

SMAI-JCM
SMAI JOURNAL OF
COMPUTATIONAL MATHEMATICS

Modified parareal method for
solving the two-dimensional
nonlinear shallow water equations
using finite volumes

JOAO G. CALDAS STEINSTRASSER, VINCENT GUINOT & ANTOINE ROUSSEAU
Volume 7 (2021), p. 159-184.

http://smai-jcm.centre-mersenne.org/item?id=SMAI-JCM_2021__7__159_0

© Société de Mathématiques Appliquées et Industrielles, 2021
Certains droits réservés.



creative
commons



Publication membre du

Centre Mersenne pour l'édition scientifique ouverte

<http://www.centre-mersenne.org/>

l'Inria

INVENTEURS DU MONDE NUMÉRIQUE

Soumission sur <https://smai-jcm.centre-mersenne.org/ojs/submission>





Modified parareal method for solving the two-dimensional nonlinear shallow water equations using finite volumes

JOAO G. CALDAS STEINSTRÄESSER¹
VINCENT GUINOT²
ANTOINE ROUSSEAU³

¹ Inria, IMAG, Univ Montpellier, CNRS, Montpellier, France

E-mail address: joao-guilherme.caldas-steinstraesser@inria.fr

² Univ Montpellier, HSM, CNRS, IRD, Inria, Montpellier, France

E-mail address: vincent.guinot@inria.fr

³ Inria, IMAG, Univ Montpellier, CNRS, Montpellier, France

E-mail address: antoine.rousseau@inria.fr.

Abstract. In this work, the POD-DEIM-based parareal method introduced in [8] is implemented for solving the two-dimensional nonlinear shallow water equations using a finite volume scheme. This method is a variant of the traditional parareal method, first introduced by [22], that improves the stability and convergence for nonlinear hyperbolic problems, and uses reduced-order models constructed via the Proper Orthogonal Decomposition - Discrete Empirical Interpolation Method (POD-DEIM) applied to snapshots of the solution of the parareal iterations. We propose a modification of this parareal method for further stability and convergence improvements. It consists in enriching the snapshots set for the POD-DEIM procedure with extra snapshots whose computation does not require any additional computational cost. The performances of the classical parareal method, the POD-DEIM-based parareal method and our proposed modification are compared using numerical tests with increasing complexity. Our modified method shows a more stable behaviour and converges in fewer iterations than the other two methods.

2020 Mathematics Subject Classification. 68W10, 76B07.

Keywords. Parareal method, POD-DEIM, reduced-order model, finite volume, shallow water equations.

1. Introduction

The trade-off between high-fidelity results and affordable computational costs is a real challenge encountered by many mathematical models for physical problems. Computing approximate solutions close enough to analytical or experimental results usually requires fine temporal and/or spatial discretizations, which may be linked by stability conditions, and high-order numerical schemes. The combination of these factors may lead the computational time and memory cost to a prohibitive magnitude, limiting potential applications of these mathematical models.

Over the past decades, parallel-in-time (PinT) methods have arisen as a promising strategy for overcoming part of such challenge. Various types of methods have been proposed, including predictor-corrector iterative algorithms, methods based on spatial domain decomposition and noniterative, direct solvers. We refer the reader to [17, 25] for detailed reviews. One of the most popular PinT algorithms, included in the first category mentioned above, is the parareal method, first developed by [22] and whose name stands for “parallel in real-time”. It has been applied to a wide range of problems. Examples are fluid dynamics [15, 32], molecular dynamics [4], chemistry [12], geodynamics [30] and finance [26]. The popularity of the method is explained by its simple formulation and by the fast convergence and substantial reduction of the computational cost in many problems, especially parabolic ones. However, when hyperbolic problems are dealt with, even very simple ones such as the one-dimensional advection equation, the method suffers from a slow convergence and a lack of stability. It is thus of limited practical interest for this type of application [28].

A number of alternatives are available from the literature. Some of them propose to enrich the predictor-corrector parareal iteration with the solution computed in previous parareal iterations. In the case of linear problems, one can cite the Krylov subspace-enhanced parareal method [14, 18]; in a more general framework, [8] propose to use reduced-order models (ROMs) constructed on-the-fly from the parareal solution. ROMs are, in themselves, an approach to the challenge presented in the beginning of this introduction. Their purpose is to replace an expensive, high-dimensional problem by a low-dimensional one. In order to tackle nonlinear problems, [8] propose to introduce, in the parareal iterations, a ROM constructed using two combined model reduction techniques: the Proper Orthogonal Decomposition (POD) and the Empirical Interpolation Method (EIM) [5]. This combined method is usually known as POD-Discrete Empirical Interpolation Method (POD-DEIM) [7].

In this work, we solve the nonlinear two-dimensional shallow water equations (SWE), discretized with a finite volume scheme, using parareal methods. This set of depth-averaged flow equations, used in many applications, from urban floods [19] to atmospheric circulation [31], may have prohibitive computational costs due to the Courant-Friedrichs-Lewy (CFL) stability condition in explicit schemes. We propose a modification of the POD-DEIM-based method proposed in [8] for improving the speed of convergence and the stability. This modification consists in enriching the snapshot sets used as inputs for the POD-DEIM procedures. It does not require extra computational cost for computing the additional snapshots. However, it increases the cost of the POD-DEIM itself, and the method should provide convergence in very few iterations.

This paper is structured as follows: in Section 2, we present an overview of some parareal methods, namely the original one and its adaptation, suitable for treating nonlinear hyperbolic problems, using reduced-order models; in Section 3, we briefly discuss the impact of the snapshot set on the quality of the reduced-order models and we propose our modification of the POD-DEIM-based parareal method for enriching the computed ROMs; a speedup estimation is presented in Section 4; numerical tests are reported in Section 5, focusing on the comparison of three parareal methods, the classical one, the POD-DEIM-based and our proposed modification, in terms of convergence and computational cost; and a discussion and a conclusion are presented in Section 6.

2. Overview of parareal methods

The parareal method, first developed by [22], is an iterative predictor-corrector algorithm that involves parallel computation in time for solving time-dependent differential equations with a smaller computational cost. It is based on the use of two time-integration solvers (also called propagators). The coarser of the two, acting as a predictor, is computed sequentially along the entire time domain. It yields an approximate solution at a relatively low computational cost. The finer propagator produces corrections for the predicted solution. Being more expensive, it is computed in parallel within each time step of the coarse propagator [15]. The acceleration of the computational time by the parareal method is based on the joint use of a coarse and a fine time steps, since the fine computation can be parallelized using as many processors as there are coarse time steps.

The method stands out for a very simple problem-independent implementation. It is known to be computationally efficient when applied to complex diffusive problems. In contrast, it is inefficient (i.e. it is not faster than a full fine simulation) and may become unstable when applied to hyperbolic or advection-dominated problems, even the simplest ones, such as the 1D linear advection equation with a constant speed. This problem has been identified and studied in many works, e.g. [10, 16, 28]; [28] indicates that this lack of stability is due to the different discrete phase speeds in the coarse and fine propagators.

Many works, e.g. [8, 10, 13, 29], propose adaptations and developments of the parareal method in order to improve its performance when applied to hyperbolic problems. The present work focuses on

the method introduced by [8], which can be efficiently implemented for solving nonlinear problems by using reduced-order models (ROMs). In this section we briefly present the original and the ROM-based parareal algorithms.

2.1. The original parareal method

We follow [15] by presenting the original or classical parareal algorithm applied to the simple time-dependent problem

$$\begin{aligned} \frac{d}{dt}\mathbf{y}(t) + A\mathbf{y}(t) &= 0, \quad \text{in } [0, T] \\ \mathbf{y}(0) &= \mathbf{y}_0 \end{aligned} \quad (2.1)$$

where A is assumed time-independent for the sake of simplicity.

Let us define two discrete integration schemes F_t and G_t , called *fine* and *coarse propagators* respectively, that solve (2.1) numerically using fixed time steps (respectively t and Δt). $\Delta t \ll t$, therefore a full time integration over $[0, T]$ is much cheaper using the coarse propagator than the fine one. We also assume that $\rho := \Delta t / t$ is an integer. We remark that G_t can also be characterized by a coarser spatial discretization, a cheaper integration scheme or a simplified mathematical model [2]. When convenient, we denote by M_f and M_c the size of the spatial discretization (e.g. the number of cells in a finite volume scheme) of F_t and G_t , respectively.

Consider a homogeneous temporal discretization of $[0, T]$ using a constant time step Δt and resulting in $N_t + 1$ discrete time instants:

$$t_n = n\Delta t, \quad n = 0, \dots, N_t$$

with $T = t_{N_t}$. Following the notation in [29], we denote the propagation of a solution \mathbf{y} from t_0 to t_1 by F_t , using a constant time step t , and by G_t , using a constant time step Δt , respectively by $F_t(\mathbf{y}, t_1, t_0)$ and $G_t(\mathbf{y}, t_1, t_0)$.

The computation of approximate fine solutions $\mathbf{y}_n = \mathbf{y}(t_n)$ defined on the time instants of the coarse temporal discretization, with $\mathbf{y}_n := F_t(\mathbf{y}_{n-1}, t_n, t_{n-1}) = F_t(\mathbf{y}_0, t_n, 0)$, can be very expensive in terms of computational time, since it must be done sequentially along the fine time steps. The parareal algorithm allows this problem to be overcome by introducing an iterative predictor-corrector procedure that uses the coarser propagator G_t :

$$\mathbf{y}_{n+1}^{k+1} = \underbrace{G_t(\mathbf{y}_n^{k+1}, t_{n+1}, t_n)}_{\text{Prediction}} + \underbrace{F_t(\mathbf{y}_n^k, t_{n+1}, t_n) - G_t(\mathbf{y}_n^k, t_{n+1}, t_n)}_{\text{Correction}} \quad (2.2)$$

where the superscript k indicates the iteration number.

Note that, given a previously computed solution $(\mathbf{y}_n^k)_{n=0}^{N_t}$ in all instants of the coarse temporal discretization during iteration k , the only term in (2.2) that must be done sequentially for the computation of the next iteration's solution is the low-expensive, coarse prediction. On the other hand, the correction terms in each coarse time step, including the expensive fine propagation, can be computed in parallel, since they depend on $(\mathbf{y}_n^k)_{n=0}^{N_t}$. Therefore, assuming that there are as much processors as coarse time steps for the simulation, each parallel processor performs per iteration only ρ fine time steps using the fine propagator F_t (instead of ρN_t time steps as in the full fine sequential simulation).

To make it clearer, the iterative process (2.2) is described in detail in Algorithm 16. The steps of the algorithm are named with “prediction” and “correction” interpreted as in (2.2), following [15]. It differs from the nomenclature adopted in [29] and [8]. A discussion on the different interpretations of the predictor-corrector procedure in the parareal method is proposed in [23].

