) approach [7] which can be written as

$$\mathcal{J}(v, \lambda)_j := d\Delta x^3 \left\{\lambda_{j+1}(\mathbb{D}_2 v)_{j+1} - 2\lambda_j(\mathbb{D}_2 v)_j + \lambda_{j-1}(\mathbb{D}_2 v)_{j-1}\right\} \tag{28}$$

where $d \in \mathbb{R}^+$ is a stabilization parameter, which in practice we have set to 1 in all numerical simulations. Also (27) can be reshaped into (6) simply defining $\mathbb{M}^\Phi = \mathbb{M}^u := \mathbb{I}$, $\mathbb{X}^t := \alpha\mathbb{D}_2$, $\mathbb{X}^x := \omega\mathbb{D}_3$, $\mathbb{F}(\eta) := (\gamma + \delta\eta)\mathbb{D}\eta + \mathcal{J}(\eta, \lambda)$ and $\mathbb{S}^{ex}(\eta) := \nu\eta$.

Starting from (27), we build a second order SSPRK(2,2) IMEX approximation. The time-step is computed from the CFL condition: $\Delta t \leq \texttt{CFL}\Delta x/(\max_i \lambda_i^n)$, with $\texttt{CFL} < 1$.

3.4. Benchmarks for the modified BBM-KdV equation

In this section we describe some problems that we will use to test the reduced algorithms.

3.4.1. Propagation of a periodic monochromatic wave

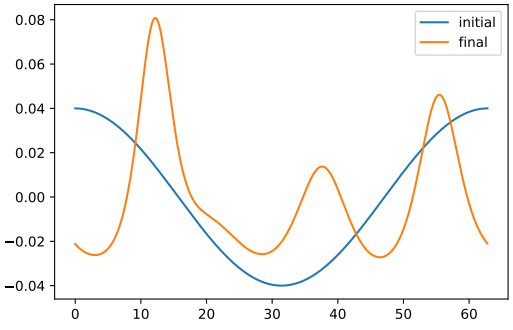
The first one starts with a long wave on the whole domain, i.e., a cosine on a periodic domain. This cosine deforms and splits into different waves that travel at different speeds. We use in (17) with the dimensionalized coefficients passed through (19): $h_0 = 1$, $g = 9.81$, $a_0 = 0.04$, $u_0 = a_0 \cos(x/10)$ on the domain $[0, 20\pi]$ and final time is $T = 200$. The FOM will use $N_h = 2000$ degrees of freedom and time step will be set with a `CFL` number of 0.2. In Figure 2(a) the initial and the final solution are depicted.

3.4.2. Undular bore propagation

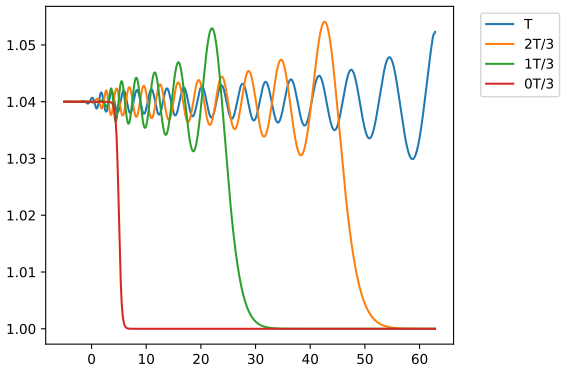
This test emulates a dam break which develops into several waves. The initial condition is

$$u_0 = a_0\left(1 - \frac{1}{1 + e^{-4(x-5)}}\right)$$

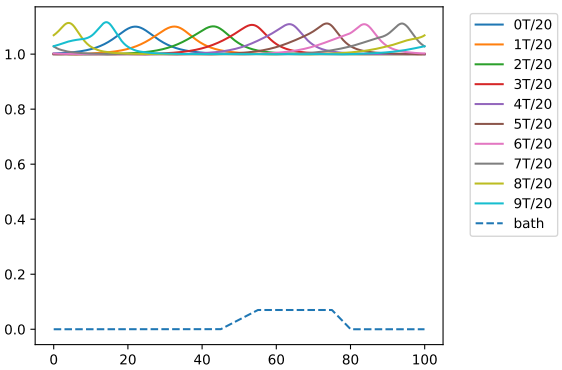
and the parameters in (14) are $h_0 = 1$, $g = 9.81$, $a_0 = 0.04$, domain $[0, 20\pi]$, $\texttt{CFL} = 0.1$ and final time $T = 20$. Initial and final simulations are shown in Figure 2(b).



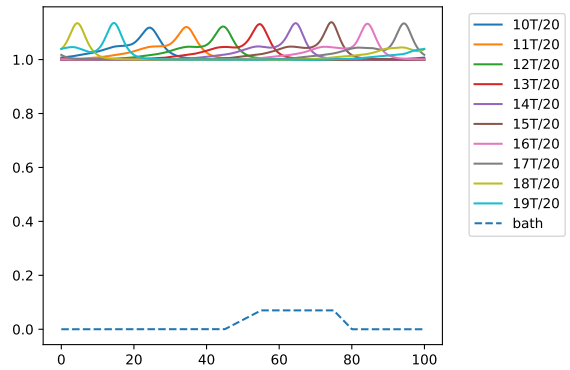
(a) Propagation of a periodic monochromatic wave



(b) Undular bore propagation



(c) Solitary waves interacting with a submerged bar until $T/2$



(d) Solitary waves interacting with a submerged bar from $T/2$ to T

Figure 2: Initial and final solution for tests.

Table 1: Estimate percentage of computational time for solving the linear systems for the three benchmark problems: comparison of the implementation of Thomas algorithm all in `numba` framework and sparse solver by `scipy` with fluxes computed in `numba`

	`numba` and Thomas			`scipy` and `numba`		
N_h	3.4.1	3.4.2	3.4.3	3.4.1	3.4.2	3.4.3
2000	86.4%	63.4%	85.2%	86.6%	84.4%	86.6%
4000	84.1%	62.4%	83.9%	86.2%	83.5%	88.8%
8000	86.1%	61.5%	85.0%	94.1%	85.3%	91.0%

Table 2: Estimate of ratio of computational time of the linear system over computational time of the nonlinear fluxes for the three benchmark problems: comparison of the implementation of Thomas algorithm all in `numba` framework and sparse solver by `scipy` with fluxes computed in `numba`

	`numba` and Thomas			`scipy` and `numba`		
N_h	3.4.1	3.4.2	3.4.3	3.4.1	3.4.2	3.4.3
2000	11.2	3.5	11.3	72.4	52.1	71.8
4000	11.8	3.7	11.6	73.4	54.9	74.3
8000	11.2	3.7	12.8	78.4	56.1	75.8

3.4.3. Solitary waves interacting with a submerged bar

This test consists of a soliton evolving on a ramp bathymetry defined by

$$b(x) = 0.07 \begin{cases} \frac{x-45}{10} & \text{if } 45 < x \leq 55, \\ 1 & \text{if } 55 < x \leq 75, \\ -\frac{x-80}{5} & \text{if } 75 < x \leq 80, \\ 0 & \text{else.} \end{cases}$$

The initial condition is defined by $u_0 = a_0 \cosh^{-1}(\frac{x-22}{5})$, final time is $T = 60$, the parameters are $h_0 = 1$, $g = 9.81$, $a_0 = 0.1$, `CFL` $= 0.1$ and the domain is $[0, 100]$. In Figures 2(c) and 2(d) the evolution of the solution in time is shown.

Now, we compare the computational times of the solution of the linear systems for the three benchmark problems with respect to other costs in the problem solution. In Table 1 we compare the costs of the linear systems with respect to the whole simulation, while in Table 2 we see the ratio of the cost of the solution of the linear systems over the cost of the nonlinear fluxes. Let us focus on Thomas algorithm first, we notice that boundary conditions make a huge difference. Indeed, periodic boundary conditions imply the solution of Thomas algorithm three times instead of just one and this increases the costs of the system solver.

On the other side, using a general solver for sparse system, like the one provided by `scipy`, increases the computational costs of the solver even by a factor of around 10. In particular, the distinction between different boundary conditions is less evident in this case. This algorithm is more interesting in perspective of a more general discretization or in case of multidimensional problem, but less efficient.

3.5. Projection based reduction

As said before, the computational operation in (27) that we mainly aim at reducing is the solution of the linear system, while the flux and source terms are easily parallelized.

As before, we introduce the reduced variable of dimension $N_{RB} \ll N_h$ so that we approximate

$$\eta^n \approx V \hat{\eta}^n, \qquad \hat{\eta}^n \in \mathbb{R}^{N_{RB}} \tag{29}$$

with the columns of $V \in \mathbb{R}^{N_h \times N_{RB}}$ containing the basis vectors, and $\hat{\eta}^n$ the vector of the reduced basis coefficients.

Two projection based model order reduction (MOR) strategies exist [19], equivalent under certain conditions. The first one is the projection with a test matrix $W \in \mathbb{R}^{N_h \times N_{RB}}$ and it is obtained by projecting the whole equation (27), i.e.,

$$W^T \left(V\hat{\eta}^{n+1} - r(V\hat{\eta}^n, \Phi^n(V\hat{\eta}^n)) \right) = 0, \tag{30a}$$

$$r(\eta, \Phi) = \eta - \Delta t \left((\gamma + \delta\eta)\mathbb{D}\eta + \mathcal{J}(\eta, \lambda) - \Phi \right). \tag{30b}$$

This opens the question on which matrix W we should consider. If we choose $W = V$ we obtain a Galerkin projection, while different choices lead to Petrov-Galerkin projections.

An alternative approach, allowing to answer this question, is to construct the reduced space using a minimization problem. Using in particular the scalar product (22), we can for example define

$$\hat{\eta}^{n+1} = \arg\min_x \left\langle Vx - r(V\hat{\eta}^n, \Phi^{n+1}(V\hat{\eta}^n), Vx - r(V\hat{\eta}^n, \Phi^n(V\hat{\eta}^n) \right\rangle_{\mathcal{E}}. \tag{31}$$

Using the discrete form of (22) (in dimensional variables), we can introduce the scalar product matrix

$$\Theta := \Delta x(\mathbb{I} + \alpha\mathbb{D}^T\mathbb{D}),$$

where $\mathbb{I} \in \mathbb{R}^{N_h \times N_h}$ is the identity matrix and $\mathbb{D} \in \mathbb{R}^{N_h \times N_h}$ is the (first) derivative matrix, and such that at the discrete level

$$\langle u, v \rangle_{\mathcal{E}} = u^T \Theta v.$$

Then, we can write the minimization (31) as the solution of

$$2V^T\Theta V\hat{\eta}^{n+1} - 2V^T\Theta r(V\hat{\eta}^n, \Phi^{n+1}(V\hat{\eta}^n)) = 0, \tag{32a}$$

suggesting that minimizing the energy in the projected space requires testing with $W = \Theta^T V$ in (30a).

3.5.1. Approximation accuracy of the problems

For the benchmarks involving the modified BBM-KdV equation we have applied the POD on 1000 snapshots in time, with fixed parameters. The FOM dimension is here $N_h \approx 10^3$, which is a representative order of magnitude. In practice, we have used 2000 points. Concerning the potential for reduction, we need to look at the error decay in terms of singular values. For the tests considered here, the decay has been plotted in Figures 3(a) to 3(c). As we can see, the energy drop is not exceedingly fast, still being exponential from the very beginning. In particular, for compact travelling waves as in the last benchmark, the advection dominated regime slows down the convergence of the eigenvalues. Nevertheless, if we want to keep the error of the order of 10^{-2} or 10^{-3} the number of eigenmodes needed is in between 50 and 70, which is much below the typical mesh size used in the FOM and can already reduce the computational cost of the linear system (even if full). We remark that the computational reduction is not proportional to the dimension as the original FOM has a sparse linear system to be solved, while the ROM has a full linear system.

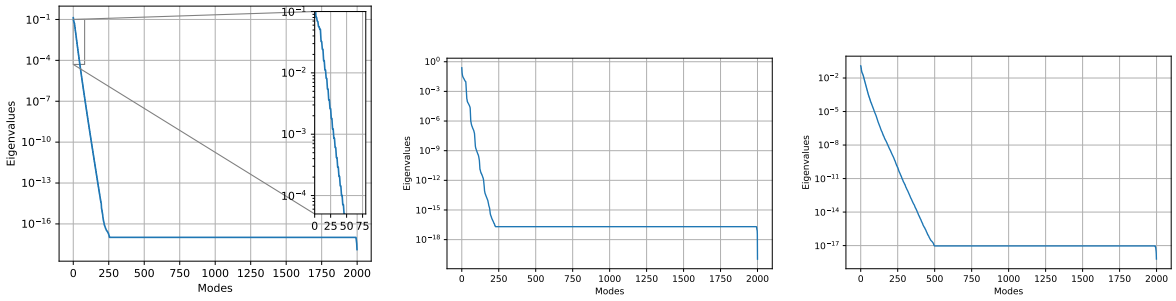
3.5.2. Reduction of the linear operator

We consider here the reduction of the linear system which gives Φ in (27). In general this operation can scale as an $\mathcal{O}({N_h}^2)$, unless particularly optimized algorithms are implemented. Indeed, in our FOM simulations we will use Thomas algorithm which only scales as $\mathcal{O}(N_h)$. When the linear operator is the only one reduced, we still need to assemble the discretization of the nonlinear part of the PDE. For this reason we will refer to this approach as to a partial discrete reduced order model or pdROM.

The pdROM discrete evolution equation can be written as

$$W^T V\hat{\eta}^{n+1} = W^T V\hat{\eta}^n - W^T \Delta t \left((\gamma + \delta V\hat{\eta}^n)\mathbb{D}V\hat{\eta}^n + \mathcal{J}(V\hat{\eta}^n, \lambda) \right) + \Delta t W^T \Phi^{n+1}, \tag{33a}$$

$$W^T \Phi^{n+1} = -\omega W^T (\mathbb{I} - \alpha\mathbb{D}_2)^{-1}\mathbb{D}_3 V(\hat{\eta}^n + \nu\hat{\eta}^n). \tag{33b}$$



(a) Propagation of a periodic monochromatic wave

(b) Undular bore propagation

(c) Solitary waves interacting with a submerged bar

Figure 3: Singular values decay

Also here, we can see the connection with (8), where all the terms can be matched. In practice, for N_{RB} sufficiently small, the $N_{RB} \times N_{RB}$ matrix $A_{N_{RB}} := W^T(\mathbb{I} - \alpha\mathbb{D}_2)^{-1}\mathbb{D}_3 V$ can be pre-computed and stored. Denoting with $M_{N_{RB}} := W^T V \in \mathbb{R}^{N_{RB}\times N_{RB}}$, the pdROM update can be effectively implemented as

$$\hat{\Phi}^{n+1} = -\omega A_{N_{RB}}^{-1}(\hat{\eta}^n + \nu\hat{\eta}^n), \tag{34a}$$

$$\hat{\mathbb{F}}(\eta) = W^T\left((\gamma + \delta\eta)\mathbb{D}\eta + \mathcal{J}(\eta, \lambda)\right), \tag{34b}$$

$$\hat{\eta}^{k+1} = \hat{\eta}^n + \Delta t M_{N_{RB}}^{-1}\left(-\hat{\mathbb{F}}(V\hat{\eta}^n) + \hat{\Phi}^{n+1}\right). \tag{34c}$$

It is clear that, with this approach, the parameters γ, δ, ω can be easily modified in the online computations. Also α can be decoupled from the reduction process and used to assemble the linear reduced system, but in this context α represent the topography parameters and we do not aim to change it. In the system case we will see how to modify a parameter in the LHS matrix.

