THÈSE POUR OBTENIR LE GRADE DE DOCTEUR
DE L'UNIVERSITE DE MONTPELLIER

En Mathématiques et Modélisation

École doctorale : Information, Structures, Systèmes

Unité de recherche Inria - Équipe LEMON / IMAG

Coupling large and small scale shallow water models with porosity in the presence of anisotropy

Présentée par João Guilherme CALDAS STEINSTRAESSER
Le 1$^{\text{er}}$ octobre 2021

Sous la direction de Vincent GUINOT et Antoine ROUSSEAU

Devant le jury composé de

Laurent DEBREU, Directeur de recherche, Inria	Examinateur
Cristián ESCAURIAZA, Professeur associé, Pontifícia Universidad Católica de Chile	Rapporteur
Martin GANDER, Professeur, Université de Genève	Rapporteur
Vincent GUINOT, Professeur, Université de Montpellier	Co-directeur de thèse
Olga MULA, Maître de conférences, Université Paris-Dauphine	Examinatrice
Antoine ROUSSEAU, Chargé de recherche, Inria	Co-directeur de thèse
Sandra SOARES-FRAZÃO, Professeure, Université Catholique de Louvain	Présidente du jury

UNIVERSITÉ
DE MONTPELLIER

MERCI ! ¡GRACIAS! OBRIGADO!

Au moment où j'écris ces lignes, cela fait à peu près sept ans que j'ai atterri en France pour mes études, dont les trois dernières années à Montpellier dans le cadre de cette thèse. Une période assez mouvementée, avec beaucoup d' allers et retours entre le Brésil, le Chili et la France, mais (ou donc) très enrichissante. Je voudrais exprimer ma reconnaissance à tous ceux qui ont rendu possibles ces années d'études et qui m'ont permis de travailler dans le monde de la recherche scientifique, me motivant et m'inspirant à poursuivre ce métier.

Tout d'abord, un grand merci à mes directeurs de thèse. Antoine, qui me guide depuis des années (plus de cinq ans déjà ?) dans ces aventures académiques, en France et au Chili, et qui m'a beaucoup appris sur la science, le métier de chercheur, la course à pied et plein d'autres choses. Et Vincent, qui a toujours apporté de la rigueur et des bonnes idées à mes travaux de thèse, et qui m'a introduit à des sujets et modèles permettant de réaliser cette recherche. Merci à tous les deux pour tout ce travail ensemble, votre disponibilité et vos enseignements. Je vous suis très reconnaissant de m'avoir donné l'occasion de faire ce travail.

Je remercie également Cristián Escauriaza et Martin Gander d'avoir accepté d'être rapporteurs de cette thèse, et les membres de mon jury, Laurent Debreu, Olga Mula et Sandra Soares-Frazão.

Merci à tous les membres de l'équipe LEMON, au sein de laquelle cette thèse s'est déroulée, qui m'ont permis de connaître le travail dans une équipe de recherche, autour de réunions, sessions de codage, déjeuners et cafés, toujours avec de la bonne humeur : Annie, Carole, Gwladys, Pascal, Cécile, Vita, Joseph, Fátima, José, Yassine. Merci également à Inria, qui m'a donné les moyens de m'initier dans la recherche scientifique, non seulement dans ma thèse, mais aussi dans des stages réalisés auparavant, à Bordeaux dans l'équipe CARDAMOM et au Chili avec MERIC. Merci à tous mes encadrants et collègues de travail dans ces années.

Merci à Louis Emerald, Martin Parisot, Emmanuel Audusse, Alexandre Paris et Philippe Heinrich pour le travail enrichissant au CEMRACS 2019 et qui m'ont permis de m'immerger dans des sujets de recherche en dehors du cadre de ma thèse.

J'exprime mes très profonds remerciements et ma reconnaissance aux professeurs du département de Mécanique de la Faculté des Sciences de l'Université de Montpellier, qui m'ont accueilli en tant qu'enseignant dans ces deux dernières années. Et merci surtout à tous mes étudiants en TD et TP de la Fac des Sciences et de Polytech, qui étaient toujours présents et motivés (même sous des conditions pas très faciles, avec des cours en distanciel et un enseignant avec un accent assez curieux) et qui m'ont permis d'exercer ma plus grande passion, celle que me motive le plus dans le monde académique.

Un merci très spécial à Cécile pour son amitié, les déjeuners au soleil, les sorties à la plage ou à la piscine, les discussions à distance pendant les longues semaines de confinement et pour tous ses enseignements et exemples me permettant de faire des petits mais importants changements dans mon quotidien.

En faisant un détour vers l'autre côté de l'Atlantique, muchísimas gracias también a mis amigos de Chile. Aunque no los haya visto mucho en estos tres últimos años, fueron muy importantes en mi camino hacia aquí. Tengo todavía muy buenos recuerdos de todos los momentos de risa, fútbol y frío en las montañas cerca de Santiago. Gracias a Felipe, José, Angeline, Pancho, Salo, Cristóbal, Tristan, para mencionar algunos de los mejores weones que tuve el placer de conocer en Chile.

Meus agradecimentos a todos que vieram comigo estudar na França e que são meus amigos e minha família aqui. Aos que ainda estão em Paris mas não esqueceram do cara que resolveu morar no sul e sempre que possível me chamaram pra viajar, pegar uma praia, fazer uma randonnée, comer uma boa comida (parabéns Ric!), ou então passar horas em uma boa partida de age (sdds age): PV, Ric, Gabi, Baldi, Panda, Mari, Chico, André, Nathi, Ricardo. E também aos que já voltaram pro Brasil mas de quem guardo muitas boas lembranças: Marília, Renata, Allan, Edu, Caio, Mota e muitos outros. E, claro, ao Bola (Boleta para os íntimos), companheiro de muitas pedaladas, centenas (milhares?) de quilômetros ao longo do Canal du Midi ou subindo o Mont Aigoual, boa música e observação de estrelas.

Muito obrigado ao Vinícius, um grande amigo cuja companhia fez a vida em Montpellier ser muito mais fácil e agradável, entre almoços, idas à praia e voltas de bicicleta.

À Babi e sua família e ao Gabriel, grandes amigos de vários e vários anos, sempre presentes e interessados no meu doutorado e em saber como andam as coisas.

A todos os professores que me guiaram e me permitiram chegar até aqui. Em especial ao Saulo, que me inspirou a seguir na área de matemática aplicada e me guiou nos meus primeiros passos no mundo da pesquisa acadêmica.

Um obrigado muito especial a todos os alunos que tive em todos esses anos, que tanto me ensinaram e que me dão alegria e motivação pra seguir nessa jornada.

Às companhias inesperadas em tempos não muito fáceis: Flávio & seus amigos, Cláudia, Clodoaldo e Carlos, Anderson, Jeremias, Pedro, Milton e tantos outros.

À minha família por sempre me apoiar. Ao meu pai, sempre preocupado, atencioso e disposto a ajudar. As minhas irmãs e irmão, Gabi, Mari, Daniel. À Gabi por sempre dar bons conselhos e por ser uma inspiração por sua trajetória acadêmica e profissional. E obrigado também à família que me acolheu e me aguentou por bastante tempo, Victor, Marina, Márcia, Laerte, Malu, Nicole e Amarula.

E muito obrigado à Katia por todo o apoio e por sempre estar presente, apesar da distância.

The author is grateful to the OPAL infrastructure from Université Côte d'Azur and the NEF computing cluster platform from Inria Sophia Antipolis-Méditerranée center for providing resources and support.

ABSTRACT

This PhD thesis aims to study the coupling of nonlinear shallow water models at different scales, with application to the numerical simulation of urban floods. Accurate simulations in this domain are usually prohibitively expensive due to the small mesh sizes necessary for the spatial discretization of the urban geometry and the associated small time steps constrained by stability conditions. Porosity-based shallow water models have been proposed in the past two decades as an alternative approach, consisting of upscaled models using larger mesh sizes and time steps and being able to provide good global approximations for the solution of the fine shallow water equations, with much smaller computational times. However, small-scale phenomena are not captured by this type of model. Therefore, we seek to formulate a numerical model coupling the fine and upscaled ones, in order to obtain more accurate solutions inside the urban zone, always with reduced computational costs relatively to the simulation of the fine model. The guideline for this objective lays on the use of predictor-corrector iterative parallel-in-time numerical methods, which naturally fit to this fine/coarse formulation. We focus on the parareal, one of the most popular parallel-in-time methods. As a main challenge, temporal parallelization suffers from instabilities and/or slow convergence when applied to hyperbolic or advection-dominated problems, such as the shallow water equations. Therefore, we consider a variant of the method using reduced-order models (ROMs) formulated on-the-fly along parareal iterations, using Proper Orthogonal Decomposition (POD) and the Empirical Interpolation Method (EIM), being able to improve the stability and convergence of the parareal method for solving nonlinear hyperbolic problems. We investigate the limitations of this ROM-based parareal method and we propose a number of modifications that provide further stability and convergence improvements: enrichment of the input snapshot sets used for the model reduction procedure; formulation of local-in-time ROMs; and incorporation of an adaptive parareal approach recently presented in the literature. The original and ROM-based parareal methods, including the proposed improvements, are compared and evaluated in terms of stability, convergence towards the fine solution and numerical speedup obtained in a parallel implementation. In a first part, the methods are formulated, studied and implemented considering a set of numerical simulations coupling the classical shallow water equations (without the porosity concept) at different scales. After this initial study, we implement them for coupling the classical and the porosity-based shallow water models, for the simulation of urban floods.

Keywords: porosity-based shallow water equations, multiscale models, parallel-in-time, parareal method, reduced-order models, urban floods

RÉSUMÉ

Cette thèse de doctorat porte sur l'étude du couplage des modèles de Saint-Venant non linéaires à différentes échelles, appliqué à la simulation numérique d'inondations urbaines. Des simulations précises dans ce domaine ont, en général, un coût computationnel prohibitif dû aux petites tailles de maille nécessaires pour la discrétisation spatiale de la géométrie urbaine et, par conséquent, les petits pas de temps restreints par des conditions de stabilité. Au cours des deux dernières décennies, des modèles de Saint-Venant à porosité ont été proposés comme une alternative, s'agissant des modèles à échelle élargie utilisant des mailles et des pas de temps plus grands, et capables de fournir des bonnes approximations globales aux solutions des modèles fins, avec un temps de calcul considérablement plus petit. Néanmoins, certains phénomènes à petite échelle ne sont pas capturés par ce type de modèle. Nous cherchons donc à formuler un modèle numérique couplant les modèles à petite et à large échelle, afin d'obtenir des solutions plus précises à l'intérieur des zones urbaines, toujours avec des temps de calcul plus petits par rapport à la simulation des modèles fins. La ligne directrice de ce travail est l'utilisation de méthodes itératives de parallélisation en temps, du type prédicteur-correcteur, qui s'adaptent naturellement à cette formulation fin/grossier. Nous nous concentrons sur la méthode Parareel (*parareal method*), une des méthodes de parallélisation en temps les plus connues. Comme défi majeur, la parallélisation en temps présente en général des instabilités et une convergence lente dans le cadre de la résolution de problèmes hyperboliques ou dominés par l'advection, comme les équations de Saint-Venant. Nous considérons donc une variante de la méthode qui incorpore des modèles d'ordre réduit (*Reduced-Order models* - ROMs) formulés à la volée au cours des itérations de la méthode Parareel, utilisant la decomposition orthogonale aux valeurs propres (*Proper Orthogonal Decomposition* - POD) et la méthode d'interpolation empirique (*Empirical Interpolation Method* - EIM), et capable d'améliorer la stabilité et la convergence de la méthode pour la résolution de problèmes hyperboliques non linéaires. Nous étudions les limitations de cette méthode Parareel basée sur des ROMs et nous proposons des modifications qui fournissent des améliorations additionnelles à la stabilité et la convergence : l'enrichissement des données d'entrée pour les techniques de réduction de modèle ; la formulation des ROMs locaux en temps ; et l'incorporation d'une méthode Parareel adaptative proposée récemment dans la littérature. La méthode Parareel originale et celle basée sur des modèles réduits, y compris les modifications proposées, sont comparées et évaluées en termes de stabilité, convergence vers la solution du modèle fin et accélération de la simulation numérique obtenue dans une implémentation parallèle. Dans un premier temps, les méthodes et améliorations proposées sont formulées, étudiées et implémentées en considérant des simulations numériques couplant les équations de Saint-Venant classiques (sans le concept de porosité) à différentes échelles. Après cette étude initiale, nous appliquons les méthodes au couplage des équations classiques et des équations à porosité, pour la simulation d'inondations urbaines.

Mots-clés: équations de Saint-Venant à porosité, modèles multi-échelles, parallélisation en temps, méthode Parareel, modèles d'ordre réduit, innondations urbaines

CONTENTS

Résumé détaillé en français **xv**

1 Introduction **1**

1.1 Context and challenges of urban flood modelling . . . 1

1.1.1 Urban floods: an increasing issue . . . 1

1.1.2 Mathematical and numerical modelling as components of urban flood management 2

1.1.3 Porosity-based shallow water models . . . 3

1.2 Parallelization-in-time . . . 3

1.2.1 A further direction for parallelization . . . 3

1.2.2 Classification and examples of parallel-in-time methods . . . 5

1.2.3 Challenges and alternatives for hyperbolic problems . . . 6

1.3 Objectives and methodology . . . 7

1.4 The SW2D-LEMON software . . . 7

1.5 Publications and conferences . . . 8

1.6 Summary of the thesis . . . 9

2 Shallow water models at different scales for the simulation of urban floods **11**

2.1 Introduction . . . 11

2.1.1 Notation . . . 12

2.2 The classical shallow water equations (SWE) . . . 13

2.3 An overview on model upscaling . . . 14

2.4 The Single Porosity (SP) model . . . 15

2.4.1 The porosity parameter . . . 15

2.4.2 Derivation of the SP shallow water equations . . . 16

2.4.3 Final formulation . . . 18

2.5 The Multiple Porosity (MP) model . . . 18

2.6 The Integral Porosity (IP) model . . . 19

2.7 The Dual Integral Porosity (DIP) model . . . 21

2.8 Other developments and porosity models . . . 22

2.9 Hyperbolicity of the shallow water models . . . 23

2.9.1 Classical SWE . . . 23

2.9.2 SP model . . . 24

2.9.3 IP model . . . 24

2.9.4 DIP model . . . 24

2.10 Finite volumes discretization . . . 25

2.10.1 Notations . . . 25

2.10.2 Discretization . . . 27

2.10.3 Global-in-space matrix formulation of the finite volume scheme . . . 28

2.11 Numerical examples . . . 31

2.12 Conclusion of the chapter . . . 34

3 The parareal method and some adaptations for hyperbolic problems 37

3.1 Introduction . . . 38

3.1.1 Basic notation and definitions . . . 39

3.2 The original parareal method . . . 40

3.2.1 Presentation of the algorithm . . . 40

3.2.2 Speedup estimation . . . 42

3.2.3 Exact convergence in a finite number iterations . . . 44

3.2.4 Performance of the classical parareal method . . . 45

3.2.5 Other interpretations of the parareal method . . . 47

3.3 The Krylov subspace-enhanced parareal method for linear hyperbolic problems . . . 48

3.4 The ROM-based parareal method for nonlinear hyperbolic problems . . . 49

3.4.1 The Proper Orthogonal Decomposition (POD): reduction of the linear term . . . 51

3.4.2 The empirical interpolation method (EIM): reduction of the nonlinear term . . . 52

3.4.3 Introduction of ROMs in the parareal method . . . 55

3.4.4 Speedup estimation . . . 58

3.5 Application of the ROM-based parareal method to the SWE . . . 61

3.5.1 Formulation of the POD-EIM-based reduced SWE . . . 61

3.5.2 The ROM-based parareal method for solving the SWE . . . 63

3.6 Numerical examples . . . 63

3.6.1 Definition of the test cases . . . 64

3.6.2 Simulation using the classical parareal method . . . 66

3.6.3 Simulation using the ROM-based parareal method . . . 67

3.6.4 Some indications for understanding the parareal performance . . . 75

3.6.5 A first discussion on the influence of the length time slices . . . 79

3.6.6 Influence of nonlinearities . . . 80

3.6.7 Influence of the type of computational mesh . . . 83

3.6.8 Influence of the interpolation procedure . . . 84

3.6.9 Some conclusions on the numerical simulations . . . 87

3.7 Conclusion of the chapter . . . 89

4 Improvements of the ROM-based parareal method 91

4.1 Introduction . . . 91

4.2 Discussion on the quality of the reduced models . . . 92

4.3 Enrichment of the snapshots sets . . . 95

4.3.1 Proposed modification . . . 95

4.3.2 Speedup estimation . . . 96

4.3.3 Numerical examples . . . 96

4.4 A principal interval decomposition (PID) approach . . . 104

4.4.1 The principal interval decomposition and extension to the POD-EIM . . . 104

4.4.2 A PID-ROM-based parareal method . . . 105

4.4.3 Speedup estimation . . . 105

4.4.4 Numerical examples . . . 107

4.5 An adaptive approach . . . 110

4.5.1 An adaptive ROM-based parareal method . . . 111

4.5.2 Speedup estimation . . . 113

4.5.3 Numerical examples . . . 116

4.6 Combination of the proposed improvements . . . 121

4.7 Conclusion of the chapter . . . 123

5 Strategies for longer parareal simulations 129

5.1 Introduction . . . 129

5.2 Motivating numerical examples . . . 130

5.3 Choice of the length of time slices . . . 132

5.4 Local-in-time parareal . . . 136

5.4.1 Proposed approach . . . 136

5.4.2 Speedup estimation . . . 137

5.4.3 Numerical examples . . . 139

5.5 Conclusion of the chapter . . . 144

6 Coupling between the classical and the porosity-based shallow water equations 147
6.1 Introduction . . . 147
6.2 A simple and challenging example . . . 148
6.3 Description of a base test case . . . 151
6.3.1 Definitions . . . 151
6.3.2 Comparison between the models . . . 154
6.4 Influence of the coarsening between the classical and the porosity-based SWE . . . 157
6.4.1 Study for a fixed coarse propagator . . . 157
6.4.2 Study for a fixed fine, reference propagator . . . 159
6.4.3 Some general conclusions . . . 161
6.5 Setting parareal configurations for improving its performance . . . 161
6.5.1 Comparison between configurations of the ROM-based parareal method . . . 162
6.5.2 Comparison between the classical and ROM-based parareal methods . . . 163
6.5.3 Combining the classical and ROM-based parareal methods . . . 165
6.5.4 Some comments on the use of the adaptive approach . . . 169
6.6 Towards more challenging examples . . . 172
6.6.1 Initial water depth $h_0 = 0.5$ . . . 173
6.6.2 Initial water depth $h_0 = 0.1$ . . . 182
6.6.3 Initial water depth $h_0 = 0$ . . . 183
6.7 Conclusion of the chapter . . . 184

7 Conclusion and perspectives 187
7.1 On the performance of the ROM-based parareal method . . . 187
7.1.1 Synthesis and conclusions . . . 187
7.1.2 Perspectives . . . 189
7.2 On the performance of adaptive parareal methods . . . 190
7.2.1 Synthesis and conclusions . . . 190
7.2.2 Perspectives . . . 190
7.3 On the temporal parallelization of urban floods simulations . . . 190
7.3.1 Summary and conclusions . . . 190
7.3.2 Perspectives . . . 191

A Some details on the parallel implementation and execution 195
A.1 Parallelization using OpenMP . . . 195
A.2 Software execution . . . 195
A.3 Measurement of computational times . . . 196

B Implementation of the POD-EIM model reduction 197
B.1 Implementation using MKL linear algebra functions . . . 197
B.2 Model reduction workflow . . . 198
B.3 Complexity of the POD-EIM model reduction procedures . . . 199
B.3.1 Complexity of the POD-EIM procedure . . . 199
B.3.2 Complexity of the computation of ROM matrices . . . 200

C Spatial interpolation procedures 201
C.1 Delaunay triangulation . . . 201
C.2 Linear interpolation . . . 202
C.3 Cubic interpolation . . . 203

Bibliography 206

RÉSUMÉ DÉTAILLÉ EN FRANÇAIS

Chapitre 1: Introduction

Dans cette thèse de doctorat, on s'intéresse au couplage de modèles de Saint-Venant (en anglais, "Shallow Water Equations" - SWE) à différentes échelles pour la simulation d'inondations urbaines. En général, ce type de simulation requiert l'utilisation de maillages fins pour la représentation de la géométrie urbaine, avec des cellules de l'ordre du mètre, ce qui implique que les pas de temps doivent être également petits, à cause des conditions de stabilité du type CFL. Par conséquent, les temps de calcul sont élevés et impraticables pour des simulations en temps réel à l'échelle des villes. Une approche alternative consiste à utiliser des modèles à échelle élargie, appelés modèles de Saint-Venant à porosité, dans lesquels des coefficients de porosité sont définis pour chaque cellule et interface du maillage afin d'indiquer la présence ou non d'obstacles (bâtiments, etc.). Cela permet d'utiliser des mailles beaucoup plus grandes (et aussi des pas de temps plus grands, à cause des conditions de stabilité), conduisant à des temps de calcul considérablement plus petits (entre deux et trois ordres de grandeur moins élevés).