We remark that, by construction, the parareal algorithm provides, at iteration k , exact convergence to the fine, reference solution at time t_k [16]. It implies that the parareal method always provide exact convergence in a finite number of iterations. However, the algorithm is interesting in terms of computational time only if the convergence is obtained within a number of iterations k_{cvg} that is

```

1 Initialization: initial guess given by the coarse propagator:
2  $y_0^0 = y_0$ 
3 for  $n = 0$  to  $N_t - 1$  do
4   |  $y_{n+1}^0 = G_t(y_n^0, t_{n+1}, t_n)$ 
5 end
6 Iterations:
7 for  $k = 0$  to  $N_{itermax} - 1$  do
8   | Compute the fine term of the correction (in parallel):
9   | for  $n = 0$  to  $N_t - 1$  do
10  | |  $y_{n+1}^k = F_t(y_n^k, t_{n+1}, t_n)$ 
11  | | end
12  | | Compute the coarse predictions and correct them to obtain the final solution in the iteration
13  | | (sequentially):
14  | | for  $n = 0$  to  $N_t - 1$  do
15  | | |  $y_{n+1}^{k+1} = G_t(y_n^{k+1}, t_{n+1}, t_n) + y_{n+1}^k - G_t(y_n^k, t_{n+1}, t_n)$ 
16  | | | end
17 end

```

Algorithm 1: Original parareal algorithm

substantially smaller than N_t : if $k_{cvg} = N_t$, the parareal simulation is necessarily more expensive than the full fine sequential one. This convergence property also implies that the number of time steps to be computed decreases along iterations, and a convergence criterion (not explicitly introduced in Algorithm 16 for the sake of conciseness) can be established for deciding on the first time step to be computed. In practice, we follow [2] by considering a criterion based on the distance between the correction term and the iteration's solution:

$$k_n := \frac{y_n^k - y_n^k}{y_n^k} < \text{TOL} \quad (2.3)$$

2.2. The POD-DEIM-based parareal algorithm for nonlinear problems

A family of modifications of the parareal method proposed in the literature and aiming to overcome its issues when applied to hyperbolic problems consists in improving the coarse propagator by using information provided by previous iterations. In the framework of the Parallel Implicit Time-integration Algorithm (PITA), which is equivalent to parareal in linear problems [18], [14] proposed to project the solution into a subspace spanned by the previous parareal solution at all time instants, an approach called by [18] as Krylov subspace-enhanced parareal method. However, this adaptation introduces the fine, expensive propagator in the sequential step of the parareal algorithm, which can only be efficiently computed (i.e. less expensively than the reference solution) with a linearity assumption of the problem. In the case of nonlinear problems, an efficient approach is proposed by [8]. It consists in replacing the coarse propagator by a reduced-order model (ROM), formulated on-the-fly along the parareal iterations using the previous parareal solutions. Several approaches can be considered for the model reduction; we consider here the combined Proper Orthogonal Decomposition - Discrete Empirical Interpolation Method (POD-DEIM), which is briefly described in the following paragraphs, based on [7].

2.2.1. The POD-DEIM reduction method

Consider the following system of nonlinear ODEs, that can be obtained by the spatial discretization of a scalar PDE:

$$\frac{d}{dt}\mathbf{y}(t) = A\mathbf{y}(t) + \mathbf{F}(\mathbf{y}(t)) \quad (2.4)$$

with $\mathbf{y} \in \mathbb{R}^{M_f}$, $A \in \mathbb{R}^{M_f \times M_f}$ being a constant-in-time matrix and \mathbf{F} a nonlinear function with values in \mathbb{R}^{M_f} . We choose here the notation M_f , referring to the fine discretization, to show that this quantity is possibly too large, in which case solving (2.4) may be expensive. Thus, one may seek to approximate it by a problem with smaller dimension.

Reduction of the linear term. The original system (2.4) is approximated with a reduced-order system of order $q \ll M_f$, via the construction of a subspace S_q of \mathbb{R}^{M_f} with dimension q . Let $V_q = [v_1, \dots, v_q] \in \mathbb{R}^{M_f \times q}$ be a matrix whose columns are the vectors of an orthonormal basis of S_q . The reduction of system (2.4) consists in replacing $\mathbf{y}(t)$ by its approximation $V_q\mathbf{y}(t)$ (with $\mathbf{y} \in \mathbb{R}^q$) and projecting the system onto S_q :

$$\frac{d}{dt}\mathbf{y}(t) = V_q^T A V_q \mathbf{y}(t) + V_q^T \mathbf{F}(V_q \mathbf{y}(t)) \quad (2.5)$$

The reduced basis vectors $\{v_i\}_{i=1}^q$ of S_q can be obtained via a Proper Orthogonal Decomposition (POD) of a snapshot matrix $Y = [\mathbf{y}(t_1), \dots, \mathbf{y}(t_{n_s})] \in \mathbb{R}^{M_f \times n_s}$, containing the solution in chosen n_s time instants. $\{v_i\}_{i=1}^q$ are solution of a minimization problem (we refer the reader to [7] for details) and are the first q left singular vectors of Y (i.e. those associated to the largest singular values). Therefore, the POD can be performed via a Singular Value Decomposition (SVD) of Y :

$$Y = V \Sigma W^T$$

where $V = [v_1, \dots, v_r] \in \mathbb{R}^{M_f \times r}$ and $W = [w_1, \dots, w_r] \in \mathbb{R}^{n_s \times r}$ are respectively the left and right singular vectors of Y , and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r) \in \mathbb{R}^{r \times r}$, $\sigma_1 \geq \dots \geq \sigma_r > 0$ is a diagonal matrix containing their respective singular values. A threshold σ_{sv} must be set for choosing the $q \ll r$ singular values to be kept, in order to obtain the associated basis vectors $v_i = v_i, i = 1, \dots, q$.

If the nonlinear term $V_q^T \mathbf{F}(V_q \mathbf{y}(t))$ in (2.5) is neglected, it is effectively a reduced problem w.r.t. the original one (2.4) ($q \ll M_f$), since the matrix $V_q^T A V_q$ does not depend on time and can be precomputed. Nevertheless, the nonlinear term limits the reduction of the model, since it must be computed on M_f points at each time step (because $V_q \mathbf{y}(t) \in \mathbb{R}^{M_f}$). Therefore, a separate reduction approach must be defined for the nonlinear term.

Reduction of the nonlinear term. The limitation of the reduction technique for nonlinear problems can be overcome by combining the POD method with the Empirical Interpolation Method (EIM), introduced by [5]. This combined technique is called as POD-Discrete Empirical Interpolation Method (DEIM) by [7] and is a particular and simplified framework of the EIM. Let $\mathbf{f}(t)$ be a generic nonlinear function with values in \mathbb{R}^{M_f} . The POD-DEIM model reduction consists of two steps: firstly, a POD is performed on a set of snapshots of the nonlinear term, $Y = [\mathbf{f}(t_1), \dots, \mathbf{f}(t_{n_s})] \in \mathbb{R}^{M_f \times n_s}$, in order to obtain a subspace S_m of \mathbb{R}^{M_f} with dimension $m \ll M_f$, whose basis vectors are the columns of a matrix $V_m = [v_1, \dots, v_m] \in \mathbb{R}^{M_f \times m}$. Secondly, a greedy algorithm is performed in order to choose m spatial points on which the nonlinear term will be computed. These spatial points are identified by the indices P_1, \dots, P_m and the algorithm provides a matrix $P = [e_{P_1}, \dots, e_{P_m}] \in \mathbb{R}^{M_f \times m}$, where e_i are vectors of the canonical basis of \mathbb{R}^{M_f} .

Then, $\mathbf{f}(t)$ is approximated by

$$\mathbf{f}(t) \approx V_m P^T V_m^{-1} P^T \mathbf{f}(t) \quad (2.6)$$

We refer the reader to [7] for details on the greedy DEIM algorithm and on the derivation of (2.6). Note that a left-hand multiplication of a vector $\mathbf{v} \in \mathbb{R}^{M_f}$ by P^T is equivalent to choosing the components of \mathbf{v} indexed by P_1, \dots, P_m . Therefore, equation (2.6) shows that we can approximate $\mathbf{f}(t) \in \mathbb{R}^{M_f}$ by computing it at $m = M_f$ points and interpolating over the full set of M_f points.

Using (2.6) in (2.5) with $\mathbf{f}(t) = \mathbf{F}(V_q \mathbf{y}(t))$, we obtain the POD-DEIM reduced model for (2.4):

$$\frac{d}{dt} \mathbf{y}(t) = V_q^T A V_q \mathbf{y}(t) + V_q^T V_m P^T V_m^{-1} P^T \mathbf{F}(V_q \mathbf{y}(t)) \quad (2.7)$$

We notice that the matrix $V_q^T V_m P^T V_m^{-1} \in \mathbb{R}^{q \times m}$ does not depend on time and can be precomputed, and that $P^T \mathbf{F}(V_q \mathbf{y}(t))$ is a vector of size $m = M_f$.

2.2.2. Application of the POD-DEIM reduction method in the parareal algorithm

The POD-DEIM approach is introduced in the framework of the parareal method by [8]. The idea is to replace, in the parareal iteration (2.2), the coarse model G_t by the ROM

$$F_{r,t}^k(\mathbf{P}^k \mathbf{y}, t_1, t_0)$$

defined by the subspaces S_q^k and S_m^k and the same time step τ as F_t ; and \mathbf{P}^k is the projection onto S_q^k . Therefore, the coarse step of the algorithm uses a fine temporal discretization, but with only $q = M_f$ degrees of freedom, $m = M_f$ components to be computed in the nonlinear term and expensive matrices that do not depend on time and can be computed only once per iteration. In what follows, we omit, for the sake of clearness, the subscript denoting the dimension of the subspaces, and we keep only the superscript indicating the parareal iteration. The resulting parareal method is named in [8] as reduced basis parareal method, in a more generic way, since other model reduction procedures are also considered. We name it hereafter as POD-DEIM-based parareal method (or PD method, for simplification) for focusing on the POD-DEIM technique. The algorithm is presented in Algorithm 2, in which the modifications w.r.t. the original parareal method (Algorithm 16) are highlighted. Note that the coarse propagator G_t is still used in the 0-th iteration (in which there are no snapshots available), for producing the (low-expensive) initial prediction, but is no longer used in the following iterations, being replaced by the reduced-order model. Also note that the ROMs are reformulated at each iteration, using as snapshots the parareal solution computed at all coarse time steps and all previous iterations. In lines 16 and 17 of Algorithm 2, we include in the definition of the spaces S^k and S^k the snapshots set and the singular value threshold they depend on. These thresholds may not be the same in the two cases, and, with some abuse of nomenclature, we call $\tau_{sv, linear}$ the one used for computing the spaces spanned by snapshots of the solution, in contrast to the spaces spanned by nonlinear snapshots. Moreover, we remark that, depending on the problem and the formulation of the reduced model, more than one space of each type may be computed at each iteration. The reduced-order model for the shallow water equations, described in Section 5, is an example.

3. A modification of the POD-DEIM-based parareal algorithm for improving the reduced model

```

1 Initialization: initial guess given by the coarse propagator:
2  $y_0^0 = y_0$ 
3 for  $n = 0$  to  $N_t - 1$  do
4   |  $y_{n+1}^0 = G_t(y_n^0, t_{n+1}, t_n)$ 
5 end
6 Iterations:
7 for  $k = 0$  to  $N_{itermax} - 1$  do
8   Compute the fine term of the correction (in parallel):
9   for  $n = 0$  to  $N_t - 1$  do
10    |  $y_{n+1}^k = F_t(y_n^k, t_{n+1}, t_n)$ 
11  end
12  Define the snapshots sets:
13   $Y^k = \{y_n^j, j = 0, \dots, k; n = 0, \dots, N_t\}$ 
14   $Y^k = \{F(y_n^j), j = 0, \dots, k; n = 0, \dots, N_t\}$ 
15  Compute the spaces and define the reduced model  $F_{r,t}^k$ :
16   $S^k(Y^k, sv, linear)$  (using POD)
17   $S^k(Y^k, sv, nonlinear)$  (using POD-DEIM)
18  Compute the coarse term of the correction (in parallel):
19  for  $n = 0$  to  $N_t - 1$  do
20    |  $y_{n+1}^k = F_{r,t}^k(P^k y_n^k, t_{n+1}, t_n)$ 
21  end
22  Compute the coarse predictions and correct them to obtain the final solution in the iteration
    (sequentially):
23  for  $n = 0$  to  $N_t - 1$  do
24    |  $y_{n+1}^{k+1} = F_{r,t}^k(P^k y_n^{k+1}, t_{n+1}, t_n) + y_{n+1}^k - \bar{y}_{n+1}^k$ 
25  end
26 end

```

Algorithm 2: POD-DEIM-based parareal algorithm.