Remark 3 (Modification to nondispersive code)**.** *It is very important to stress that the computational results, as well as all conclusions concerning cost reduction can be obtained keeping a classical solver for nondispersive nonlinear equations, to which we add the reduced dispersive term, i.e.,*

$$\eta^{k+1} = \eta^n - \Delta t\left((\gamma + \delta\eta^n)\mathbb{D}\eta^n + \nu\eta^n + \mathcal{J}(\eta^n, \lambda)\right) + \Delta t\beta V\hat{\Phi}^{n+1}, \tag{35a}$$

$$\hat{\Phi}^{n+1} := -(W^T(\mathbb{I} - \alpha\mathbb{D}_2)V)^{-1}W^T\mathbb{D}_3\eta^n, \tag{35b}$$

where $B_{N_{RB}} := W^T(\mathbb{I} - \alpha\mathbb{D}_2)V \in \mathbb{R}^{N_{RB}\times N_{RB}}$ *can be stored and inverted at low computational cost.* Nevertheless, working on two different spaces contemporary does not guarantee the same accuracy as in the previously presented algorithm. We have observed that reducing and projecting only the Φ term leads to much more inaccurate methods. We believe that this is due to spurious modes that the reconstruction of Φ generates and that are not appropriately damped in the full dimensional space, while they are in the pdROM approach. In the test section we will show that both algorithms have comparable computational costs, but (34) provides more stability.

Remark 4 (EIMROM stability)**.** *As shown in Section 2.2.2, we can reduce the computational costs of the source and flux terms with a hyper-reduction technique. The extra reduction of the EIM algorithm does not guarantee anymore the energy minimization* (14)*. This is noticeable when the dimension of the EIM space* N_{EIM} *is smaller or close to the dimension of the RB space* N_{RB}*. In that regime, it is easy to see instabilities and Runge phenomena. In the simulation section, we will observe in which regime we should stay with the dimension* N_{EIM}*.* Other works already shown that the dimension of the EIM space should be larger that the reduced space one to obtain stability of the reduced model [36, 49, 17, 2, 10]. Some novel algorithms have been proposed in recent papers [10, 36] based on different techniques as the

over-collocation method and variants of the Gappy-POD. We will not investigate in deep this behavior in this work.

Remark 5 (Fully implicit)**.** *We want to remark that even for the pdROM simulations, even if the solution is energy stable up to an $\mathcal{O}(\varepsilon\mu^2, \mu^4)$, oscillations may arise due to the under-resolution in the reduced space. This would not lead to blow ups of the solution, but it might have some unphysical behaviors. In the case of the EIM algorithm the oscillations might also lead to instabilities and blow ups when the EIM space does not properly approximate the nonlinear flux manifold. In order to alleviate this issue, it might be helpful to consider a fully implicit discretization or, even better, an implicit Euler time discretization. On the other side, this would lead to extra computational costs due to nonlinear solvers that might heavily slow down the reduced formulation, both in the pdROM and EIMROM contexts. In particular, the ROM computational costs with respect of the FOM ones, where the benchmark algorithms are explicit for the hyperbolic equation, might be disadvantageous, at least for one dimensional simulations.*

4. Simulations for KdV-BBM

4.1. Modus operandi

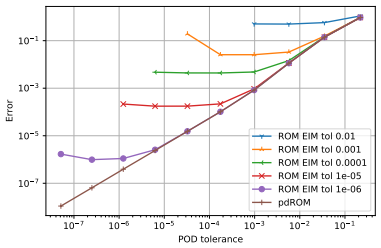
In this section, we will test the pdROM and EIMROM in different regimes, showing their error and the computational costs with respect to the FOM solutions.

1. We will proceed studying a time-only manifold of solutions, in order to understand which errors our algorithms can reach with different dimensions of reduced basis spaces. In this first phase, we will fix the parameters of the problem, we will compute the FOM solution on a mesh with $N_h = 2000$ points and we will collect 1000 snapshots in time $u(t^k)$ and the respective nonlinear fluxes $F(u(t^k))$. These snapshots will form the training set used to choose the reduced basis space and the EIM space, setting either the dimension of the space or the tolerance. Then, different simulations of the original problem are run with pdROM and EIMROM, recording their errors and their computational times. We will picture also one significant simulation of the reduced algorithms for a visual comparison for each test.
2. The second stage will consider a solution manifold generated by time and parameters. In practice, we will vary h_0, on which all the parameters depend in the BBM-KdV problem, see (24). In particular, we build the training sample using 10 randomly sampled $h_0 \in [h_{min}, h_{max}] = [0.7\bar{h}_0, 1.3\bar{h}_0]$, and for each of these parameters, we collect 1000 snapshots at uniform timesteps. We populate the ROM and EIM spaces using these 10.000 snapshots and their fluxes. Then we test the reduced algorithms for $\bar{h}_0$ for a larger final time (for every test we will specify which one), checking again how the reduced procedures perform with different tolerances for EIM and POD. Finally, we test the methods for a parameter $h_0 = 0.63 \cdot \bar{h}_0$ outside the training set and, when sensible, with a different initial condition (for every test we will specify which one), to assess the property of the reduced algorithms in the extrapolation regime.

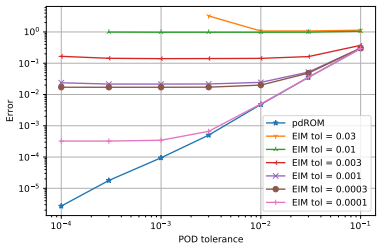
4.2. Propagation of a periodic monochromatic wave

For this test, we use $\bar{h}_0 = 1$ and final time $T = 200$. This test is fairly simple and it can be well compressed with few basis functions. Nevertheless, it already shows what can be achieved with and without EIM. We collect samples for different times of one simulation obtaining a time-only reduction. In Figure 4(a) the decay of the relative error is exponential with respect to the dimension N_{RB} of the reduced space (different points correspond to $N_{RB} = 10k$ for $k = 1, \dots, 10$) and with few basis functions we can reach a very small error. On the other side, this error could be worsen or even the simulation might not run until the final time if we add the EIM algorithm with not enough basis functions. The error for those methods stagnates around the tolerance of EIM and if $N_{EIM} \lessapprox N_{RB}$ the method becomes unstable.

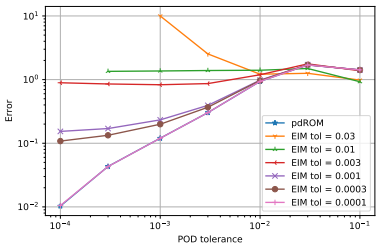
The computational time for the KdV-BBM pdROM compared to the FOM using Thomas method is between 20% and 30% in Figure 4(d). The hyper-reduction allows to further reduce the in between 4%



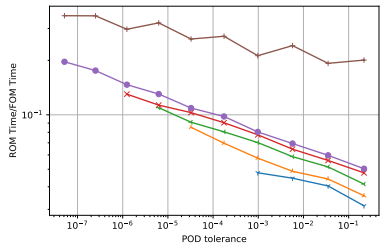
(a) EIM ROM errors time-only reduction



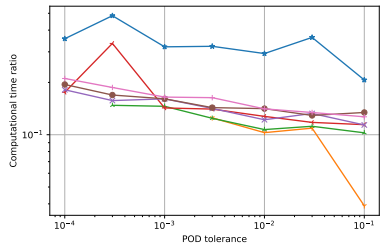
(b) ROM errors time-parameter reduction on parameter and η_0 in the training set



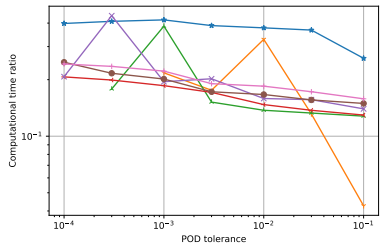
(c) ROM errors time-parameter reduction on parameter and η_0 outside the training set



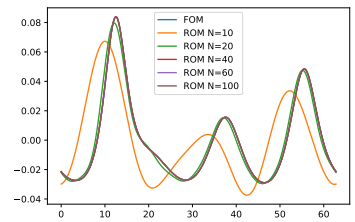
(d) EIM ROM computational times time-only reduction (legend above)



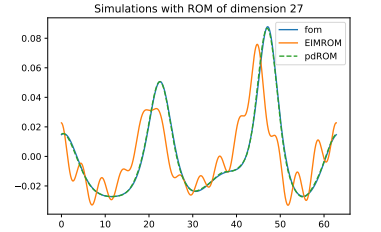
(e) ROM computational time time-parameter reduction on parameter and η_0 in the training set (legend above)



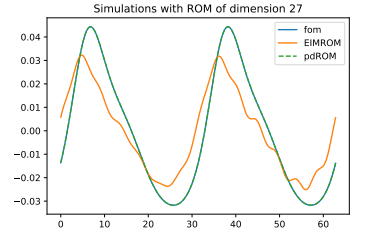
(f) ROM computational time time-parameter reduction on parameter and η_0 outside the training set (legend above)



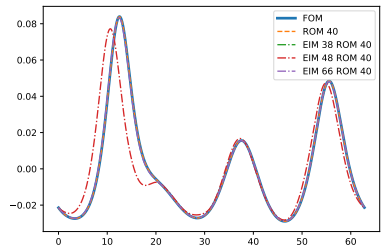
(g) ROM solutions time-only reduction



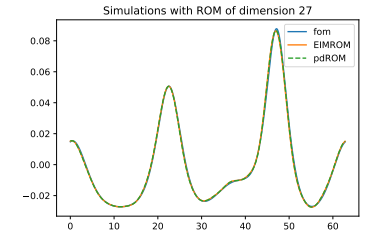
(h) ROM solutions time-parameter reduction with $tol_{POD} = 0.3$, $N_{RB} = 27$ and $tol_{EIM} = 0.03$, $N_{EIM} = 72$ on parameter and η_0 in the training set



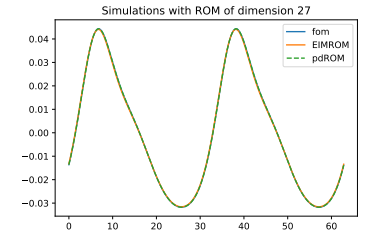
(i) ROM solutions time-parameter reduction with $N_{RB} = 27$ and $N_{EIM} = 72$ on parameter and η_0 outside the training set



(j) ROM solutions time-only reduction



(k) ROM solutions time-parameter reduction with $tol_{POD} = 0.3$, $N_{RB} = 27$ and $tol_{EIM} = 0.0003$, $N_{EIM} = 119$ on parameter and η_0 in the training set



(l) ROM solutions time-parameter reduction with $N_{RB} = 27$ and $N_{EIM} = 119$ and $tol_{EIM} = 0.0003$, $N_{EIM} = 119$ on parameter and η_0 outside the training set

Figure 4: **Propagation of a periodic monochromatic wave.** pdROM and EIM reduction: simulation, errors and computational time, varying POD and EIM dimensions

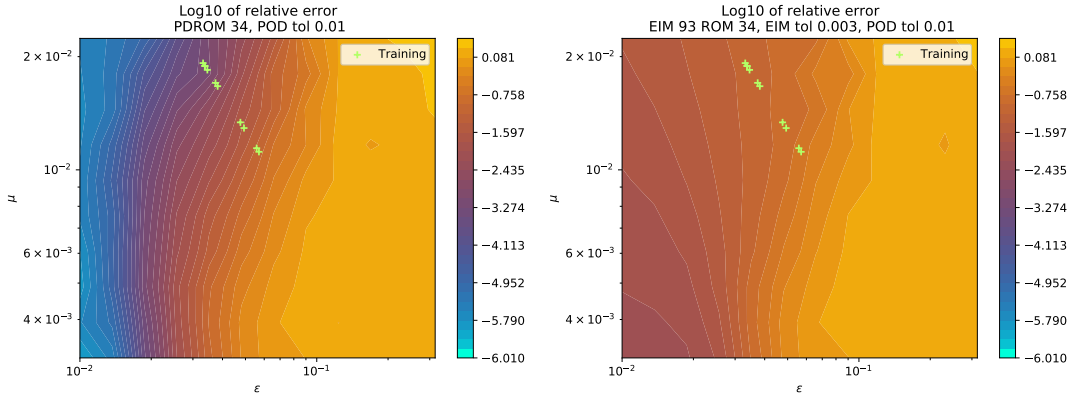


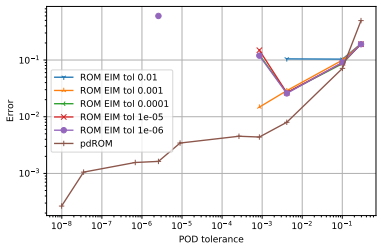
Figure 5: **Propagation of a periodic monochromatic wave.** Error plot varying the parameters a_0 and h_0 for pdROM $N_{RB} = 34$ and EIMROM with $N_{EIM} = 93$

and 10% of the original computational time. In Figure 4(g) we see the simulations run with pdROM with different dimensions of the reduced space. Already with $N_{RB} = 20$ we obtain an accurate solution. In Figure 4(j) we see that a minimum amount of EIM basis is necessary not to have unstable solutions and to be accurate.

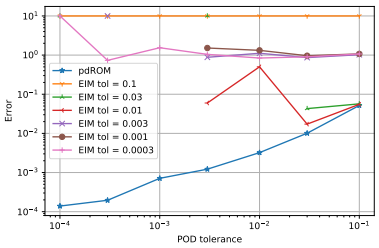
For the second phase, we introduce the reduction in the parameter space. For Figures 4(b), 4(e), 4(h) and 4(k), we test the obtained reduced spaces and the respective algorithms onto the parameter $h_0 = 1$ and for final time $T = 250$. The error decay in Figure 4(b) is similar to the one of the time-only reduced case. The largest differences stays in the differences of the number of basis functions needed to reach a certain tolerance. In the example in Figure 4(h), with $N_{RB} = 27$ we reach a tolerance of 0.3. With time-only reduction $N_{RB} = 10$ are enough to reach the same tolerance, though not reaching the same accuracy in the simulation see Figure 4(a). The same holds for the EIM algorithm. This is noticeable in the computational times, that increase for all algorithms given certain thresholds, see Figures 4(d) and 4(e), this is a measure of the increased difficulty of the problem. In Figure 4(h) a representative simulation is shown, where we see that with $N_{RB} = 27$ basis function we get an already accurate approximation with pdROM, while $N_{EIM} = 72$ is still not enough to obtain a good accuracy for EIMROM. Increasing $N_{EIM} = 119$ in Figure 4(k), we can obtain an accurate solution also for EIMROM. This is accordance with the observations made in [36, 49, 17, 2, 10]. In this case, the computational time of EIMROM is 3 times faster than the pdROM, and decreasing tol_{EIM} we can achieve even smaller errors. A particular attention must be observed in choosing the tolerance of EIM small enough with respect to tol_{POD}, as one can see in Figure 4(b) that the EIMROM simulations can explode if tol_{EIM} is not small enough.