Bien que les modèles de Saint-Venant à porosité soient capables de fournir des bonnes approximations globales des solutions du modèle fin (les équations de Saint-Venant classiques, résolues avec une discrétisation spatio-temporelle fine), certains phénomènes à petite échelle ne sont pas bien représentés. On cherche donc à coupler ces deux modèles afin de garder les meilleures caractéristiques de chacun : la précision du modèle fin et le coût computationnel réduit du modèle grossier.

Pour faire ce couplage, on utilise des méthodes itératives de parallélisation en temps ("parallel-in-time" - PinT) du type prédicteur-correcteur, qui sont propices à cette formulation fin-grossier. Plus précisément, on s'intéresse à la méthode "Pararéel", une des méthodes PinT les plus populaires. Néanmoins, il est bien connu que les méthodes PinT, le Pararéel compris, sont peu efficaces pour la résolution des problèmes hyperboliques (comme les équations de shallow water). Nous explorons donc l'utilisation d'une variante de la méthode Pararéel utilisant des modèles d'ordre réduit et conçue pour améliorer la performance lors de la résolution des problèmes hyperboliques. Nous proposons des améliorations à la méthode et comparons sa performance et celle de la méthode Pararéel originale (ou classique) pour résoudre les équations de Saint-Venant et le couplage proposé dans ce travail.

Chapitre 2 - Les modèles de Saint-Venant à différentes échelles

Ce chapitre est consacré à une revue des modèles de Saint-Venant à différentes échelles. Nous présentons d'abord les équations classiques et, après une revue du concept de passage à large échelle ("upscaling"), nous présentons différents modèles à porosité qui ont été développés au cours des deux dernières décennies et qui sont obtenus en moyennant en espace les équations classiques.

Les modèles présentés en détail sont :

- Le modèle à **porosité unique (Single Porosity - SP)** (Guinot and Soares-Frazão, 2006), où les équations sont modifiées en introduisant un seul paramètre de porosité, représentant la fraction de l'aire d'un domaine ou d'une maille qui n'est pas couverte par des obstacles (donc disponible pour le fluide)
- Le modèle à **porosité multiple (Multiple Porosity - MP)** (Guinot, 2012), où le domaine est

divisé en régions où l'écoulement a différentes propriétés (*e.g.* des zones d'eau mobile, d'eau stagnante, des intersections de rues dans la géométrie urbaine, etc.). Dans chaque région, l'écoulement est décrit par des équations et des coefficients de porosité spécifiques.

- Le modèle à **porosité intégrale (Integral Porosity - IP)** (Sanders et al., 2008) : contrairement aux deux modèles précédents, ce modèle est proposé dans une formulation intégrale, ce qui permet de définir deux coefficients de porosité : un coefficient de stockage, représentant la fraction d'une aire qui n'est pas couverte par des obstacles et défini pour chaque maille (analogue au coefficient de porosité dans le modèle SP) et un coefficient de passage, défini pour chaque interface du maillage et représentant la fraction de l'interface disponible pour le fluide. Avec cette formulation, le modèle est capable de mieux représenter des phénomènes anisotropes que dans le modèle SP.

- Le modèle à **porosité intégrale duale (Dual Integral Porosity - DIP)** (Guinot et al., 2017) : ce modèle améliore le modèle IP en définissant deux ensembles de variables conservées, une pour les mailles et l'autre pour les interfaces du maillage. Un modèle de fermeture est proposé pour lier ces deux ensembles. Cette approche permet de mieux représenter les vitesses d'onde. En outre, des conditions sur les porosités de stockage et de passage sont définies pour assurer que le modèle reste hyperbolique et valable physiquement.

Après cette présentation des modèles, on décrit un schéma en volumes finis (FV) explicite en temps pour leur résolution numérique. Ce schéma est valable pour les équations de Saint-Venant classiques et à porosité. On propose notamment une reformulation globale en espace du schéma, ce qui sera utile pour la formulation des modèles réduits dans le chapitre suivant.

On conclut le chapitre avec l'application des équations de Saint-Venant classiques et des modèles SP et DIP pour la résolution d'un cas test simple, illustrant un écoulement dans une zone urbaine fictive, comme motivation pour la suite du travail. On constate que les modèles à porosité sont beaucoup moins coûteux (avec un temps de calcul cent fois plus petit par rapport aux équations classiques), mais ne fournissent qu'une approximation globale de la solution fine. En comparant les résultats des modèles SP et DIP, on constate aussi que le dernier fournit des meilleures approximations.

Chapitre 3 - La méthode Pararéel et quelques adaptations pour des problèmes hyperboliques

Dans ce chapitre, on présente les méthodes de parallélisation en temps qui seront utilisées pour le couplage proposé dans ce travail.

On présente d'abord la méthode Pararéel classique (Lions et al., 2001). Il s'agit d'une méthode itérative, du type prédicteur-correcteur, s'appuyant sur l'utilisation simultanée d'un modèle (propagateur) fin et un modèle (propagateur) grossier. Le premier, qu'on considère comme modèle de référence, est précis et avec un coût computationnel très élevé (en général, par l'utilisation d'un petit pas de temps), tandis que le deuxième est beaucoup moins cher mais moins précis (en utilisant par exemple des pas de temps plus grands). La méthode se construit en divisant le domaine temporel en un certain nombre d'intervalles. À chaque itération, des prédictions sont calculées en utilisant le modèle grossier de façon séquentielle au long de tout le domaine temporel, et des corrections sont fournies par le modèle fin calculé de façon parallèle, la propagation fine au long de chaque intervalle de temps étant calculée par un processeur. Par conséquent, en parallélisant l'évolution coûteuse du modèle fin, la méthode permet de réduire le temps nécessaire pour calculer la solution précise.

Il est facile de montrer que la solution de la méthode Pararéel converge exactement vers la solution de référence (celle du modèle fin) en un nombre fini d'itérations (égal au nombre d' intervalles de temps). Néanmoins, la méthode n'est intéressante (*i.e.* moins coûteuse qu'une simulation séquentielle du modèle fin) que si elle fournit une convergence beaucoup plus rapide. Cela est bien vérifié pour des problèmes paraboliques et diffusifs, mais pas pour des problèmes hyperboliques ou dominés par des phénomènes advectifs, pour lesquels la méthode présente des instabilités et une convergence lente. Ce comportement est dû aux différences de vitesse de phase discrète entre le modèle fin et le modèle grossier (Ruprecht, 2018).

On s'intéresse donc à une variante de la méthode Pararéel où le modèle grossier est remplacé par des modèles d'ordre réduit (ROMs) formulés à la volée au cours des itérations, à partir des réalisations (snapshots) de la solution fournies par les itérations précédentes (Chen et al., 2014). Ce modèle réduit est résolu avec le même petit pas de temps que celui du propagateur fin, ce qui permet d'espérer des meilleures approximations des vitesses de phase. Pour le problème considéré ici (les équations de Saint-Venant), la réduction de modèle est effectuée en combinant deux techniques, la décomposition orthogonale aux valeurs propres (Proper orthogonal decomposition - POD) et la méthode d'interpolation empirique (Empirical Interpolation Method - EIM), ce qui convient à des problèmes non linéaires (Barrault et al., 2004; Chaturantabut and Sorensen, 2010).

Des premiers résultats numériques, réalisés sur des problèmes relativement petits et simples, et couplant les équations de Saint-Venant classiques à différents échelles (*i.e.* sans encore utiliser le concept de porosité), montrent que la méthode Pararéel basée sur des modèles réduits est plus performante que la méthode classique dans des situations où celle-ci présente des instabilités et/ou une convergence lente. En outre, à travers une implémentation parallèle (même sur des petites dimensions), on est capable de comparer les variantes de la méthode Pararéel en termes de temps de calcul. La méthode utilisant des modèles réduits présente des coûts beaucoup plus grands, en mettant en évidence l'importance d'une implémentation efficace de la réduction de modèle.

Ces résultats montrent aussi que la performance de la méthode basée sur des modèles réduits est influencée par plusieurs paramètres et configurations, par exemple la troncature des modèles réduits, le maillage et la régularité de la solution. De façon plus générale, on peut conclure que cette performance est conditionnée par la qualité de modèles réduits formulés à la volée et par leur efficacité à reproduire la dynamique du modèle fin, ce qui motive le travail présenté dans le prochain chapitre.

Chapitre 4 - Améliorations de la méthode Pararéel basée sur des modèles d'ordre réduit

On commence le chapitre en investiguant de façon plus détaillée la qualité du modèle réduit formulé à la volée au cours des itérations. Comme difficulté majeure, ces modèles réduits sont construits à partir des snapshots peu précis (puisque fournis au cours des itérations de la méthode Pararéel). Il est donc nécessaire d'améliorer la qualité de la réduction de modèle. Néanmoins, ces améliorations ne doivent pas entraîner des coûts importants, sinon la méthode Pararéel perdrait son intérêt. On propose trois approches, qui sont étudiées en utilisant les mêmes cas test définis dans le chapitre précédent :

- **Enrichissement des ensembles de snapshots utilisés pour la réduction de modèle :** dans la formulation originale de la méthode Pararéel basée sur des modèles réduits, les snapshots sont pris dans les instants définissant les intervalles de temps. Néanmoins, il existe des solutions calculées dans des pas de temps intermédiaires (lors de la correction fine dans l'algorithme Pararéel) qui ne sont pas utilisées. On propose donc d'enrichir les ensembles de snapshots avec un certain nombre de ces solutions intermédiaires (ce qui n'implique pas des coûts additionnels pour les calculer, puisqu'elles sont déjà disponibles). Néanmoins, la réduction de modèle elle-même devient plus coûteuse, à cause de sa complexité quadratique par rapport au nombre de snapshots. On constate que des enrichissements simples (avec peu de snapshots additionnels) permettent d'améliorer la convergence et stabilité de la méthode Pararéel, avec des temps de calcul qui restent raisonnables. Pour des enrichissements plus importants, les améliorations sont moins notables mais les temps de calcul deviennent prohibitifs.

- **Formulation des modèles réduits locaux en temps :** cette approche, inspirée de la méthode de décomposition en intervalles principaux (Principal Intervals Decomposition - PID) (Ijzerman, 2000), consiste à diviser le domaine temporel em fenêtres (qui peuvent elles mêmes contenir plusieurs intervalles de temps de la méthode Pararéel) et faire des réductions de modèle dans chacune d'elles (au lieu d'une réduction globale en temps), afin de mieux capturer des phénomènes locaux. La PID a été originellement conçue dans le cadre de la POD ; on l'étend au cadre de la POD-EIM et on l'introduit dans la méthode Pararéel. Cette approche fournit quelques améliorations à la stabilité de la méthode ; néanmoins, dans des cas où la réduction de modèle globale est déjà efficace, cette formulation locale a des effets négatifs sur la convergence et la stabilité. Concernant le coût numérique, les temps de calcul deviennent prohibitifs quand le nombre de fenêtres augmente, à cause de la formulation de plusieurs modèles réduits à chaque itération.

- **Utilisation d'une approche adaptative :** basée sur la méthode Pararéel adaptative proposée par Maday and Mula (2020) dans le cadre de la méthode Pararéel classique, cette approche consiste à utiliser, au cours des itérations, des modèles fins qui se raffinent progressivement, au lieu d'un modèle fixe et très cher (celui de référence). On étend cette approche au cadre de la méthode Pararéel basée sur des modèles réduits. On constate que des améliorations de la stabilité et notamment des temps de calcul sont obtenus. La méthode fournit des solutions avec des précisions donnés avec un temps computationnel moins important, même si plus d'itérations s'avèrent nécessaires, puisque les itérations intermédiaires sont considérablement moins chères.

On conclut le chapitre en étudiant l'utilisation combinée de ces approches. Notamment, l'enrichissement des ensembles de snapshots, avec peu de snapshots additionnels, combiné avec l'approche adaptative, se révèle être une bonne solution pour établir un compromis entre convergence et temps de calcul.

Chapitre 5 - Stratégies pour des simulations plus longues utilisant la méthode Pararéel

Dans les chapitres précédents, la performance des variantes de la méthode Pararéel a été évaluée en utilisant des exemples relativement simples, et il a été illustré que, sous les mêmes configurations, la méthode basée sur des modèles réduits peut être plus performante que la méthode classique. Dans ce chapitre, on étudie comment les deux méthodes se comportent dans des problèmes de taille plus grande. On considère des problèmes analogues à ceux étudiés auparavant, mais définis dans des domaines spatiaux et temporels plus grands. Dans un premier temps, à titre de motivation, il est constaté que, en gardant les mêmes configurations qui se sont montrées efficaces pour les petits problèmes, la méthode basée sur des modèles réduits présente des instabilités et une convergence plus lente pour les problèmes de taille plus grande. On compare donc deux stratégies pour résoudre ces problèmes :

- **Utilisation des intervalles de temps plus grands :** dans les simulations des chapitres précédents, les intervalles de temps (sur lesquels la parallélisation de la méthode Pararéel s'appuie) ont été considérés comme égaux aux pas de temps associés au propagateur grossier (ce qui était un choix naturel vu la petite dimension des problèmes). On définit donc des intervalles plus grands, contenant plusieurs pas de temps grossiers, et on étudie le comportement des variantes de la méthode Pararéel en fonction des tailles des intervalles. On vérifie que la stabilité et la convergence de la méthode classique sont considérablement améliorées en choisissant des intervalles plus grands. Pour la méthode basée sur des modèles réduits, on observe aussi des améliorations, mais la relation entre la performance et la taille des intervalles est moins claire.

- **Définition des simulations locales en temps de la méthode Pararéel :** dans cette approche, on garde les petits intervalles de temps, mais on fait des simulations de la méthode Pararéel dans des fenêtres de temps (au lieu d'une simulation couvrant tout le domaine temporel). Dans chaque sous-intervalle, un certain nombre d'itérations est réalisé, fournissant la solution initiale pour la fenêtre suivante. Il est vérifié que, en définissant des fenêtres à chaque fois plus petites, la méthode Pararéel basée sur des modèles réduits devient plus stable et converge plus vite. Cela est aussi vrai pour la méthode classique mais, dans ce cas, si les intervalles restent petits, la convergence et la stabilité sont limitées.

Comme conclusion de ce chapitre, la performance de la méthode Pararéel classique semble plutôt conditionnée par la taille des intervalles de temps distribués à des processeurs. D'autre part, la méthode basée sur des modèles réduits est plus efficace en effectuant des sous-simulations. Cela indique que la réduction de modèle, réalisée au cours des itérations en utilisant des snapshots fournis par la méthode, est assez difficile quand elle est faite sur des longs intervalles de temps ; donc, la méthode basée sur des modèles réduits est moins appropriée pour résoudre des problèmes définis dans des grands domaines temporels.

Il faut noter que les deux approches étudiées dans ce chapitre limitent l'accélération de la simulation computationnelle qui peut être fournie par la méthode Pararéel. Dans le premier cas, en définissant des intervalles de temps plus grands, le nombre de processeurs qui peuvent être utilisés en parallèle devient plus petit. Dans le deuxième cas, on sérialise partiellement la simulation de la méthode Pararéel, et chaque sous simulation contient un petit nombre d'intervalles de temps (donc moins parallélisable). Pour cette deuxième approche, il existe aussi une limitation de la convergence de la solution obtenue: puisque chaque sous-simulation reçoit comme condition initiale une solution fournie par les simulations précédentes, sa convergence est limitée par la qualité de cette solution initiale.

Chapitre 6 - Couplage entre les équations de Saint-Venant classiques et à porosité

Après des études initiales où la méthode Pararéel classique et celle basée sur des modèles réduits ont été appliquées au couplage des équations de Saint-Venant classiques à différentes échelles, on étudie, dans ce chapitre, leur application à de simulations d'inondations urbaines, où des modèles de Saint-Venant à porosité jouent le rôle de propagateur grossier.

D'après les résultats observés dans les chapitres précédents, on peut s'attendre à des difficultés importantes dans cette application. Des simulations d'inondations urbaines sont en général caractérisées par des grands domaines en espace et temps, et par des solutions fortement discontinues due à la présence d'obstacles. Un premier exemple illustre bien cette difficulté: il s'agit d'un des cas tests considérés dans les chapitres précédents et pour lequel la méthode Pararéel basée sur des modèles réduits est particulièrement efficace. En modifiant le cas test par l'introduction des obstacles dans le domaine, la méthode, sous les mêmes configurations, devient fortement instable.

La majorité de l'étude dans ce chapitre se base sur un cas test représentant un écoulement dans une zone urbaine fictive. On s'intéresse d'abord à l'influence du grossissement entre les modèles fin et grossier sur la convergence et la stabilité des méthodes Pararéel. Pour cela, on définit différents modèles de référence (*i.e.* différentes discrétisations des équations de Saint-Venant classiques), en plus de deux discrétisations des modèles à porosité.

Dans un premier temps, on fixe le modèle grossier (une discrétisation des équations à porosité) et on considère plusieurs modèles de référence. Dans le cadre de la méthode Pararéel classique, on observe une relation très claire entre la vitesse de convergence et le raffinement du modèle fin : quand celui-ci est plus grossier (donc plus proche du modèle à porosité), l'erreur de la solution Pararéel décroît plus rapidement. Cette relation est moins claire dans la méthode basée sur des modèles réduits.

Dans une deuxième approche, on fixe le modèle fin (la plus fine des discrétisations des équations de Saint-Venant classiques) et on utilise différents modèles grossiers (les modèles classique et à porosité avec différentes tailles de maille et de pas de temps). Pour la méthode Pararéel classique, on observe une convergence plus rapide quand le modèle grossier est plus fin, et notamment quand il s'agit d'une discrétisation du modèle classique. Dans la méthode basée sur des modèles réduits, on observe un comportement instable dans toutes les configurations, indépendamment du raffinement du propagateur grossier, surtout à la fin du domaine temporel, où le profil de la solution est plus complexe et présente de fortes discontinuités.

On conclut que, tandis la méthode Pararéel classique est clairement influencée par le grossissement entre les modèles fin et grossier, la performance de la méthode basée sur des modèles réduits semble plutôt conditionnée par la capacité des modèles réduits à représenter la dynamique du modèle de référence, ce qui se révèle particulièrement difficile quand celui-ci est très fin, défini dans des grands domaines en espace et en temps et caractérisé par des solutions présentant des fortes variations et discontinuités. Comme ce scénario est typique de la simulation des inondations urbaines, la méthode Pararéel basée sur des modèles réduits convient mal à cette application.

Après cette étude initiale, où on s'intéressait à la stabilité et la convergence des méthodes, on analyse ses performances, en termes d'accélération de la simulation, en fonction des paramètres choisis. Des bonnes approximations de la solution de référence sont obtenues avec la méthode Pararéel classique, mais avec des configurations assez restrictives (choix de grands intervalles de temps, définition des sous-simulations) afin d'assurer la stabilité, ce qui impose des fortes limitations à l'accélération attendue. En effet, en général la simulation de la méthode Pararéel est au plus deux ou trois fois plus rapide que la simulation du modèle fin, ce qui représente une petite efficacité parallèle puisque toutes les simulations ont été exécutées avec vingt processeurs. Des bonnes approximations peuvent aussi être obtenues avec la méthode basée sur des modèles réduits, mais avec des configurations encore plus restrictives, ce qui rend la méthode moins intéressante que la méthode classique.

Des résultats intéressants sont aussi obtenus en utilisant l'approche adaptative, décrite dans le Chapitre 4. En utilisant des modèles fins qui se raffinent progressivement au cours des itérations, les performances des méthodes Pararéel classique et basée sur des modèles réduits sont considérablement améliorées. Néanmoins, en pratique, il s'avère plus efficace d'utiliser un des modèles fins intermédiaires directement comme propagateur grossier : même si les itérations sont plus chères, la convergence est plus rapide, et des solutions avec la même précision sont obtenues avec un temps de calcul moins important. Malgré cela, l'amélioration de la stabilité et de la convergence indiquent que des méthodes de parallélisation en temps à plusieurs niveaux peuvent être envisagées comme une approche plus performante.