3.1. Influence of the snapshot sets in the definition of the reduced-order models

The choice of the snapshot set is a key factor in the formulation of reduced-order models. Indeed, for a given snapshot set, the reduced basis constructed via the POD is optimal, i.e. it minimizes an approximation error concerning the snapshots [7], but if the snapshots are not a meaningful representation of the solution of a problem, the reduced-order model will be a low-quality approximation for this problem.

Moreover, as discussed in [27], even if the POD is optimal in approximating a given dataset, it does not necessarily provide the best possible description for the dynamics that generate the snapshots. In this case, including more POD modes (i.e. retaining more basis vectors) may degrade the quality of the reduced model.

Many approaches are presented in the literature for improving snapshot selection and/or overcoming misrepresentations and instabilities of POD-based reduced-order models. Examples are: subtracting the mean from the snapshots so as to consider only the fluctuations in the solution [24]; enriching the snapshot set with finite difference quotients [24] or with the gradient [1] of the solution; greedy approaches for snapshot selection [9]; and the balanced POD method [27], that requires solving a dual problem.

As shown in the numerical tests presented in Section 5, the POD-DEIM-based parareal is unstable for a number of test cases, specially when the time step and/or spatial discretization of the coarse propagator G_t is much coarser than that of the fine propagator F_t ($\Delta t \gg \tau$). This suggests two possible causes for the observed instability:

- (1) Since the snapshots are selected at every coarse time step Δt , if $\Delta t \gg \tau$ there are not enough snapshots for properly representing the fine solution;
- (2) The errors due to the coarse spatial and temporal discretization make the coarse solution a poor approximation for the fine one.

3.2. Proposed modification

We propose a modification of Algorithm 2 for minimizing the first cause. It consists in enriching the snapshots sets with intermediate snapshots of the fine correction term (computed in line 10 of Algorithm 2) between the instants of the coarse temporal mesh. The intermediate snapshots are taken every τ , $t < \Delta t < \Delta t$, and stored in a set Υ_k^n defined for each iteration and coarse time step:

$$\Upsilon_k^n = \{F_t(\mathbf{y}_n^k, t_n + l\tau, t_n), l = 1 \text{ s.t. } l\Delta t < \Delta t\}, \quad n = 0, \dots, N_t - 1$$

Therefore, the snapshots sets in each iteration k of the proposed modified algorithm read

$$Y^k = \{\mathbf{y}_n^j, j = 0, \dots, k; n = 0, \dots, N_t\} \cup \bigcup_{n=0}^{N_t-1} \bigcup_{j=0}^{N_t-1-k} \Upsilon_n^k$$

$$Y^k = \{F(\mathbf{y}_n^j), j = 0, \dots, k; n = 0, \dots, N_t\} \cup \bigcup_{n=0}^{N_t-1} \bigcup_{j=0}^{N_t-1-k} F \Upsilon_n^k$$

where $F \Upsilon_n^k$ is a set containing the nonlinear function F computed on every element of Υ_n^k .

We remark that, in the framework of linear problems, using POD-only reduced-order models, results presented by [21] suggest that the convergence of the ROM-based parareal method can be accelerated when there is enough input snapshots. However, no discussion on this topic is proposed by the authors. We propose here to investigate how the snapshot enrichment can improve the convergence and stability of the POD-DEIM based parareal algorithm.

Note that the proposed modification of the algorithm, to which we refer hereafter as modified POD-DEIM-based (MPD) parareal method, does not require any extra computational cost for generating the snapshots. Indeed, in Algorithm 2, the intermediate snapshots are computed but not used. Due to this non-intrusive aspect, we propose this approach instead of focusing on the second cause of instability mentioned above (i.e. modifying the coarse propagator, what in some applications one may not be able to choose freely, or may be limited by stability conditions).

However, as discussed in the speedup estimation in Section 4, the cost of the POD procedure grows quadratically with the number of snapshots. We also recall that the number of snapshots is proportional to the number of iterations. Therefore, the number of intermediate snapshots may not be excessive. By defining $\Delta t = \Delta t / \nu$, $\nu \in \mathbb{N}$, one should naturally begin by setting $\nu = 2$.

4. Speedup estimation

We estimate in this section the speedup provided by the modified POD-DEIM-based parareal method. In this estimation, we neglect overhead costs associated to the parallelization, e.g. initialization of processors and communication between them. As discussed in Section 5, the numerical implementation

considered in this work makes use of a shared memory parallelization, such that communication costs are not present.

We recall that Δt and t are the time steps respectively in the coarse (G_t) and in the fine models (F_t), and that the fine reduced model ($F_{r,t}^k$) also propagates with the fine time step t . For a simulation with total length T , we have $N_t = T/\Delta t$ coarse and $N_t = T/t$ fine time steps, with N_t, N_t and $\rho = N_t/N_t$ supposed to be integers. Concerning the spatial discretization, we denote by M_f and M_c the number of degrees of freedom in the fine and coarse propagators respectively. $m(k)$ is the number of degrees of freedom of $F_{r,t}^k$, in iteration k of the parareal algorithm.

Let t_c be the computational time for advancing one time step Δt on the coarse model G_t ; and t_f and $t_r(k)$ the computational times for advancing one time step t using respectively F_t and $F_{r,t}^k$. Since the reduced model $F_{r,t}^k$ is reformulated at each iteration k , its computational time depends on the iteration.

We also denote by $t_S(k)$ the computational time for computing a subspace S^k (obtained from snapshots of the solution) or S^k (obtained from snapshots of the nonlinear terms). Since the former corresponds to a POD (via a SVD) and the latter corresponds to a POD and a subsequent DEIM, we majorize $t_S(k)$ by the computational time for computing the subspaces S^k . We denote by N_{spaces} the number of subspaces to be computed in each iteration, and by \hat{m} the highest dimension among all subspaces and all iterations. Since the subspaces S and S are independent, their computation can also be parallelized. We suppose that there are $N_{p,S}$ processors, or groups of processors, for performing the POD and the POD-DEIM procedures.

The reference computational time (i.e. for performing a full fine sequential simulation) is $T_{\text{ref}} = N_t t_f$. Concerning the MPD parareal simulation with \hat{k} iterations and using N_p processors, its total computational time $T_{\text{par}}(\hat{k}, N_p)$, as detailed in Algorithm 2, involves:

- an initial full coarse prediction taking $T_c = N_t t_c$;
- within each iteration k :
 - the parallel computation of the fine term of the correction, taking

$$T_{\text{corr},f} = \frac{N_t N_t}{N_p N_t} t_f$$

- the possibly parallel computation of the subspaces, taking

$$T_{\text{spaces}} = \frac{N_{\text{spaces}}}{N_{p,S}} t_S(k)$$

- the parallel computation of the coarse term of the correction, taking

$$T_{\text{corr},c} = \frac{N_t N_t}{N_p N_t} t_r(k)$$

- the sequential computation of the predictions, taking

$$T_{\text{pred}} = N_t t_r(k)$$

Therefore,

$$\begin{aligned} T_{\text{par}}(\hat{k}, N_p) &= T_c + \hat{k} (T_{\text{corr},f} + T_{\text{spaces}} + T_{\text{corr},c} + T_{\text{pred}}) \\ &= N_t t_c + \hat{k} \left(\frac{N_t}{N_p} t_f + t_r(\hat{k}) + \frac{N_{\text{spaces}}}{N_{p,S}} t_S(\hat{k}) + N_t t_r(\hat{k}) \right) \end{aligned}$$

Note that $s(k)$ and $r(k)$ are supposed to increase along iterations, since the spaces are constructed using an increasing number of snapshots and may have an increasing dimension. Therefore, both these parameters are majorized by their value at iteration \hat{k} .

Therefore, the speedup provided by the POD-DEIM parareal algorithm can be estimated as

$$\begin{aligned} s(\hat{k}, N_p) &= \frac{T_{\text{ref}}}{T_{\text{par}}(\hat{k}, N_p)} \\ &= \frac{1}{\frac{N_{t,c}}{N_{t,f}} + \frac{\hat{k}}{N_p} \frac{f + r(\hat{k})}{f} + \hat{k} \frac{r(\hat{k})}{f} + \hat{k} \frac{N_{\text{spaces}} s(\hat{k})}{N_{p,s} N_{t,f}}} \end{aligned} \quad (4.1)$$

We refer the reader to [8, 29] for the derivation and interpretation of various bounds to the speedup estimate (4.1). We focus here on the bound due to the last term in the denominator of (4.1) and accounting to the ROM formulation, since it is directly related to the modification proposed here to the ROM-based parareal method. We have

$$s(\hat{k}, N_p) < \frac{N_{p,s} N_{t,f}}{\hat{k} N_{\text{spaces}} s(\hat{k})} \quad (4.2)$$

Let us develop the estimation for the computational time $s(\hat{k})$. As said above, it is majorized by the computational time for computing the subspaces \hat{S} , since this computation is more expensive than the one for the subspaces S . Moreover, in the framework of the modified POD-DEIM-based parareal algorithm, we enrich the snapshots set by keeping intermediate snapshots at every $\Delta t = \Delta t / \hat{k}$, $\hat{k} \leq N$. Within a given iteration k , the computation of \hat{S} involves:

- (1) A POD (via a SVD) for obtaining the subspace's basis, using as input a matrix of size $M_f \times k$ ($N_{t,c} + 1$). The obtained subspace has dimension $m \leq M_f$.
- (2) A DEIM for obtaining the m spatial indices for the reduced model, consisting in the sequential solve of $m - 1$ linear systems and m lookups for the maximum element in vectors of size M_f . The i -th linear system has a size $i \times i$ (see [7] for details).

The SVD and the solution of the linear systems are obtained using respectively the functions *dgesvd* and *dgesv* of the *MKL-LAPACK* suite. The lookup for the maximum element is performed using the function *max_element* from the *C++ Standard Library*.

The *MKL-LAPACK* function *dgesvd* applied to a matrix in $\mathbb{R}^{q \times n}$ has a complexity $O(qn^2 + n^3)$ [11]. Concerning *dgesv*, the complexity for a matrix in $\mathbb{R}^{n \times n}$ is $O(n^3)$ ¹. For the function *max_element*, the complexity for looking up a vector in \mathbb{R}^n is $O(n)$ ².

Therefore, the computational cost for the subspace computation can be estimated by

$$\begin{aligned} s(\hat{k}) &= O \left(M_f \hat{k}^2 N_{t,c}^2 + \hat{k}^3 N_{t,c}^3 + \sum_{i=2}^{\hat{m}} i^3 + \hat{m} M_f \right) \\ &= O \left(\hat{k}^2 N_{t,c}^2 (M_f + \hat{k} N_{t,c}) + \frac{\hat{m}^2 (\hat{m} + 1)^2}{4} - 1 + \hat{m} M_f \right) \\ &= O \left(M_f \hat{k}^2 N_{t,c}^2 + \frac{\hat{m}^4}{4} + \hat{m} M_f \right) \end{aligned}$$

¹<http://www.netlib.org/lapack/lug/node71.html>

²https://en.cppreference.com/w/cpp/algorithm/max_element#Complexity

where we assume $M_f \hat{k} N_t$ (which is expected in parareal algorithms) by keeping reasonably small and achieving a fast convergence. Therefore, from (4.2) and by considering $\tau_f = O(M_f)$ (which is expected in an explicit FV scheme), we have

$$s(\hat{k}, N_p) < \frac{N_{p,s} N_t M_f}{\hat{k} N_{\text{spaces}} M_f \hat{k}^2 - 2N_t^2 + \frac{\hat{m}^4}{4} + \hat{m} M_f} \quad (4.3)$$

Bound (4.3) shows firstly a quadratic dependence on the number of snapshots (at iteration k , there are $k(N_t + 1)$ snapshots), showing that both \hat{k} and the number of iterations should be kept small. Moreover, it reveals a strong dependence of the speedup on the dimension \hat{m} of the reduced model. Assuming indeed that this dimension is small, so that $\hat{m}^4/4$ can be neglected w.r.t. $M_f \hat{k}^2 - 2N_t^2$, we conclude that this bound does not depend on the spatial dimension M_f of the problem, but is strongly affected by the number of iterations (by a factor $1/\hat{k}^3$) and by the number of input snapshots. Finally, this bound can be improved by computing the subspaces in parallel, thus approaching $N_{p,s}/N_{\text{spaces}}$ to 1. However, since a parallel speedup is also expected in each execution of the *MKL-LAPACK* functions, a trade-off may be found in the distribution of processors and depends on the problem size.