In Figures 4(c), 4(f), 4(i) and 4(l) we test the algorithms with $h_0 = 0.63 \notin [h_{min}, h_{max}]$, $a_0 = 0.05$ instead of 0.04 and initial conditions $u_0(x) = -1.2a_0 \cos(\frac{4}{10}\pi(x-2)) + \frac{a_0}{10}\cos(\frac{8}{10}\pi x)$. Similar conclusion can be drawn also in this case. The main difference is a larger error, due to the higher wave amplitude of the solution.

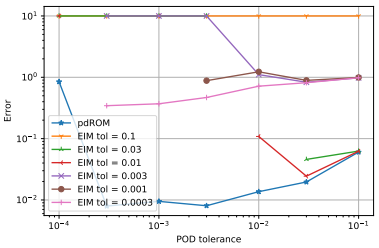
In Figure 5 we can see the relative error of pdROM $N_{RB} = 34$ and EIMROM with $N_{EIM} = 93$ varying a_0 and h_0 in the initial conditions and the problem settings. We plot the error as a function of ε and μ, in order to understand how the nonlinearity and the dispersion affect the solutions. We know that as h_0 gets smaller or a_0 gets bigger (ε larger), the nonlinearity of the problem becomes more pronounced, hence leading to more oscillations and steeper gradients. In the strongly nonlinear regime ($\varepsilon = a_0/h_0 \gtrapprox 10^{-1}$) the hypothesis of having small ε are not valid anymore and the error increases rapidly. On the other side, the dispersion coefficient $\mu = h_0^2/L^2$ does not affect much the error for a fixed ε, in Figure 5. Overall we can see that, as long as nonlinearity does not become predominant, the pdROM provides lower errors over the parameter space showing potential for a more robust approximation outside the training set.



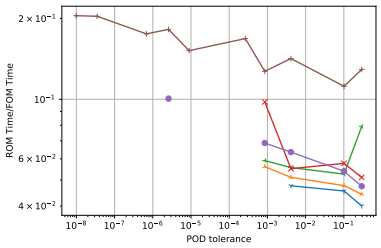
(a) EIM ROM errors time-only reduction



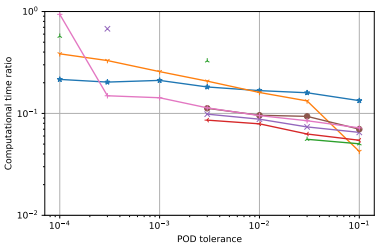
(b) ROM errors time-parameter reduction on parameter and η_0 in the training set



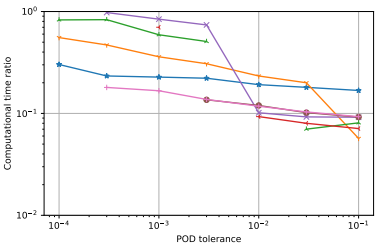
(c) ROM errors time-parameter reduction on parameter and η_0 outside the training set



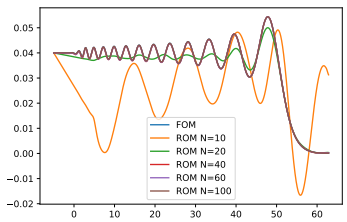
(d) EIM ROM computational times time-only reduction (legend above)



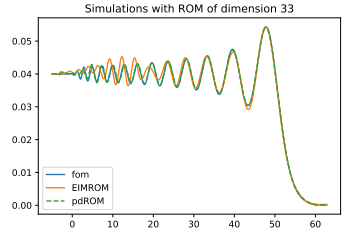
(e) ROM computational time time-parameter reduction on parameter and η_0 in the training set (legend above)



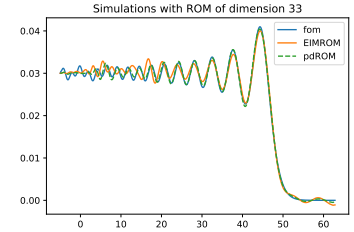
(f) ROM computational time time-parameter reduction on parameter and η_0 outside the training set (legend above)



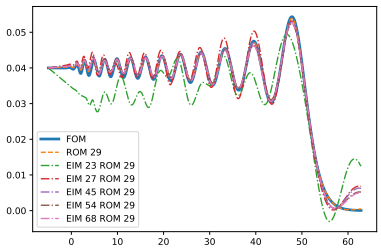
(g) ROM solutions time-only reduction



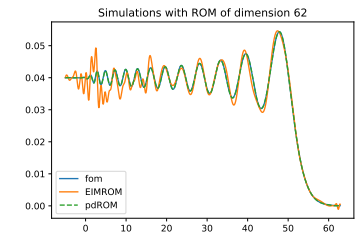
(h) ROM solutions time-parameter reduction with $tol_{POD} = 0.03$, $N_{RB} = 33$ and $tol_{EIM} = 0.03$, $N_{EIM} = 52$ on parameter and η_0 in the training set



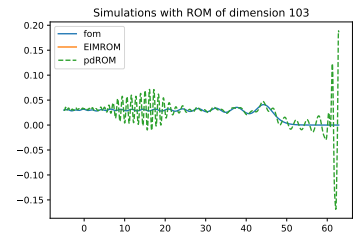
(i) ROM solutions time-parameter reduction with $tol_{POD} = 0.03$, $N_{RB} = 33$ and $tol_{EIM} = 0.03$, $N_{EIM} = 52$ on parameter and η_0 outside the training set



(j) EIMROM solutions time-only reduction



(k) ROM solutions time-parameter reduction with $tol_{POD} = 0.003$, $N_{RB} = 62$ and $tol_{EIM} = 0.01$, $N_{EIM} = 77$ on parameter and η_0 in the training set



(l) ROM solutions time-parameter reduction with $tol_{POD} = 0.0001$, $N_{RB} = 103$ on parameter and η_0 outside the training set

Figure 6: **Undular bore propagation.** pdROM and EIM reduction: simulation, errors and computational time, varying POD and EIM dimensions

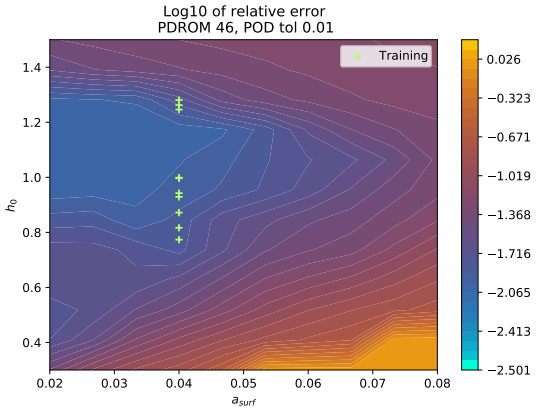


Figure 7: **Undular bore propagation.** Error plot varying the parameters a_0 and h_0 for pdROM $N_{RB} = 46$

4.3. Undular bore propagation

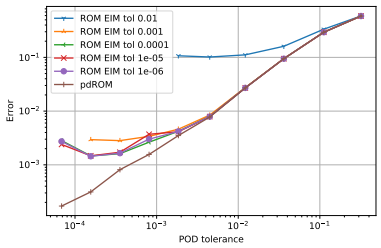
This is a very complex test for model reduction. We start by choosing $\bar{h}_0 = 1$, $a_0 = 0.04$ and final time $T = 20$. In order to take into account the non homogeneous Dirichlet boundary condition at the left boundary, we subtract from the method the residual equation computed on a lift solution u_{lift} that verify the Dirichlet boundary conditions. In this way, the resulting scheme can be applied with homogeneous Dirichlet boundary condition on the left.

This tests is way more challenging than the previous one, in particular because flat zones and oscillating ones coexist and move around the domain. In particular, the EIM method easily fails in capturing this behavior without falling into Gibbs phenomena in the flat area. Indeed, in Figure 6(a) we see the error decay for only-time reduction both for EIMROM and pdROM and the former become very oscillatory when too many POD bases are used. The pdROM uses between 10% and 20% of the FOM computational time, while the EIMROM, in the few situations where it does not explode, uses around 5% of the FOM time, see Figure 6(d). In Figure 6(g) we see for different POD basis functions how we approach the exact solution, while in Figure 6(j) we fix the number of POD basis functions very low and we change the EIM ones. In this stable situation as the EIM bases increase we slowly converge towards the pdROM solution, even if the right part of the solution struggles with obtaining the right behavior.

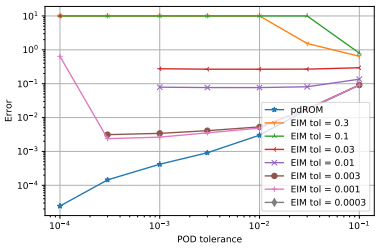
When exploring the parameter domain in the training set $h_0 \in [0.7, 1.3]$, we observe a similar situation. The decay of the error on a parameter inside the training domain computed at time $T = 15$ for pdROM decays exponentially in Figure 6(b), while EIMROM is affected by oscillations and struggles with obtaining reasonable results. The computational time of the pdROM is between the 15% and the 25% of the FOM computational time as shown in Figure 6(e), while the EIMROM, when the simulation is reasonable for few basis functions, needs only around the 5% of the FOM computational time. In Figure 6(h) we see for a tolerance of the POD of $3 \cdot 10^{-2}$ we need $N_{RB} = 33$ and the pdROM is quite close to the FOM solution, while EIMROM with same tolerance and $N_{EIM} = 52$ start oscillating around the original position of the dam. In Figure 6(k) the situation worsen for the EIMROM which becomes very oscillatory for $N_{RB} = 62$ and $N_{EIM} = 77$ and this explains why the error is so large.

Taking the parameter out of the training set, considering $h_0 = 0.63$ and $a_0 = 0.03$, the EIMROM becomes even more oscillatory and its error often explodes, but also the pdROM suffer of the larger flat zone in the part of the domain, leading to instabilities. The error decay is indeed affected by this and when too many POD bases are considered the error grows, see Figure 6(c) and the simulation in Figure 6(l). Computational times stays similar to the previous case Figure 6(f), and, if properly choosing the number of basis functions, both pdROM and EIMROM gives decent solutions, though the EIMROM oscillates around the flat area, see Figure 6(i).

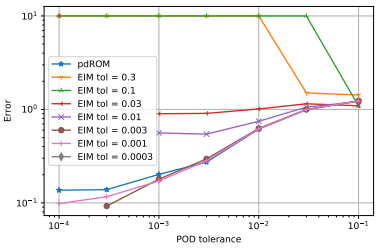
Clearly this test case is pushing over the limit of the method as the advection dominated part is leading the computations and the contrast between flat and oscillating region leads very easily to Gibbs



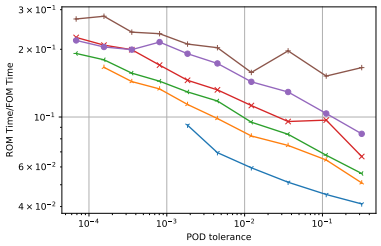
(a) EIM ROM errors time-only reduction



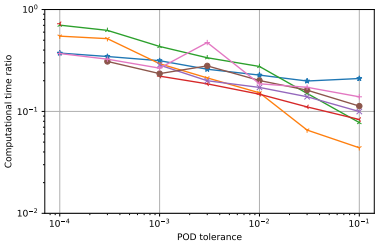
(b) ROM errors time-parameter reduction on parameter and η_0 in the training set



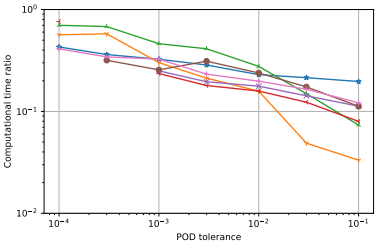
(c) ROM errors time-parameter reduction on parameter and η_0 outside the training set



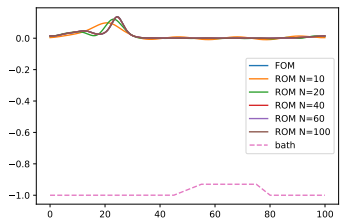
(d) EIM ROM computational times time-only reduction (legend above)



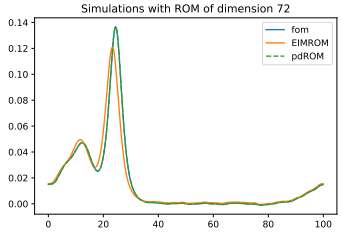
(e) ROM computational time time-parameter reduction on parameter and η_0 in the training set (legend above)



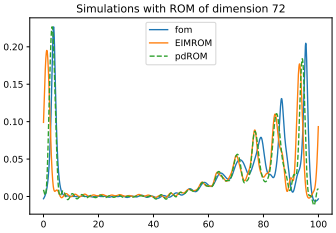
(f) ROM computational time time-parameter reduction on parameter and η_0 outside the training set (legend above)



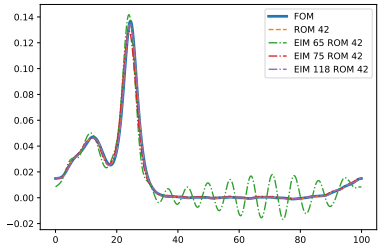
(g) ROM solutions time-only reduction



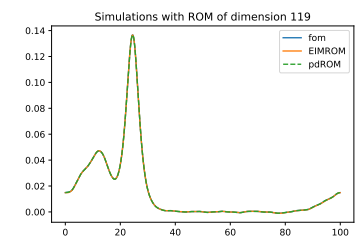
(h) ROM solutions time-parameter reduction with $tol_{POD} = 0.01$, $N_{RB} = 72$ and $tol_{EIM} = 0.03$, $N_{EIM} = 152$ on parameter and η_0 in the training set



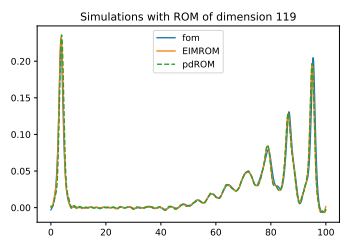
(i) ROM solutions time-parameter reduction with $tol_{POD} = 0.01$, $N_{RB} = 72$ and $tol_{EIM} = 0.03$, $N_{EIM} = 152$ on parameter and η_0 outside the training set



(j) EIMROM solutions time-only reduction



(k) ROM solutions time-parameter reduction with $tol_{POD} = 0.001$, $N_{RB} = 119$ and $tol_{EIM} = 0.001$, $N_{EIM} = 259$ on parameter and η_0 in the training set



(l) ROM solutions time-parameter reduction with $tol_{POD} = 0.001$, $N_{RB} = 119$ and $tol_{EIM} = 0.001$, $N_{EIM} = 259$ on parameter and η_0 outside the training set

Figure 8: **Solitary waves interacting with a submerged bar.** pdROM and EIM reduction: simulation, errors and computational time, varying POD and EIM dimensions

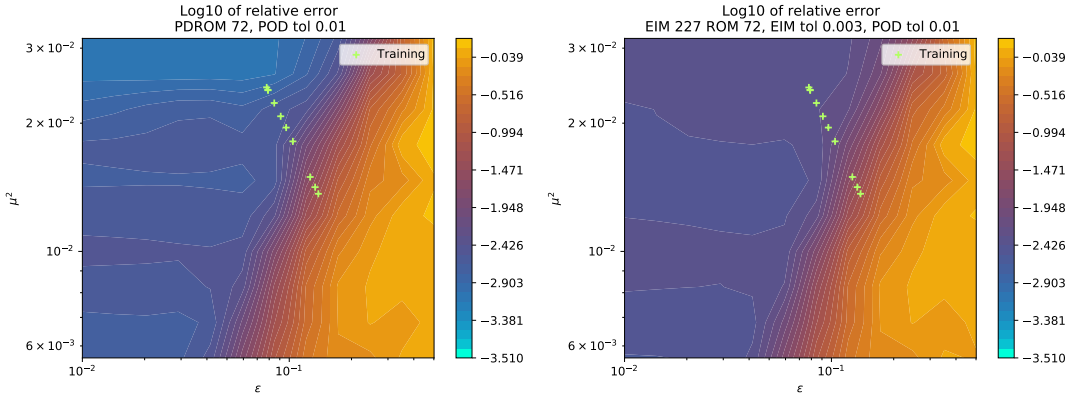


Figure 9: **Solitary waves interacting with a submerged bar.** Error plot varying the parameters a_0 and h_0 for pdROM $N_{RB} = 72$ and EIMROM with $N_{EIM} = 227$

phenomena. Hence, it is important to stay close to the training set area for this case and not to introduce extra interpolation error with the EIM.