Chapitre 7 - Conclusions et perspectives

Dans ce travail, on a étudié l'application de méthodes de parallélisation en temps pour le couplage des modèles de Saint-Venant à différentes échelles pour la simulation d'inondations urbaines. À cause des difficultés d'application de ces méthodes à des problèmes hyperboliques, on a considéré la méthode Pararéel dans sa formulation originale ainsi qu'une de ses variantes utilisant des modèles réduits formulés au cours des itérations et résolus avec le même discrétisation temporelle du modèles de référence.

Il a été vérifié que cette variante peut apporter d'importantes améliorations à la méthode Pararéel dans des configurations où la méthode classique présente des instabilités et une convergence lente. Néanmoins, ces améliorations ne se vérifient que pour des problèmes simples, résolus dans de petits domaines en espace et en temps et avec des solutions relativement lisses qui peuvent être capturées de façon efficace par la réduction de modèle. Pour des problèmes plus complexes, comme dans le cas des inondations urbaines, la méthode utilisant des modèles réduits présente des fortes instabilités qui limitent son utilisation aux applications envisagées dans ce travail.

En conclusion, il est plus efficace d'utiliser la méthode parallèle classique. Néanmoins, afin d'assurer sa stabilité et sa convergence, il se révèle nécessaire d'utiliser des configurations très restrictives, ce qui limite fortement l'accélération qui peut être fournie par la méthode. Donc, dans l'état actuel de ce travail, la méthode Pararéel n'est pas envisageable comme seul outil pour fournir d'importantes accélérations des simulations d'inondations urbaines. Pour cela, il faut l'utiliser, par exemple, en complément à des méthodes de décomposition de domaine spatial, donc dans un cadre de parallélisation en espace et en temps.

CHAPTER 1

INTRODUCTION

Contents

1.1 Context and challenges of urban flood modelling 1

1.1.1 Urban floods: an increasing issue 1

1.1.2 Mathematical and numerical modelling as components of urban flood management 2

1.1.3 Porosity-based shallow water models 3

1.2 Parallelization-in-time 3

1.2.1 A further direction for parallelization 3

1.2.2 Classification and examples of parallel-in-time methods 5

1.2.3 Challenges and alternatives for hyperbolic problems 6

1.3 Objectives and methodology 7

1.4 The SW2D-LEMON software 7

1.5 Publications and conferences 8

1.6 Summary of the thesis 9

The objective of this PhD thesis is to develop a numerical model coupling shallow water models at different scales for the simulation of urban floods. More precisely, we aim to couple the classical shallow water equations, an accurate and computationally expensive model solved within fine spatial and temporal discretizations, with the so-called *porosity-based shallow water equations*, consisting in upscaled and low-expensive approximations to the fine model. The idea is to keep the best properties of shallow models at each scale: the accuracy of the fine model and the low computational cost of the upscaled one. We explore here the use of iterative predictor-corrector parallel-in-time methods, namely the *parareal method* and some of its variations, which naturally fit to the proposed coupling between fine and coarse models.

1.1 Context and challenges of urban flood modelling

1.1.1 Urban floods: an increasing issue

Floods are increasingly important and frequent events whose extensive human and socio-economic impacts pose real challenges for societies and governments. According to a 2015 report by the Centre for Research on the Epidemiology of Disasters (CRED) and the United Nations Office for Disaster Risk Reduction (UNDRR, formely UNISDR), floods were the most frequent weather-related natural disasters in the period 1995-2015, accounting for 47% of the reported events and affecting 2.3 billion people (CRED and UNISDR, 2015). In the period 1997-2006, the number of flood events doubled, the largest increase among all water-related disasters (Adikari and Yoshitani, 2009). Data obtained from the CRED's Emergency Events Database (EM-DAT)[1] attest the increasing frequency of flood events (Figure 1.1), a scenario that

[1] EM-DAT, CRED / UCLouvain, Brussels, Belgium – www.emdat.be (D. Guha-Sapir)

tends to get worse in the following decades due to climate changes and the consequent larger frequency of extreme weather events (Jha et al., 2011; CRED and UNISDR, 2015).

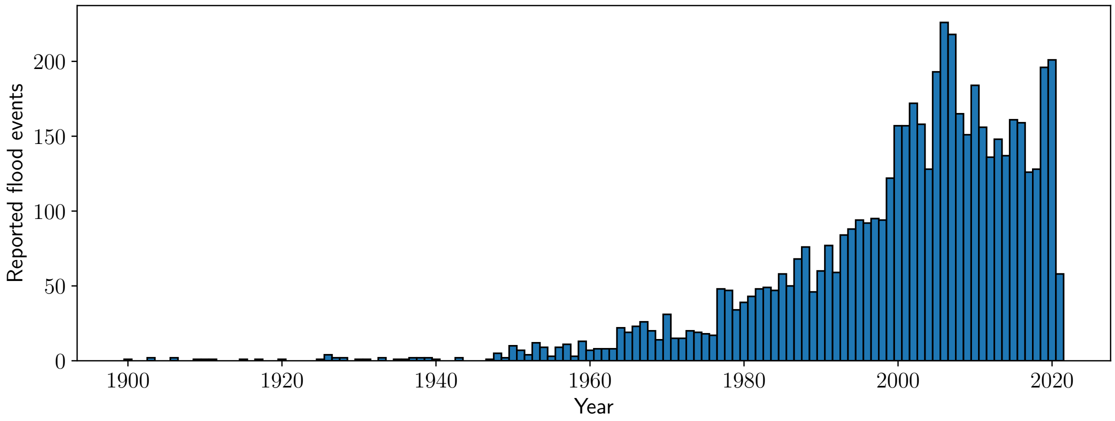


Figure 1.1: Number of reported flood events per year since 1900. Data extracted from the CRED's Emenergency Events Database (EM-DAT).

Urban floods are protagonist in this context, with many of major flood events taking place in highly dense urban areas (Jha et al., 2011). The urbanization process itself is a major factor for increasing the frequency and intensity of flood events, due to the land use leading to less impermeable surfaces, obstructions of flow paths and generation of urban microclimates, among other factors that affect local hydrological characteristics (Jacobson, 2011). Fast and unplanned urbanization processes contribute to increase these impacts. The consequences of flood hazards also tend to be more drastic in urban areas, because of their highly concentrated and growing population, with 55% of the world population living in urban areas, a rate projected to reach 68% by 2050 (United Nations, 2019), in addition to other indirect impacts such as water-borne diseases, water pollution and interruption of essential supply chains (Tucci, 2007; Jha et al., 2011). Therefore, urban floods are an increasing issue requiring an integrated risk management for reducing its impacts.

1.1.2 Mathematical and numerical modelling as components of urban flood management

A large variety of structural and non-structural measures may be adopted for the management of urban flood hazards, in order to avoid or minimize their impacts (Tucci, 2007; Jha et al., 2011). Structural measures are infrastructural works for containing discharges, increasing drainage and reducing and delaying flood peaks. Non-structural ones include *e.g.* forecasting, monitoring and warning systems, civil defence actions and educational campaigns.

Mathematical and numerical modelling of hydrodynamics processes, also referred as hydroinformatics, are essential non-structural components of urban flood management, either by providing real-time forecasting or pre-simulating scenarios to guide decision-making (Hénonin et al., 2013). Various types of models exist, *e.g.* one-dimensional modeling of drainage systems, two-dimensional shallow-water-based free-surface models and coupled free-surface and drainage models, with different advantages, disadvantages and applicability contexts (Hénonin et al., 2013; Guo et al., 2020).

Two-dimensional free-surface models are able to provide detailed flood simulations and have been an intensely studied research topic. A large variety of models are developed, with different numerical schemes, spatial and temporal discretizations and shock-capturing algorithms, aiming to provide more accurate and realistic numerical simulations and overcome challenges such as wetting-drying problems and flows over discontinuous bathymetries (Teng et al., 2017). However, these models present a high computational complexity, which limits their application as real-time forecasting models or when several simulations need to be performed. Indeed, reasonable balances between accuracy and computational effort, ensuring proper representation of small-scale phenomena, require fine computational meshes, with cells in the order of meters, or even smaller near singularities (Guinot, 2012), which also limits the time step due to stability conditions (Gallegos et al., 2009), leading to prohibitive simulation times in large spatial and temporal domains. Domains larger than 1000 km^2 are usually considered impracticable to be modelled with two-dimensional hydrodynamic models when resolutions smaller than 10 m are required (Teng et al., 2017). As a result, while 1D drainage models are operational for real-time application, 2D surface models and coupled 1D drainage - 2D surface models are in research stage (Hénonin et al., 2013; Guo et al., 2020).

Research efforts have been made for accelerating the numerical simulation of 2D urban free-surface models. A research guideline is to use high-performance computing (HPC), with CPU- or GPU-based parallelization and spatial domain decomposition techniques, features included in many of the available software packages for 2D hydraulic modelling (Neelz and Pender, 2013; Guo et al., 2020). Recent works prove the potentials in using HPC for urban flood modelling, *e.g.* with high-resolution city-scale simulations completed in nearly real-time in a multi-GPU implementation (Xing et al., 2018). Another approach for reducing the computational cost of 2D shallow water simulations is considering simplified models (Guo et al., 2020). It can be done for example by neglecting specific terms of the governing equations, which gives reasonable approximations under certain flow hypothesis but may lead to inaccurate results in more general frameworks. This approach can be useful for obtaining simplified information in large scale applications, such as the flood extent and water of level, but at the cost of inaccurate or no representation of dynamic processes and velocity fields (Teng et al., 2017).

1.1.3 Porosity-based shallow water models

An alternative approach for simplifying and reducing the complexity of 2D shallow water models consists in looking the urban zone from a macroscopic point of view. Instead of a detailed representation of buildings, obstacles, streets and other small-scale (relative to the city scale) elements of the urban geometry, one seeks a statistical, large-scale description: for example, the fraction of a certain area or cross section of the urban zone that is not covered by buildings or obstacles, thus being available for the flow.

In this sense, the urban geometry can be seen as a porous media, where buildings and obstacles are the impermeable solid fraction, and streets and avenues are voids. Macroscopic representations of porous media have been studied for a long time, using different techniques known as *upscaling*, and this idea could be translated to the modelling of urban flows. It gave rise to the so-called *porosity-based shallow water models*. First proposed by (Defina et al., 1994) for treating wet and dry regions with uneven bathymetries, this approach was later introduced in the context of urban floods by (Hervouet et al., 2000) and (Guinot and Soares-Frazão, 2006). Under this porosity approach, fine computational meshes are no longer required, since the urban geometry is not physically discretized (*i.e.* as holes in the mesh); instead, coarser meshes can be used and the urban geometry is taken into account by defining spatial porosity parameters, attributed to each computational cell and introduced in the governing equations. It is illustrated with an example in Figure 1.2, showing the computational meshes used for the simulation of a flood in the neighborhood of West Sacramento (California, USA).

The use of coarser meshes, with cells sizes of about one order magnitude larger than in the fine, classical shallow water models, and the consequent larger (in the same proportion) time steps allowed by stability conditions lead to much smaller computational times (between two and three orders of magnitude smaller) (Guinot et al., 2017). Therefore, porosity-based shallow water models provide low-computational approximations to the fine ones, and, through a steady evolution along the past two decades, these approximate models have become more accurate and physically meaningful.

However, although these constant developments and improvements, fine scale phenomena cannot be accurately represented by porosity-based models, due to its upscaled nature. This motivates the work developed in this thesis. By coupling fine and upscaled shallow water models, we aim to construct a model able to provide accurate solutions, including fine scale features, within smaller computational costs compared to the fine model.

1.2 Parallelization-in-time

1.2.1 A further direction for parallelization

Possibilities and limits in scientific computing are determined by the current computing capacity and performance, with numerical simulations needing to satisfy a trade-off between accuracy and computational cost. Along decades, computer development has been guided by the well-known Moore's law (Moore, 1965, 1975), which stated that the number of transistors in computing chips would double every

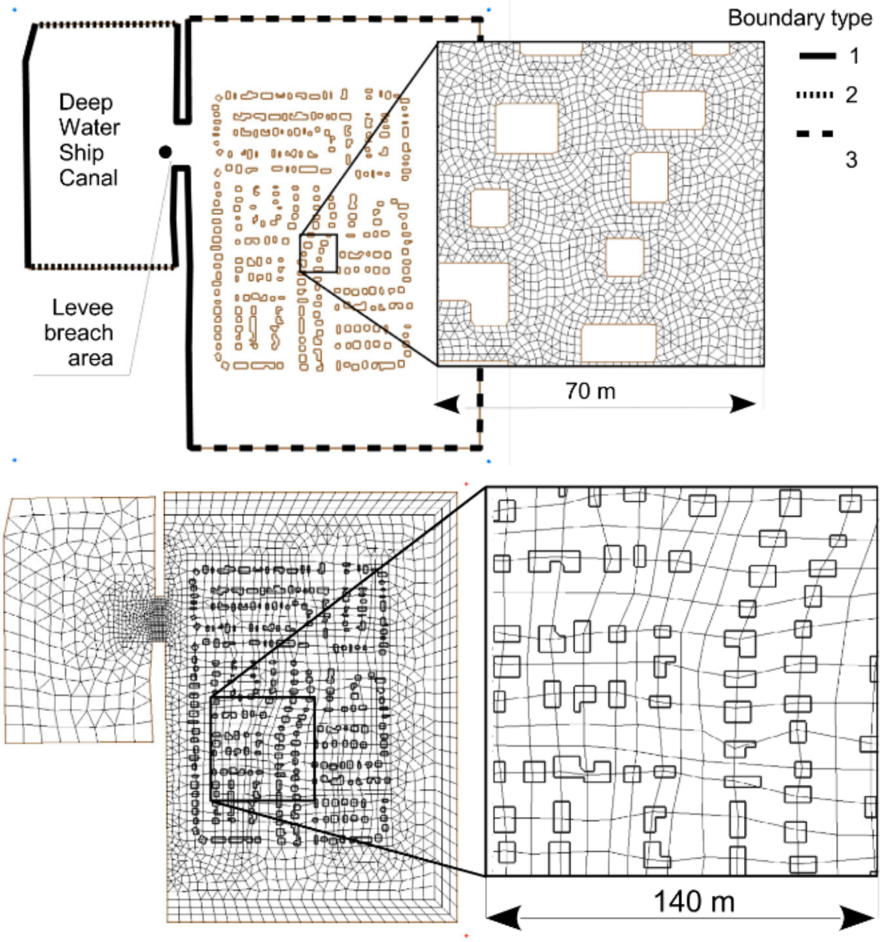


Figure 1.2: Example of computational meshes for the simulation of an urban flood at different scales. The meshes discretize the neighborhood of West Sacramento (California, USA) Top: mesh used for the classical shallow water model, with small cells and the buildings represented as holes in the mesh. Bottom: mesh used for the porosity-based shallow water models, with larger cells and the buildings not physically represented in the mesh. Figures extracted from (Guinot et al., 2017).

two years. This prediction has and still is verified, but with decreasing growth rates that may lead to a stagnation in the following years. Moreover, increasing the number of transistors is not translated into a gain of performance at the same rate (Shalf, 2020). Summed up to manufacturing and energy consumption issues (Markov, 2014), these factors indicate an obsolescence of Moore's law. It explains a movement, from the 2000's, towards a multicore computing paradigm, in which performance gains are obtained rather by increasing the number of parallel processors than improving the processors themselves (Parkhurst et al., 2006; Dolean et al., 2015).

Scientific computing has naturally followed this trend, and research efforts have been made for developing numerical methods and algorithms able to take advantage of parallelism (Dolean et al., 2015). In the context of the numerical simulation of partial differential equations, spatial domain decomposition methods have a long history in this direction. First proposed by (Schwarz, 1870) and formulated in a parallel framework by (Lions, 1988), this class of numerical methods consists in dividing the spatial domain in many subdomains and solving the governing equations iteratively in each of them, in parallel relative to each other.

Parallel-in-time (PinT) methods are more recent, being proposed and developed over almost 60 years, since the work of Nievergelt (1964), and having gained special importance over the past two decades, due to the emergence and improvement of massively parallel computers. The increasing number of scientific publications on the subject over the past years (Figure 1.3) confirms its growing importance. The parallelization of the temporal direction shows itself as a new path for further computational speedup when spatial parallelization saturates. Evidently, the causality principle, *i.e.* the exclusively forward advance of time, imposes a conceptual challenge for the development of PinT algorithms (Gander, 2015).

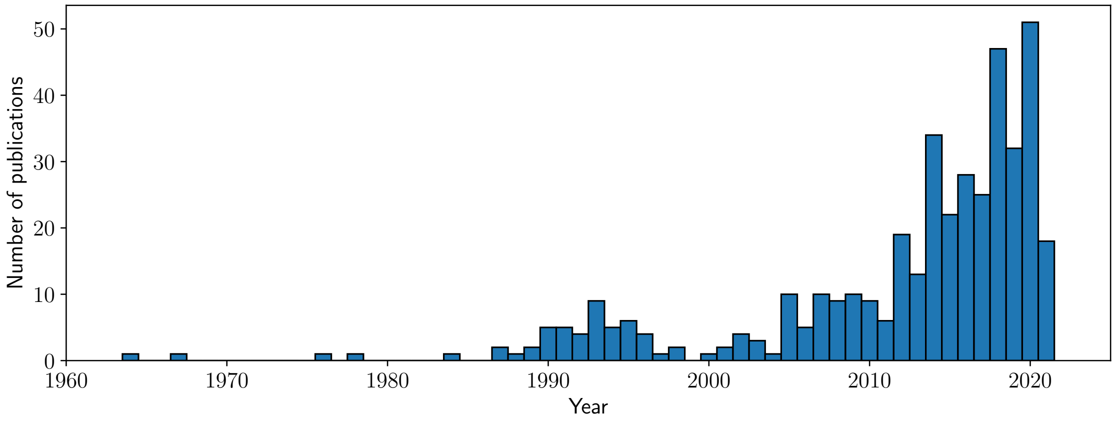


Figure 1.3: Number of scientific publications related to parallel-in-time methods per year. Data obtained from Parallel-in-Time website[2].

1.2.2 Classification and examples of parallel-in-time methods

Various PinT methods have been presented in the literature, and we refer to Gander (2015) and Ong and Schroder (2020) for detailed reviews. Following (Bellen and Zennaro, 1989), they can be classified into three categories :

- *Parallelization across the method*: it consists in parallelizing the numerical scheme used for updating the solution within each time step. These methods are also referred as *direct solvers* by (Gander, 2015) since they usually do not involve an iterative process. Some examples are Runge-Kutta schemes with independent steps computed in parallel (Burrage and Suhartanto, 1997) and the Revisionist Integral Deferred Correction (RIDC) (Christlieb et al., 2010);
- *Parallelization across the system*, or *across the problem*, in which the problem is distributed among processors. For example, for a system of ordinary differential equations, each equation is computed by a parallel processor in an iterative procedure, which is the principle of the waveform relaxation method (Lelarasmee et al., 1982); its extension to space-time partial differential equation, called Schwarz waveform relaxation method (Gander, 1996), is a parallel-in-time algorithm based on spatial domain decomposition, performing an iterative simulation of the problem in each spatial subdomain;
- *Parallelization across the time*, or *across the steps*, in which the problem is solved simultaneously over several time steps. It is performed via iterative processes using a low-expensive and less accurate coarse model along with an expensive and accurate fine one, the former being computed sequentially along the entire temporal domain and the latter in parallel among time steps. These methods are either based on multiple shooting or multigrid ideas, and some examples are the parareal (Lions et al., 2001), the Parallel Implicit Time-integrator Algorithm (PITA) (Farhat and Chandesris, 2003), the Multigrid Reduction In Time (MGRIT) (Friedhoff et al., 2013; Falgout et al., 2014) and the Parallel Full Approximation Scheme in Space and Time (PFASST) (Emmett and Minion, 2012).

Figure 1.4 schematizes the basic ideas behind each iteration of methods proposing parallelization across the system and across the time. We remark that this classification is not rigid, and methods may be combined for further improvements. We can cite, for example, combined parareal and waveform relaxation method (Gander et al., 2013), thus mixing parallelization across the time and across the system, and a recent proposed PFASST method with inner parallel solvers (Schöbel and Speck, 2020), thus combining parallelization across the time and across the method.