5. Numerical results

We present in this section some numerical tests for solving the 2D nonlinear SWE using parareal methods. Our objective is to compare the performance of the original parareal method (Algorithm 16), the POD-DEIM-based parareal method (Algorithm 2) and its proposed variation, in terms of computational time and convergence to the reference solution, which is given by a full fine sequential simulation:

$$\mathbf{y}_{\text{ref},n} := F_t(\mathbf{y}_0, t_n, 0), \quad n = 1, \dots, N_t$$

We consider here the 2D nonlinear SWE as formulated in [19]. In fact, in that work a porosity-based shallow water model is proposed, in which the fine discretization of a spatial domain with obstacles (e.g. an urban zone) can be replaced by a coarse discretization of an analogous porous media. A porosity parameter is defined for each cell and inserted in the equations, representing the presence of obstacles. We consider here an unitary porosity, which leads to the classical SWE. They read

$$-\frac{\partial}{\partial t} \mathbf{U}(t) + \frac{\partial}{\partial x} \mathbf{F}(\mathbf{U}(t)) + \frac{\partial}{\partial y} \mathbf{G}(\mathbf{U}(t)) = \mathbf{S}(\mathbf{U}(t)) \quad (5.1)$$

with

$$\mathbf{U} = \begin{pmatrix} h \\ hu_x \\ hu_y \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} hu_x \\ hu_x^2 + gh^2/2 \\ hu_x u_y \end{pmatrix},$$

$$\mathbf{G} = \begin{pmatrix} hu_y \\ hu_x u_y \\ hu_y^2 + gh^2/2 \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} 0 \\ S_{0,x} + S_{f,x} \\ S_{0,y} + S_{f,y} \end{pmatrix}$$

where g is the gravitational acceleration, h is the water depth, u_x and u_y are respectively the velocities in the x and y directions, $S_{0,x}$ and $S_{0,y}$ are source terms related to spatial variations of the topography, and $S_{f,x}$ and $S_{f,y}$ are friction source terms.

Eq. (5.1) is discretized using an explicit-in-time finite volume (FV) scheme, for which a POD-DEIM reduced-order model can be straightforwardly derived following the procedure described in Section 2.2.1. For the sake of conciseness, the derivation of both the FV scheme and the POD-DEIM ROM are not presented here and we refer the reader to [6] for all the details. We only highlight here, as it will be useful in the discussion presented along this section, that the reduced model is constructed by computing three reduced subspaces $S_{q_i}^{(i)}$, $i = 1, 2, 3$ of dimension $q_i \ll M_f$ (with M_f the number of

cells of the spatial mesh used by F_t), respectively spanned by snapshots of the three components of the solution, and five reduced subspaces $\hat{S}_{m_l}^{(l)}$, $l = 1, \dots, 5$ of dimension $m_l = l_f$ (with l_f the number of interfaces of the mesh), respectively spanned by snapshots of the nonlinear term (three spaces accounting for the fluxes and the remaining two accounting for the source terms).

In this section, we propose a one-dimensional test case (in fact, a 2D one but with only one direction of propagation) and two-dimensional cases, the second of them using a coarse propagator also coarsened in space. We intend to observe empirically the influence of the initial coarse prediction on the construction of the reduced-order models and, as a consequence, on the convergence and stability behaviour of the POD-DEIM-based parareal method. Then, the eventual need for enriching the snapshots sets can be assessed.

Using the same notation as in the parareal algorithms presented in Section 2, \mathbf{y}_n^k represents the parareal solution at time t_n and iteration k . For the 2D nonlinear SWE, we have

$$\mathbf{y}_n^k = \begin{pmatrix} (\mathbf{U}^{(1)})_n^k \\ (\mathbf{U}^{(2)})_n^k \\ (\mathbf{U}^{(3)})_n^k \end{pmatrix} \in \mathbb{R}^{3M_f}$$

with $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)} \in \mathbb{R}^{M_f}$ containing respectively the values of h , hu_x and hu_y in all cells of the mesh associated to the fine propagator F_t .

By denoting as $[\mathbf{y}]_i$ the i -th component of the vector \mathbf{y} , we define the following errors between parareal and the reference solution

- Error at time t_n and iteration k , computed in the entire fine mesh:

$$e_n^k := \frac{\overline{M}_f \sum_{i=1}^{\overline{M}_f} |[\mathbf{y}_n^k]_i - [\mathbf{y}_{\text{ref}, n}]_i|}{\overline{M}_f \sum_{i=1}^{\overline{M}_f} |[\mathbf{y}_{\text{ref}, n}]_i|}$$

- Error at iteration k , computed in the entire fine mesh:

$$\bar{e}^k := \max_{n=0, \dots, N_t} e_n^k$$

where $\overline{M}_f = 3M_f$ for the SWE.

The code is parallelized using *OpenMP* directives and computational times are measured using the *OpenMP* function `omp_get_wtime`. Linear Algebra operations are performed using the *Intel MKL* suite, and, when the coarse and fine spatial meshes are different, the interpolation weights are given by a piecewise linear interpolation using Delaunay triangulations, which is made by the open source library *delaunay_linterp*³. These interpolation weights are precomputed to be reused in several simulations, such that the computational times presented in this section does not include the time to compute them. I/O operations are also excluded from the computational times. Note that there are no parallel communication costs involved in the simulations, since a shared memory parallelization paradigm is considered. All simulations, including the reference ones, are performed five times and their computational times are averaged to give the results presented here.

In all simulations, the convergence criterion (2.3) is used, with $\text{TOL} = 10^{-10}$, and we set a maximal number of iterations $N_{\text{itermax}} = 5$. For the POD-DEIM based parareal simulations, we use $\text{sv}_{\text{linear}} = \text{sv}_{\text{nonlinear}} = 10^{-3}$, unless explicitly specified.

All simulations are performed in a dual-Xeon E5-2680 v2 with a frequency of 2.80GHz, with 256GB of RAM capacity. For taking advantage of the shared memory parallelization, all parallel processors are in the same node, with 20 cores available per node. Therefore, in each simulation the number of processors is set to $N_p = \min\{N_t, 20\}$. In each iteration k , the computation of the subspaces $S^{(l),k}$, $l =$

³http://rncarpio.github.io/delaunay_linterp/

1, 2, 3 and $S^{(l),k}$, $l = 1, \dots, 5$, is done sequentially (relatively to each other), which corresponds to take $N_{p,S} = 1$ as defined in Section 4. Therefore, the parallelization of the ROM formulation is done inside the *MKL-LAPACK* functions.

In order to have an easier identification of the results presented in what follows, we propose the following nomenclature for the test cases:

$$\text{SWE}\{\text{nb.dimensions}\}\text{D-}\{\text{method}\}\text{-}\{N_t\}\text{-}[c]$$

where

- *nb.dimensions* can be 1 (for the 1D problem) or 2 (for the 2D problem);
- *method* can be *cl* (for the classical parareal method, presented in Algorithm 16), *pd* (for the POD-DEIM-based parareal method, presented in Algorithm 2) or *mpd* (for the modified POD-DEIM method using enriched snapshots sets);
- N_t is the number of coarse time steps, as defined previously;
- the last term, *c*, is used when there is spatial coarsening between the fine and the coarse meshes.

For example, *SWE1D-cl-25* stands for a 1D test case using the classical parareal method with 25 coarse time steps and no spatial coarsening, and *SWE2D-mpd-20-c* stands for the a 2D simulation using the modified POD-DEIM-based parareal method with 20 coarse time steps and spatial coarsening. Moreover, when necessary, the reference solution is denoted, for example, as *SWE1D-ref*.

5.1. First test case: 1D problem (SWE1D)

We consider in this test case a rectangular domain $\Omega = [0, 20]^2$. The fine propagator is described by a time step $\tau = 0.001$ and a Cartesian homogeneous mesh with $\Delta x = \Delta y = 1$, containing 400 cells and 840 interfaces. The initial solution is a lake-at-rest, with initial water depth $h = 1$ and a flat bottom. No friction is considered. A unitary and constant mass flux is imposed at the western boundary $x = 0$. The other three boundaries are closed (imposed null mass flux). The simulation length is $T = 5$, starting on $t = 0$. The average time for computing the sequential fine solution is of approximately $T_{\text{ref}} = 2.2\text{s}$.

We define a coarse propagator G_τ with $\Delta t = 0.2$ and no spatial coarsening. Therefore, there are $N_t = 25$ coarse time steps, and 20 processors are used for the simulations, such that some processors may compute two coarse time steps in the parallel computation of the correction terms. Following the nomenclature defined above, the three simulations are named *SWE1D-cl-25*, *SWE1D-pd-25* and *SWE1D-mpd-25*. The MPD method is defined by additional snapshots taken at every $\Delta t = 0.1$ (i.e. by taking $\nu = 2$, as defined previously).

The evolution of the errors e_n^k and \bar{e}^k is presented in Figure 5.1. We notice that the classical parareal method is very unstable, and the error increases along iterations. For the PD method, we observe a fast convergence to the reference solution and, for the MPD method, the convergence is even faster: in only one parareal iteration, the error in all the 25 timesteps is in the order of 10^{-11} , and the exact convergence (up to the chosen convergence threshold) is attained in two iterations.

Figure 5.2 exemplifies the behaviour of the methods by showing the evolution of water height h , compared to the reference solution, in a given point of the domain $((x, y) = (10, 5))$. We observe that the solution given by the classical parareal iterations is highly oscillatory and unstable, whereas both the POD-DEIM-based methods give, in only one iteration, a visually converged solution, indicating that the model reduction is able to well capture the fine dynamics in this relatively simple problem.

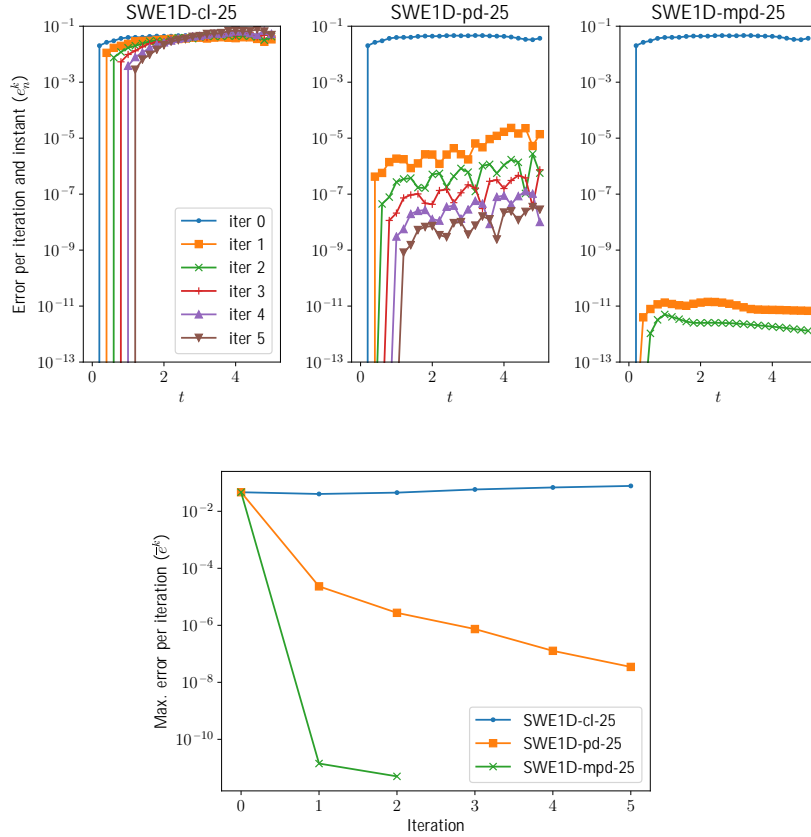


FIGURE 5.1. Evolution of the errors e_n^k along iterations and time for test cases *SWE1D-cl-25* (first plot), *SWE1D-pd-25* (second plot) and *SWE1D-mpd-25* (third plot); and evolution of the errors ζ^k along iterations for the same test cases (fourth plot). For *SWE1D-mpd-25*, the algorithm converges in 2 parareal iterations. The legend is the same in the first three plots.