In Figure 7 we see how the error of the pdROM with $N_{RB} = 46$ varies with respect to a_0 and h_0. We can see a strong gradient when h_0 and a_0 increase in the area of unstable solutions. Again, the main factor here is the relevance of nonlinearity. As long as the waves remain dispersive and we are far from very shallow bores, we observe an error plateau which seems to indicate a certain robustness with respect to the variation of parameters. It must be remarked however, that outside this area, the error grows relatively fast also for pdROM. Some improved treatment of nonlinearity is necessary to handle this kind of problems when getting close the the shallow/highly nonlinear range.

4.4. Solitary waves interacting with a submerged bar

For this test, we use $\bar{h}_0 = 1$, final time $T = 60$ and $a_0 = 0.1$. This test has similar results to the **Propagation of a periodic monochromatic wave**. Its advection dominated character is more pronounced than in the **Propagation of a periodic monochromatic wave**, but the error decays quickly enough as we can see in Figure 8(a). The pdROM solution with $N_{RB} = 40$ basis function is indistinguishable from the FOM one in Figure 8(g) and, adding the EIM approximation, we can see different results. In Figure 8(j) for the EIMROM simulation with $N_{RB} = 42$ and $N_{EIM} = 65$ we still do not have enough accuracy and the solution oscillates, with $N_{EIM} = 75$ the solution is much more stable, but we can still see differences in the peak of the wave, finally for $N_{EIM} = 118$ observe a very accurate solution. Similarly to before, the computational costs scale between 15% and 30% of FOM for pdROM and between 5% and 25% for EIMROM, see Figure 8(d).

In the second phase, we consider also here 10 simulations for randomly chosen $h_0 \in [0.7\bar{h}_0, 1.3\bar{h}_0]$ and we use them to compute the reduced space with the POD. The error decay with respect to the POD tolerance in Figure 8(b) is similar to the first phase, but the dimension of the spaces are larger as we notice in the computational time plot in Figure 8(e) up to a 40% of the FOM computational time for pdROM. In Figures 8(h) and 8(k) we see accurate pdROM simulations with $N_{RB} = 72$ for a POD tolerance of 0.01 and a computational time of around 25% of the FOM time, while we need $N_{EIM} = 259$ to get very accurate EIMROM solutions for computational costs around 20% of the FOM ones. Again, the EIM space should be very large in order to obtain accurate and stable results [36, 49, 17, 2, 10].

When testing on a parameter outside the training domain, i.e., $a_0 = 0.15$ and $h_0 = 0.63$, we have that the exact solution is more oscillatory, see Figures 8(i) and 8(l), and to catch properly the solution we need more basis functions in both spaces. Indeed, the errors are larger in Figure 8(c), while the computational times stays similar.

In Figure 9 we see how the error varies for pdROM with $N_{RB} = 72$ and EIMROM with $N_{EIM} = 227$ for different parameters h_0 and a_0, plotting the error with respect to ε and μ. As before we see that

nonlinearity really plays a major role, with the shallow water/hyperbolic limit being poorly handled by both approaches. Away from this limit (left part of the figures) the EIMROM error is larger than the pdROM one.

We remark that the computational cost of the linear system is the largest one in the FOM and, while doing a pdROM, we obtain already a great reduction. In Appendix Appendix B we compare the pdROM costs with the reduction of only the linear system for Φ (evolving the full FOM) as suggested in Remark 3. Essentially, the cost of the two methods are identical. Moreover, changing the linear solver into something more generic results in an even stronger reduction of the computational costs. We compare the costs of the Thomas algorithm and of a generic sparse solver in Appendix Appendix B.

5. An extension to a Boussinesq system

We propose in this section a possible extension to well known dispersive shallow water model by Madsen and Sørensen [30]. In the dimensional form the model equations read [30, 38]

$$\begin{cases} \partial_t \eta + \partial_x q = 0, \\ \partial_t q - \mathcal{T}^t[\partial_t q] + \partial_x(qu) + gh\partial_x \eta - \mathcal{T}^x[\eta] = 0, \end{cases} \tag{36}$$
$$\text{with } \mathcal{T}^t[\cdot] := B_1 \bar{h}^2 \partial_{xx}[\cdot] + \frac{1}{3}\bar{h}\partial_x \bar{h}\partial_x[\cdot] \text{ and } \mathcal{T}^x[\cdot] := gB_2\bar{h}^2\left(2\partial_x \bar{h} + \bar{h}\partial_x\right)\partial_{xx}[\cdot],$$

where, with the notation of Figure 1, $\eta(x,t)$ is the water level variation from the reference value, $q(x,t)$ is the discharge, $b_0 b(x)$ is the bathymetry profile, h_0 is the reference value for water height, $\bar{h}(x) = h_0 - b_0 b(x)$ is the water height at rest, $h(x,t) = \eta(x,t) + \bar{h}(x)$ is the water height, $u(x,t)$ is the depth averaged speed and it is linked to q by $q(x,t) = h(x,t)u(x,t)$. System (36) provides a $\mathcal{O}(\varepsilon\mu^2, \mu^4)$ approximation of the full nonlinear wave equations. It contains the nonlinear terms belonging also to shallow water equations, and they are the only nonlinear terms of the model. Similarly to Boussinesq equation and to the previous scalar model there are other *linear* dispersion terms which scale with constants B_1 and B_2. The model constants B_1 and B_2 are set to $B_2 = 1/15$ and $B_1 = B_2 + 1/3$ to optimize the linear characteristics of the model (phase and shoaling) with respect to the full Euler equations [30].

As before, we manipulate the equations for the purpose of their numerical solution. First we recast the model in dimensionless form:

$$\begin{cases} \partial_t \eta + \partial_x q = 0, \\ \partial_t q - \mu^2\mathcal{T}^t[\partial_t q] + \varepsilon\partial_x(qu) + h\partial_x \eta - \mu^2\mathcal{T}^x[\eta] = 0, \end{cases} \tag{37}$$
$$\text{with } \mathcal{T}^t[\cdot] := B_1 \bar{h}^2 \partial_{xx}[\cdot] + \frac{1}{3}\bar{h}\partial_x \bar{h}\partial_x[\cdot] \text{ and } \mathcal{T}^x[\cdot] := B_2\bar{h}^2\left(2\partial_x \bar{h} + \bar{h}\partial_x\right)\partial_{xx}[\cdot].$$

We then isolate the nonlinear hyperbolic operator and write the system as

$$\begin{cases} \partial_t \eta + \partial_x q = 0, \\ (1 - \mu^2\mathcal{T}^t)[\partial_t q + \varepsilon\partial_x(qu) + h\partial_x\eta] + \mu^2\mathcal{T}^t[\varepsilon\partial_x(qu) + h\partial_x\eta] - \mu^2\mathcal{T}^x[\eta] = 0. \end{cases} \tag{38}$$

In this case we do not neglect any of the small order terms to keep the original model, often used in literature (see e.g. [6, 38, 3]) and references therein. The final model can be written as a system of 3 equations, given by

$$\begin{cases} \partial_t \eta + \partial_x q = 0, \\ (1 - \mu^2\mathcal{T}^t)[\psi] = \mu^2\left\{\mathcal{T}^x[\eta] - \mathcal{T}^t[\varepsilon\partial_x(qu) + h\partial_x\eta]\right\}, \\ \partial_t q + \varepsilon\partial_x(qu) + h\partial_x\eta = \psi. \end{cases} \tag{39}$$

Also for this model, we show the analogy with the model defined in (2). Indeed, if we consider

$$u := (\eta, q)^T, \quad \partial_x F(u) := \left(\partial_x q, \varepsilon\partial_x(qu) + h\partial_x\eta\right)^T, \quad S(u) = 0,$$
$$\mathcal{X}^x(u) := \left(0, -\mu^2\left\{\mathcal{T}^x[\eta] - \mathcal{T}^t[\varepsilon\partial_x(qu) + h\partial_x\eta]\right\}\right)^T, \quad \mathcal{X}^t(u) := \left(0, \mu^2\mathcal{T}^t\right)^T,$$

one obtains (39). For this model, one cannot show an exact conservation for a total energy. For other dispersive nonlinear systems one can show the conservation of an energy of the form $\mathcal{E} = \frac{gh^2}{2} + \frac{hu^2}{2} + \frac{h^3(u_x)^2}{6}$ [28]. Differently from the BBM-KdV equation, the relation between this energy and the main unknowns is nonquadratic, and it is not straightforward to use it to build the reduced basis and the test reduced space as the one studied before. Based on regularity requirements, we will instead investigate a formal extension of the method proposed in the scalar case.

5.1. Wave generation and boundary conditions

We will consider two families of solutions: solitary waves, monochromatic periodic waves. Exact solitary waves for (36) can be obtained assuming $\eta = \eta(x - Ct)$, and $q = C\eta$, which allows to obtain an ODE from the second equation in (36). The solution is uniquely defined given amplitude and depth at rest (a_0, h_0). In particular, setting $C_0^2 = gh_0$, the celerity C verifies the consistency condition

$$\frac{C^2}{C_0^2} = \frac{a_0^2}{2h_0^2} \frac{1 + \frac{a_0}{3h_0}}{\frac{a_0}{h_0} - \log(1 + \frac{a_0}{h_0})} \tag{40}$$

Details can be found in [38, 34]. The ODE is integrated here with a built-in solver in `scipy`.

To generate monochromatic waves, we have included an internal wave generator. Different definitions exist in literature [44, 32, 46, 47]. Following [46, 47], we add to the η equation a compact forcing term with periodic variation in time:

$$\partial_t(\eta + h_{\text{iwg}}) + \partial_x q = 0, \tag{41}$$

where the internal generator is defined by

$$h_{\text{iwg}}(x,t) := A_{\text{iwg}} f_{\text{iwg}}(x) \sin(\omega t), \tag{42}$$

with $\omega = 2\pi/T$ the frequency of the signal and T the period, f_{iwg} the spatial damping function

$$f_{\text{iwg}}(x) = \Gamma_{\text{iwg}} e^{-(x - x_{\text{iwg}})^2 / d_{\text{iwg}}^2}. \tag{43}$$

We refer to [38, 47] for the tuning of the parameters and we will use the following ones

$$\Gamma_{\text{iwg}} = 0.185\sqrt{g/h_0}T, \quad d_{\text{iwg}}^2 = \frac{g\alpha_{\text{iwg}}^2 h_0 T^2}{80}, \quad T = 2.525, \quad \alpha_{\text{iwg}} = 4, \tag{44}$$

and x_{iwg} the center of the generator and $\alpha_{\text{iwg}} \in [0, 4]$ is a parameter controlling the amplitude of the wave and β_{iwg} is just a rescaling factor. Also A_{iwg} is a rescaling factor that will be changed along the simulations to test the problem for different parameters.

Finally, outflow conditions are imposed through *sponge layers*. Following [38] these terms are written as diffusion operators which will be discretized implicitly and added to the mass matrix. At the continuous level they have the form

$$S(v) = -\nu \partial_{xx} v, \qquad \nu(x) := \begin{cases} n_1 \frac{1 - e^{n_2 \frac{x - X_{s1}}{X_{s2} - X_{s1}}}}{1 - e}, & X_{s1} \le x \le X_{s2}, \\ 0, & \text{else.} \end{cases} \tag{45}$$

Here n_1 and n_2 are two coefficients that should be tuned with the problem, we will use the values $n_1 = 10^{-3}$ and $n_2 = 10$ as suggested in [38]. The points X_{s1} and X_{s2} are fixed close to the boundary and X_{s2} will coincide with the boundary itself. Also here, both these linear source terms can be added in the formulation to match (2) defining $\mathbb{S}^{im}$ accordingly.

5.2. Full order discrete model

The approach used to discretize (36) is similar to the one used in the scalar case. We combine a linear continuous Galerkin method (see [38] and references therein) with an SSPRK(2,2) time integrator [18]. To introduce a little dissipation we again make use of a continuous interior penalty operator (CIP) [7]. The resulting discrete equations can be written as

$$\mathbb{M}\left((\boldsymbol{\eta}+h_{\text{iwg}})^{(s)}-\sum_{r=0}^{s-1}\rho_r^s(\boldsymbol{\eta}+h_{\text{iwg}})^{(r)}\right)+\Delta t\mathbb{S}\boldsymbol{\eta}^{(s)}=-\Delta t\sum_{r=0}^{s-1}\theta_r^s\left(\mathcal{N}^\eta(\boldsymbol{\eta}^{(r)},\boldsymbol{q}^{(r)})+\mathcal{J}(\boldsymbol{\eta}^{(r)},\lambda^{(r)})\right) \tag{46a}$$

$$(\mathbb{M}-\mathbb{T}^t)\boldsymbol{\psi}=-\sum_{r=0}^{s-1}\theta_r^s\left(\mathbb{T}^t\mathbb{M}^{-1}\mathcal{N}^q(\boldsymbol{\eta}^{(r)},\boldsymbol{q}^{(r)})+\mathbb{T}^x\boldsymbol{\eta}^{(r)}\right) \tag{46b}$$

$$\mathbb{M}\left(\boldsymbol{q}^{(s)}-\sum_{r=0}^{s-1}\rho_r^s\boldsymbol{q}^{(r)}\right)+\Delta t\mathbb{S}\boldsymbol{q}^{(s)}=\Delta t\mathbb{M}\boldsymbol{\psi}-\Delta t\sum_{r=0}^{s-1}\theta_r^s\left(\mathcal{N}^q(\boldsymbol{\eta}^{(r)},\boldsymbol{q}^{(r)})+\mathcal{J}(\boldsymbol{q}^{(r)},\lambda^{(r)})\right) \tag{46c}$$

where the index in bracket notation is to denote the Runge–Kutta stages, i.e., $\{\eta^{(s)},q^{(s)}\}_{s=1,2}$, and ρ_r^s, θ_r^s are the Runge–Kutta coefficients in the Shu–Osher formulation as in the previous sections. The matrix $\mathbb{M}\in\mathbb{R}^{N\times N}$ is the usual $\mathbb{P}^1$ mass matrix $\mathbb{M}_{ij}=\Delta x(1+3\delta_{ij})/6$, while the nonlinear operators $\mathcal{N}^\eta$ and $\mathcal{N}^q$ are given by

$$\begin{bmatrix}\mathcal{N}^\eta\\ \mathcal{N}^q\end{bmatrix}_j=\Delta x\mathbb{D}\begin{bmatrix}q\\ q^2/h\end{bmatrix}_j+\frac{g}{6}\begin{bmatrix}0\\ (2h_j+h_{j+1})(\eta_{j+1}-\eta_j)-(2h_j+h_{j-1})(\eta_j-\eta_{j-1})\end{bmatrix}.$$

The CIP stabilization $\mathcal{J}$ is defined exactly as in (28), with $\lambda_j=u_j+\sqrt{gh_j}$. The operators $\mathbb{T}^t$ and $\mathbb{T}^x$ are the finite element approximation of the corresponding $\mathcal{T}$ operators appearing in (36), while $\mathbb{S}$ is the linear diffusion operator obtained from the discretization of the sponge layer terms. These are reported in Appendix A for completeness.