[2] https://parallel-in-time.org

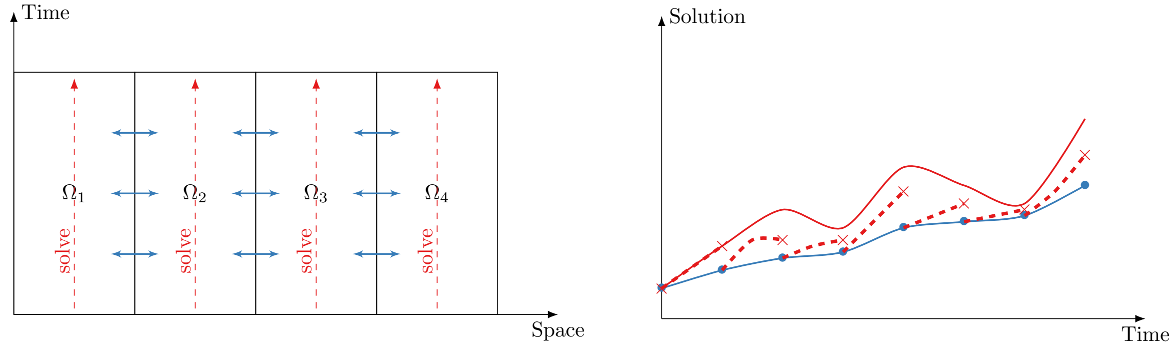


Figure 1.4: Basic ideas behind each iteration of methods proposing parallelization across the system (left) and parallelization across the time (right). In parallelization across the system, the spatial domain is divided into several subdomains and the problem is solved in each of them (red, dashed arrows), and information (interface boundary conditions) are exchanged between the subdomains (blue, horizontal arrows). In parallelization across the time, the expensive, fine simulation along the entire temporal domain (red, full line) is replaced by fine computations in parallel along smaller time slices (red, dashed lines), relying on a less expensive and less accurate solution computed sequentially (blue, dotted line).

In this work, we focus on the parareal method, one the most popular PinT methods and whose name stands for “parallel in real-time”. It is an iterative predictor-corrector method that consists in sequentially computing predictions using a coarse model, whereas the corrections, given by an accurate and expensive fine model, are computed in parallel along time slices dividing the temporal domain. Through iterations, the parareal solution converges to the reference solution, given by a sequential simulation of the fine model. This formulation using simultaneously a coarse and a fine models is a well-fitting approach for the purposes of this work, aiming to couple the porosity-based shallow water models, a low-expensive and less accurate model solved within larger time steps and computational cells, with the classical shallow water equations, which are more accurate and expensive since they are solved using finer discretizations.

1.2.3 Challenges and alternatives for hyperbolic problems

Parallel-in-time methods have been applied to a large variety of problems. For mentionining some few, recent examples, we cite applications to fluid-structure interactions (Margenberg and Richter, 2021), atmospheric circulation models (Hamon et al., 2020), turbulent flow (Lunet et al., 2018), molecular dynamics (Rosa-Raíces et al., 2019), thermodynamics (Stump and Plotkowski, 2020), electrodynamics (Bolten et al., 2020), poromechanics (Borregales et al., 2019), plasma dynamics (Samaddar et al., 2019), optimal control problems (Wu and Liu, 2020) and blood flow modelling (Blumers et al., 2021), with many implementations in massively parallel HPC systems. The most successful applications are for parabolic, diffusive problems, for which PinT methods are able to provide substantial reduction of the computational costs. It is not the case, however, of hyperbolic or advection-dominated problems, even the simplest ones, for which this class of methods, including the parareal, usually presents stability and convergence issues (Ruprecht, 2018). Therefore, the application of PinT for fluid dynamics problems, described by hyperbolic and advective phenomena, is a substantial challenge. Since the shallow water equations are an hyperbolic model, such challenges have to be faced in this work.

Adaptations of PinT methods are presented in the literature proposing alternatives for improving their performance when applied to hyperbolic problems. In the framework of predictor-corrector iterative methods, using coarse and fine models, a class of adaptations consists in modifying and improving the coarse model by reusing information computed in the previous iterations. In the case of parareal, proposed methods consist in formulating reduced spaces spanned by solutions of previous iterations (the Krylov-subspace-enhanced parareal method, initially proposed in the PITA framework by Farhat et al. (2006) and showed to be equivalent to the parareal case, for linear problems, by Gander and Petcu (2008)) and reduced-order models (ROMs) also constructed from previous solutions (Chen et al., 2014). While the former approach is efficient only for linear problems, the latter can also be applied to nonlinear ones, by using suitable model reduction techniques.

In this work, we give a special attention to the parareal variant using reduced-order models, to which we refer here as *ROM-based parareal method*. By applying it to a large variety of problems, Chen et al.

(2014) showed that it is able to greatly improve the stability and convergence of the parareal method in some situations, but fail in others. We then investigate its performance for solving the two dimensional nonlinear shallow water equations, and, in a second moment, for coupling the classical and porosity-based shallow water models.

We notice that model order reduction is a vast domain, comprising a large variety of methods. As a common goal, these methods aim to reduce the complexity, thus the computational time, of an expensive problem. Their application in the parareal framework consists in replacing the low-expensive and low-accurate coarse model by a low-expensive ROM approximating the dynamics of the fine model. Even if various model reduction methods could be considered, we focus here in a method combining two techniques, the Proper Orthogonal Decomposition (POD) (Kosambi, 1943) and the Empirical Interpolation Method (EIM) (Barrault et al., 2004), which is suitable for nonlinear problems and can be efficiently implemented by using well performing linear algebra software libraries.

1.3 Objectives and methodology

We investigate in this work the application of the parareal method for coupling the classical and porosity-based shallow water equations (which act respectively as fine and coarse models) to the simulation of urban floods. Here, in the context of predictor-corrector iterative parallel-in-time methods, the term "coupling" should be understood as the simultaneous use of different discretizations (*i.e.* at different scales) of a mathematical model, or the use of different models for the same physical problem, in order to obtain an approximate numerical solution to it. It is different from the usual meaning *e.g.* in climate simulations in Earth System Models (Washington et al., 2009), in which models describing different physical processes and possibly defined in different spatial domains (*e.g.* atmospheric and oceanic circulation models) are simulated simultaneously and exchange information. Therefore, along this work we refer to a coupling between classical and porosity-based shallow water models, or even to a coupling between classical shallow water models with different discretization sizes.

Due to well known issues of the parareal method when applied to hyperbolic problems, we consider the ROM-based parareal method as alternative approach and we compare its performance with the original (or classical) parareal. However, the introduction of model reduction techniques in the parareal iteration may lead to additional computational costs that limit the effective interest in using this approach, even if possible improvements in terms of stability and convergence can be obtained. Thus, we propose here to evaluate and compare the variants of the parareal method also in terms of computational cost, via real parallel implementations (even if in small scales).

Therefore, along this work we perform a detailed comparison, in terms of convergence, stability and computational cost, between the classical and ROM-based parareal methods for solving the problem proposed here. This study is performed by considering a gradually increasing complexity. In a first moment, we evaluate the performance of the methods in relatively simple problems, solved in small spatial and temporal domains and coupling the classical shallow water equations at different scales (*i.e.* the porosity-based models are not considered at this point). Based on these first observations, we investigate if further modifications of the ROM-based parareal can improve its performance. In a second moment, we still couple the classical model at different scales, but within larger spatial and temporal domains, and we investigate how the classical and ROM-based parareal methods behave in function of chosen parameters. Finally, we investigate the performance of both methods when used for coupling the classical and porosity-based models, in problems that are naturally more challenging, since they are usually solved in large domains and are characterized by highly discontinuous flows, due to the presence of obstacles in the urban geometry.

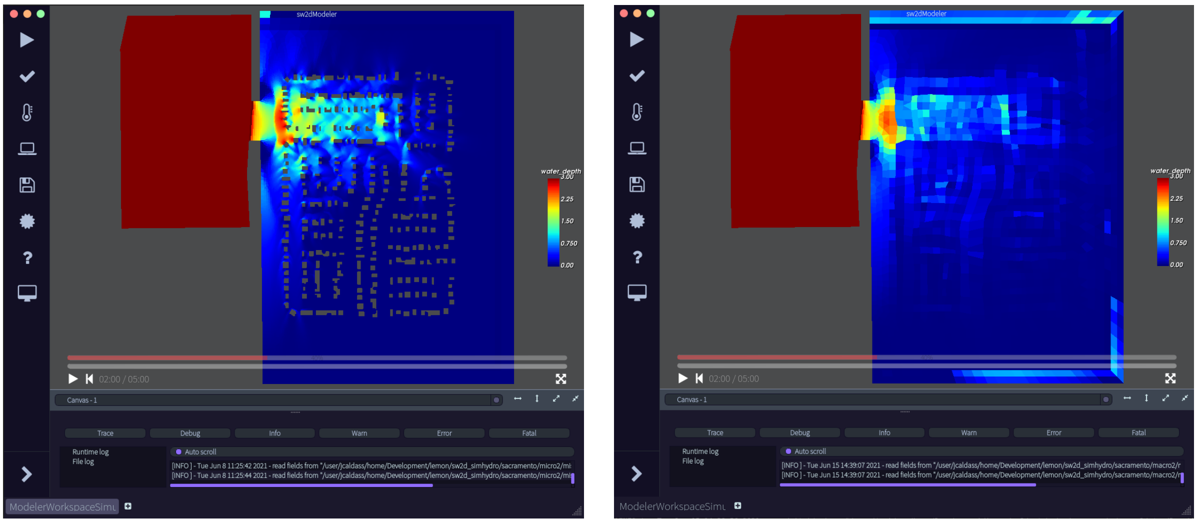
1.4 The SW2D-LEMON software

This PhD work took place in LEMON, a research team between Montpellier antenna from Inria Sophia Antipolis-Méditerranée center and the laboratories HydroSciences Montpellier (HSM) and Institut Montpelliérain Alexander Grothendieck (IMAG) from Université of Montpellier. LEMON (*Littoral, Environment: MOdels and Numerics*)[3] is a interdisciplinary team working on the mathematical modelling and numerical simulation of nearshore natural processes, including the development and implementation of porosity-based shallow water models.

[3] https://team.inria.fr/lemon/

Among its research activities, LEMON team develops *SW2D-LEMON*[4] (Caldas Steinstraesser et al., 2021a), a C++ multi-platform software for the simulation of shallow water models, including the upscaled, porosity-based ones. The software can be executed either from command line or a graphical user interface (GUI) allowing the visualization of output results, and is used both for research and educational purposes. Figure 1.5 presents an example of numerical simulations performed using SW2D-LEMON and visualized using its GUI. These simulations correspond to the West Sacramento test case, whose computational meshes are presented in Figure 1.2, and are solved using both the classical and porosity-based shallow water models.

The development of SW2D-LEMON was part of this PhD work. In order to facilitate the study and development of the parallel-in-time methods, a standalone software has been developed, into which the main features of the classical and porosity-based shallow water models were incorporated. A merge of the two softwares, *i.e.* the implementation of the developed PinT methods into SW2D-LEMON, is intended. Part of this merge has already been performed. Namely, the formulation of a matricial, global-in-space finite volume (FV) scheme, described in Section 2.10.3 and required for constructing reduced-order models, was implemented, prestoring time-independent geometric information and replacing the more traditional FV algorithm that loops on cells and interfaces of the computational mesh. This reformulation showed to be able to reduce the computational time (from factors between 2 and 4 in the performed simulations), even if other relatively important costs are present in SW2D-LEMON (*e.g.* the online storage of output results). In the standalone software developed in this work, which has simplified features such that the FV algorithm represents a more important fraction of the execution time, the reduction of the computational cost reached the order of tenths.



```
[INFO ] - Tue Jun 8 11:25:42 2021 - read fields from "/user/jcaldass/home/Development/lemon/sw2d_simhydro/sacramento/micro2/mi
[INFO ] - Tue Jun 8 11:25:44 2021 - read fields from "/user/jcaldass/home/Development/lemon/sw2d_simhydro/sacramento/micro2/mi
```

```
[INFO ] - Tue Jun 15 14:39:07 2021 - read fields from "/user/jcaldass/home/Development/lemon/sw2d_simhydro/sacramento/macro2/r
[INFO ] - Tue Jun 15 14:39:07 2021 - read fields from "/user/jcaldass/home/Development/lemon/sw2d_simhydro/sacramento/macro2/r
```

Figure 1.5: Examples of numerical simulations performed using the SW2D-LEMON software and visualized using its GUI. This problem correspond to the simulation of an urban flood in the neighborhood of West Sacramento (see Figure 1.2), using the classical shallow water model (left) and a porosity-based model (right). Figures present the water depth at a given time instant of simulation.

1.5 Publications and conferences

The following publications and conference presentations concern the work developed in this PhD thesis:

Publication in journals

- Caldas Steinstraesser, J. G., Guinot, V., and Rousseau, A. (2020a). Modified parareal method for solving the two-dimensional nonlinear shallow water equations using finite volumes. *Journal of Computational Mathematics*. Submitted;

[4] http://sw2d.inria.fr/

Conference proceedings

- Caldas Steinstraesser, J. G., Vincent, and Rousseau, A. (2021c). Application of a modified parareal method for speeding up the numerical resolution of the 2D shallow water equations. In *Simhydro 2021 – 6th International Conference Models for complex and global water issues – Practices and expectations*;
- Caldas Steinstraesser, J. G., Delenne, C., Finaud-Guyot, P., Guinot, V., Kahn Casapia, J. L., and Rousseau, A. (2021a). SW2D-LEMON: a new software for upscaled shallow water modeling. In *Simhydro 2021 – 6th International Conference Models for complex and global water issues – Practices and expectations*;

Conference presentations

- Caldas Steinstraesser, J. G., Guinot, V., and Rousseau, A. (2020b). A modified ROM-based parareal method for solving the two-dimensional nonlinear shallow water equations. CAN-J 2020 (Congrès d'Analyse Numérique pour les Jeunes). Online;
- Caldas Steinstraesser, J. G., Guinot, V., and Rousseau, A. (2021b). Coupling shallow-water models at different scales using parallel-in-time methods. 8ème école EGRIN (Écoulements Gravitaires et RIsques Naturels). Online.

CEMRACS 2019

Moreover, the author participated in the research session of CEMRACS 2019[5]. CEMRACS (*Centre d'Eté Mathématique de Recherche Avancée en Calcul Scientifique* - Summer Center in Mathematics and Advanced Research on Scientific Computing) is a scientific event in Applied Mathematics organized by the French Applied and Industrial Mathematical Society (SMAI), gathering young and senior scientists to work in research projects linked to a given theme. The 2019 edition of CEMRACS focused on geophysical fluids and gravity flows, and the author worked in a project on the simulation of landslide-generated tsunamis, in collaboration with Emmanuel Audusse (Université Paris 13), Louis Emerald (Université de Rennes), Philippe Heinrich (CEA), Alexandre Paris (Université de Pau) and Martin Parisot (Inria, CARDAMOM team). The project aimed to compare the performance of shallow water and Boussinesq models in simulating the generation and propagation of waves formed by landslides, w.r.t. to reference simulations provided by the 3D Navier-Stokes equations. In a second moment, the inverse problem, *i.e.* the recovering of a landslide displacement from a given generated wave, is tackled via optimization techniques. This work is presented in the following published proceedings article:

- Audusse, E., Caldas Steinstraesser, J.G., Emerald, L., Heinrich, P., Paris, A., and Parisot, M. (2021). Comparison of models for the simulation of landslide generated tsunamis. *ESAIM: ProcS*, 70:14–30.

1.6 Summary of the thesis

This thesis is organized as follows:

Chapter 2 is devoted to a review on shallow water models at different scales, namely the classical shallow water equations (the fine model) and the porosity-based shallow water equations (the coarse one). After the introduction of the main concepts of volume averaging for model upscaling, we present a historical evolution of some of the main porosity-based models, ranging from models in which a single porosity parameter is defined to models defining a dual set of porosities and able to more accurately represent flow anisotropies and wave velocities. An explicit finite-volume (FV) discretization of the classical and porosity-based models is presented, and we introduce a global-in-space form of the FV scheme that will be useful to the formulation of reduced-order models in the next chapter. A numerical example of a flow on a fictitious urban zone is presented for illustrating the accuracies and computational costs of the classical and porosity-based models.

In **Chapter 3**, we present a review of the parareal method and some of its variants for tackling hyperbolic problems. Basic definitions, properties and challenges of the original or classical parareal

[5] http://smai.emath.fr/cemracs/cemracs19/

method are presented. We then review the parareal variant using reduced-order models (the ROM-based parareal method), with ROMs formulated using the combined POD-EIM method. A speedup estimation of the parareal methods is presented and interpreted, providing guidelines for their efficient application. A set of numerical tests with increasing complexity is introduced for comparing the classical and ROM-based parareal methods, in terms of convergence, stability and computational cost, and for identifying factors that may influence their performance. We remark that the test cases in this and the following two chapters consider a coupling between fine and coarse discretizations of classical shallow water equations, *i.e.* at this moment we still do not consider the porosity-based models.

Chapter 4 focuses on the improvement of the ROM-based parareal method. After identifying some factors that limit the quality of the formulated reduced-order models, and thus of the parareal simulations in which they are used, we propose three modifications for the method: the enrichment of the input snapshots sets used for the model reduction, with extra snapshots whose computation does not require any additional computational cost, but make the model reduction more expensive; the formulation of local-in-time ROMs, an approach known as principal interval decomposition (PID), for better capturing local phenomena; and the incorporation of an adaptive approach recently proposed in the framework of the classical parareal method, and consisting in the use of progressively refined propagators through parareal iterations. In a first moment, these modification are studied and evaluated alone, independently of each other, by using the same test cases as in Chapter 3. A combined application of the proposed modifications is studied in a second moment.

In **Chapter 5**, we study the behaviour of the classical and ROM-based parareal methods for solving problems defined in larger temporal domains, for which convergence and stability are more challenging. We compare their performances under two strategies: firstly, by increasing the length of the time slices dividing the temporal domain; secondly, by performing local-in-time parareal simulations, instead of a global one. This study is still performed using numerical tests coupling the classical shallow water equations at different scales.

In **Chapter 6**, we study the application of the parareal methods for coupling the classical and porosity-based shallow water models, for the simulation of urban floods. We begin by identifying some challenges that may arise due to the introduction of obstacles in the domain and the resulting more complex flow profile (*e.g.* with discontinuities of the velocity fields). We then define a set of numerical tests simulating a flow in a fictitious urban zone. By considering various discretizations of the problem (mesh and time step sizes), we investigate the influence of the coarsening between the fine and coarse models on the performance of the classical and ROM-based parareal methods. An investigation of the achieved convergence and numerical speedups in function of the chosen parareal configurations is also performed. After this initial study, we tackle a more challenging problem, defined in a larger temporal domain, by solving the numerical test presented in Chapter 2 for illustrating the classical and shallow water models.

A conclusion is presented in **Chapter 7**. Some details of the parallel implementation, model reduction procedures and spatial interpolation are provided respectively in **Appendices A, B** and **C**.

CHAPTER 2

SHALLOW WATER MODELS AT DIFFERENT SCALES FOR THE SIMULATION OF URBAN FLOODS

Contents

2.1 Introduction **11**

2.1.1 Notation 12

2.2 The classical shallow water equations (SWE) **13**

2.3 An overview on model upscaling **14**

2.4 The Single Porosity (SP) model **15**

2.4.1 The porosity parameter 15

2.4.2 Derivation of the SP shallow water equations 16

2.4.3 Final formulation 18

2.5 The Multiple Porosity (MP) model **18**

2.6 The Integral Porosity (IP) model **19**

2.7 The Dual Integral Porosity (DIP) model **21**

2.8 Other developments and porosity models **22**

2.9 Hyperbolicity of the shallow water models **23**

2.9.1 Classical SWE 23

2.9.2 SP model 24

2.9.3 IP model 24

2.9.4 DIP model 24

2.10 Finite volumes discretization **25**

2.10.1 Notations 25

2.10.2 Discretization 27

2.10.3 Global-in-space matrix formulation of the finite volume scheme 28

2.11 Numerical examples **31**

2.12 Conclusion of the chapter **34**

2.1 Introduction

A commonly faced issue when modeling urban floods using shallow water equations is the high computational cost for their numerical simulation. Accurate simulations w.r.t. experimental and/or analytical solutions require a precise representation of the urban geometry (streets and buildings), which is achieved by using very fine computational meshes. This fact, summed to the Courant–Friedrichs–Lewy (CFL) stability condition that arises in explicit temporal discretizations and imposes very small time steps due to the small mesh sizes, make it impractical to perform numerical simulation covering large areas and large time periods.