Indeed, using only the first formulated ROM (i.e. not in the parareal iteration) would already provide a high-quality approximation to the fine solution, but the parareal iteration, by construction, is required for attaining exact convergence.

The speedup provided by each method along iterations is presented in Figure 5.3. The 0 – t h iteration (initial coarse prediction) is not shown for the sake of readability, since this iteration is much faster than the reference solution and the following iterations. As expected, the classical parareal method is the cheapest one, and the modified POD-DEIM-based method, the most expensive one. As said above, both the PD and MPD methods give a very good result in one iteration, so we can consider for them a speedup of approximately 5.0 and 3.9, respectively.

5.2. Second test case: 2D problem (SWE2D)

For the 2D test case, we consider a rectangular domain $\Omega = [0, 100]^2$, with a fine propagator described by $\tau = 0.001$ and a Cartesian homogeneous mesh with $x = y = 2$, containing 2500 cells and 5100 interfaces. The sequential fine simulation takes approximately $T_{\text{ref}} = 12.7\text{s}$, on average. The initial

MODIFIED PARAREAL METHOD FOR SOLVING THE SWE

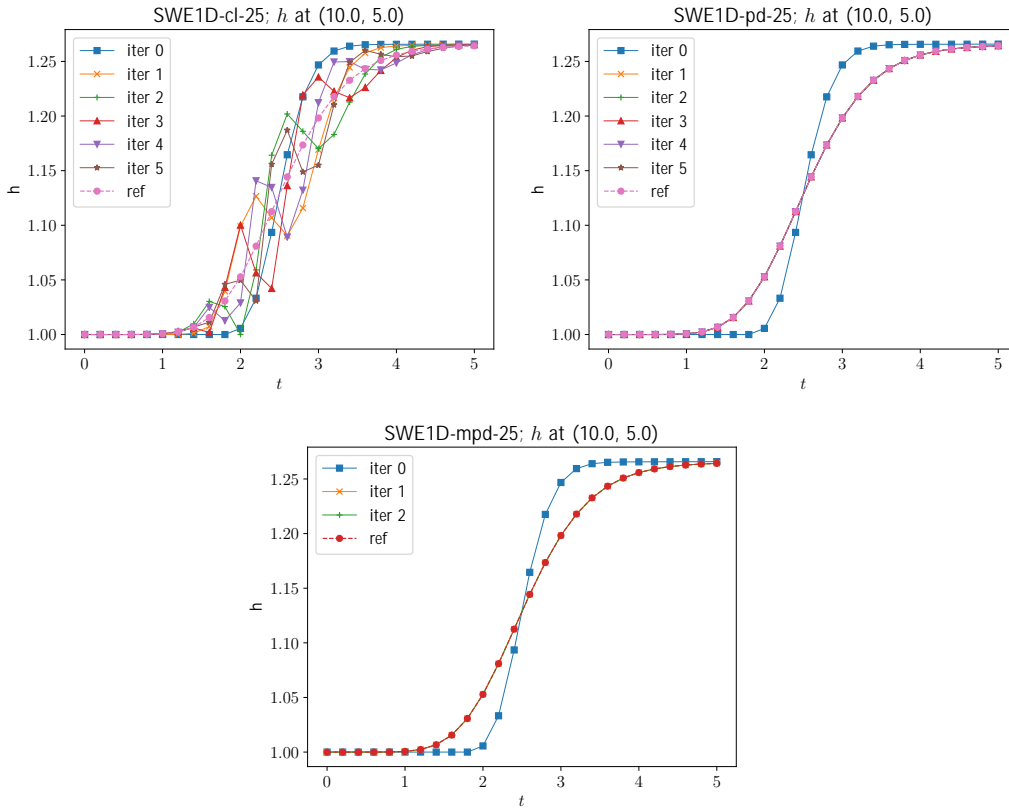


FIGURE 5.2. Evolution of h along time in position $(x, y) = (10, 5)$ for test cases $SWE1D-cl-25$ (top left), $SWE1D-pd-25$ (top right) and $SWE1D-mpd-25$ (bottom). Circles and squares represent respectively the reference solution and the initial prediction. For the second and third figures, the reference solution and the parareal iterations superimpose for $k = 1$.

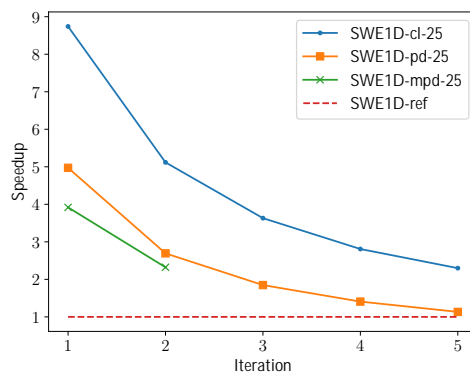


FIGURE 5.3. Evolution of the average speedup along iterations, given by the ratio between the reference and the parareal computational times, for test cases $SWE1D-cl-25$ (circles), $SWE1D-pd-25$ (squares) and $SWE1D-mpd-25$ (crosses). The dashed line represents the reference speedup, equal to 1.

condition is given by

$$\begin{aligned} h(t=0, x, y) &= h_0 + \exp\left[-\frac{(x-x_0)^2}{2\sigma_x^2} + \frac{(y-y_0)^2}{2\sigma_y^2}\right] \\ u_x(t=0, x, y) &= 0, \\ u_y(t=0, x, y) &= 0 \end{aligned} \quad (x, y) \in [0, 100]^2 \quad (5.2)$$

with $h_0 = 1$, $x_0 = y_0 = 50$ and $\sigma_x = \sigma_y = 7.5$.

All boundaries are closed (imposed null mass flux), the bottom is flat and no friction is considered. The simulation is performed from $t = 0$ to $T = 5$.

For this test case, we perform two sets of parareal simulations, respectively without and with spatial coarsening.

5.2.1. Simulations without spatial coarsening

We begin by defining a coarse propagator with $\Delta t = 0.25$ and no spatial coarsening. The tests are named *SWE2D-cl-20*, *SWE2D-pd-20* and *SWE2D-mpd-20*. For the MPD method, additional snapshots are taken at every $\Delta t = 0.125$ ($\ell = 2$).

Figure 5.4 shows the relative error for three parareal methods. We notice that, in this test case, the classical parareal method behaves better than in the previous one, but it converges slower than the POD-DEIM-based methods. We also notice that the MPD, by constructing a reduced model using twice the number of snapshots, provides a great improvement on the quality of the solution in the first iteration, for almost all the time steps.

The evolution of the speedup along iterations for each method is presented in Figure 5.5. We notice, for the three methods, a higher speedup in comparison to the test case *S1* presented above. As indicated in the speedup estimate (4.1), this may be due to the increase in the number of spatial points in the fine propagator ($M_f = 400$ in *SWE1D* and $M_f = 2500$ in *SWE2D*) and the reduction in the number of coarse time steps ($N_t = 25$ in *SWE1D* and $N_t = 20$ in *SWE2D*).

Figure 5.6 shows the evolution of the parareal solutions h given by each method at the point $(x, y) = (40, 40)$. For both the POD-DEIM-based methods, we obtain in one iteration a very good approximation to the reference solution in this point. For the classical parareal method, a good solution is obtained in two iterations. Therefore, in this test case, as shown in Figure 5.5, the classical method is able to give fast convergence, with a speedup factor of approximately 9.2, whereas the PD and the MPD have respectively a speedup of approximately 11.0 and 8.5.

5.2.2. Simulations with spatial coarsening

Using the same reference solution, i.e. the same fine propagator F_t , we perform a second set of parareal simulations for test case *SWE2D*. We set now a lower-quality initial prediction, by defining a coarse propagator G_t with a coarser Cartesian mesh ($\Delta x = \Delta y = 5$), but the same time step $\Delta t = 0.25$ as before, such that there are $N_t = 20$ coarse time steps. The tests are named *SWE2D-cl-20-c*, *SWE2D-pd-20-c* and *SWE2D-mpd-20-c*. For the MPD method, additional snapshots are taken at every $\Delta t = 0.125$ ($\ell = 2$).

The evolution of the errors e_n^k and \bar{e}^k (Figure 5.7) and the evolution of the solution in position $(x, y) = (40, 40)$ (Figure 5.9) show that both the classical and the PD method present a troubled convergence behaviour. For the classical method, the error increases in the first iteration and in the following iterations the solution approaches slowly to the reference one. For the PD method, the first iteration provides a good error decrease, but the following iterations become unstable and the error increases. The stability can be improved by reducing the subspaces' dimensions (i.e. increasing $\ell_{sv, \text{linear}}$ and $\ell_{sv, \text{nonlinear}}$), but it may strongly affect the speed of convergence of the method, as discussed with

MODIFIED PARAREAL METHOD FOR SOLVING THE SWE

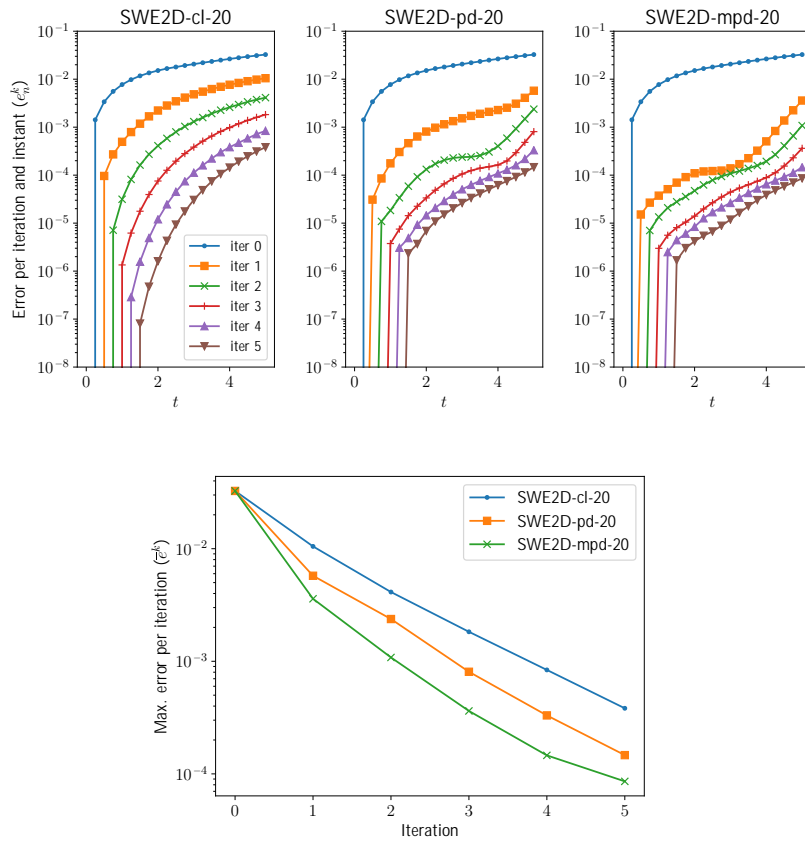


FIGURE 5.4. Evolution of the errors e_n^k along iterations and time for test cases *SWE2D-cl-20* (first plot), *SWE2D-pd-20* (second plot) and *S0-mpd-20* (third plot); and evolution of the errors e^k along iterations for the same test cases (fourth plot). The legend is the same in the first three plots.