Also for this discrete model we can see the matching with (6), with few differences in the discretization. The mass matrices can be rewritten as $\mathbb{M}^u=\mathbb{M}^\Phi:=\begin{pmatrix}\mathbb{M}&0\\0&\mathbb{M}\end{pmatrix}$, the dispersion terms and the fluxes can be defined as

$$\mathbb{X}^t:=\begin{pmatrix}0&0\\0&\mathbb{T}^t\end{pmatrix},\quad \mathbb{X}^xu:=\begin{pmatrix}0,&-\mu^2\left\{\mathbb{T}^x\eta-\mathbb{T}^t\mathbb{M}^{-1}\mathcal{N}^q(\boldsymbol{\eta},\boldsymbol{q})\right\}^T,\end{pmatrix}\quad \mathbb{F}(u):=\begin{bmatrix}\mathcal{N}^\eta+\mathcal{J}(\boldsymbol{\eta},\lambda)\\ \mathcal{N}^q+\mathcal{J}(\boldsymbol{q},\lambda)\end{bmatrix} \tag{47}$$

and the source terms can be written as

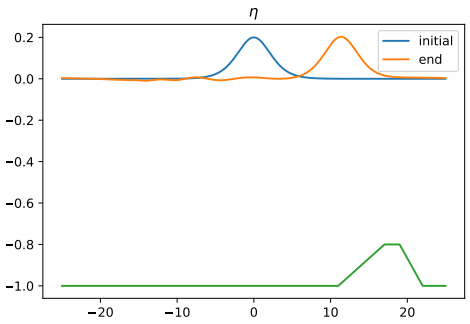
$$\mathbb{S}^{im}=\begin{pmatrix}\mathbb{S}&0\\0&\mathbb{S}\end{pmatrix},\qquad \mathbb{S}^{ex}(u)=\begin{bmatrix}\partial_t h_{\text{iwg}}\\0\end{bmatrix}. \tag{48}$$

The main difference with the KdV-BBM model is that here the explicit dispersive terms are nonlinear as they contain a flux term. For more details on the discretization of the spatial operators we refer to [38].

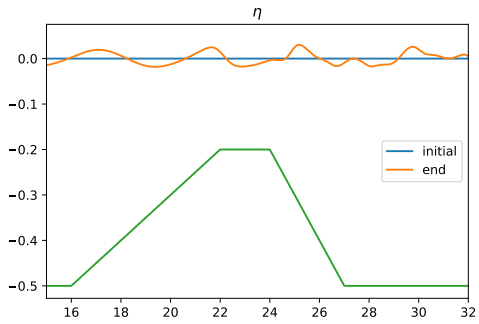
5.3. Benchmarks for weakly dispersive free surface waves

5.3.1. Solitary waves interacting with a submerged bar

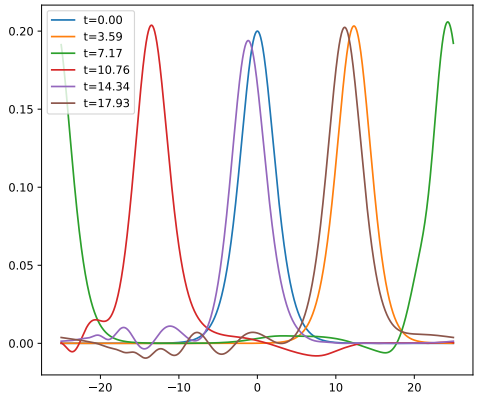
We consider the simulation of solitary waves propagating over a trapezoidal bar, defined as in Section 3.4.3, but with maximal height equal to $b_0=0.2$ and points of change of slope equal to $\{11,17,19,22\}$. We choose periodic boundary conditions on the domain $\Omega=[-20,30]$ and we center the initial solitary wave at the initial condition in $x=5$. Even though this is a traveling solution, the shape of the soliton is smooth and large enough not to encounter problems in the reduction due to the advection character of the solution. We use the following parameters $a_0=0.2$, $h_0=1$, $g=9.81$, $\texttt{CFL}=0.5$, $T=18$. Simulations at various times are displayed in Figures 10(a) and 10(c).



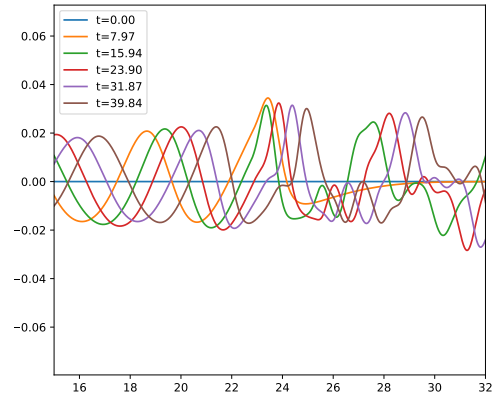
(a) Solitary waves interacting with a submerged bar



(b) Periodic waves over a submerged bar



(c) Solitary waves interacting with a submerged bar: η at different timesteps



(d) Periodic waves over a submerged bar: η at different timesteps

Figure 10: Full order model solutions for EB system tests.

Table 3: Estimate percentage of computational time for solving the linear systems for the three benchmark problems: comparison of the implementation of Thomas algorithm all in `numba` framework and sparse solver by `scipy` with fluxes computed in `numba`

	`numba` and Thomas		`numba` and `scipy`	
N_h	5.3.1	5.3.2	5.3.1	5.3.2
2000	75.8%	39.1%	74.3%	77.7%
4000	72.1%	32.8%	87.4%	80.8%
8000	69.1%	27.7%	94.5%	81.1%

Table 4: Estimate of ratio of computational time of the linear system over computational time of the nonlinear fluxes for the three benchmark problems: comparison of the implementation of Thomas algorithm all in `numba` framework and sparse solver by `scipy` with fluxes computed in `numba`

	`numba` and Thomas		`numba` and `scipy`	
N_h	5.3.1	5.3.2	5.3.1	5.3.2
2000	6.8	2.25	39.8	31.5
4000	6.7	2.18	41.1	31.5
8000	6.9	2.15	39.0	32.1

5.3.2. Periodic waves over a submerged bar

We consider monochromatic waves propagating in the domain $\Omega = [0, 35]$ on a trapezoidal submerged bar of maximum height 0.3. An internal wave generator is positioned at $x_{\text{iwg}} = 10$. Sponge layers are set on both ends of the domain. The wave parameters a_0 and h_0 will be changed along the simulations. For the first test we use $a_0 = 0.027$, as in [38] and $h_0 = 0.5$ (original was $h_0 = 0.4$ that we will leave for the final simulations). As before, $g = 9.81$ and $\texttt{CFL} = 0.5$, while we set $T = 40$. FOM simulations for different times are depicted in Figures 10(b) and 10(d). This test is particularly complicated to be reduced, as one can heuristically see from the simulations. Indeed, the gradients get steep after the trapezoid and travels all along the right domain.

Here, we compare as for the BBM-KdV benchmarks the computational cost of the system solution with respect to the whole simulation. As before, we observe that the cost of the system is a large part of the simulation. In Table 1 we list the time percentage devoted to the solution of the linear system, while in Table 2 we show the ratio between the solver time and the flux evaluation. For the very optimized Thomas algorithm in `numba` we observe that the linear solver is not so expensive as in the previous cases, in particular for the non periodic boundary condition case. On the other side, for a generic sparse solver the computational costs of the linear solver are huge with respect to the other operations and it is noticeable comparing it with the cost of the explicit flux evaluation.

5.4. Reduction of the enhanced Boussinesq model

For the reduction of the FOM we proceed analogously to the scalar case. The main difference is that, in absence of a clear energy statement allowing a linear reduction, each variable has its own reduced basis associated to the minimization of the L^2 norm as in classical Galerkin projection methods. We discuss the reduction main steps hereafter.

5.4.1. Projection on reduced spaces and linear reduction

We consider two reduced spaces for the two variables: $V^\eta \in \mathbb{R}^{N_h \times {N_{RB}}^\eta}$ and $V^q \in \mathbb{R}^{N_h \times {N_{RB}}^q}$. The spaces are obtained again with the POD algorithm, using the same tolerance on both variables. Then, we

project the equations (46) onto these spaces, obtaining

$$\begin{aligned}
W^{\eta,T}(\mathbb{M}+\Delta t\mathbb{S})V^{\eta}\hat{\boldsymbol{\eta}}^{(s)} =& W^{\eta,T}\mathbb{M}\left(\sum_{r=0}^{s-1}\rho_r^s(V^{\eta}\hat{\boldsymbol{\eta}}^{(r)}+h_{\text{iwg}}^{(r)})-h_{\text{iwg}}^{(s)}\right)\\
&-\Delta t\sum_{r=0}^{s-1}\theta_r^s W^{\eta,T}\left(\mathcal{N}^{\eta}\left(V^{\eta}\hat{\boldsymbol{\eta}}^{(r)},V^{q}\hat{\boldsymbol{q}}^{(r)}\right)+\mathcal{J}(V^{\eta}\hat{\boldsymbol{\eta}}^{(r)},\lambda^{(r)})\right)\\
W^{q,T}(\mathbb{M}-\mathbb{T}^t)V^{q}\hat{\boldsymbol{\psi}} =& -\sum_{r=0}^{s-1}\theta_r^s W^{q,T}\left(\mathbb{T}^t\mathbb{M}^{-1}\mathcal{N}^{q}(V^{\eta}\hat{\boldsymbol{\eta}}^{(r)},V^{q}\hat{\boldsymbol{q}}^{(r)})+\mathbb{T}^x V^{\eta}\hat{\boldsymbol{\eta}}^{(r)}\right)\\
W^{q,T}(\mathbb{M}+\Delta t\mathbb{S})V^{q}\hat{\boldsymbol{q}}^{(s)} =& W^{q,T}\mathbb{M}V^{q}\sum_{r=0}^{s-1}\rho_r^s\hat{\boldsymbol{q}}^{(r)}+\Delta t W^{q,T}\mathbb{M}^{q}V^{q}\hat{\boldsymbol{\psi}}\\
&-\Delta t\sum_{r=0}^{s-1}\theta_r^s W^{q,T}\left(\mathcal{N}^{q}(V^{\eta}\hat{\boldsymbol{\eta}}^{(r)},V^{q}\hat{\boldsymbol{q}}^{(r)})+\mathcal{J}(V^{q}\hat{\boldsymbol{q}}^{(r)},\lambda^{(r)})\right)
\end{aligned}\tag{49}$$

for every stage s of SSPRK2. The reduced variables are denoted with the $\hat{\cdot}$ symbol. As already remarked several times, we have no energy allowing to define a minimization problem leading to an efficient linear reduction strategy. Hence we will make use here of classical L^2 projection arguments, and consider Θ to be the identity, and $W = V$.

Similarly to the scalar case, we can precompute the projected operators and use them in the reduced simulation to save computational time. In this first stage, we obtain the pdROM algorithm. Let us define $\hat{\mathbb{M}}^{\eta} := W^{\eta,T}\mathbb{M}V^{\eta}$, $\hat{\mathbb{M}}^{q} := W^{q,T}\mathbb{M}V^{q}$, $\hat{\mathbb{T}}^{t} := W^{q,T}\mathbb{T}^{t}V^{q}$, $\hat{\mathbb{T}}^{x} := W^{q,T}\mathbb{T}^{x}V^{\eta}$, $\hat{\mathbb{S}}^{\eta} := W^{\eta,T}\mathbb{S}V^{\eta}$, $\hat{\mathbb{S}}^{q} := W^{q,T}\mathbb{S}V^{q}$, recalling that $h_{\text{iwg}} = \mu_{\text{iwg}}(t,a_0)f_{\text{iwg}}$, we also define $\hat{f}_{\text{iwg}} := W^{\eta,T}f_{\text{iwg}}$. We obtain substituting in (49)

$$(\hat{\mathbb{M}}^{\eta}+\Delta t\hat{\mathbb{S}}^{\eta})\hat{\boldsymbol{\eta}}^{(s)} =\hat{\mathbb{M}}^{\eta}\sum_{r=0}^{s-1}\rho_r^s\hat{\boldsymbol{\eta}}^{(r)}-\left(\mu_{\text{iwg}}^{(s)}-\sum_{r=0}^{s-1}\rho_r^s\mu_{\text{iwg}}^{(r)}\right)\hat{\mathbb{M}}^{\eta}\hat{f}_{\text{iwg}}-\Delta t\sum_{r=0}^{s-1}\theta_r^s\hat{\mathcal{N}}^{\eta,(r)},\tag{50a}$$

$$(\hat{\mathbb{M}}^{q}-\hat{\mathbb{T}}^{t})\hat{\boldsymbol{\psi}} =-\sum_{r=0}^{s-1}\theta_r^s\left(\hat{\mathbb{T}}^{t}\hat{\mathbb{M}}^{q,-1}\hat{\mathcal{N}}^{q,(r)}+\hat{\mathbb{T}}^{x}\hat{\boldsymbol{\eta}}^{(r)}\right),\tag{50b}$$

$$(\hat{\mathbb{M}}^{q}+\Delta t\hat{\mathbb{S}}^{q})\hat{\boldsymbol{q}}^{(s)} =\hat{\mathbb{M}}^{q}\sum_{r=0}^{s-1}\rho_r^s\hat{\boldsymbol{q}}^{(r)}+\Delta t\hat{\mathbb{M}}^{q}\hat{\boldsymbol{\psi}}-\Delta t\sum_{r=0}^{s-1}\theta_r^s\hat{\mathcal{N}}^{q,(r)},\tag{50c}$$

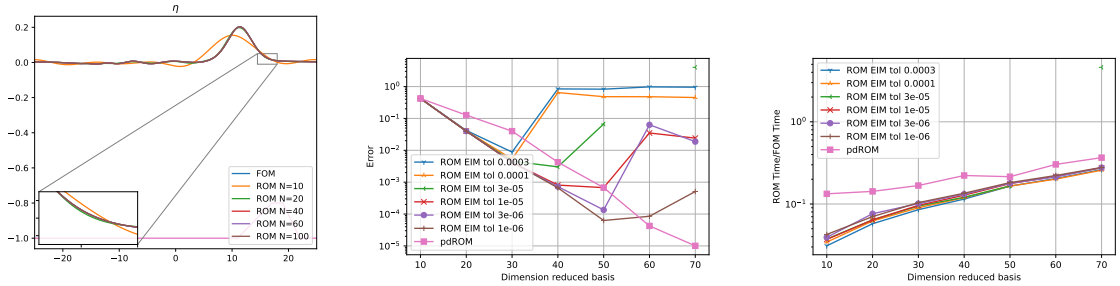
where the reduced nonlinear fluxes are defined as

$$\begin{aligned}
\hat{\mathcal{N}}^{\eta,(r)} :=& W^{\eta,T}\left(\mathcal{N}^{\eta}\left(V^{\eta}\hat{\boldsymbol{\eta}}^{(r)},V^{q}\hat{\boldsymbol{q}}^{(r)}\right)+\mathcal{J}(V^{\eta}\hat{\boldsymbol{\eta}}^{(r)},\lambda^{(r)})\right),\\
\hat{\mathcal{N}}^{q,(r)} :=& W^{q,T}\left(\mathcal{N}^{q}(V^{\eta}\hat{\boldsymbol{\eta}}^{(r)},V^{q}\hat{\boldsymbol{q}}^{(r)})+\mathcal{J}(V^{q}\hat{\boldsymbol{q}}^{(r)},\lambda^{(r)})\right).
\end{aligned}\tag{51}$$

Notice that there has been a further projection inside the matrix multiplication

$$W^{q,T}\mathbb{T}^{t}\mathbb{M}^{-1}\mathcal{N}^{q}\approx W^{q,T}\mathbb{T}^{t}V^{q}V^{q,T}\mathbb{M}^{-1}W^{q}W^{q,T}\mathcal{N}^{q}=\hat{\mathbb{T}}^{t}\hat{\mathbb{M}}^{q,-1}\hat{\mathcal{N}}^{q},\tag{52}$$

in order to have an efficient implementation of the operations without the need of too many operators. Hence, all the matricial operations are reduced and only the nonlinear fluxes have computational costs that scale as an $\mathcal{O}(N_h)$. The solution of the reduced system can be done after the simple assembly of the reduced mass and sponge matrices, with dimension N_{RB}. As before, this method will be denoted with pdROM in the benchmarks section and all the terms can be matched with (8) following the same definitions of the FOM discretization.