Shallow water models with porosity have gained popularity in the past two decades as an alternative for overcoming this difficult. In a very general description, they upscale the classical shallow water equations and introduce the concept of porosity, for taking into account the presence of obstacles in the domain (*i.e.* the reduction of storage areas and exchange sections). Therefore, the computational mesh does not need to represent accurately the obstacles' geometry and can be much coarser compared to the mesh required by the classical model. As a consequence, time steps are also allowed to be much larger, and porosity-based models are able to reduce the the computational time by two to three orders of magnitude, when compared to the classical shallow water equations (Guinot, 2017).

Porosity-type parameters were firstly introduced in the shallow water equations by Defina et al. (1994), as a "*storativity coefficient* analogous to the one used in groundwater hydraulics" for taking into account dry and wetted parts of the computational domain, and, in a more general framework, by Defina (2000), Hervouet et al. (2000) (in which the term "porosity" is introduced) and Guinot and Soares-Frazão (2006). In these approaches, known as Single Porosity (SP) models, a single isotropic porosity parameter is introduced in the differential form of the shallow water equations. Since then, many modeling and numerical improvements have been proposed and more complex porosity models have been formulated: for example, the multiple porosity (MP) model (Guinot, 2012), which divides the domain into regions with different flow properties, defining a porosity for each region and exchange parameters between them; the Integral Porosity (IP) model (Sanders et al., 2008), which, by considering the integral formulation of the equations and distinguishing a storage cell-defined porosity and a conveyance edge-based porosity, is able to take into account anisotropic phenomena; and the Dual Integral Porosity (DIP) model (Guinot et al., 2017), that improves the IP model with a better resolution of wave propagation speeds.

In this chapter, we present an overview of the main porosity-based shallow water models. The classical shallow water equations are briefly recalled in Section 2.2. As an introduction to the porosity-based models, we present an overview of upscaling techniques in Section 2.3. In Section 2.4, we make a detailed description of the SP approach, for illustrating the derivation of porosity-based models. For the other models (MP, IP, and DIP, respectively in Sections 2.5, 2.6 and 2.7), their main concepts, motivations and properties are presented. Note that the porosity-based models are not presented in a strict chronological order: we first present those formulated in a differential form (SP and MP), and then those formulated in an integral one (IP and DIP), for keeping a more continuous sequence of ideas. Other developments and porosity approaches, including models designed for applications others than the simulation of urban floods, are briefly presented in Section 2.8. A discussion on the hyperbolicity of the classical and porosity-based shallow water models is proposed in Section 2.9, since it is a main issue for their simulation using parallel-in-time methods, as discussed in the next chapter. A finite volume (FV) discretization of the models is presented in Section 2.10. Finally, some numerical examples for illustrating the results of the SWE, SP and DIP models are presented in Section 2.11, and a conclusion is given in Section 2.12.

2.1.1 Notation

Let us introduce some basic notation used along this and the following chapters. We denote by $\boldsymbol{x} = (x, y)$ the two horizontal coordinates. The conserved variables of the classical and porosity-based shallow water equations are arranged in the vector

$$\boldsymbol{U} = \boldsymbol{U}(\boldsymbol{x}, t) := \begin{pmatrix} h \\ hu_x \\ hu_y \end{pmatrix}$$

where $h = h(\boldsymbol{x}, t)$ is the water depth and u_x and u_y are respectively the $x-$ and $y-$components of the velocity vector $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t) = \left(u_x,\ u_y\right)^T$. Therefore, hu_x and hu_y are the unit discharges (volumetric discharge per unit of length) respectively in the $x-$ and $y-$directions. The intensity of the gravitational acceleration and the fluid density are denoted respectively by g and ρ. The bottom elevation is considered constant in time and defined as $z_b = z_b(\boldsymbol{x})$ w.r.t. an arbitrary reference $z = 0$ along the vertical coordinate. The free surface elevation is defined as $\eta = \eta(\boldsymbol{x}, t) = z_b(\boldsymbol{x}) + h(\boldsymbol{x}, t)$. Some of these definitions are presented along the $x - z$ plan in Figure 2.1.

In order to formulate a unique and general finite volume discretization in Section 2.10, valid for all the

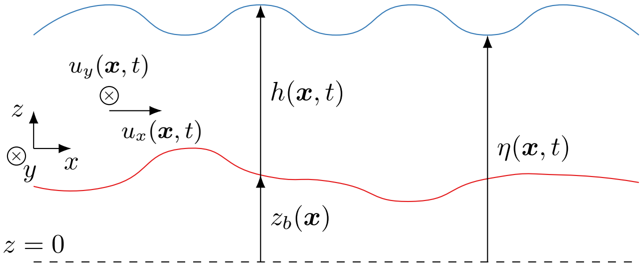


Figure 2.1: Schematic representation in the $x-z$ plan of some definitions used in the classical and porosity-based shallow water models. The red and the blue lines represent respectively the bottom elevation and the free surface elevation.

porosity-based models and also for the classical shallow water equations (differing only on the formulation of the numerical fluxes), all the models are presented in the conservative form

$$\frac{\partial}{\partial t}\left(\underline{\boldsymbol{U}}^{\text{MODEL}}(\boldsymbol{x},t)\right)+\frac{\partial}{\partial x}\left(\boldsymbol{F}^{\text{MODEL}}(\boldsymbol{U}(\boldsymbol{x},t))\right)+\frac{\partial}{\partial y}\left(\boldsymbol{G}^{\text{MODEL}}(\boldsymbol{U}(\boldsymbol{x},t))\right)=\boldsymbol{S}^{\text{MODEL}}(\boldsymbol{U}(\boldsymbol{x},t)) \quad (2.1)$$

where $\underline{\boldsymbol{U}}^{\text{MODEL}}$ is the solution $\boldsymbol{U}$ possibly scaled by porosity coefficients, $\boldsymbol{F}^{\text{MODEL}}$ and $\boldsymbol{G}^{\text{MODEL}}$ are flux terms and $\boldsymbol{S}^{\text{MODEL}}$ is a source term. Finally, “MODEL” can be “SWE”, “SP”, “MP”, “IP” or “DIP”, even if the validity of a differential formulation for the porosity model is questioned in the integral models (IP and DIP), as discussed in Section 2.6. In any case, the FV discretization for the proposed equations is the same that would be obtained for the integral formulation.

2.2 The classical shallow water equations (SWE)

First presented by Barré de Saint-Venant (1871), the shallow water equations (SWE), also known as Saint-Venant equations, is a two-dimensional depth-averaged approximation of the Navier-Stokes equations widely used in many fluid dynamics applications, *e.g.* river hydrodynamics (Churuksaeva and Starchenko, 2015), transport of polluants (Audusse and Bristeau, 2003), nearshore ocean modeling (Brocchini and Dodd, 2008), erosional dambreak flows (Fraccarollo and Capart, 2002) and atmospheric dynamics (Verkley, 2009).

The SWE equations are obtained from the Navier-Stokes by assuming an hydrostatic pressure (a derivation is presented by Gerbeau and Perthame (2001)), which implies that the velocity field is constant along the vertical direction. Therefore, it consists of a system of equations depending only on the two horizontal dimensions. In its inviscid form, and following the proposed notation (2.1), the SWE can be written as

$$\frac{\partial}{\partial t}\left(\underline{\boldsymbol{U}}^{\text{SWE}}(\boldsymbol{x},t)\right)+\frac{\partial}{\partial x}\left(\boldsymbol{F}^{\text{SWE}}(\boldsymbol{U}(\boldsymbol{x},t))\right)+\frac{\partial}{\partial y}\left(\boldsymbol{G}^{\text{SWE}}(\boldsymbol{U}(\boldsymbol{x},t))\right)=\boldsymbol{S}^{\text{SWE}}(\boldsymbol{U}(\boldsymbol{x},t)) \quad (2.2)$$

where

$$\underline{\boldsymbol{U}}^{\text{SWE}}=\boldsymbol{U}=\begin{pmatrix} h \\ hu_x \\ hu_y \end{pmatrix}, \qquad \boldsymbol{F}^{\text{SWE}}(\boldsymbol{U})=\begin{pmatrix} hu_x \\ hu_x^2+gh^2/2 \\ hu_xu_y \end{pmatrix},$$
$$\boldsymbol{G}^{\text{SWE}}(\boldsymbol{U})=\begin{pmatrix} hu_y \\ hu_xu_y \\ hu_y^2+gh^2/2 \end{pmatrix}, \qquad \boldsymbol{S}^{\text{SWE}}(\boldsymbol{U})=\begin{pmatrix} 0 \\ S_{0,x}^{\text{SWE}}+S_{f,x}^{\text{SWE}} \\ S_{0,y}^{\text{SWE}}+S_{f,y}^{\text{SWE}} \end{pmatrix}$$

.

The first equation of (2.2) is called the *continuity equation* and the remaining two equations are called respectively the $x-$ and $y-$*momentum equations.* Moreover,

$$S_{0,x}^{\text{SWE}}=-gh\frac{\partial z_b}{\partial x}, \qquad S_{0,y}^{\text{SWE}}=-gh\frac{\partial z_b}{\partial y}$$

are source terms due to topography variations and

$$S_{f,x}^{\mathrm{SWE}} = -ghu_x \left\|\boldsymbol{u}\right\| \left[\frac{1}{K^2 h^{4/3}} + s_x\right], \qquad S_{f,y}^{\mathrm{SWE}} = -ghu_y \left\|\boldsymbol{u}\right\| \left[\frac{1}{K^2 h^{4/3}} + s_y\right]$$

are friction source terms for which we adopt the same formulation proposed by Guinot and Soares-Frazão (2006) for the SP model and presented in detail in Section 2.4.

2.3 An overview on model upscaling

The porosity-based shallow water models are obtained via an upscaling of the classical SWE (2.2). Techniques of upscaling differential equations are widely studied in various applications, *e.g.* groundwater flow (Wen and Gómez-Hernández, 1996), thermal transport and chemical reactions in porous media (Whitaker, 1999; Yang et al., 2015) , composite materials (Otero et al., 2015), elastodynamics (Willis, 1997) and seismology (Capdeville et al., 2020). As discussed by Capdeville et al. (2020), several types of uspcaling methods have been proposed in the literature. As a common goal, they seek a macroscopic, effective description of physical processes characterized by small-scale features. For mentioning two different upscaling approaches, we cite the asymptotic homogenization analysis (Bensoussan et al., 1978) and the volume averaging procedure (Whitaker, 1999).

In the asymptotic homogenization analysis, two spatial variables are introduced, for describing the large and small scales, and the upscaled or homogenized equations are obtained via an asymptotic analysis. As described by Allaire (2010), asymptotic homogenization is “a rigorous version of what is known as averaging” and is equipped with a solid mathematical theory. However, its application to porous media is not straightforward, because assumptions on which most of the homogenization theory relies, such as the periodicity of the medium and large separation between the fine and coarse scales, are not always valid (Auriault et al., 2005). Moreover, Auriault (2005) shows that advection problems in homogeneous media are not homogenizable, due to their dependence on macroscopic boundary conditions.

On the other hand, the volume averaging approach is based on a more physical and intuitive idea. It consists in averaging the governing equations over a given volume, which is supposed to be large enough to provide a representative average but small enough to capture phenomena of interest (Arce et al., 2005). This approach is based on the so-called spatial averaging theorem, formulated independently by (Anderson and Jackson, 1967; Slattery, 1967; Whitaker, 1967). Let Ω be a volume control composed by two phases, α- and β-phase, defined respectively in the subdomains Ω_α and Ω_β of Ω (see Figure 2.2), and ψ_β a quantity defined in Ω_β. The volume averaging theorem is a three-dimensional formulation of the Leibniz integral rule, stating that the integral of a gradient over Ω_β is the sum of the gradient of the integral with an integral along the boundary:

$$\langle \nabla \psi_\beta \rangle = \nabla \langle \psi_\beta \rangle + \frac{1}{|\Omega|} \int_{\Gamma_{\beta,\alpha}} \boldsymbol{n}_{\beta,\alpha} \psi_\beta d\Gamma \tag{2.3}$$

where

$$\langle \cdot \rangle = \frac{1}{|\Omega|} \int_{\Omega_\beta} \cdot \, d\Omega \tag{2.4}$$

is usually known as *phase average*, $\Gamma_{\beta,\alpha}$ is the interface between Ω_α and Ω_β, and $\boldsymbol{n}_{\beta,\alpha}$ is the unit normal vector to $\Gamma_{\beta,\alpha}$ pointing outwards Ω_β. Reviews on the derivation of (2.3) are presented by (Howes and Whitaker, 1985; Whitaker, 1999). Moreover, when the spatial derivatives are replaced by temporal ones, (2.3) is usually referred to as Reynolds transport theorem (Takatsu, 2017).

The application of the volume averaging theorem to a given space-dependent problem provides a spatially smoothed governed equation. As pointed out by Whitaker (1999), this smoothed equation depends explicitly on the interface boundary conditions between the phases, due to the last term of (2.3). The interface integral can only be computed by establishing a relation between the values of ψ_β in Ω_β and $\Gamma_{\beta,\alpha}$, which is called a *closure model.*

As described in details in the following sections, the porosity-based shallow water models upscale the SWE via a volume averaging procedure. The conserved variables and fluxes are averaged for taking into account the presence of obstacles, and thus the availability or not for the flow, which is translated by the introduction of porosity parameters. The averaging procedures and obtained equations take different forms in each of the porosity-based models. It will also be discussed that the notion of closure model is

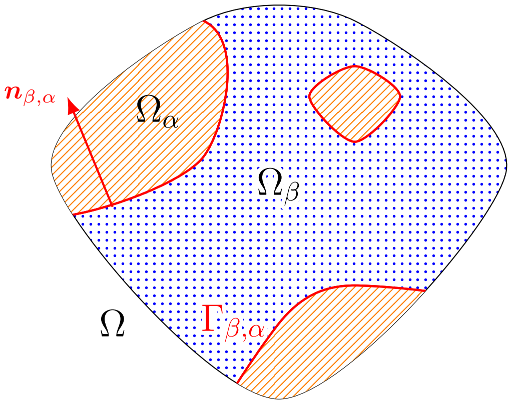


Figure 2.2: Definitions for the volume averaging theorem. The α- and $\beta-$ phases are defined respectively in Ω_α (orange regions) and Ω_β (blue, dotted region). Eq. (2.4) is an average over Ω of quantities defined in Ω_β.

explicitly introduced and developed only in the DIP model. In all precedent porosity-based approaches, a somewhat natural closure model is implicity considered in the derivation of the equations.

Finally, we remark that upscaling can be seen as as a filtering problem (Farmer, 2002), in which the spatial average (2.4) acts as filter. Indeed, a large variety of averaging procedures, alternatively to eq. (2.4), is proposed and studied in the literature and the choice of the one to be used should rely on desired properties of the average fields and equations and attributes of the domain (Wen and Gómez-Hernández, 1996; Davit and Quintard, 2017). Thus, these different average formulations correspond to different filtering procedures. It is also worth noting that these averages are performed only in space; averaging or filtering in time could be possible paths to be explored in the context of upscaling.

2.4 The Single Porosity (SP) model

2.4.1 The porosity parameter

In the single porosity approach for the SWE equations, a spatial porosity parameter $\phi = \phi(\boldsymbol{x})$ is defined for representing the presence of obstacles in the domain, *e.g.* buildings in an urban zone. More precisely, this parameter represents the fraction of the domain available for the storage or passage of the fluid, *i.e.* the fraction of the domain not containing obstacles, as illustrated in Figure 2.3.

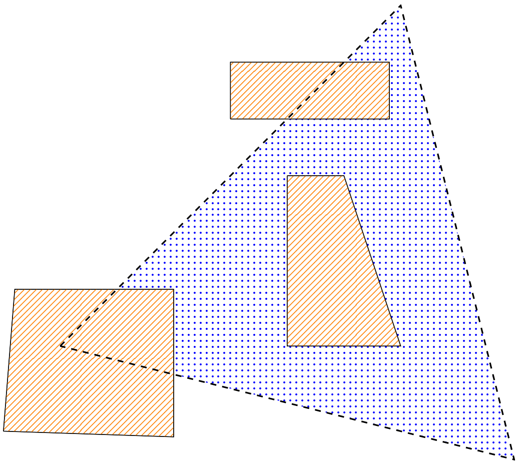


Figure 2.3: Definition of the porosity parameter ϕ. The triangle with dashed borders is a cell of the computational mesh and the orange quadrilaterals with full borders are obstacles (*e.g.* buildings in an urban zone). The blue, dotted area is available for the flow, and ϕ is the fraction this area represents relative to the cell's area.

In the first work developing this approach, Defina et al. (1994) propose a number of formulations for a depth-dependent porosity parameter $\phi = \phi(\boldsymbol{x}, \eta)$ in the case of uneven bathymetries in a given small and finite region Ω of the domain, based on the bottom elevation $z_b(\boldsymbol{x})$, the mean bottom elevation $\overline{z}_b$ in Ω, the highest bottom elevation z_{lim} above $\overline{z}_b$, the free surface elevation $\eta(\boldsymbol{x})$, the area $|\Omega|$ of Ω and the wetted area $|\Omega_{\text{wet}}(\eta)|$, with $\Omega_{\text{wet}}(\eta) := \{\boldsymbol{x} \in \Omega | \eta(\boldsymbol{x}) > z_b(\boldsymbol{x})\}$:

$$\phi(\boldsymbol{x}, \eta) = \frac{|\Omega_{\text{wet}}(\eta)|}{|\Omega|} \tag{2.5a}$$

$$\phi(\boldsymbol{x}, \eta) = \begin{cases} e^{-0.7\left(1 - \dfrac{\eta - \overline{z}_b}{z_{\text{lim}}}\right)}, & \eta - \overline{z}_b < z_{\text{lim}} \\ 1, & \eta - \overline{z}_b \geq z_{\text{lim}} \end{cases} \tag{2.5b}$$

$$\phi(\boldsymbol{x}, \eta) = P(\eta > z_b) \tag{2.5c}$$

The first definition (2.5a) corresponds to the wetted fraction of Ω (as represented in Figure 2.3); the second one (2.5b) is derived from the examination of real topography profiles; and the last one (2.5c) is a statistical approach taking into account the wetting probability. These quantities may vary in time, since they depend on the water depth, but the time-dependance is omited for simplification.

The definition of ϕ as the wetted fraction is presented more formally by Defina (2000) as

$$\phi(\boldsymbol{x}, \eta) = \nu(\boldsymbol{x}, \eta)$$

where ν is the integral over Ω of a phase function φ indicating wet zones:

$$\nu(\boldsymbol{x}, z) = \frac{1}{|\Omega|} \int_{\Omega} \varphi(\boldsymbol{x}, z) d\Omega \tag{2.6a}$$

$$\varphi(\boldsymbol{x}, z) = \begin{cases} 1, & z > z_b(\boldsymbol{x}) \\ 0, & z \leq z_b(\boldsymbol{x}) \end{cases} \tag{2.6b}$$

i.e. ν is the phase average (2.4) of ϕ.

These concepts are introduced in the modeling of urban floods by Hervouet et al. (2000) and Guinot and Soares-Frazão (2006). In these works, the porosity parameter $\phi = \phi(\boldsymbol{x})$ is considered independent of the water depth, which corresponds to consider the emerged structures to have a constant horizontal transverse section along the vertical axis. Moreover, since the urban geometry is considered constant in time, so is the porosity parameter. Alternatively to the definition of the porosity parameter as the fraction of the urban zone, in numerical tests presented by (Guinot and Soares-Frazão, 2006) the porosity parameter is computed as the percentage of the cross-section normal to the flow direction that is available for the fluid, representing the fraction of this section not covered by buildings, or contractions of a channel's width in the case of flows in a channel.

We present the SP model by considering the formulation proposed by Guinot and Soares-Frazão (2006). They validated the model by comparing its results with analytical or semi-analytical solutions of the classical SWE in one- and two-dimensional dambreak problems, and also with experimental data. A more detailed experimental validation is presented by Soares-Frazão et al. (2008).

2.4.2 Derivation of the SP shallow water equations

In the following, we briefly present the derivation of the single porosity-based shallow water equations as proposed by Guinot and Soares-Frazão (2006).