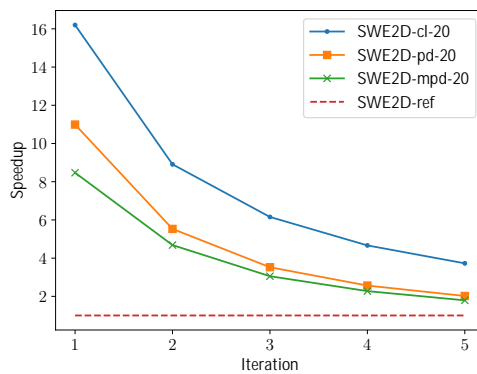


FIGURE 5.5. Evolution of the average speedup along iterations, given by the ratio between the reference and the parareal computational times, for test cases *SWE2D-cl-20* (circles), *SWE2D-pd-20* (squares) and *SWE2D-mpd-20* (crosses). The dashed line represents the reference speedup, equal to 1.

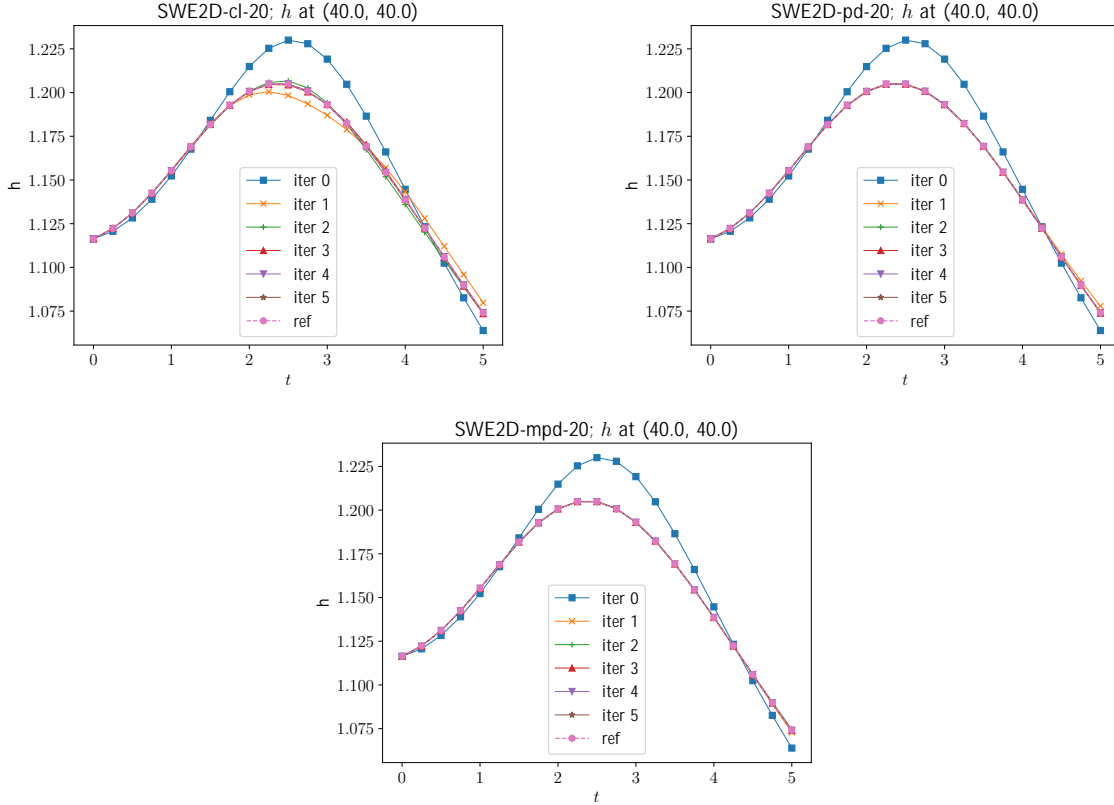


FIGURE 5.6. Evolution of h along time in position $(x, y) = (40, 40)$ for test cases $SWE2D-cl-20$ (top left), $SWE2D-pd-20$ (top right) and $SWE2D-mpd-20$ (bottom). Circles and squares represent respectively the reference solution and the initial prediction. For the second and third figures, the reference solution and the parareal iterations nearly superimpose for $k = 1$.

more details below in Section 5.2.3. The MPD provides the better approximation among the three methods. As in the PD, we observe an error increase from the first to the second iterations in more advanced times steps, but with much smaller magnitude, and the following iterations are able to control the instabilities. Concerning the speedup (Figure 5.8), we can consider for the point $(40, 40)$, in the MPD method, a good convergence in two iterations (speedup factor of approximately 4.5). For the classical method, we have good convergence in four iterations (speedup factor of approximately 4.8). We notice that the classical parareal method remains faster than the MPD, and further improvements of this last one must be done for reducing its computational cost. However, the gains in stability are a positive feature of the MPD.

5.2.3. Discussion on the influence of model reduction thresholds

The influence of the model reduction thresholds $\nu_{sv,linear}$ and $\nu_{sv,nonlinear}$ on the convergence and stability of the ROM is illustrated in Figure 5.10, showing the evolution of the relative error e_n^k in test case $SWE2D-pd-20-c$ with $\nu_{sv} = \nu_{sv,linear} = \nu_{sv,nonlinear} \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$. With larger thresholds (i.e. with lower-dimensional ROMs), the instabilities near the end of the temporal domain are less important, but the error decrease is slower in the first half of the simulation, when compared to smaller thresholds (higher-dimensional ROMs), with which, however, the unstable behaviour become

MODIFIED PARAREAL METHOD FOR SOLVING THE SWE

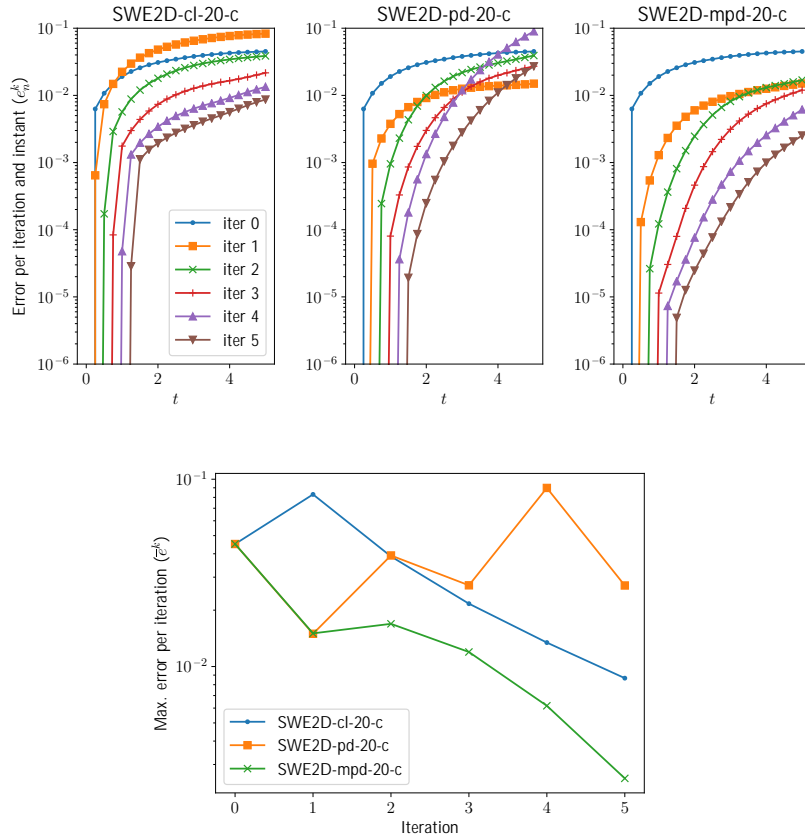


FIGURE 5.7. Evolution of the errors e_n^k along iterations and time for test cases *SWE2D-cl-20-c* (first plot), *SWE2D-pd-20-c* (second plot) and *SWE2D-mpd-20-c* (third plot); and evolution of the errors \bar{e}^k along iterations for the same test cases (fourth plot). The legend is the same in the first three plots.

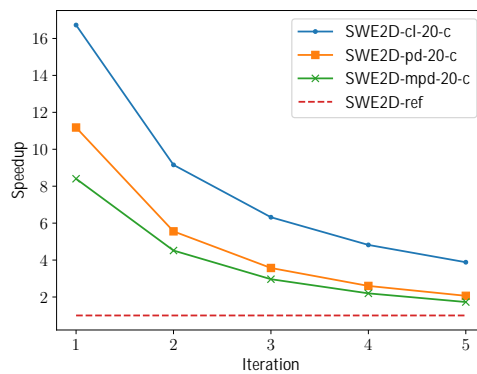


FIGURE 5.8. Evolution of the average speedup along iterations, given by the ratio between the reference and the parareal computational times, for test cases *SWE2D-cl-20-c* (circles), *SWE2D-pd-20-c* (squares) and *SWE2D-mpd-20-c* (crosses). The dashed line represents the reference speedup, equal to 1.

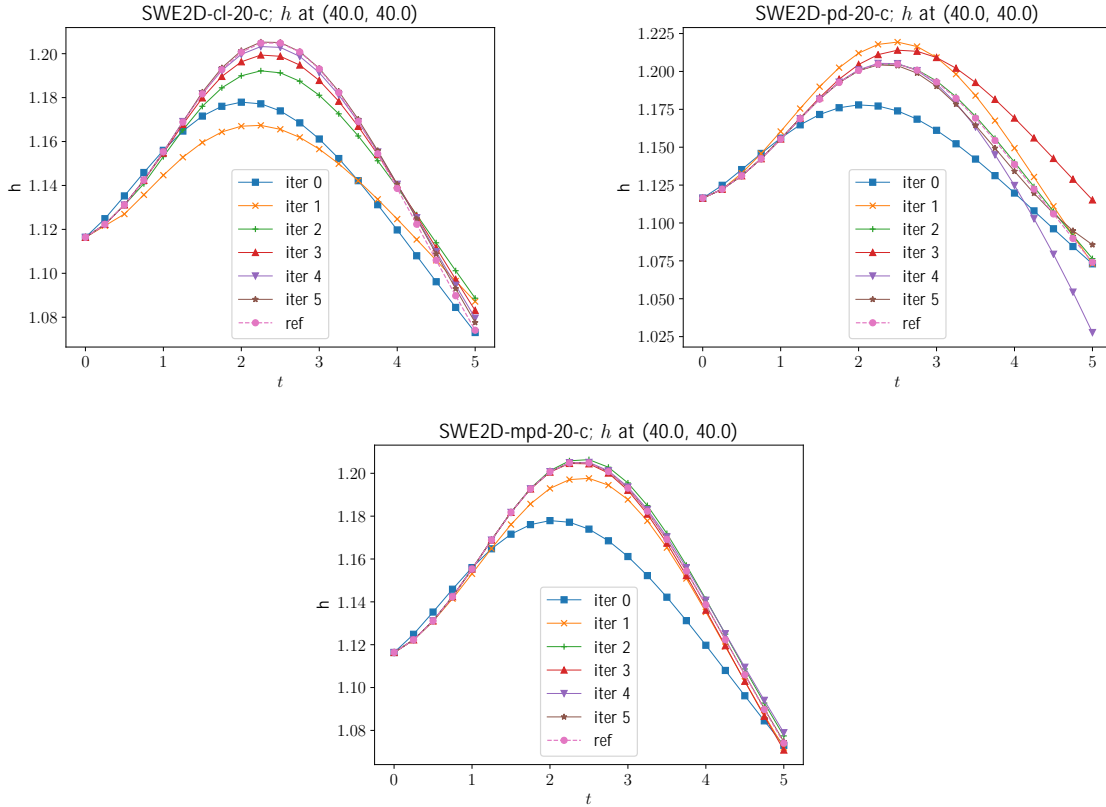


FIGURE 5.9. Evolution of h along time in position $(x, y) = (40, 40)$ for test cases *SWE2D-cl-20-c* (top left), *SWE2D-pd-20-c* (top-right) and *SWE2D-mpd-20-c* (bottom). Circles and squares represent respectively the reference solution and the initial prediction. For the third figure, the reference solution and the parareal iterations nearly superimpose for $k = 2$.