(a) ROM solutions time-only reduction (b) EIM ROM errors time-only reduction (c) EIM ROM computational times time-only reduction

Figure 11: **Solitary waves over a submerged bar.** pdROM and EIM (time-only) reduction: simulation, errors and computational time, varying POD and EIM dimensions

Remark 6 (A more efficient implementation)**.** *All of the above manipulations retain the structure of a perturbation of the nonlinear shallow water solver via the use of the auxiliary variable $\boldsymbol{\psi}$. This is very interesting from the practical point of view for several reasons as in perspective it could be used to enhance an existing shallow water solver. It also allows to more easily embed certain wave breaking closures (see e.g. [16, 24]). However, this reduced model can be more efficiently implemented using a global form that does not isolate the shallow water equations. In particular, instead of* (50c) *and* (50b) *we could use the following*

$$(\hat{\mathbb{M}}^q - \hat{\mathbb{T}}^t + \Delta t\hat{\mathbb{S}}^q)\hat{\boldsymbol{q}}^{(s)} - (\hat{\mathbb{M}}^q - \hat{\mathbb{T}}^t)\sum_{r=0}^{s-1}\rho_r^s\hat{\boldsymbol{q}}^{(r)} + \sum_{r=0}^{s-1}\theta_r^s\Delta t\left(\hat{\mathcal{N}}^{q,(r)} + \hat{\mathbb{T}}^x\hat{\boldsymbol{\eta}}^{(r)}\right) = 0. \tag{53}$$

Avoiding the explicit evaluation of $\boldsymbol{\psi}$ allows to avoid several projections and reduce the online cost.

5.4.2. Hyper-reduction

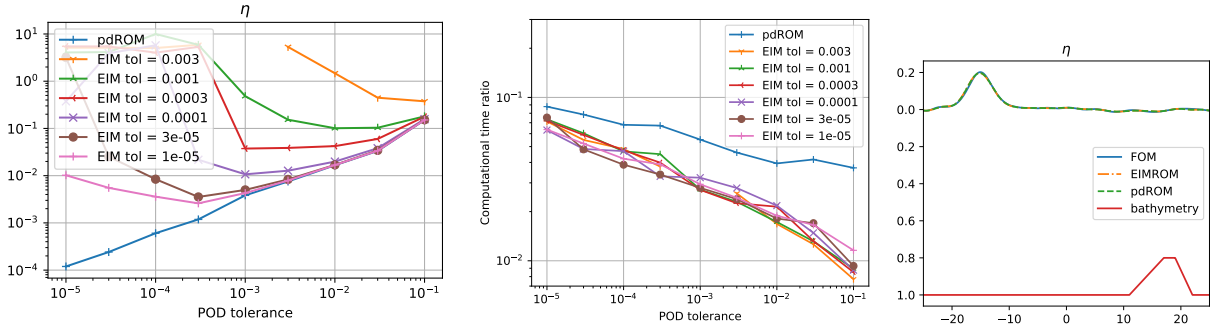
The final reduction that can be applied is the EIM on the nonlinear fluxes $\mathcal{N}^\eta, \mathcal{N}^q$. The same procedures presented in the scalar case can be applied in this case, obtaining an extra level of approximation which allows to get rid of all the computations that scale as an $\mathcal{O}(N_h)$. Same caveats hold for this problem: the tuning of the tolerance of the EIM algorithm must be done accordingly to the POD one otherwise instabilities and Runge phenomena might appear, see [36, 49, 17, 2, 10].

6. Simulations for enhanced Boussinesq

In this section we study the simulations for the EB model. We follow the *modus operandi* of Section 4.1 We will show results mainly for the variable η, omitting the ones for q for brevity. In the second stage we will let not only h_0 vary to form the training set, but also a_0, we will specify their range in the different tests. To make a comparison with a reasonable solver, being the problem more complicated, we will use a sparse `scipy` solver for the FOM, where the tridiagonal structure of the matrices is not exploited *a priori*.

6.1. Solitary waves over a submerged bar

For this test we set $a_0 = 0.2$ and $\bar{h}_0 = 1$. For the first phase we just compress the solutions for a simulation until $T = 25$ with the POD. The problem is relatively simple and the wave does not show much dispersion. We can see with different choices of ${N_{RB}}^\eta = {N_{RB}}^q$ the behavior of the pdROM approximated solutions in Figure 11(a). Adding also the EIM to the algorithm we obtain expected results as we can see both in the error behavior in Figure 11(b), and in computational times of Figure 11(c), where we can achieve around 4% to 10% of FOM computational time for pdROM and between 1% and 8% for

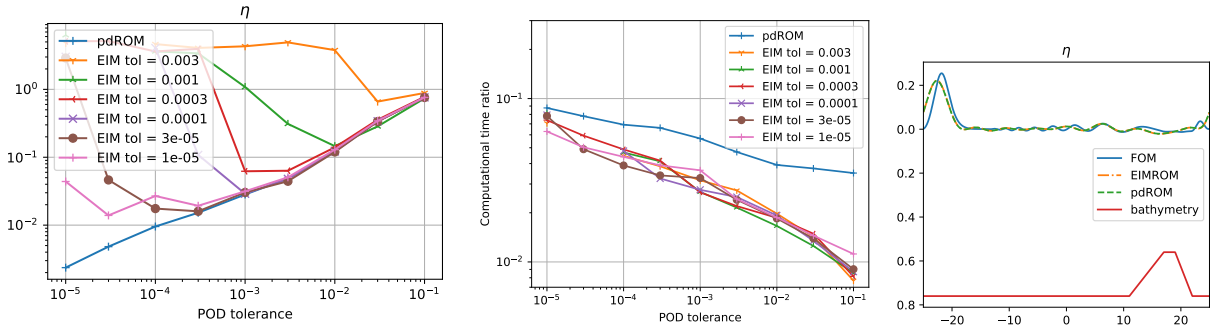


(a) ROM errors time-parameter reduction on parameter in the training set

(b) ROM computational time time-parameter reduction on parameter and u_0 in the training set

(c) ROM solutions time-parameter reduction in the training set

Figure 12: **Solitary waves over a submerged bar.** pdROM and EIM (time-parameter) reduction: simulation, errors and computational time, varying POD and EIM dimensions. Parameter inside the training domain.



(a) ROM errors time-parameter reduction on parameter and u_0 outside the training set

(b) ROM computational time time-parameter reduction on parameter and u_0 outside the training set

(c) pdROM and EIMROM solutions time-parameter reduction outside the training set

Figure 13: **Solitary waves over a submerged bar.** pdROM and EIM (time-parameter) reduction: simulation, errors and computational time, varying POD and EIM dimensions. Parameter outside the training domain.

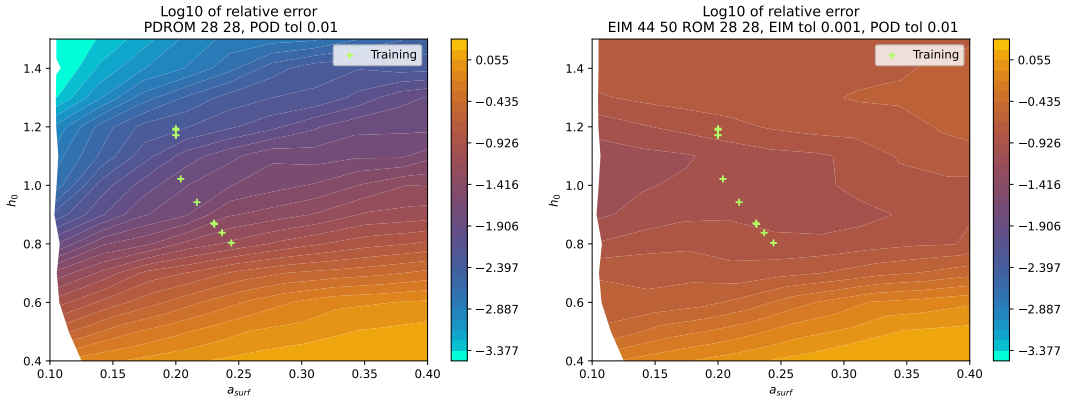
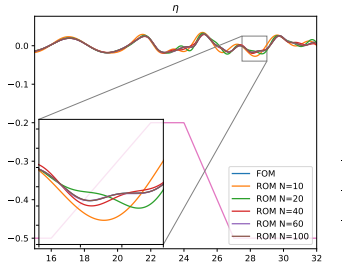
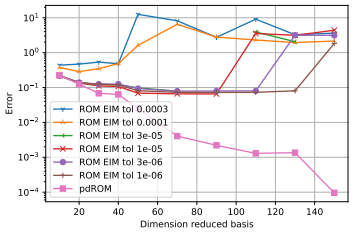


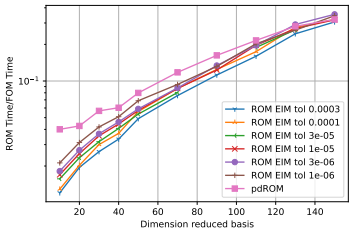
Figure 14: **Solitary waves over a submerged bar.** Error plot varying the parameters a_0 and h_0 for pdROM $N_{RB}{}^{\eta} = 31$ and $N_{RB}{}^{q} = 30$ and EIMROM with $N_{EIM}{}^{\eta} = 40$ and $N_{EIM}{}^{q} = 51$



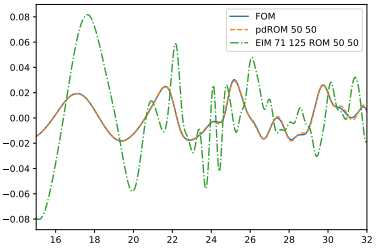
(a) ROM solutions time-only reduction

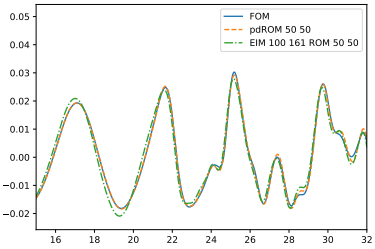


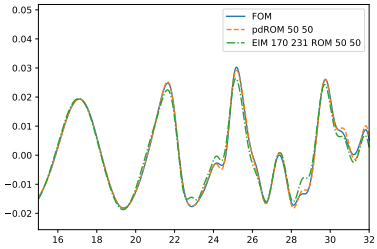
(b) EIM ROM errors time-only reduction



(c) EIM ROM computational times time-only reduction







(d) EIM ROM simulation time-only reduction with $N_{RB}{}^{\eta} = N_{RB}{}^{q} = 50$ and $tol_{EIM} = 10^{-4}, 3 \cdot 10^{-5}, 10^{-6}$ respectively from left to right

Figure 15: **Monochromatic waves on a submerged bar.** pdROM and EIM (time-only) reduction: simulation, errors and computational time, varying POD and EIM dimensions

EIMROM. In this case, the system of the FOM requires more computational time than the scalar case one, as it is composed of more terms.

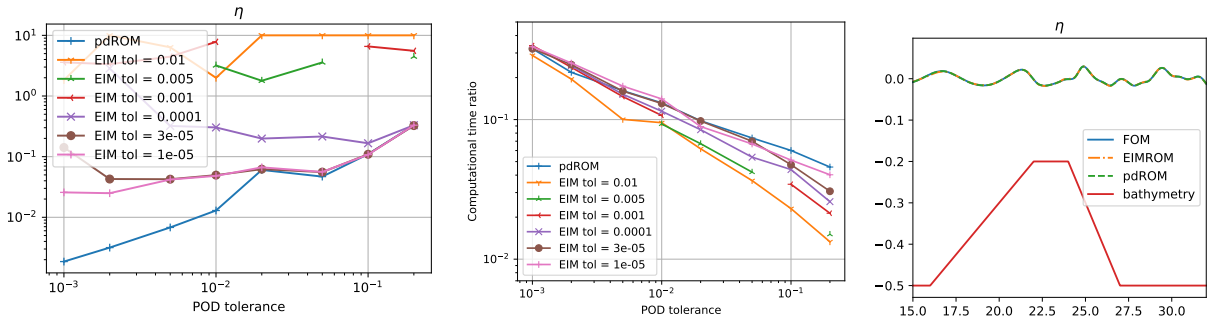
Now, we consider 10 snapshots with randomly chosen parameters $h_0 \in [0.8, 1.2]$ and $a_0 \in [0.16, 0.24]$. This training set is used to compute the POD and EIM basis functions. Simulating pdROM and EIMROM for $a_0 = 0.2$ and $h_0 = 1$, with relatively few basis functions the error is already low, see Figure 12(a) and the computational costs stay in the same range between 1% and 10% of FOM computational times, see Figure 12(b). In Figure 12(c) we see that with only $N_{RB}{}^{\eta} = 20$, $N_{RB}{}^{q} = 20$ for 0.03 POD tolerance the pdROM already approximates very well the solution and adding the EIM with $N_{EIM}{}^{\eta} = 47$ and $N_{EIM}{}^{q} = 60$ for 0.0003 EIM tolerance results in an accurate approximation of the FOM in just 7% of the FOM computational time.

Considering $h_0 = 0.76$ and $a_0 = 0.252$ slightly outside the training set we can already see larger errors (one order of magnitude) in Figure 13(a) and comparable computational times in Figure 13(b). In Figure 13(c) we see that the FOM is a bit more oscillating than in the previous case and that the pdROM and EIMROM, for the same parameters used above, struggle more with approximating accurately the solution, still being not too far from the FOM solution.