The continuity equation

The continuity equation is obtained by defining the fluid volume in an infinitesimal rectangular control volume $[x_0, x_0 + \delta x] \times [y_0, y_0 + \delta y]$ by

$$V = \int_{x_0}^{x_0+\delta x} \int_{y_0}^{y_0+\delta y} (\phi h)(\boldsymbol{x}) dy dx$$

and by imposing the volume balance over this control volume (balance between the volume variation in time and the volumetric fluxes across the boundaries). The fluxes are defined for the four boundaries of the infinitesimal control volume and read, *e.g.* for the western boundary $x = x_0$,

$$F_W = \int_{y_0}^{y_0+\delta y} (\phi h u_x)(x_0, y) dy$$

By taking the limits $\delta x, \delta y \to 0$ in the volume balance, the continuity equation reads

$$\frac{\partial}{\partial t}(\phi h) + \frac{\partial}{\partial x}(\phi h u_x) + \frac{\partial}{\partial y}(\phi h u_y) = 0 \tag{2.7}$$

The momentum equations

The momentum equations in the horizontal directions are derived analogously. The total $x-$ and $y-$momentum in the control volume are defined respectively by

$$M_x = \int_{x_0}^{x_0+\delta x} \int_{y_0}^{y_0+\delta y} \phi h u_x(\boldsymbol{x}) dy dx, \qquad M_y = \int_{x_0}^{x_0+\delta x} \int_{y_0}^{y_0+\delta y} \phi h u_y(\boldsymbol{x}) dy dx$$

In the case of the $x-$momentum equation, four forces act on the control volume:

- The pressure force on the western and eastern boundaries; for example, for the western one:

$$P_W = \frac{\rho g}{2} \int_{y_0}^{y_0+\delta y} (\phi h^2)(x_0, y) dy$$

- A reaction due to the variation of the porosity in the $x-$direction:

$$W_x = \frac{\rho g}{2} \int_{x_0}^{x_0+\delta x} \int_{y_0}^{y_0+\delta y} \left(\frac{\partial \phi}{\partial x} h \right)(\boldsymbol{x}) dy dx$$

- A reaction due to the variation of the topography in the $x-$direction:

$$B_x = -\rho g \int_{x_0}^{x_0+\delta x} \int_{y_0}^{y_0+\delta y} \left(\phi h \frac{\partial z_b}{\partial x} \right)(\boldsymbol{x}) dy dx$$

- A friction force, represented by a classical Strickler law with a Strickler coefficent K:

$$R_x = -\rho g h \int_{x_0}^{x_0+\delta x} \int_{y_0}^{y_0+\delta y} \left(\frac{\|\boldsymbol{u}\|}{K^2 h^{4/3}} \phi u_x \right)(\boldsymbol{x}) dy dx$$

  where $\|\boldsymbol{u}\| = \sqrt{u_x^2 + u_y^2}$. Other friction formulations (*e.g.* Manning, Chéey) could also be considered (Guinot and Soares-Frazão, 2006).

Proceeding as in the derivation of the continuity equation, we impose the momentum balance in the $x-$direction, by summing up the moment fluxes and the force contributions, and we take the limits $\delta x, \delta y \to 0$ for obtaining the $x-$momentum equation

$$\frac{\partial}{\partial t}(\phi h u_x) + \frac{\partial}{\partial x}\left(\phi \left(h u_x^2 + \frac{g h^2}{2} \right) \right) + \frac{\partial}{\partial y}(\phi h u_x u_y) = S_{0,x}^{\text{SP}} + S_{f,x}^{\text{SP}} \tag{2.8}$$

A similar procedure leads to the $y-$momentum equation

$$\frac{\partial}{\partial t}(\phi h u_y) + \frac{\partial}{\partial x}(\phi h u_x u_y) + \frac{\partial}{\partial y}\left(\phi \left(h u_y^2 + \frac{g h^2}{2} \right) \right) = S_{0,y}^{\text{SP}} + S_{f,y}^{\text{SP}} \tag{2.9}$$

In equations (2.8) and (2.9), we have the source terms

$$S_{0,x}^{\text{SP}} = -\phi g h \frac{\partial z_b}{\partial x} + g \frac{h^2}{2} \frac{\partial \phi}{\partial x}, \qquad S_{0,y}^{\text{SP}} = -\phi g h \frac{\partial z_b}{\partial y} + g \frac{h^2}{2} \frac{\partial \phi}{\partial y}$$

$$S_{f,x}^{\mathrm{SP}} = -\phi g h u_x \left\|\boldsymbol{u}\right\| \left[\frac{1}{K^2 h^{4/3}} + s_x\right], \qquad S_{f,y}^{\mathrm{SP}} = -\phi g h u_y \left\|\boldsymbol{u}\right\| \left[\frac{1}{K^2 h^{4/3}} + s_y\right]$$

where the head loss coefficients s_x and s_y account for singular head losses, respectively in the $x-$ and $y-$directions, due to singularities of the urban geometry (see Guinot and Soares-Frazão (2006, Appendix B) for a justification).

2.4.3 Final formulation

Grouping the continuity equation (2.7) and the momentum equations (2.8) and (2.9), we can write the SP model under the form (2.1) as

$$\frac{\partial}{\partial t}\left(\underline{\boldsymbol{U}}^{\mathrm{SP}}(\boldsymbol{x},t)\right) + \frac{\partial}{\partial x}\left(\boldsymbol{F}^{\mathrm{SP}}(\boldsymbol{U}(\boldsymbol{x},t))\right) + \frac{\partial}{\partial y}\left(\boldsymbol{G}^{\mathrm{SP}}(\boldsymbol{U}(\boldsymbol{x},t))\right) = \boldsymbol{S}^{\mathrm{SP}}(\boldsymbol{U}(\boldsymbol{x},t)) \tag{2.10}$$

where

$$\underline{\boldsymbol{U}}^{\mathrm{SP}} = \phi\underline{\boldsymbol{U}}^{\mathrm{SWE}} = \phi\boldsymbol{U} = \begin{pmatrix} \phi h \\ \phi h u_x \\ \phi h u_y \end{pmatrix}, \qquad \boldsymbol{F}^{\mathrm{SP}}(\boldsymbol{U}) = \phi\boldsymbol{F}^{\mathrm{SWE}}(\boldsymbol{U}) = \phi\begin{pmatrix} h u_x \\ h u_x^2 + g h^2/2 \\ h u_x u_y \end{pmatrix},$$

$$\boldsymbol{G}^{\mathrm{SP}}(\boldsymbol{U}) = \phi\boldsymbol{G}^{\mathrm{SWE}}(\boldsymbol{U}) = \phi\begin{pmatrix} h u_y \\ h u_x u_y \\ h u_y^2 + g h^2/2 \end{pmatrix}, \qquad \boldsymbol{S}^{\mathrm{SP}}(\boldsymbol{U}) = \begin{pmatrix} 0 \\ S_{0,x}^{\mathrm{SP}} + S_{f,x}^{\mathrm{SP}} \\ S_{0,y}^{\mathrm{SP}} + S_{f,y}^{\mathrm{SP}} \end{pmatrix}$$

.

Note that, by taking an uniform unitary porosity parameter $\phi \equiv 1$, we recover the classical shallow equations (2.2).

2.5 The Multiple Porosity (MP) model

The multiple porosity (MP) model, introduced by Guinot (2012), proposes a more complex formulation of the porosity approach for the shallow water equations by dividing the computational domain Ω into a number of non-overlapping regions. In each of them, the flow dynamics has different properties, thus being described by different flow variables, equations and porosity parameters. In the following, $\boldsymbol{U}_\rho^{\mathrm{MP}} := \left(h_\rho,\ h_\rho u_{x,\rho},\ h_\rho u_{y,\rho}\right)^T$ denotes the solution in a generic region Ω_ρ, with $\boldsymbol{u}_\rho := \left(u_{x,\rho},\ u_{y,\rho}\right)^T$. Additional source terms are included in the equations for taking into account the mass and moment exchange between the regions.

The proposed regions in the MP model are:

- Isotropic mobile water region (Ω_m): zone available for the water to flow, governed by the classical shallow water equations and occupying a fraction ϕ_m of the domain's area;
- Stagnant water region (Ω_s): region, occupying a fraction ϕ_s in which the flow has zero speed $\boldsymbol{u}_s = \boldsymbol{0}$. Variations of the water depth h_s in this region are exclusively induced by exchanges with Ω_m;
- M anisotropic, connected mobile water regions $\Omega_k, k = 1, \dots, M$, occupying a fraction ϕ_k of the domain, in which the flow is parallel to a preferential direction $\boldsymbol{e}_k \in \mathbb{R}^2$ ($\|\boldsymbol{e}_k\| = 1$) induced by the urban geometry (thus corresponding to the flow in streets). The equations are projected and solved in the direction $\boldsymbol{e}_k$;
- $(M-1)M/2$ non-connected isotropic mobile water regions $\Omega_{(k,p)}, k, p = 1, \dots, M, k \neq p$, occupying a fraction $\phi_{(k,p)}$ of the domain and corresponding to crossroads and intersections between the anisotropic regions Ω_k and Ω_p and exchanging water exclusively with them.
- Buildings region (Ω_b): region in which there is no flow, occupying a fraction $1 - \phi$ of the domain's area, with

$$\phi = \phi_m + \phi_s + \sum_{k=1}^{M}\phi_k + \sum_{\substack{k,p=1,\dots,M \\ k\neq p}}\phi_{(k,p)}$$

Numerical tests presented by Guinot (2012) show that the MP model outperforms the SP one in representing the flow inside urbans zones containing stagnant zones and preferential directions. Therefore, the model stands out for introducing an alternative approach for taking into account anisotropic properties of the flow. As presented in the following two sections, flow anisotropy is a capital motivation in the integral porosity models, first introduced by Sanders et al. (2008).

As in the SP model, the governing equations of the MP model are formulated in a differential form. Following (2.1), we write

$$\frac{\partial}{\partial t}\left(\underline{\boldsymbol{U}}^{\mathrm{MP}}(\boldsymbol{x},t)\right)+\frac{\partial}{\partial x}\left(\boldsymbol{F}^{\mathrm{MP}}(\boldsymbol{U}(\boldsymbol{x},t))\right)+\frac{\partial}{\partial y}\left(\boldsymbol{G}^{\mathrm{MP}}(\boldsymbol{U}(\boldsymbol{x},t))\right)=\boldsymbol{S}^{\mathrm{MP}}(\boldsymbol{U}(\boldsymbol{x},t))$$

where $\underline{\boldsymbol{U}}^{\mathrm{MP}}=\underline{\boldsymbol{U}}^{\mathrm{MP}}_{\rho}$ if $\boldsymbol{x}\in\Omega_\rho$ and the formulation of the fluxes and source terms depends on the region. We refer the reader to Guinot (2012) for details. As said above, we highlight that $\boldsymbol{S}^{\mathrm{MP}}_{\rho}$ defined for each region Ω_ρ includes, besides the friction source term and those arising from topography and porosity variations, an additional source term $\boldsymbol{Q}_\rho$ representing the exchange between different regions.

2.6 The Integral Porosity (IP) model

An integral formulation of the porosity-based shallow water equations is proposed by Sanders et al. (2008). In the continuous level, the presence of obstacles or empty zones in space is represented via a binary density function $\varepsilon(\boldsymbol{x})$, equal to 0 or 1 if $\boldsymbol{x}$ correspond to a void or an obstacle, respectively, analogously to the phase function (2.6b) defined by Defina (2000). In the discrete level, two porosity parameters are defined: a volumetric or storage porosity ϕ_Ω, and an areal or conveyance porosity ϕ_Γ. The former, defined for each cell of the mesh, represents the fraction available to store fluid. The latter, defined for each interface of the mesh, represents the fraction of the interface available for exchanging fluxes. Also analogously to the definition (2.6a) proposed by Defina (2000), these two porosities are defined, respectively to the $i-$th mesh's cell Ω_i (with area A_i) and to to the $j-$th mesh's interface σ_j (with length w_j), as

$$\phi_{\Omega_i}:=\frac{1}{A_i}\int_{\Omega_i}\varepsilon(\boldsymbol{x})d\Omega,\qquad \phi_{\Gamma j}:=\frac{1}{w_j}\int_{\sigma_j}\varepsilon(\boldsymbol{x})d\Gamma$$

An illustration of these definitions is presented in Figure 2.4.

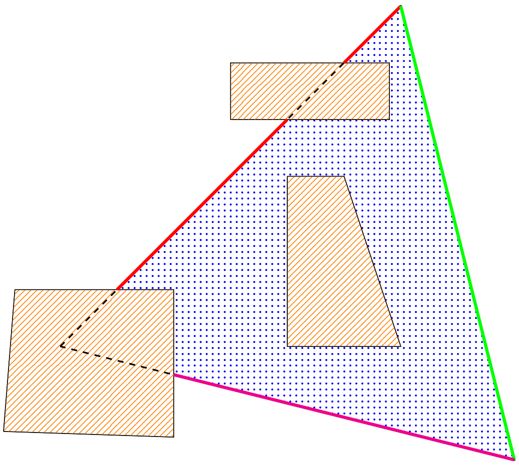


Figure 2.4: Definition of the storage and conveyance porosity parameters, respectively ϕ_Ω and ϕ_Γ. The triangle is a cell of the computational mesh and the orange quadrilaterals are obstacles (*e.g.* buildings in an urban zone). The blue, dotted area and the coloured, full lines are available for the flow. ϕ_Ω is the fraction the dotted area represents relative to the cell's area, and, for each cell's interface, ϕ_Γ is the fraction the coloured, full segment represents relative to the interface's length.

The continuity and momentum equations in the IP model are derived by applying the Reynolds transport theorem and read, for an arbitrary control volume Ω with boundary $\partial\Omega$,

$$\frac{\partial}{\partial t}\int_{\Omega} \varepsilon h d\Omega + \int_{\partial\Omega} \varepsilon h(\boldsymbol{u}\cdot\boldsymbol{n})d\Gamma = 0 \tag{2.11}$$

$$\frac{\partial}{\partial t}\int_{\Omega} \varepsilon h\boldsymbol{u} d\Omega + \int_{\partial\Omega} \varepsilon \left[h\boldsymbol{u}(\boldsymbol{u}\cdot\boldsymbol{n}) + \frac{gh^2}{2}\boldsymbol{n}\right] d\Gamma = \int_{\partial\Omega} \varepsilon \boldsymbol{s}_{\Gamma} d\Gamma + \int_{\Omega} \varepsilon \boldsymbol{s}_{\Omega} d\Omega \tag{2.12}$$

where $\boldsymbol{n}$ is the unit normal vector along $\partial\Omega$ pointing outwards Ω and $\boldsymbol{s}_\Omega$ and $\boldsymbol{s}_\Gamma$ are respectively the domain and boundary source terms, given by

$$\boldsymbol{s}_{\Omega} = -g\left(h - h|_{z_0}\right)\nabla z_b - (C_D^f + C_D^b)\boldsymbol{u}\left\|\boldsymbol{u}\right\|, \qquad \boldsymbol{s}_{\Gamma} = \frac{gh|_{z_0}^2}{2}\boldsymbol{n} \tag{2.13}$$

where η_0 is the mean surface elevation over Ω, $h|_{\eta_0}(\boldsymbol{x}) = \eta_0 - z_b(\boldsymbol{x})$, and C_D^f and C_D^b are respectively a friction and a building drag coefficients.

By distinguishing these two porosity parameters, the IP model is able to take anisotropic effects into account, since the conveyance porosity ϕ_Γ is direction-dependent. Therefore, a better representation of preferential flow directions can be done. Such anisotropic effects may arise due to irregularities of the urban geometry (asymmetric shapes, spaces and alignment of buildings).

As a drawback, the integral formulations of porosity-based shallow water equations present a strong dependence on the mesh configuration, since the conveyance porosity depends on the direction of the interfaces, whereas the isotropic, differential models (SP and MP) are almost insensitive to the mesh (Guinot, 2017). Sanders et al. (2008) state that the position of the mesh nodes should be chosen in order to well capture the anisotropies induced by the buildings configuration; more precisely, the best configuration, called by the authors as *gap-conforming*, is the one in which mesh nodes are placed in the center of the buildings, and the edges between nodes are the gaps between the buildings. Guinot (2017) proposes a detailed consistency study explaining this sensitivity on the mesh in the case of triangular and quadrilateral ones and states that ϕ_Γ should be finely defined in function of the edges orientation, and not only the buildings configuration.

This integral formulation of the porosity-based SWE is motivated by a discussion on the non-existence of a representative elemental volume (REV). By definition, the REV is the smallest size of a subset of a given domain in which the statistical properties of the heterogeneous, porous media can be considered constant, *i.e.* representative of the whole domain (Bear, 1988; Sanders et al., 2008; Guinot, 2012). A REV smaller than the scale in which the porosity is defined is a necessary assumption for the derivation of porosity-based models in differential form (*e.g.* the SP model), since it assumes the continuity of the solution. However, the REV imposes $\phi_\Omega = \phi_\Gamma$, not allowing to represent anisotropic phenomena. Guinot (2012) proposes a more detailed discussion on the REV and assesses via numerical tests that the REV is usually larger than the urban zone, thus theoretically invalidating the differential formulation of porosity-based models; nevertheless, as shown with numerical examples, it does not impede their successful application in practice.

Even if the REV hypothesis is not valid, we consider here the differential form (2.1) for equations (2.11)-(2.12). It can be easily done by using the divergence theorem, taking the limit of an infinitesimal control volume Ω, and considering the solution and flux terms scaled respectively by ϕ_Ω and ϕ_Γ. We then obtain

$$\frac{\partial}{\partial t}\left(\underline{\boldsymbol{U}}^{\mathrm{IP}}(\boldsymbol{x},t)\right) + \frac{\partial}{\partial x}\left(\boldsymbol{F}^{\mathrm{IP}}(\boldsymbol{U}(\boldsymbol{x},t))\right) + \frac{\partial}{\partial y}\left(\boldsymbol{G}^{\mathrm{IP}}(\boldsymbol{U}(\boldsymbol{x},t))\right) = \boldsymbol{S}^{\mathrm{IP}}(\boldsymbol{U}(t)) \tag{2.14}$$

where

$$\underline{\boldsymbol{U}}^{\mathrm{IP}} = \phi_\Omega \boldsymbol{U} = \begin{pmatrix} \phi_\Omega h \\ \phi_\Omega h u_x \\ \phi_\Omega h u_y \end{pmatrix}, \qquad \boldsymbol{F}^{\mathrm{IP}}(\boldsymbol{U}) = \phi_\Gamma \begin{pmatrix} hu_x \\ hu_x^2 + gh^2/2 \\ hu_x u_y \end{pmatrix},$$
$$\boldsymbol{G}^{\mathrm{IP}}(\boldsymbol{U}) = \phi_\Gamma \begin{pmatrix} hu_y \\ hu_x u_y \\ hu_y^2 + gh^2/2 \end{pmatrix}, \qquad \boldsymbol{S}^{\mathrm{IP}}(\boldsymbol{U}) = \begin{pmatrix} 0 \\ S_{0,x}^{\mathrm{IP}} + S_{f,x}^{\mathrm{IP}} \\ S_{0,y}^{\mathrm{IP}} + S_{f,y}^{\mathrm{IP}} \end{pmatrix}$$

.

and

$$S^{\mathrm{IP}}_{0,x} = -\phi_\Omega g \left(h - h|_{\eta_0}\right) \frac{\partial z_b}{\partial x} + \frac{g}{2} \frac{\partial}{\partial x} \left(\phi_\Gamma h|^2_{\eta_0}\right), \qquad S^{\mathrm{IP}}_{0,u} = -\phi_\Omega g \left(h - h|_{\eta_0}\right) \frac{\partial z_b}{\partial y} + \frac{g}{2} \frac{\partial}{\partial y} \left(\phi_\Gamma h|^2_{\eta_0}\right)$$

$$S^{\mathrm{IP}}_{f,x} = -\phi_\Omega g (C^f_D + C^b_D) u_x \left\|\boldsymbol{u}\right\|, \qquad S^{\mathrm{IP}}_{f,y} = -\phi_\Omega g (C^f_D + C^b_D) u_y \left\|\boldsymbol{u}\right\|$$

Note that, by considering the isotropic case $\phi_\Omega \equiv \phi_\Gamma$ in equation (2.14), one recovers the SP model (2.10) (except for minor differences in the formulation of the source terms). Moreover, we highlight that the FV discretization of (2.14) is the same that would be obtained for the integral formulation (2.11)-(2.12).