more important. As discussed in Section 3, keeping more POD modes may degrade the quality of the ROM when the snapshots are not a good representation of the problem’s dynamics, which explains the observed instabilities; however, by keeping too few POD modes, the formulated ROM has not enough information for representing the fine propagator, thus slowing down the parareal convergence. Therefore, a trade-off should be established between stability and convergence when choosing the model reduction thresholds. As proposed in this work and illustrated in Figure 5.7, an alternative approach for overcoming the instabilities is to improve the set of input snapshots. Evidently, this compromise should also be taken into account when using the MPD method, but we can expect it to be less restrictive, as illustrated in Figure 5.11, showing the error in test *SWE2D-pd-20-c* with various threshold values: with large thresholds, we still note a stable but slow-converging behaviour; with smaller ones, only small instabilities are observed in the very last time steps of the simulation, but they are effectively controlled along iterations.

5.3. Discussion on the computational time of the POD-DEIM-based parareal methods

We consider the last test case presented above (*SWE2D* with spatial coarsening) for detailing the computational cost of the POD-DEIM-based parareal method and its modification.

MODIFIED PARAREAL METHOD FOR SOLVING THE SWE

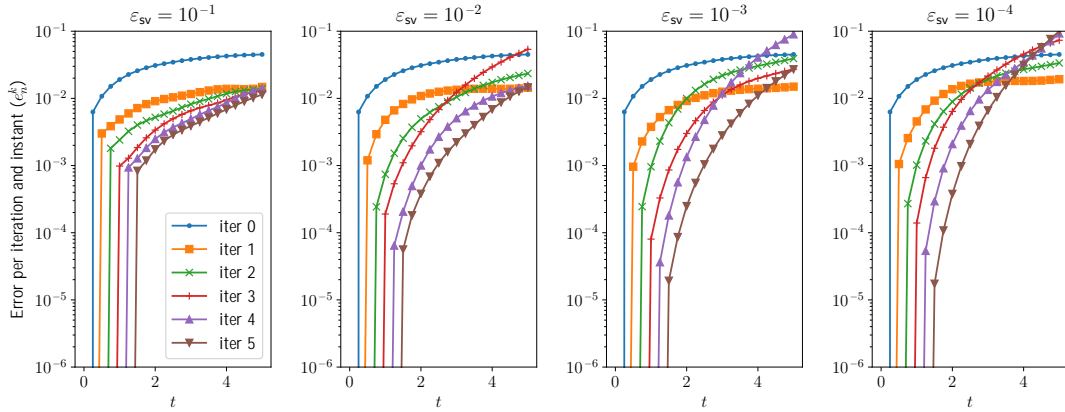


FIGURE 5.10. Evolution of the errors e_n^k along iterations and time for test case $SWE2D\text{-}pd\text{-}20\text{-}c$ with various thresholds $\varepsilon_{sv} = \varepsilon_{sv,linear} = \varepsilon_{sv,nonlinear}$ for the model reduction. The legend is the same in all plots.

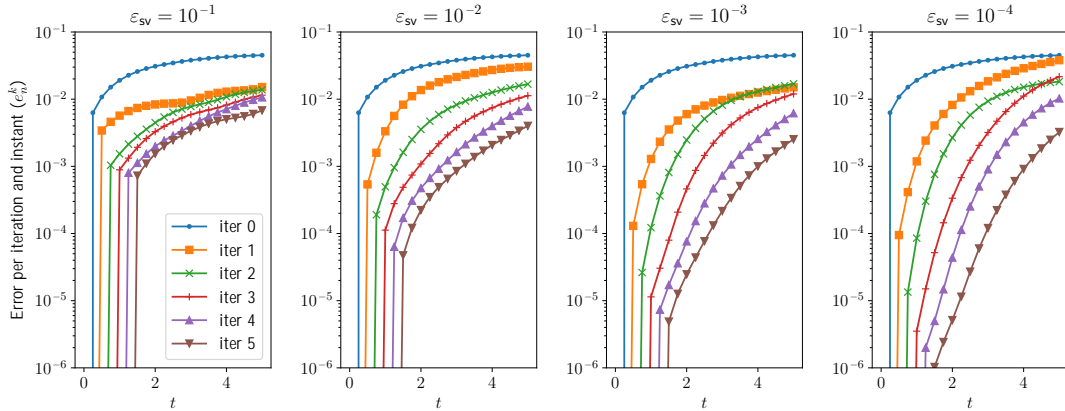


FIGURE 5.11. Evolution of the errors e_n^k along iterations and time for test case $SWE2D\text{-}mpd\text{-}20\text{-}c$ with various thresholds $\varepsilon_{sv} = \varepsilon_{sv,linear} = \varepsilon_{sv,nonlinear}$ for the model reduction. The legend is the same in all plots.

Figure 5.12 presents the average time spent in each step of PD and MPD along the parareal iterations. In these graphs, we denote:

- **FC**: time for computing the fine term of the correction (line 10 of Algorithm 2), including the storage of the snapshots;
- **SP**: time for computing the subspaces (lines 16 and 17 of Algorithm 2);
- **CC**: Time for computing the coarse term of the correction (line 20 of Algorithm 2);
- **PU**: Time for computing the predictions and updating the final solution of the iteration (line 24 of Algorithm 2), or, in the initial prediction (0-th iteration), the computation by the coarse propagator (line 4 of Algorithm 2);
- **Total**: Total time spent in the iteration.

Other iteration steps, e.g. the convergence checking and the computation of the matrices of the reduced-order model do not have their computational time shown explicitly in Figure 5.12 since they are relatively small, but are included in the total iteration time.

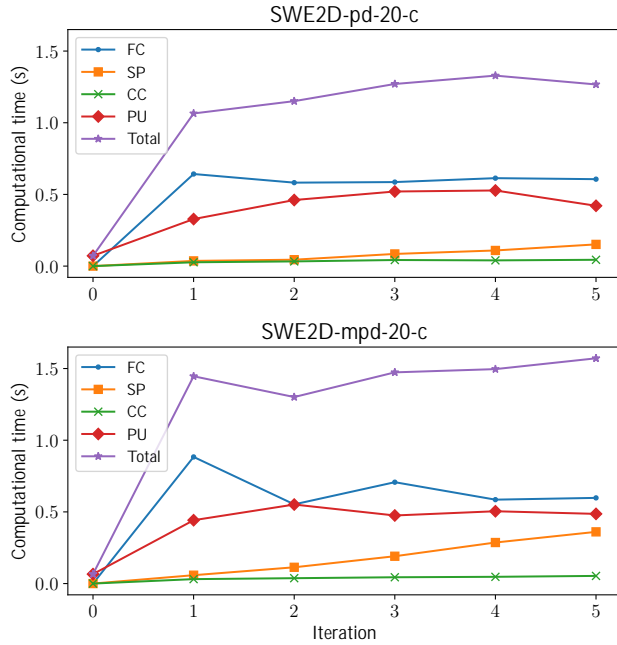


FIGURE 5.12. Evolution along iterations of the computational time spent in each step of the POD-DEIM-based parareal method (top, test case $SWE2D-pd-20-c$) and its modification (bottom, test case $SWE2D-mpd-20-c$). FC , SP , CC , PU and $Total$ stands respectively for the computation of the fine correction term, the computation of the subspaces, the computation of the coarse correction term, the computation of the prediction and update of the solution, and the total iteration time.

Several conclusions can be made from the results presented in Figure 5.12. We notice, as expected, that the 0 – th iteration, consisting uniquely of a full sequential coarse simulation, has a very small computational cost. Moreover, the computation of the coarse term correction (CC), consisting on a parallel simulation of the reduced-order model, has also a negligible computational time.

Concerning the computation of the fine correction term (FC), which is done in parallel, we notice that the spent time is similar for both POD-DEIM-based parareal methods. The only exception is the first iteration in MPD, which is a little more expensive, due to the creation of the structure to store the extra snapshots. In the following iterations, the same structure is reused, and the computational time for storing the snapshots is smaller. We recall that there is no extra cost for computing these extra snapshots, since they are provided by the parallel fine simulation.

Figure 5.12 also provides some conclusions concerning the computation of the subspaces. As discussed in Section 4, the POD procedure for computing the basis can be estimated to have a quadratic dependence on the number of snapshots, which, in its turn, is proportional to the number of iterations and the number of time steps for taking the snapshots. Therefore, its cost increases rapidly along the parareal iteration and if we keep a large number of intermediate snapshots in the MPD. Moreover, by using more snapshots as input for the POD, we expect to have a basis with higher dimensions. It affects directly the cost of the DEIM procedure, for which we estimate a dependence with exponent four on the basis dimension. The sum of these factors can be clearly observed in Figure 5.12: the SP

curve increases much faster in the MPD than the PD method. It reinforces the importance of having a fast convergence of the method, in very few iterations.

Finally, the influence of the number of snapshots on the dimension of the reduced-order model can be observed in Figure 5.13, which represents, for the same test cases *SWE2D-pd-20-c* and *SWE2D-mpd-20-c*, the evolution, along the parareal iterations, of the subspaces $S^{(l)}$, $l = 1, 2, 3$, computed from snapshots of the three components of the SWE solution, and the subspaces $\hat{S}^{(l)}$, $l = 1, 2, 3$, computed from snapshots of the three first flux terms. Concerning the two remaining fluxes, accounting for source terms, their subspaces have dimension zero since a flat bottom is considered in the simulations, so they are not represented in the graphs. In all iterations, all the subspaces computed in the MPD, using twice the number of snapshots, have a higher dimension than the subspaces computed in the PD, for the same thresholds sv,linear and sv,nonlinear . As said above, it affects the computational time for the DEIM procedure. Nevertheless, the spaces remain relatively small and we do not see remarkable differences in the computational time for solving the reduced-order models (which are included in curves *CC* and *PU* in Figure 5.12). We recall that the dimension of the FV reduced-order model is given by the dimension of the spaces $S^{(l)}$, $l = 1, 2, 3$, with fluxes computed on a number of interfaces given by the dimension of $\hat{S}^{(l)}$, $l = 1, 2, 3$. Both these quantities remain in the order of tenths, whereas the mesh for the fine propagator contains 2500 cells and 5100 interfaces.