For $N_{RB}{}^{\eta} = 28$, $N_{RB}{}^{q} = 28$ relative to a POD tolerance of 0.01 and $N_{EIM}{}^{\eta} = 44$, $N_{EIM}{}^{q} = 50$ for EIM tolerance of 0.001, we see in Figure 14 how the parameters a_0 and h_0 influence the relative error of the pdROM and EIMROM approximations. The structure is similar to the scalar case, where increasing the nonlinearity of the problem the oscillations rise and the error follows. Again, we see that the EIMROM do not obtain such an accurate solution as pdROM for problems close to the training set, though using much less computational time (around 3 times faster than pdROM).

6.2. Monochromatic waves on a submerged bar

The last test is very challenging as its solutions are very close to show steep gradients and transport phenomena. Moreover, the water level is often close to the bathymetry and, if not well represented, the

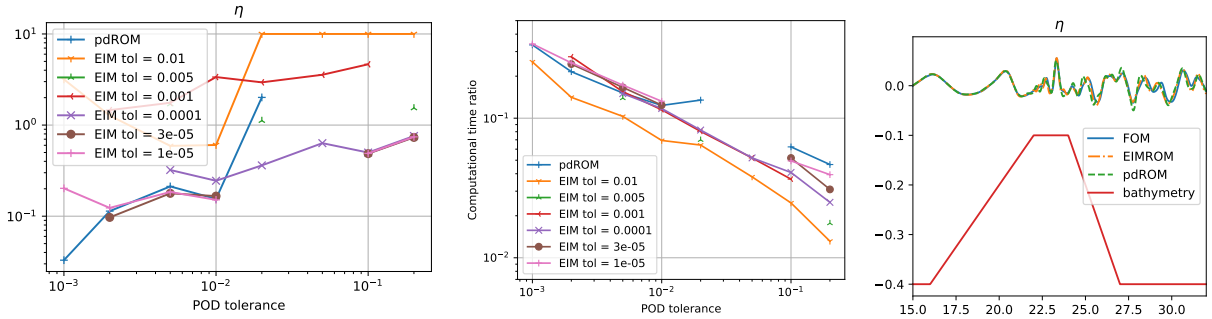


(a) ROM errors time-parameter reduction on parameter in the training set

(b) ROM computational time time-parameter reduction on parameter in the training set

(c) pdROM and EIMROM solutions time-parameter reduction

Figure 16: **Monochromatic waves on a submerged bar.** pdROM and EIM (time-parameter) reduction: simulation, errors and computational time, varying POD and EIM dimensions. Parameter inside the training domain.



(a) ROM errors time-parameter reduction on parameter outside the training set

(b) ROM computational time time-parameter reduction on parameter and u_0 outside the training set

(c) pdROM and EIMROM solutions time-parameter reduction

Figure 17: **Monochromatic waves on a submerged bar.** pdROM and EIM (time-parameter) reduction: simulation, errors and computational time, varying POD and EIM dimensions. Parameter outside the training domain.

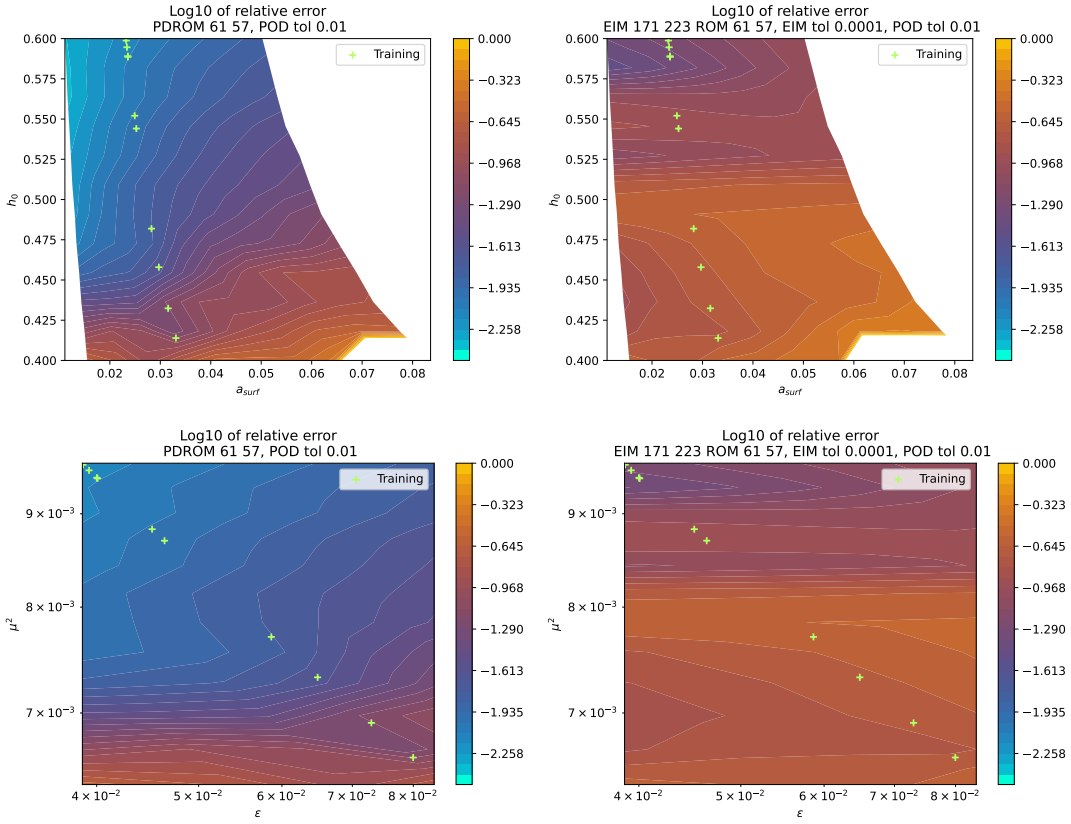


Figure 18: **Monochromatic waves on a submerged bar.** Error plot varying the parameters a_0 and h_0 for pdROM $tol_{POD} = 10^{-2}$ ${N_{RB}}^{\eta} = 61$ and ${N_{RB}}^{q} = 57$ and EIMROM with $tol_{EIM} = 10^{-4}$ and ${N_{EIM}}^{\eta} = 121$ and ${N_{EIM}}^{q} = 235$

approximation can result into invalid values. Fortunately, the steep gradients are all close to the end of the trapezoid and with not so many basis functions it is possible to approximate them. We take $a_0 = 0.027$, $\bar{h}_0 = 0.5$ and final time $T = 40$. Taken the POD over the time evolution of this FOM, we approximate in Figure 15(a) with pdROM the same evolution for different ${N_{RB}}^{\eta} = {N_{RB}}^{q}$ of the same problem. We see that with 40 basis functions the pdROM approximation is close to the FOM one, but only for 60 basis functions we do not observe any large deformation. In Figure 15(b) we see more precisely the error decay for different N_{RB} and different EIM tolerances. The error is quite large, and the decay with the EIM approximation is very slow. It must be noticed that in many of these EIMROM simulations, the water height reaches negative values and stop at an earlier time, in those cases the error is not reported or it is larger than the scale. We see for an EIM tolerances of 10^-4, $3 \cdot 10^{-5}$ and 10^{-6} in Figure 15(d) how the error for the first one is very large, but also for the other two cases the error does not decrease a lot in particular in the area after the trapezoid, where the steepest part of the solutions are located. The computational costs for this problem range between 4% and 30% for pdROM and between 1% and 30% for EIMROM as shown in Figure 15(c).

Introducing more parameters in the training set, with $h_0 \in [0.4, 0.6]$ and $a_0 \in [0.0216, 0.0324]$, leads to a much more rich manifold of solutions. This implies that the eigenvalue decay of the POD gets slower. For the test parameter $\bar{h}_0$ and $a_0 = 0.027$, we observe in Figure 16(b) that the computational costs for pdROM and EIMROM are very similar and they are around the 5% of the FOM computational time to obtain an error of 10^{-1}, around 15% of the time for an error of 10^{-2} and around 30% for an error of 10^{-3}. For very accurate results with error lower than 10^{-4} a computational time of more of the 30% of the FOM computational time may be required. The pdROM simulation in Figure 16(a) shows the accurate result for $tol_{POD} = 0.001$, ${N_{RB}}^{\eta} = 99$, ${N_{RB}}^{\eta} = 92$ and $tol_{EIM} = 0.00003$, ${N_{EIM}}^{\eta} = 213$ ${N_{EIM}}^{q} = 291$. The number of EIM basis functions is large in order to obtain a good approximation of the solution and almost not gaining anything in terms of computational time with respect to the pdROM. This number must be large in order to avoid the simulation to oscillate so much that it hits negative values and it crashes the simulation. Again, this behavior has been observed in many other works [36, 49, 17, 2, 10] and we do not aim at finding better hyper reduction algorithm in this work.

For the parameter outside the training set, we chose $a_0 = 0.027$ and $h_0 = 0.4$, which correspond to the test case in [38] which was validated with experimental data. This simulation is quite challenging also for pdROM and when it is not enough resolved it can have oscillations in h going below 0. This happens for example for $tol_{POD} = 5 \cdot 10^{-2}$, as shown from the incomplete curves in Figure 17(a). Increasing the dimension of the reduced space we overcome this issue. Computational costs are similar to the ones discussed above. In Figure 17(c) we plot the simulation for $tol_{POD} = 0.005$, ${N_{RB}}^{\eta} = 99$, ${N_{RB}}^{\eta} = 92$ and $tol_{EIM} = 0.00003$, ${N_{EIM}}^{\eta} = 213$ ${N_{EIM}}^{q} = 291$. Here it is the EIMROM simulation with the largest tolerance that does not crashes along the simulation. The computational time reduction due to the EIM at this stage is almost negligible with respect to the pdROM and the approximation is very similar. Both simulations oscillate more than the original FOM after the trapezoid at this resolution, though using only 15% of the computational time of the FOM. To reach errors of the order of 10^{-1} we need computational costs of around the 25% of the FOM ones as one can observe comparing Figure 17(a) and Figure 17(b) for $tol_{POD} = 2 \cdot 10^{-3}$.

In Figure 18 we see the strong influence that h_0 and a_0 have on these simulations. In this case, it is crucial the level of h_0. When this level is too low, we fall in another regime and all the hypotheses made on the dispersive equations are not valid anymore. Moreover, the simulations risk to hit negative values and break in this regime. Above this regime, for pdROM we can simply say that as the oscillations and the nonlinearity increase, the error increases. On the other side, the dispersion term is actually helping in reducing the error in the reduced space. For EIMROM we observe more areas where the algorithm struggles with obtaining good results, also for not too small h_0 values.

7. Conclusion

In this work we have proposed some strategies for model order reduction for dispersive waves equations. Applications have been shown to a BBM-KdV type model, and to the enhanced Boussinesq equations of Madsen and Sørensen. Both models contain hyperbolic and dispersive terms that can be decoupled in the

numerical solver. The dispersive terms can be obtained solving an elliptic and linear problem, which is well suited to be reduced with standard projection based reduction techniques after the choice of a reduced basis space. We apply the proper orthogonal decomposition on some snapshots obtained for different times and parameters to obtain a reduced space. When the gradients of the solutions are not so steep and the nonlinear character is not the dominant one, we can obtain large reduction with the pdROM in computational times, up to 20 times less, and still obtain a reliable solution. Introducing a second level of approximation with the empirical interpolation method, this factor can increase up to 100, but stability issues may arise. Hence, it is not always obvious how to choose how many interpolation functions are needed. There are some clear bounds that cannot be overtaken when using these simple reduction techniques: the nonlinearity cannot be too pronounced and the water level must be far enough from zero. Nevertheless, the results are encouraging in particular knowing that already for one dimensional problems we can can obtain good reduction.

This idea can be easily applied to a pre-existent hyperbolic algorithm, as a shallow water code, with a cheap additional term which takes into account of the dispersive effect. This can, in fact, be used without the burden of having to solve a large linear system. In the future, we aim to apply this algorithm to more complicated two dimensional problems, where we expect larger reduction in computational times (as the FOM will become more expensive), and to selectively switch on and off the dispersive term into a shallow water code, in order to allow to pass between different regimes, according to the solution shape.

To our knowledge this is the first work in this direction, and the initial results are quite promising. Several future improvements and developments are foreseen both involving improved reduction strategies, and more complex wave dynamics and models. Notable the extension to multiple space dimensions, as well as breaking waves are under investigation.

Acknowledgments

D. T. has been funded by an Inria Postdoc in Team Cardamom and by a SISSA Mathematical Fellowship. M. R. is a permanent member of Inria Team Cardamom.

Declarations of interest: none.

Appendix A. $\mathbb{P}^1$ finite elements for the MS model: full expressions

We report here the expressions of the finite element approximations of some of the operators arising in the Madsen and Sørensen model. The first is $\mathbb{T}^t \in \mathbb{R}^{N\times N}$, representing the matrix discretization of the terms $\mathcal{T}^t$. It can be split into two matrices. The first one is simply defined as a tridiagonal matrix with entries for the jth row $B\bar{h}_j^2[1 \quad -2 \; 1]$, while the second one has the following entries for the jth row

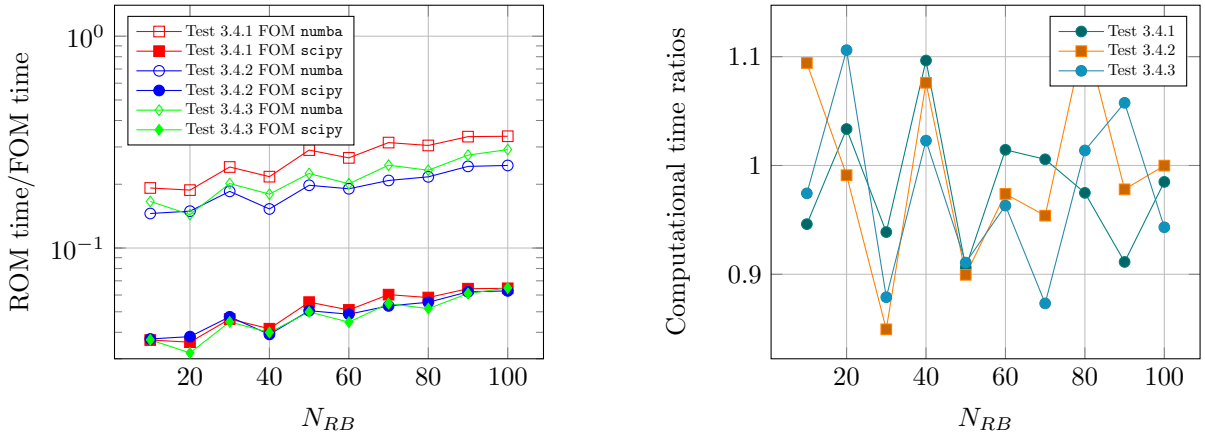
$$-\frac{1}{18}\begin{bmatrix} (\bar{h}_j-\bar{h}_{j-1})(2\bar{h}_j+\bar{h}_{j-1}) \\ -(\bar{h}_j-\bar{h}_{j-1})(2\bar{h}_j+\bar{h}_{j-1})+(\bar{h}_{j+1}-\bar{h}_j)(2\bar{h}_j+\bar{h}_{j+1}) \\ -(\bar{h}_{j+1}-\bar{h}_j)(2\bar{h}_j+\bar{h}_{j+1}) \end{bmatrix}^T. \tag{A.1}$$