2.7 The Dual Integral Porosity (DIP) model

The dual integral porosity (DIP) model, proposed by Guinot et al. (2017), improves the IP model by defining a dual set of flow variables, $\boldsymbol{U}_\Omega = \left(h_\Omega,\ h_\Omega u_{x,\Omega},\ h_\Omega u_{y,\Omega}\right)^T$ and $\boldsymbol{U}_\Gamma = \left(h_\Gamma,\ h_\Gamma u_{x,\Gamma},\ h_\Gamma u_{y,\Gamma}\right)^T$, defined respectively for the interior $\Omega \backslash \Gamma$ of a control volume Ω and for its boundary Γ. By replacing the dual variables in equations (2.11)-(2.12), we obtain

$$\frac{\partial}{\partial t} \int_\Omega \varepsilon h_\Omega d\Omega + \int_{\partial\Omega} \varepsilon h_\Gamma (\boldsymbol{u}_\Gamma \cdot \boldsymbol{n}) d\Gamma = 0 \tag{2.15}$$

$$\frac{\partial}{\partial t} \int_\Omega \varepsilon h_\Omega \boldsymbol{u}_\Omega d\Omega + \int_{\partial\Omega} \varepsilon \left[h_\Gamma \boldsymbol{u}_\Gamma (\boldsymbol{u}_\Gamma \cdot \boldsymbol{n}) + \frac{g h_\Gamma^2}{2} \boldsymbol{n} \right] d\Gamma = \int_{\partial\Omega} \varepsilon \boldsymbol{s}_\Gamma d\Gamma + \int_\Omega \varepsilon \boldsymbol{s}_\Omega d\Omega \tag{2.16}$$

where $\boldsymbol{u}_\Omega := \left(u_{x,\Omega},\ u_{y,\Omega}\right)^T$ and $\boldsymbol{u}_\Gamma := \left(u_{x,\Gamma},\ u_{y,\Gamma}\right)^T$.

These two sets of variables are linked via closure models. Firstly, the closure is established by imposing the continuity of the flux Q_{AB} across a segment AB lying on Γ and the flux $Q_{A'B'}$ across a segment $A'B'$, with the same length of AB and obtained by an infinitesimal shift of AB towards the interior of Ω:

$$Q_{AB} = \phi_\Gamma h_\Gamma (\boldsymbol{u}_\Gamma \cdot \boldsymbol{n}) = \phi_\Omega h_\Omega (\boldsymbol{u}_\Omega \cdot \boldsymbol{n}) = Q_{A'B'} \tag{2.17}$$

where $\boldsymbol{n}$ is the unit normal vector to AB and $A'B'$. Eq. (2.17) yields

$$\boldsymbol{u}_\Gamma = \frac{\phi_\Omega}{\phi_\Gamma} \frac{h_\Omega}{h_\Gamma} \boldsymbol{u}_\Omega \tag{2.18}$$

Secondly, concerning the dual water depths, it is assumed that

$$h_\Omega = h_\Gamma \tag{2.19}$$

under the hypothesis that obstructions due to buildings affect mainly the flow velocity, having negligible effects on the surface elevation.

As already discussed, DIP is the first porosity-based model that explicitly introduces a closure model between inner domain and boundary variables. In the previous models, in which this dual set of variables is not proposed, it is implicitly supposed that domain and boundary variables are equivalent.

Guinot et al. (2017) also states that the conveyance porosity cannot be larger than the storage porosity, *i.e.* one must have $\phi_\Gamma \leq \phi_\Omega$, what was not established in the IP model. The authors show that the non-respect of this conditions leads to non-physical behaviours, such as wave propagation speeds larger than the ones obtained in the classical SWE. Moreover, as detailed in Section 2.9, $\phi_\Gamma > \phi_\Omega$ can lead to a loss of hyperbolicity and make the initial value problem ill-posed.

Moreover, Guinot et al. (2017) propose a momentum dissipation mechanism for a more realistic representation of the solution and wave propagation speeds, specially in the case of rising water depths. This mechanism is formulated by defining momentum dissipation parameters $\mu_{xx}, \mu_{xy}, \mu_{yx}$ and μ_{yy} depending on the orientation of the urban layout and that should be calibrated from fine simulations using the classical SWE or experimental results. An anisotropic model is also proposed for the building drag, with coefficients $C^b_{D,xx}, C^b_{D,xy}, C^b_{D,yx}$ and $C^b_{D,yy}$ and a frontal area parameter a.

The final proposed model, comprising these anistropic formulations, already considering the closure relations (2.18) and (2.19) and omitting the sub-index Ω for h_Ω, $u_{x,\Omega}$ and $u_{y,\Omega}$, reads

$$\frac{\partial}{\partial t}\phi_\Omega \boldsymbol{U} + \int_{d\Gamma} \widehat{M}\widehat{F}\boldsymbol{n} d\Gamma = \int_\Omega \varepsilon\left(\boldsymbol{s}_\Omega + \boldsymbol{s}'_\Omega\right) d\Omega + \int_\Gamma \varepsilon \boldsymbol{s}_\Gamma d\Gamma \tag{2.20}$$

with

$$\widehat{F} = \phi_\Gamma \begin{pmatrix} \beta h u & \beta h v \\ \beta h u_x^2 + g\frac{h^2}{2} & \beta h u_x u_y \\ \beta h u_x u_y & \beta h u_y^2 + g\frac{h^2}{2} \end{pmatrix} \qquad \widehat{M} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1-\mu_{xx} & -\mu_{xy} \\ 0 & -\mu_{yx} & 1-\mu_{yy} \end{pmatrix}, \qquad \beta = \frac{\phi_\Omega}{\phi_\Gamma} \geq 1$$

$$\boldsymbol{s}'_\Omega = -\begin{pmatrix} 0 & 0 \\ C^b_{D,xx} & C^b_{D,xy} \\ C^b_{D,yx} & C^b_{D,yy} \end{pmatrix} a h \boldsymbol{u} \left\|\boldsymbol{u}\right\|$$

with $\boldsymbol{s}_\Omega$ and $\boldsymbol{s}_\Gamma$ defined as in the IP model (eq. (2.13)).

We consider here a simplified framework in which the momentum dissipation and the anisotropic building drag effects described above are neglected. Then, as done for the IP model, a differential formulation for (2.15)-(2.16) under the form (2.1) reads

$$\frac{\partial}{\partial t}\left(\underline{\boldsymbol{U}}^{\text{DIP}}(\boldsymbol{x},t)\right) + \frac{\partial}{\partial x}\left(\boldsymbol{F}^{\text{DIP}}(\boldsymbol{U}(\boldsymbol{x},t))\right) + \frac{\partial}{\partial y}\left(\boldsymbol{G}^{\text{DIP}}(\boldsymbol{U}(\boldsymbol{x},t))\right) = \boldsymbol{S}^{\text{DIP}}(\boldsymbol{U}(t)) \tag{2.21}$$

where

$$\underline{\boldsymbol{U}}^{\text{DIP}} = \phi_\Omega \boldsymbol{U} = \begin{pmatrix} \phi_\Omega h \\ \phi_\Omega h u_x \\ \phi_\Omega h u_y \end{pmatrix}, \qquad \boldsymbol{F}^{\text{DIP}}(\boldsymbol{U}) = \phi_\Gamma \begin{pmatrix} \beta h u_x \\ \beta^2 h u_x^2 + g h^2/2 \\ \beta^2 h u_x u_y \end{pmatrix},$$

$$\boldsymbol{G}^{\text{DIP}}(\boldsymbol{U}) = \phi_\Gamma \begin{pmatrix} \beta h u_y \\ \beta^2 h u_x u_y \\ \beta^2 h u_y^2 + g h^2/2 \end{pmatrix}, \qquad \boldsymbol{S}^{\text{DIP}}(\boldsymbol{U}) = \boldsymbol{S}^{\text{IP}}(\boldsymbol{U})$$

.

2.8 Other developments and porosity models

Many works propose more detailed analyses of the porosity models presented in the previous sections. Schubert and Sanders (2012) compare, in terms of quality of the results and complexity of implementation (production of the mesh, parametrization, computational time for the simulation, etc.), four approaches for representing the presence of buildings in the simulation of urban floods: the porosity model (more specifically the IP model); the "buildings-hole" method, in which buildings are not meshed; the "building-resistance" method, in which a large resistance parameter is assigned to cells in buildings regions; and the "building-block" method, in which buildings correspond to elevations in the topography. The authors conclude that flow velocities are very sensitive to the chosen method and list the positive and negative aspects of each one that should be taken into account depending on the objectives and available resources for the simulation.

Kim et al. (2015) performs a detailed study on the sources of errors in the simulation of the shallow-water porosity models w.r.t. real solutions and divide them in three types: structural model errors, due to the hypothesis for deriving the shallow water model; scale errors, originated by the upscaling performed by the porosity models (fine to pore scale); and parametrization errors, coming from the determination of porosity parameters for representing obstacles in the sub-grid scale, in addition to other parameters, *e.g.* head loss and friction coefficients. By comparing numerical simulations to laboratory experiments, the authors conclude that parametrization errors are smaller than structural ones but much larger than scale errors. Therefore, there is still room for improving the porosity models, specially concerning the representation of flow velocities. The authors also confirm the superiority of anisotropic porosity approaches. Detailed experimental validations of porosity-based models are also presented by Soares-Frazão et al. (2008) and Velickovic et al. (2017).

Besides the models presented here, other porosity approaches are proposed in the literature for the simulation of urban floods. Chen et al. (2012a) propose a model considering two types of porosity coefficients, called building covering ration (BCR) and conveyance reduction factor (CRF), defined analogously to the storage ϕ_Ω and conveyance ϕ_Γ porosities in the IP model (in fact, their complementary to the unity), but modifying only the continuity equation. This approach fails to represent situations in which buildings bisect a computational cell, since the internal flow is blocked by the CRF. This problem is overcame in (Chen et al., 2012b) by considering a multi-layer approach, in which the equations are solved in multiple layers for representing individual flowpaths separated by buildings inside a cell.

More recent works propose the formulation of depth-dependent porosities, paving the way to applications others than urban floods. This depth-dependent approach is considered in the initial works in the domain, *e.g.* by Defina (2000), but are not considered in the models presented in this chapter. Özgen et al. (2016a) proposes an integral porosity model depending on the water depth and allowing the full inundation of computational cells. An application to urban floods is presented in Özgen et al. (2016b), in which the porosity parameters are defined using a probability mass function of the fine bathymetry, unresolved in the pore scale. A depth-dependent porosity model based on the DIP approach is proposed by Guinot et al. (2018), for representing meandering channels and submerged buildings, for example. In the discrete level, the authors propose a tabulation of the porosity in function of the water depth using simple predetermined profiles (linear, constant by pieces, etc.). Finally, a depth-dependent SP model with anisotropic resistance and application to agriculture is proposed by Viero and Valipour (2017) for simulating the flow in irrigation and draining channels.

2.9 Hyperbolicity of the shallow water models

We briefly discuss in this section the hyperbolicity of the shallow water models presented along this chapter, since it is a major issue for their simulation using parallel-in-time methods, as described in Chapter 3. We refer to (Lhomme, 2006; Guinot, 2012; Guinot et al., 2017) for more detailed analyses of the hyperbolic structure of the classical and porosity-based shallow water equations. For simplicity, we consider the one-directional form of the governing equations (2.1), without source terms:

$$\frac{\partial}{\partial t}\underline{\boldsymbol{U}}^{\text{MODEL}} + \frac{\partial}{\partial x}\left(\boldsymbol{F}^{\text{MODEL}}(\boldsymbol{U})\right) = \boldsymbol{0} \tag{2.22}$$

where the dependence of $\boldsymbol{U}$ on $\boldsymbol{x}$ and t is omitted for the sake of clarity. A formal derivation of this unidirectional form is presented in Section 2.10 in the context of a finite volume discretization.

By supposing the continuity and differentiability of $\boldsymbol{U}$ and $\underline{\boldsymbol{U}}^{\text{MODEL}}$, we rewrite (2.22) under the non-conservative form

$$\frac{\partial}{\partial t}\underline{\boldsymbol{U}}^{\text{MODEL}} + A^{\text{MODEL}}\frac{\partial}{\partial x}\underline{\boldsymbol{U}}^{\text{MODEL}} = \boldsymbol{0} \tag{2.23}$$

where

$$\begin{aligned} A^{\text{MODEL}} &:= \frac{\partial \boldsymbol{F}^{\text{MODEL}}(\boldsymbol{U})}{\partial \underline{\boldsymbol{U}}^{\text{MODEL}}} \\ &= \frac{\partial \boldsymbol{F}^{MODEL}(\boldsymbol{U})}{\partial \boldsymbol{U}} \frac{\partial \boldsymbol{U}}{\partial \underline{\boldsymbol{U}}^{\text{MODEL}}} \in \mathbb{R}^{3\times 3} \end{aligned}$$

is the Jacobian matrix of $\boldsymbol{F}^{\text{MODEL}}(\boldsymbol{U})$ w.r.t. $\underline{\boldsymbol{U}}^{\text{MODEL}}$. The hyperbolicity of (2.23) can be determined via an eigenvalue analysis, the system being hyperbolic if A^{MODEL} has three distinct real eigenvalues. Let us perform this analysis for the classical SWE and the SP, IP and DIP models. In the following paragraphs, I_3 denotes the identity matrix in $\mathbb{R}^{3\times 3}$.

2.9.1 Classical SWE

For the classical SWE, we have $\underline{\boldsymbol{U}}^{\text{SWE}} = \boldsymbol{U}$ and

$$\boldsymbol{F}^{\text{SWE}}(\boldsymbol{U}) = \begin{pmatrix} hu_x \\ hu_x^2 + gh^2/2 \\ hu_x u_y \end{pmatrix} = \begin{pmatrix} hu_x \\ (hu_x)^2/h + gh^2/2 \\ (hu_x)(hu_y)/h \end{pmatrix}$$

Therefore,

$$\begin{aligned} A^{\mathrm{SWE}} &= \frac{\partial \boldsymbol{F}^{\mathrm{SWE}}(\boldsymbol{U})}{\partial \boldsymbol{U}} \frac{\partial \boldsymbol{U}}{\partial \underline{\boldsymbol{U}}^{\mathrm{SWE}}} = \frac{\partial \boldsymbol{F}^{\mathrm{SWE}}(\boldsymbol{U})}{\partial \boldsymbol{U}} I_3 \\ &= \begin{pmatrix} 0 & 1 & 0 \\ c^2 - u_x^2 & 2u_x & 0 \\ -u_x u_y & u_y & u_x \end{pmatrix} \end{aligned}$$

where $c := \sqrt{gh}$. The eigenvalues of A^{SWE} are

$$\lambda_1^{\mathrm{SWE}} = u_x - c, \qquad \lambda_2^{\mathrm{SWE}} = u_x, \qquad \lambda_3^{\mathrm{SWE}} = u_x + c$$

thus the classical SWE are a hyperbolic system of equations.

2.9.2 SP model

In the SP model, we have $\underline{\boldsymbol{U}}^{\mathrm{SP}} = \phi \boldsymbol{U}$ and $\boldsymbol{F}^{SP}(\boldsymbol{U}) = \phi \boldsymbol{F}^{\mathrm{SWE}}(\boldsymbol{U})$. Therefore,

$$\begin{aligned} A^{\mathrm{SP}} &= \frac{\partial \boldsymbol{F}^{\mathrm{SP}}(\boldsymbol{U})}{\partial \boldsymbol{U}} \frac{\partial \boldsymbol{U}}{\partial \underline{\boldsymbol{U}}^{\mathrm{SP}}} = \frac{\partial \left(\phi \boldsymbol{F}^{\mathrm{SWE}}(\boldsymbol{U})\right)}{\partial \boldsymbol{U}} \frac{\partial \left(\underline{\boldsymbol{U}}^{\mathrm{SP}}/\phi\right)}{\partial \underline{\boldsymbol{U}}^{\mathrm{SP}}} \\ &= \phi \frac{\partial \boldsymbol{F}^{\mathrm{SWE}}(\boldsymbol{U})}{\partial \boldsymbol{U}} \frac{1}{\phi} I_3 \\ &= A^{\mathrm{SWE}} \end{aligned}$$

then A^{SP} has the same eigenvalues as A^{SWE}:

$$\lambda_1^{\mathrm{SP}} = u_x - c, \qquad \lambda_2^{\mathrm{SP}} = u_x, \qquad \lambda_3^{\mathrm{SP}} = u_x + c$$

and the SP model is also hyperbolic.

2.9.3 IP model

For the IP model, $\underline{\boldsymbol{U}}^{\mathrm{IP}} = \phi_\Omega \boldsymbol{U}$ and $\boldsymbol{F}^{\mathrm{IP}} = \phi_\Gamma \boldsymbol{F}^{\mathrm{SWE}}$, such that

$$\begin{aligned} A^{\mathrm{IP}} &= \frac{\partial \boldsymbol{F}^{\mathrm{IP}}(\boldsymbol{U})}{\partial \boldsymbol{U}} \frac{\partial \boldsymbol{U}}{\partial \underline{\boldsymbol{U}}^{\mathrm{IP}}} = \frac{\partial \left(\phi_\Gamma \boldsymbol{F}^{\mathrm{SWE}}(\boldsymbol{U})\right)}{\partial \boldsymbol{U}} \frac{\partial \left(\underline{\boldsymbol{U}}^{\mathrm{IP}}/\phi_\Omega\right)}{\partial \underline{\boldsymbol{U}}^{\mathrm{IP}}} \\ &= \phi_\Gamma \frac{\partial \boldsymbol{F}^{\mathrm{SWE}}(\boldsymbol{U})}{\partial \boldsymbol{U}} \frac{1}{\phi_\Omega} I_3 \\ &= \frac{1}{\beta} A^{\mathrm{SWE}} \end{aligned}$$

with $\beta = \phi_\Omega/\phi_\Gamma > 0$. The eigenvalues of A^{IP} read

$$\lambda_1^{\mathrm{IP}} = \frac{1}{\beta}(u_x - c), \qquad \lambda_2^{\mathrm{IP}} = \frac{1}{\beta} u_x, \qquad \lambda_3^{\mathrm{IP}} = \frac{1}{\beta}(u_x + c)$$

and the IP model is hyperbolic.

2.9.4 DIP model

Finally, in the DIP model we have $\underline{\boldsymbol{U}}^{\mathrm{DIP}} = \phi_\Omega \boldsymbol{U}$ and

$$\boldsymbol{F}^{\mathrm{DIP}}(\boldsymbol{U}) = \phi_\Gamma \begin{pmatrix} \beta h u_x \\ \beta^2 h u_x^2 + gh^2/2 \\ \beta^2 h u_x u_y \end{pmatrix} = \phi_\Gamma \begin{pmatrix} \beta h u_x \\ \beta^2 (hu_x)^2/h + gh^2/2 \\ \beta^2 (hu_x)(hu_y)/h \end{pmatrix}$$

then

$$
\begin{aligned}
A^{\mathrm{DIP}} &= \frac{\partial \boldsymbol{F}^{\mathrm{DIP}}(\boldsymbol{U})}{\partial \boldsymbol{U}} \frac{\partial \boldsymbol{U}}{\partial \underline{\boldsymbol{U}}^{\mathrm{DIP}}} = \\
&= \phi_\Gamma \begin{pmatrix} 0 & \beta & 0 \\ c^2 - \beta^2 u_x^2 & 2\beta^2 u_x & 0 \\ -\beta^2 u_x u_y & \beta^2 u_y & \beta^2 u_x \end{pmatrix} \frac{\partial \left(\underline{\boldsymbol{U}}^{\mathrm{DIP}} / \phi_\Omega \right)}{\partial \underline{\boldsymbol{U}}^{\mathrm{DIP}}} \\
&= \begin{pmatrix} 0 & 1 & 0 \\ c^2/\beta - \beta u_x & 2\beta u_x & 0 \\ -\beta u_x u_y & \beta u_y & \beta u_x \end{pmatrix}
\end{aligned}
$$

whose eigenvalues read

$$
\lambda_1^{\mathrm{DIP}} = \beta u_x - \tilde{c}, \qquad \lambda_2^{\mathrm{DIP}} = \beta u_x, \qquad \lambda_3^{\mathrm{DIP}} = \beta u_x + \tilde{c}
$$

where

$$
\tilde{c} := \beta^{1/2} \left[(\beta - 1) u_x^2 + \frac{c^2}{\beta^2} \right]^{1/2}
$$

Note that, contrary to the previous models, imaginary eigenvalues can appear if $(\beta - 1)u_x^2 + c^2/\beta^2 < 0$, *i.e.* if no restriction is imposed to β, leading to a loss of hyperbolicity of the model. Therefore, as discussed in Section 2.7, the DIP model is always hyperbolic if the storage and conveyance porosity parameters satisfy $\beta = \phi_\Omega / \phi_\Gamma \geq 1$.

2.10 Finite volumes discretization

2.10.1 Notations

The shallow water models described in the previous sections (both the classical and the porosity-based ones) are discretized is this work using a finite volume (FV) scheme, with an explicit Euler discretization in time. The discretization presented hereafter is formulated for the generic model (2.1) and can be implemented for all shallow water models presented here, their difference being on the formulation of the numerical fluxes.