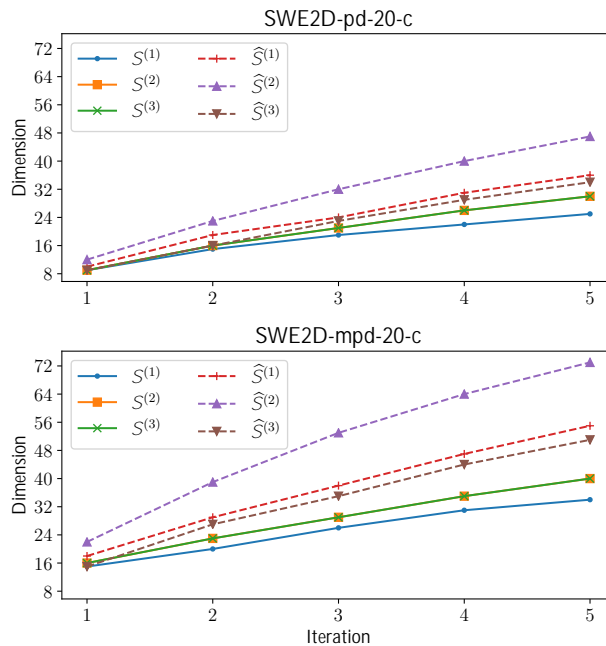


FIGURE 5.13. Evolution along iterations of the dimensions of subspaces $S^{(l)}$, $l = 1, 2, 3$ (full lines) and $\hat{S}^{(l)}$, $l = 1, 2, 3$ (dashed lines) for the POD-DEIM-based parareal method (top, test case *SWE2D-pd-20-c*) and its modification (bottom, test case *SWE2D-mpd-20-c*). Subspaces $S^{(4)}$ and $S^{(5)}$ have zero dimension in all iterations (due to the flat bottom) and are not represented. For comparison, the fine propagator's mesh has 2500 cells and 5100 interfaces.

6. Conclusion

In this paper, we implemented a POD-DEIM-based (PD) parareal method for solving the two-dimensional nonlinear shallow water equations, discretized using a finite volume scheme. The PD method, based on the Krylov-subspace-enhanced parareal method, introduces reduced-order models (ROMs) in the parareal algorithm in order to improve its stability and convergence for nonlinear hyperbolic problems. We propose a modification of the method (MPD) for improving even more both these aspects.

This modification consists in enriching the snapshots sets used for constructing the POD-DEIM reduced-order models. The motivation is the influence of the snapshot set on the quality of the ROM: if there are not enough snapshots or if they are not a good representation of the solution, the produced ROM can have a poor quality, which, in the framework of the PD parareal method, may affect its convergence and stability.

The modified POD-DEIM-based parareal method (MPD) stands out for not requiring any computational cost for obtaining the extra snapshots, since they are already computed in the fine correction step in the algorithm, but not used in the PD method. An extra cost is observed in the computation of the subspaces using the POD-DEIM procedure and can be prohibitive in too advanced parareal iterations; however, the MPD allows a convergence in very few iterations.

A preceding step for implementing the PD and the MPD is the formulation of the reduced-order model for the SWE. The nonlinearity of these equations requires a combined POD-DEIM reduction. In the context of a finite volume scheme, the obtained ROM has a dimension defined by the subspaces obtained from the POD applied to snapshots of the solution, with the nonlinear fluxes computed on interfaces chosen by the POD-DEIM applied to snapshots of the fluxes. For a test case with a reference mesh containing cells and interfaces in the order of thousands, both the ROM dimension and the number of interfaces chosen by the DEIM are in the order of tenths.

Numerical simulations were performed here for comparing the behaviour of the classical, the PD and the MPD parareal methods in terms of computational time and stability. In a first, one-dimensional test case, the classical method is unstable, whereas the PD and specially the MPD converge exactly to reference solution in very few iterations. In a second, two-dimensional test case, the classical method presents less instabilities, but the PD and the MPD converge faster. By coarsening the spatial resolution of the coarse propagator, the PD becomes unstable, suggesting that the snapshots are not a good representation of the reference solution. By using twice the number of snapshots, the MPD is able to stabilize and accelerate the convergence.

In all presented simulations, 20 parallel processors were used, and the MPD produced good approximate solutions with a good speedup factor (between 4 and 8.5 approximately, depending on the test case). However, the MPD is clearly more costly than the classical parareal method. Then, the optimization of the MPD's computational cost is a natural and necessary outlook of this work. As pointed out by [29], a task scheduling between processors as proposed in [3] is not applicable for the Krylov-subspace-enhanced parareal method (and thus for the PD and MPD methods), but similar approaches should be investigated. A coupling between spatial and temporal parallelizations, using domain decomposition and parareal techniques in an hybrid *MPI-OpenMP* implementation [20] can also be envisaged. Finally, a nested parallel computation of the subspaces, which is proposed here but not investigated, can be crucial in large problems and may be studied.

A detailed explication on the observed behaviour of the parareal methods should also be investigated. It was illustrated that the classical parareal method is highly unstable in a very simple one-dimensional test case, but behaves better in a more complex, two-dimensional problem. As discussed by [28], parareal instabilities arise in high wavenumbers due to phase speed mismatches between the propagators. Initial investigations show that the solution of the 2D test case presented here has low-magnitude high wavenumber modes compared to the 1D problem, which give some indications

on the observed behaviours. However, detailed explications should rely on diffusive and phase speed properties of the numerical discretizations considered in this work.

Concerning the applications of the work presented here, we look forward to simulations of urban floods using the SWE. In this case, using porosity-based shallow water models (e.g. the one proposed by [19]) as coarse propagator would allow to improve the quality of the initial parareal prediction and, consequently, the stability and convergence of the POD-DEIM-based parareal methods.

Acknowledgments

The authors are grateful to the OPAL infrastructure from Université Côte d’Azur and Inria Sophia Antipolis - Méditerranée “NEF” computation platform for providing resources and support.

References

- [1] N. Akkari, R. Mercier, G. Lartigue, and V. Moureau. Stable POD-Galerkin Reduced Order Models for unsteady turbulent incompressible flows. In *55th AIAA Aerospace Sciences Meeting*, 2017.
- [2] M. Astorino, F. Chouly, and A. Quarteroni. Multiscale coupling of finite element and lattice Boltzmann methods for time dependent problems. <https://hal.archives-ouvertes.fr/hal-00746942>, 2012.
- [3] E. Aubanel. Scheduling of tasks in the parareal algorithm. *Parallel Computing*, 37(3):172–182, 2011.
- [4] C. Audouze, M. Massot, and S. Volz. Symplectic multi-time step parareal algorithms applied to molecular dynamics. <https://hal.archives-ouvertes.fr/hal-00358459>, submitted to SIAM Journal of Scientific Computing, 2009.
- [5] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An ‘empirical interpolation’ method: application to efficient reduced-basis discretization of partial differential equations. *C. R. Math. Acad. Sci. Paris*, 339(9):667–672, 2004.
- [6] J. G. Caldas Steinstraesser. *Coupling large and small scale shallow water models with porosity in the presence of anisotropy*. PhD thesis, Université de Montpellier, 2021.
- [7] S. Chaturantabut and D. C. Sorensen. Nonlinear Model Reduction via Discrete Empirical Interpolation. *SIAM J. Sci. Comput.*, 32(5):2737–2764, 2010.
- [8] F. Chen, J. S. Hesthaven, and X. Zhu. On the Use of Reduced Basis Methods to Accelerate and Stabilize the Parareal Method. In *Reduced Order Methods for Modeling and Computational Reduction*, pages 187–214. Springer, 2014.
- [9] W. Chen, J. S. Hesthaven, B. Junqiang, Y. Qiu, Y. Tihao, and Z. Yang. Greedy Non-Intrusive Reduced Order Model for Fluid Dynamics. *AIAA J.*, 56:12, 2018.
- [10] X. Dai and Y. Maday. Stable Parareal in Time Method for First- and Second-Order Hyperbolic Systems. *SIAM J. Sci. Comput.*, 35(1):A52–A78, 2013.
- [11] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, and I. Yamazaki. The Singular Value Decomposition: Anatomy of Optimizing an Algorithm for Extreme Scale. *SIAM Rev.*, 60(4):808–865, 2018.
- [12] M. Duarte, M. Massot, and S. Descombes. Parareal operator splitting techniques for multi-scale reaction waves: Numerical analysis and strategies. *ESAIM, Math. Model. Numer. Anal.*, 45(5):825–852, 2011.
- [13] A. Eghbal, A. G. Gerber, and E. Aubanel. Acceleration of unsteady hydrodynamic simulations using the parareal algorithm. *J. Comput. Sci.*, 19:57–76, 2017.
- [14] C. Farhat, J. Cortial, C. Dastillung, and H. Bavestrello. Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses. *Int. J. Numer. Methods Eng.*, 67:697–724, 2006.

- [15] P. F. Fischer, F. Hecht, and Y. Maday. A Parareal in Time Semi-implicit Approximation of the Navier-Stokes Equations. In *Domain Decomposition Methods in Science and Engineering*, pages 433–440. Springer, 2005.
- [16] M. J. Gander. Analysis of the Parareal Algorithm Applied to Hyperbolic Problems Using Characteristics. *Bol. Soc. Esp. Mat. Apl.*, 42:21–35, 2008. ID: unige:6268.
- [17] M. J. Gander. 50 Years of Time Parallel Time Integration. In *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113. Springer, 2015.
- [18] M. J. Gander and M. Petcu. Analysis of a Krylov subspace enhanced parareal algorithm for linear problems. *ESAIM, Proc.*, 25:114–129, 2008.
- [19] V. Guinot and S. Soares-Frazão. Flux and source term discretization in two-dimensional shallow water models with porosity on unstructured grids. *Int. J. Numer. Methods Fluids*, 50(3):309–345, 2006.
- [20] R. Haynes. Mpi–OpenMP algorithms for the parallel space-time solution of time dependent PDEs. In *Domain decomposition methods in science and engineering XXI*, volume 98 of *Lecture Notes Computational Sciences and Engineering*, pages 179–187. Springer, 2014.
- [21] M. Iizuka and K. Ono. Influence of the phase accuracy of the coarse solver calculation on the convergence of the parareal method iteration for hyperbolic PDEs. *Comput. Vis. Sci.*, 19(3-4):97–108, 2018.
- [22] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d’EDP par un schéma en temps ‘pararéel’. *C. R. Math. Acad. Sci. Paris*, 332(7):661–668, 2001.
- [23] Yvon Maday. The ‘Parareal in Time’ Algorithm. In *Substructuring Techniques and Domain Decomposition Methods*, Computational science, engineering and technology series, pages 19–44. Saxe-Coburg Publications, 2010.
- [24] M. Müller. *On the POD method: an abstract investigation with applications to reduced-order modeling and suboptimal control*. PhD thesis, Georg-August-Universität, 2008.
- [25] B. W. Ong and J. B. Schroder. Applications of time parallelization. *Comput. Vis. Sci.*, 23(1-4), 2020.
- [26] G. Pagès, O. Pironneau, and G. Sall. The parareal algorithm for American options. *C. R. Math. Acad. Sci. Paris*, 354(11):1132–1138, 2016.
- [27] C. W. Rowley. Model reduction for fluids, using balanced proper orthogonal decomposition. *Int. J. Bifurcation Chaos Appl. Sci. Eng.*, 15(3):997–1013, 2005.
- [28] D. Ruprecht. Wave propagation characteristics of Parareal. *Comput. Vis. Sci.*, 19:1–17, 2018.
- [29] D. Ruprecht and R. Krause. Explicit parallel-in-time integration of a linear acoustic-advection system. *Computers Fluids*, 59:72–83, 2012.
- [30] H. Samuel. Time domain parallelization for computational geodynamics. *Geochemistry, Geophysics, Geosystems*, 13(1), 2012.
- [31] R. Ștefănescu and I. M. Navon. Pod/DEIM nonlinear model order reduction of an ADI implicit shallow water equations model. *J. Comput. Phys.*, 237:95–114, 2013.
- [32] J. M. F. da Trindade and J. F. Pereira. Parallel-in-time simulation of the unsteady Navier–Stokes equations for incompressible flow. *Int. J. Numer. Methods Fluids*, 45(10):1123–1136, 2004.