The sum of the two defines the dispersion matrix $\mathbb{T}^t \in \mathbb{R}^{N\times N}$. The term $\mathcal{T}^x[\eta]$ directly depends on $\partial_{xx}\eta$, is discretized using an auxiliary variable $w \approx \partial_{xx}\eta$ with the definition $w_j = \frac{\eta_{j-1}-2\eta_j+\eta_{j+1}}{\Delta x^2}$. The operator can be approximated by a matrix $\mathbb{T}^x$ with the multiplication $\mathbb{T}^x\boldsymbol{\eta}$. It can be conveniently written with two matrices $\mathbb{T}^{x,1}$ and $\mathbb{T}^{x,2}$ such that $\mathbb{T}^x\boldsymbol{\eta} = \mathbb{T}^{x,1}\boldsymbol{w} + \mathbb{T}^{x,2}\boldsymbol{w}$, where $\mathbb{T}^{x,1}$ has the following 3 entries on the j-th row in the 3 main diagonals

$$-\frac{\beta g}{3}\left[-\tfrac{(\bar{h}_j+\bar{h}_{j-1})^3}{4}-\bar{h}_j^3, \quad \tfrac{(\bar{h}_j+\bar{h}_{j-1})^3}{4}-\tfrac{(\bar{h}_j+\bar{h}_{j+1})^3}{4}, \quad \tfrac{(\bar{h}_j+\bar{h}_{j+1})^3}{4}+\bar{h}_j^3\right], \tag{A.2}$$

and where $\mathbb{T}^{x,2}$ has the following 3 entries on the j-th row in the 3 main diagonals

$$-\frac{\beta g}{6}\begin{bmatrix} (\bar{h}_j-\bar{h}_{j-1})(\bar{h}_j^2w_j+\frac{1}{4}(\bar{h}_j+\bar{h}_{j-1})^2) \\ (\bar{h}_j-\bar{h}_{j-1})(\bar{h}_j^2w_j+\frac{1}{4}(\bar{h}_j+\bar{h}_{j-1})^2)+(\bar{h}_{j+1}-\bar{h}_j)(\bar{h}_j^2w_j+\frac{1}{4}(\bar{h}_j+\bar{h}_{j+1})^2) \\ (\bar{h}_{j+1}-\bar{h}_j)(\bar{h}_j^2w_j+\frac{1}{4}(\bar{h}_j+\bar{h}_{j+1})^2) \end{bmatrix}^T. \tag{A.3}$$



(a) Ratio between computational time of pdROM with respect to FOM computed either with Thomas solver in `numba` or with the sparse solver of `scipy`

(b) Ratio between computational time of ROM with reduction only on Φ and pdROM

Figure B.19: Computational costs for different algorithms: different FOMs (left), only projection of Φ for ROM (right)

Finally, the discretization of this sponge layer term leads to an operator $\mathbb{S}$ which is a tridiagonal matrix with entries for the jth row

$$\mathbb{S}_{j,j-1} = -\frac{\nu_j + \nu_{j-1}}{8\Delta x}, \mathbb{S}_{j,j} = \frac{\nu_{j-1} + 2\nu_j + \nu_{j+1}}{8\Delta x}, \mathbb{S}_{j,j+1} = -\frac{\nu_j + \nu_{j+1}}{8\Delta x}. \tag{A.4}$$

Appendix B. Computational costs of different solvers

We compare here the computational costs of different sparse linear solvers and different strategies for the reduction. We will refer to the KdV-BBM problem (14), similar results can be drawn from the enhanced Boussinesq system (46).

First of all, let us compare different sparse linear solvers for the FOM. We have used Thomas algorithm in the previously proposed examples. This linear solver very optimized and tailored towards the problem we are solving. As soon as the discretization gets more involved or as soon as we move to more dimensions, we would lose the tridiagonal structure of the system matrices and it would become impossible to apply Thomas algorithm. In order to have a fairer comparison, in Figure 19(a) we show the ROM computational costs over the FOM computational costs when the FOM uses Thomas algorithm implemented in `numba` and when the FOM uses the sparse solver of `scipy`. We observe a huge difference between the two solvers of a factor between 4 and 6. This would make the pdROM computational costs even more appealing for a general sparse solver implementation in the FOM, passing from 20% of the computational time to only 4%!

Secondly, we want to prove again that the main computational cost of the FOM is given by the linear system and that the advantage obtained with the pdROM is focused on this step of the algorithm. Indeed, we can compare the performance of the pdROM with the scheme given by the reduction only of the Φ equation, as suggested in Remark 3. In that algorithm, Φ is computed by projecting the RHS of its equation onto the reduced basis space, then solving the reduced system and, finally, reconstructing the full Φ. Its computational costs are comparable to the one of the pdROM. In Figure 19(b) we plot the ratio of these costs for some N_{RB} for all the previous tests. We can see that they are very close to one and the fluctuations around that are probably due to the low precision in measuring the times. We remind that the ROM where only Φ is reduced has larger errors than the pdROM. This is maybe due to a mismatch of the spaces along the computations that allows the propagation of spurious modes. We are still investigating the phenomenon. Hence, in general, we would recommend a fully projected approach.

References

[1] A. Ali and H. Kalisch, *Mechanical Balance Laws for Boussinesq Models of Surface Water Waves*, J. Nonlinear Sci., 22 (2012), pp. 371–398.

[2] ——, *On the Formulation of Mass, Momentum and Energy Conservation in the KdV Equation*, Acta Appl. Math., (2014), pp. 113–131.

[3] J. Argaud, B. Bouriquet, H. Gong, Y. Maday, and O. Mula, *Stabilization of (G) EIM in presence of measurement noise: application to nuclear reactor physics*, in Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2016, Springer, 2017, pp. 133–145.

[4] P. Bacigaluppi, M. Ricchiuto, and P. Bonneton, *Implementation and Evaluation of Breaking Detection Criteria for a Hybrid Boussinesq Model*, Water Waves, 2 (2020), pp. 207–241.

[5] M. Barrault, Y. Maday, N. Nguyen, and A. Patera, *An empirical interpolation method: application to efficient reduced-basis discretization of partial differential equations*, Comptes Rendus de l'Academie des Sciences Paris, 339 (2004), pp. 667–672.

[6] T. B. Benjamin, J. L. Bona, and J. J. Mahony, *Model Equations for Long Waves in Nonlinear Dispersive Systems*, Philosophical Transactions of the Royal Society of London Series A, 272 (1972), pp. 47–78.

[7] M. Brocchini, *A reasoned overview on Boussinesq-type models: the interplay between physics, mathematics and numerics*, Proc. R. Soc. A, 469 (2013).

[8] E. Burman and P. Hansbo, *The edge stabilization method for finite elements in CFD*, in Numerical mathematics and advanced applications, Springer, 2004, pp. 196–203.

[9] N. Cagniart, Y. Maday, and B. Stamm, *Model Order Reduction for Problems with Large Convection Effects*, Springer International Publishing, Cham, 2019, pp. 131–150.

[10] A. Cauquis, M. Ricchiuto, and P. Heinrich, *Lax–Wendroff Schemes with Polynomial Extrapolation and Simplified Lax–Wendroff Schemes for Dispersive Waves: A Comparative Study*, Water Waves, (2022).

[11] Y. Chen, S. Gottlieb, L. Ji, and Y. Maday, *An EIM-degradation free reduced basis method via over collocation and residual hyper reduction-based error estimation*, Journal of Computational Physics, 444 (2021), p. 110545.

[12] M. W. Dingemans, *Water wave propagation over uneven bottoms: Linear wave propagation*, vol. 13, World Scientific, 1997.

[13] Z. Drmac and S. Gugercin, *A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions*, SIAM Journal on Scientific Computing, 38 (2016), pp. A631–A648.

[14] M. Drohmann, B. Haasdonk, and M. Ohlberger, *Reduced basis approximation for nonlinear parametrized evolution equations based on empirical operator interpolation*, SIAM Journal on Scientific Computing, 34 (2012), pp. A937–A969.

[15] M. Duruflé and S. Israwi, *A numerical study of variable depth KdV equations and generalizations of Camassa–Holm-like equations*, Journal of computational and applied mathematics, 236 (2012), pp. 4149–4165.

[16] C. Farhat, T. Chapman, and P. Avery, *Structure-preserving, stability, and accuracy properties of the energy-conserving sampling and weighting method for the hyper reduction of nonlinear finite element dynamic models*, International journal for numerical methods in engineering, 102 (2015), pp. 1077–1110.

[17] A. G. Filippini, M. Kazolea, and M. Ricchiuto, *A flexible genuinely nonlinear approach for nonlinear wave propagation, breaking and run-up*, Journal of Computational Physics, 310 (2016), pp. 381–417.

[18] F. Ghavamian, P. Tiso, and A. Simone, *POD–DEIM model order reduction for strain-softening viscoplasticity*, Computer Methods in Applied Mechanics and Engineering, 317 (2017), pp. 458–479.

[19] S. Gottlieb and C.-W. Shu, *Total variation diminishing Runge-Kutta schemes*, Mathematics of Computation, 67 (1998), pp. 73–85.

[20] S. Grimberg, C. Farhat, and N. Youkilis, *On the stability of projection-based model order reduction for convection-dominated laminar and turbulent flows*, Journal of Computational Physics, 419 (2020), p. 109681.

[21] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, *Array programming with NumPy*, Nature, 585 (2020), pp. 357–362.

[22] J. Hesthaven, G. Rozza, and B. Stamm, *Certified Reduced Basis Methods for Parametrized Partial Differential Equations*, Springer, 2016.

[23] S. Israwi, *Variable depth kdv equations and generalizations to more nonlinear regimes*, ESAIM: Mathematical Modelling and Numerical Analysis, 44 (2010), pp. 347–370.

[24] A. Karczewska and P. Rozmej, *Can simple KdV-type equations be derived for shallow water problem with bottom bathymetry?*, Communications in Nonlinear Science and Numerical Simulation, 82 (2020), p. 105073.

[25] M. Kazolea and M. Ricchiuto, *On wave breaking for Boussinesq-type models*, Ocean Modelling, 123C (2018), pp. 16–39. [pdf].

[26] K. Kunisch and S. Volkwein, *Galerkin proper orthogonal decomposition methods for parabolic problems*, Numer. Math., 90 (2001), pp. 117–148.

[27] S. K. Lam, A. Pitrou, and S. Seibert, *Numba: A llvm-based python jit compiler*, in Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, 2015, pp. 1–6.

[28] D. Lannes, *The Water Waves Problem: Mathematical Analysis and Asymptotics*, American Mathematical Society, Providence, Rhode Island, 2013.

[29] D. Lannes, *Modeling shallow water waves*, Nonlinearity, 33 (2020), p. R1.

[30] D. Lannes and P. Bonneton, *Derivation of asymptotic two-dimensional time-dependent equations for surface water wave propagation*, Phys. Fluids, 21 (2009).

[31] P. A. Madsen and O. R. Sørensen, *A new form of the Boussinesq equations with improved linear dispersion characteristics. Part 2. A slowly-varying bathymetry*, Coastal Engineering, 18 (1992), pp. 183–204.

[32] R. Mojgani and M. Balajewicz, *Lagrangian basis method for dimensionality reduction of convection dominated nonlinear flows*, arXiv e-prints, arXiv:1701.04343, (2017).

[33] O. Nwogu, *Alternative form of Boussinesq equations for nearshore wave propagation*, Journal of waterway, port, coastal, and ocean engineering, 119 (1993), pp. 618–638.

[34] M. Ohlberger and S. Rave, *Nonlinear reduced basis approximation of parameterized evolution equations via the method of freezing*, Comptes Rendus Mathematique, 351 (2013), pp. 901 – 906.

[35] J. Orszaghova, A. G. Borthwick, and P. H. Taylor, *From the paddle to the beach–A Boussinesq shallow water numerical wave tank based on Madsen and Sørensen's equations*, Journal of Computational Physics, 231 (2012), pp. 328–344.

[36] B. Peherstorfer, *Model reduction for transport-dominated problems via online adaptive bases and adaptive sampling*, arXiv e-prints, arXiv:1812.02094, (2018).

[37] B. Peherstorfer, Z. Drmac, and S. Gugercin, *Stability of discrete empirical interpolation and gappy proper orthogonal decomposition with randomized and deterministic sampling points*, SIAM Journal on Scientific Computing, 42 (2020), pp. A2837–A2864.

[38] J. Reiss, P. Schulze, J. Sesterhenn, and V. Mehrmann, *The shifted proper orthogonal decomposition: A mode decomposition for multiple transport phenomena*, SIAM Journal on Scientific Computing, 40 (2018), pp. A1322–A1344.

[39] M. Ricchiuto and A. G. Filippini, *Upwind residual discretization of enhanced Boussinesq equations for wave propagation over complex bathymetries*, Journal of Computational Physics, 271 (2014), pp. 306–341.

[40] R. Stefanescu and I. M. Navon, *POD/DEIM nonlinear model order reduction of an ADI implicit shallow water equations model*, Journal of Computational Physics, 237 (2013), pp. 95–114.

[41] R. Stefanescu, A. Sandu, and I. M. Navon, *Comparison of POD reduced order strategies for the nonlinear 2D shallow water equations*, International Journal for Numerical Methods in Fluids, 76 (2014), pp. 497–521.

[42] M. Strazzullo, F. Ballarin, R. Mosetti, and G. Rozza, *Model reduction for parametrized optimal control problems in environmental marine sciences and engineering*, SIAM Journal on Scientific Computing, 40 (2018), pp. B1055–B1079.

[43] M. Strazzullo, F. Ballarin, and G. Rozza, *POD-Galerkin model order reduction for parametrized nonlinear time dependent optimal flow control: an application to Shallow Water Equations*, Journal of Numerical Mathematics, (2021).

[44] T. Taddei, *A registration method for model order reduction: data compression and geometry reduction*, SIAM Journal on Scientific Computing, 42 (2020), pp. A997–A1027.

[45] M. Tonelli and M. Petti, *Hybrid finite volume–finite difference scheme for 2DH improved Boussinesq equations*, Coastal Engineering, 56 (2009), pp. 609–620.

[46] D. Torlo, *Model reduction for advection dominated hyperbolic problems in an ALE framework: Offline and online phases*, arXiv preprint arXiv:2003.13735, (2020).

[47] M. A. Walkley, *A numerical method for extended Boussinesq shallow-water wave equations*, PhD thesis, University of Leeds, 1999.

[48] G. Wei, J. T. Kirby, and A. Sinha, *Generation of waves in Boussinesq models using a source function method*, Coastal Engineering, 36 (1999), pp. 271–299.

[49] M. Yano, *Discontinuous Galerkin reduced basis empirical quadrature procedure for model reduction of parametrized nonlinear conservation laws*, Advances in Computational Mathematics, 45 (2019), pp. 2287–2320.

[50] R. Zimmermann, B. Peherstorfer, and K. Willcox, *Geometric subspace updates with applications to online adaptive nonlinear model reduction*, SIAM Journal on Matrix Analysis and Applications, 39 (2018), pp. 234–261.