We begin by introducing some notations for the FV discretization.

- We consider a given domain $\Omega \in \mathbb{R}^2$ discretized with a mesh $\mathcal{T}$ containing M cells $\Omega_i,\ i = 1, \ldots, M$, possibly intersecting only on their boundaries $\partial\Omega_i$ and such that $\bigcup_{i=1}^{M} \Omega_i = \Omega$;
- Each cell Ω_i is a polygon with an arbitrary number $d_i \geq 3$ of sizes (triangles, quadrilaterals, etc.). It is not required to all cells to have the same value of d_i. The edges of Ω_i are denoted by $e_i^j, j = 1, \ldots, d_i$, such that $\partial\Omega_i = \bigcup_{j=1}^{d_i} e_i^j$.
- The area and the barycenter of $\Omega_i \subset \mathcal{T}$ are denoted respectively by $A_i = |\Omega_i|$ and $\boldsymbol{c}_i(x_{c_i}, y_{c_i})$.
- The set of cells' edges lying on $\partial\Omega$ is denoted by $\partial\mathcal{T} = \{e_i^j | i = 1, \ldots, M; j = 1 \ldots d_i;$ and $|e_i^j \cup \partial\Omega| > 0\}$. Let $M_b = |\partial\mathcal{T}|$ the number of such edges.
- For properly taking into account the boundary conditions in the numerical scheme presented in the following, we define a ghost cell for each edge on $\partial\mathcal{T}$ (see Figure 2.5). These cells are numbered from $M+1$ to $M+M_b$ and we set $\widetilde{\mathcal{T}} := \mathcal{T} \cup \{\Omega_{M+1}, \ldots, \Omega_{M+M_b}\}$;
- The set of the indices of neighbours cells to $\Omega_i \subset \mathcal{T}$ is defined as $\mathcal{N}(i) := \{\Omega_j \subset \widetilde{\mathcal{T}} |\ |\partial\Omega_i \cup \partial\Omega_j| > 0,\ j \neq i\}$. This definition include ghost cells neighbour to Ω_i.
- The number of interfaces of the mesh is denoted by I. For each Ω_i and for each $j \in \mathcal{N}(i)$, we denote by $\sigma_{i,j} = \partial\Omega_i \cup \partial\Omega_j$ the interface between the cells Ω_i and Ω_j of $\widetilde{\mathcal{T}}$, with length $w_{i,j} = |\sigma_{i,j}|$;
- The unit normal vector to the interface $\sigma_{i,j}$, pointing outwards the cell Ω_i, is denoted by $\boldsymbol{n}_{i,j} = (n_{i,j}^x, n_{i,j}^y)^T$. From this definition, we have $\boldsymbol{n}_{j,i} = -\boldsymbol{n}_{i,j}$. We also define an unit normal vector $\widetilde{\boldsymbol{n}}_{i,j} = \left(\tilde{n}_{i,j}^x,\ \tilde{n}_{i,j}^y\right)^T$, but unique to each interface, corresponding to a chosen orientation for it

(hereafter, we call Ω_i and Ω_j respectively the "left" and "right" cells w.r.t. $\sigma_{i,j}$). We also define a coordinate system $(\eta_{i,j}, \xi_{i,j})$ attached to the interface. Some of these definitions are illustrated in Figure 2.6.

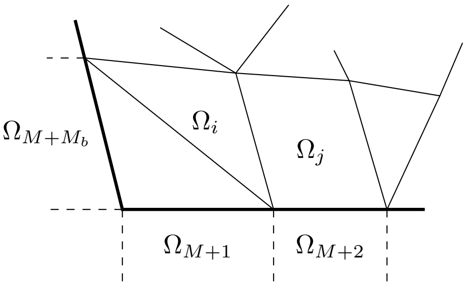


Figure 2.5: Illustration of some elements defining the mesh. Thick, full lines represent a subset of the boundary $\partial\Omega$; thin, full lines represent internal edges; dashed lines represent the interfaces of ghost cells.

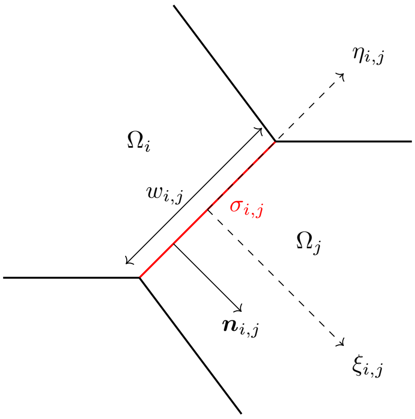


Figure 2.6: Some definitions for the finite volume discretization

- Finally, by indexing the interfaces of $\mathcal{T}$ from 1 to I, we define the maps ψ and ψ' (eq. 2.24 and 2.25, respectively) between the interfaces indices $l = 1, \ldots, I$ and the pairs (i, j) corresponding to the neighbour cells of interface σ_l. We notice that, for each $l = 1, \ldots, I$, $\psi(l)$ is unique and expresses the arbitrarily fixed orientation for the interface. For the second map, we have $\psi'(i, j) = \psi'(j, i)$.

$$\begin{aligned} \psi : \{1, 2, \ldots, I\} &\rightarrow \{1, 2, \ldots, M + M_b\}^2 \\ l &\mapsto (i, j), \qquad \sigma_l = \Omega_i \cap \Omega_j \end{aligned} \tag{2.24}$$

$$\begin{aligned} \psi' : \{1, 2, \ldots, M + M_b\}^2 &\rightarrow \{1, 2, \ldots, I\} \\ (i, j) &\mapsto l, \qquad \sigma_l = \Omega_i \cap \Omega_j \end{aligned} \tag{2.25}$$

Concerning the approximate solution given by the finite volume scheme, we denote by $\boldsymbol{U}_i$ the average value of $\boldsymbol{U} = (h,\ hu_x,\ hu_y)^T$ over the cell $\Omega_i \subset \mathcal{T}$. As presented below, we are also interested in the solution $\widetilde{\boldsymbol{U}} := (h,\ hu,\ hv)^T$ defined on the $(\xi_{i,j}, \eta_{i,j})$-coordinate system attached to the interface $\sigma_{i,j}$, where u and v are the velocity components along directions $\xi_{i,j}$ and $\eta_{i,j}$, respectively. The transformation from the (x, y) to the $(\xi_{i,j}, \eta_{i,j})$ coordinates is made by defining the rotation matrix

$$\widetilde{P}_{i,j} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \tilde{n}^x_{i,j} & -\tilde{n}^y_{i,j} \\ 0 & \tilde{n}^y_{i,j} & \tilde{n}^x_{i,j} \end{pmatrix}$$

such that $\boldsymbol{U} = \widetilde{P}_{i,j}\widetilde{\boldsymbol{U}}$ and $\widetilde{\boldsymbol{U}} = \widetilde{P}^{-1}_{i,j}\boldsymbol{U}$.

Note that $\widetilde{P}_{i,j}$ is unique for the given arbitrary orientation chosen for the interface $\sigma_{i,j}$. For the FV discretization, it will also be useful to define, for each interface $\sigma_{i,j}$, the rotation matrices

$$P_{i,j} = \begin{pmatrix} \boldsymbol{n}_{i,j} \cdot \widetilde{\boldsymbol{n}}_{i,j} & 0 & 0 \\ 0 & n^x_{i,j} & -n^y_{i,j} \\ 0 & n^y_{i,j} & n^x_{i,j} \end{pmatrix}$$

which satisfy $P_{i,j} = -P_{j,i}$.

2.10.2 Discretization

We describe now the discretization of the generic shallow water model (2.1). Instead of discretizing directly this set of equations, we consider for each interface a one-dimensional problem solved in the direction ξ normal to the interface $\sigma_{i,j}$ (we omit hereafter the indices (i,j) for $\xi_{i,j}$). By left-multiplying (2.1) by $\widetilde{P}^{-1}_{i,j}$, we obtain

$$\frac{\partial}{\partial t}\underline{\widetilde{\boldsymbol{U}}}^{\text{MODEL}}(t) + \frac{\partial}{\partial \xi}\widetilde{\boldsymbol{F}}^{\text{MODEL}}(\widetilde{\boldsymbol{U}}(t)) = \widetilde{\boldsymbol{S}}_0^{\text{MODEL}}(\boldsymbol{U}(t)) \tag{2.26}$$

where, for each model,

$$\underline{\widetilde{\boldsymbol{U}}}^{\text{SWE}} = \widetilde{\boldsymbol{U}} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \qquad \underline{\widetilde{\boldsymbol{U}}}^{\text{SP}} = \phi\widetilde{\boldsymbol{U}}, \qquad \underline{\widetilde{\boldsymbol{U}}}^{\text{IP}} = \underline{\widetilde{\boldsymbol{U}}}^{\text{DIP}} = \phi_\Omega \widetilde{\boldsymbol{U}}$$

and the average flux and source terms in the normal direction to the interface are given by

$$\widetilde{\boldsymbol{F}}^{\text{SWE}} = \begin{pmatrix} hu \\ hu^2 + gh^2/2 \\ huv \end{pmatrix}, \qquad \widetilde{\boldsymbol{F}}^{\text{SP}} = \phi\widetilde{\boldsymbol{F}}^{\text{SWE}}, \qquad \widetilde{\boldsymbol{F}}^{\text{IP}} = \phi_\Gamma\widetilde{\boldsymbol{F}}^{\text{SWE}},$$

$$\widetilde{\boldsymbol{F}}^{\text{DIP}} = \phi_\Gamma \begin{pmatrix} \beta hu \\ \beta^2 hu^2 + gh^2/2 \\ \beta^2 huv \end{pmatrix}$$

and

$$\widetilde{\boldsymbol{S}}_0^{\text{MODEL}} = \begin{pmatrix} 0 \\ S^{\text{MODEL}}_{0,\xi} \\ 0 \end{pmatrix}$$

with

$$\begin{aligned} S^{\text{SWE}}_{0,\xi} &= -gh\frac{\partial z_b}{\partial \xi}, \qquad S^{\text{SP}}_{0,\xi} = -\phi gh\frac{\partial z_b}{\partial \xi} + g\frac{h^2}{2}\frac{\partial \phi}{\partial \xi} \\ S^{\text{IP}}_{0,\xi} = S^{\text{DIP}}_{0,\xi} &= -\phi g(h - h|_{\eta_0}\frac{\partial z_b}{\partial \xi} + \frac{g}{2}\frac{\partial}{\partial \xi}\left(\phi_\Gamma h|^2_{\eta_0}\right) \end{aligned} \tag{2.27}$$

Note that friction source terms are not present in (2.27), since they are computed in a separated step, using a time splitting procedure with an exact analytical solution (see (Guinot and Soares-Frazão, 2006) for an example). In this manuscript, the friction forces are neglected. In the case of the MP model, the exchange source terms are also solved separately using an analytical solution (Guinot, 2012).

In what follows, we omit the superscript *MODEL* in order to lighten the notation. By discretizing (2.26) in space with a standard finite volume procedure and returning to the global $(x,y)-$coordinates systems, we obtain

$$\begin{aligned} \frac{d}{dt}\boldsymbol{U}_i(t) = -\frac{1}{A_i}\sum_{j\in\mathcal{N}(i)} w_{i,j} \Big[& P_{i,j}\widetilde{\boldsymbol{F}}_{i,j} \\ & + \widetilde{P}_{i,j}\left(\mathbb{1}_{\{\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j}>0\}}\boldsymbol{S}^L_{i,j} + \mathbb{1}_{\{\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j}<0\}}\boldsymbol{S}^R_{i,j}\right)\Big], \qquad i = 1,\ldots,M \end{aligned} \tag{2.28}$$

where $\mathbb{1}_{\{\text{condition}\}}$ is the indicator function for a given condition, defined as

$$\mathbb{1}_{\{\text{condition}\}} = \begin{cases} 1, & \text{if condition is true} \\ 0, & \text{else} \end{cases}$$

and

$$\widetilde{\boldsymbol{F}}_{i,j} = \begin{pmatrix} \widetilde{F}^{[1]}_{i,j} \\ \widetilde{F}^{[2]}_{i,j} \\ \widetilde{F}^{[3]}_{i,j} \end{pmatrix}, \qquad \widetilde{\boldsymbol{S}}^{L}_{i,j} = \begin{pmatrix} 0 \\ \Delta\widetilde{S}^{L}_{i,j} \\ 0 \end{pmatrix}, \qquad \widetilde{\boldsymbol{S}}^{R}_{i,j} = \begin{pmatrix} 0 \\ \Delta\widetilde{S}^{R}_{i,j} \\ 0 \end{pmatrix} \tag{2.29}$$

are respectively the average flux in the normal direction to $\sigma_{i,j}$ and the source term contributions to the "left" (L) and "right" (R) cells w.r.t. $\sigma_{i,j}$ (for the chosen arbitrary orientation), and can be determined using an approximate Riemann solver. We refer to Guinot and Soares-Frazão (2006) and Guinot et al. (2017) for details on the expressions for the numerical fluxes $\widetilde{F}^{[l]}_{i,j}$, $l = 1, 2, 3$ and source terms $\Delta\widetilde{S}^{L}_{i,j}$ and $\Delta\widetilde{S}^{R}_{i,j}$. We remark that these notations comprise both internal and boundary interfaces:

$$\widetilde{F}^{[l]}_{i,j} = \begin{cases} \widetilde{F}^{[l],\text{internal}}_{i,j}, & i, j \le M \\ \widetilde{F}^{[l],\text{boundary}}_{i,j}, & i > M \text{ or } j > M \end{cases}$$
$$\Delta\widetilde{S}^{L}_{i,j} = \Delta\widetilde{S}^{R}_{i,j} = 0, \qquad i > M \text{ or } j > M$$

The numerical flux and source terms defined in (2.29) are unique to each interface (*i.e.* , $\widetilde{\boldsymbol{F}}_{i,j} = \widetilde{\boldsymbol{F}}_{j,i}$ and similarly to the source terms). Note that, for a given interface $\sigma_{i,j}$, the contribution of the flux term $\widetilde{\boldsymbol{F}}_{i,j}$ for cells Ω_i and Ω_j have equal absolute value but opposite signs, since $P_{i,j}\widetilde{\boldsymbol{F}}_{i,j} = -P_{j,i}\widetilde{\boldsymbol{F}}_{j,i}$. On the other hand, the source term has different contributions for the two cells, which is translated in (2.28) by the multiplication by $\widetilde{P}_{i,j}$ and the separation into two terms "L" and "R".

Finally, an explicit Euler temporal discretization of (2.28) with a constant time step Δt leads to the final numerical scheme

$$\begin{aligned} \boldsymbol{U}^{n+1}_i = \boldsymbol{U}^n_i - \frac{\Delta t}{A_i} \sum_{j \in \mathcal{N}(i)} w_{i,j} \Big[P_{i,j} \widetilde{\boldsymbol{F}}_{i,j} \\ + \widetilde{P}_{i,j} \left(\mathbb{1}_{\{\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j} > 0\}} \widetilde{\boldsymbol{S}}^{L}_{i,j} + \mathbb{1}_{\{\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j} < 0\}} \widetilde{\boldsymbol{S}}^{R}_{i,j} \right) \Big], \qquad i = 1, \dots, M \end{aligned} \tag{2.30}$$

where $\boldsymbol{U}^n_i$ is an approximation of $\boldsymbol{U}$ in the cell Ω_i at time t_n.

2.10.3 Global-in-space matrix formulation of the finite volume scheme

The proposed discretizations (2.28) and (2.30) are the usual and intuitive presentation of a finite volume scheme, in which the solution in each cell is updated by the sum of contributions from all its neighbour cells. However, for the formulation of the reduced models in Section 3.5, we need to write (2.28) under a matricial, global-in-space form, corresponding of a system of M ordinary differential equations (ODEs).

For that purpose, we recall the maps (2.24) and (2.25) linking the indexes $(i, j) \in \{1, \dots, M + M_b\}^2$ and $l \in \{1, \dots I\}$ of the mesh's interfaces and we define the following vectors containing the numerical flux and source terms for all interfaces

$$\begin{aligned} &\widetilde{\boldsymbol{F}}^{(1)} := \begin{pmatrix} \widetilde{\boldsymbol{F}}_{\psi(1)} \cdot \boldsymbol{e}_1 \\ \vdots \\ \widetilde{\boldsymbol{F}}_{\psi(I)} \cdot \boldsymbol{e}_1 \end{pmatrix}, \qquad \widetilde{\boldsymbol{F}}^{(2)} := \begin{pmatrix} \widetilde{\boldsymbol{F}}_{\psi(1)} \cdot \boldsymbol{e}_2 \\ \vdots \\ \widetilde{\boldsymbol{F}}_{\psi(I)} \cdot \boldsymbol{e}_2 \end{pmatrix}, \\ &\widetilde{\boldsymbol{F}}^{(3)} := \begin{pmatrix} \widetilde{\boldsymbol{F}}_{\psi(1)} \cdot \boldsymbol{e}_3 \\ \vdots \\ \widetilde{\boldsymbol{F}}_{\psi(I)} \cdot \boldsymbol{e}_3 \end{pmatrix}, \qquad \widetilde{\boldsymbol{F}}^{(4)} := \begin{pmatrix} \widetilde{\boldsymbol{S}}^{L}_{\psi(1)} \cdot \boldsymbol{e}_2 \\ \vdots \\ \widetilde{\boldsymbol{S}}^{L}_{\psi(I)} \cdot \boldsymbol{e}_2 \end{pmatrix}, \\ &\widetilde{\boldsymbol{F}}^{(5)} := \begin{pmatrix} \widetilde{\boldsymbol{S}}^{R}_{\psi(1)} \cdot \boldsymbol{e}_2 \\ \vdots \\ \widetilde{\boldsymbol{S}}^{R}_{\psi(I)} \cdot \boldsymbol{e}_2 \end{pmatrix} \end{aligned} \tag{2.31}$$

where $\boldsymbol{e}_i, i = 1,2,3$, are the vectors of the canonical basis of $\mathbb{R}^3$ and $\widetilde{\boldsymbol{F}}^{(l)} \in \mathbb{R}^I, l = 1,\dots,5$. In what follows, $\widetilde{\boldsymbol{F}}^{(l)}_{i,j} := \widetilde{\boldsymbol{F}}^{(l)}_{\psi'(i,j)}$ refers to the $\psi'(i,j)$-th component of $\boldsymbol{F}^{(l)}$.

Similarly, we define the following vectors containing the solution's components in all the mesh's cells:

$$\boldsymbol{U}^{(1)} := \begin{pmatrix} \boldsymbol{U}_1 \cdot \boldsymbol{e}_1 \\ \vdots \\ \boldsymbol{U}_M \cdot \boldsymbol{e}_1 \end{pmatrix}, \qquad \boldsymbol{U}^{(2)} := \begin{pmatrix} \boldsymbol{U}_1 \cdot \boldsymbol{e}_2 \\ \vdots \\ \boldsymbol{U}_M \cdot \boldsymbol{e}_2 \end{pmatrix}, \qquad \boldsymbol{U}^{(3)} := \begin{pmatrix} \boldsymbol{U}_1 \cdot \boldsymbol{e}_3 \\ \vdots \\ \boldsymbol{U}_M \cdot \boldsymbol{e}_3 \end{pmatrix} \tag{2.32}$$

with $\boldsymbol{U}^{(l)} \in \mathbb{R}^M, l = 1,2,3$. Hereafter, $\boldsymbol{U}^{(l)}_i$ denotes the i-th component of $\boldsymbol{U}^{(l)}$.

Using (2.31) and (2.32), we can rewrite (2.28) under the form

$$\begin{aligned}
\frac{d}{dt}\boldsymbol{U}^{(1)}_i(t) &= -\frac{1}{A_i}\sum_{j\in\mathcal{N}(i)} && (\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j})w_{i,j}\widetilde{\boldsymbol{F}}^{(1)}_{i,j} \\
\frac{d}{dt}\boldsymbol{U}^{(2)}_i(t) &= -\frac{1}{A_i}\sum_{j\in\mathcal{N}(i)} && \Big[n^x_{i,j}\widetilde{\boldsymbol{F}}^{(2)}_{i,j} - n^y_{i,j}\widetilde{\boldsymbol{F}}^{(3)}_{i,j} \\
&&& + \tilde{n}^x_{i,j}\mathbb{1}_{\{\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j}>0\}}\widetilde{\boldsymbol{F}}^{(4)}_{i,j} \\
&&& +\tilde{n}^x_{i,j}\mathbb{1}_{\{\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j}<0\}}\widetilde{\boldsymbol{F}}^{(5)}_{i,j}\Big] w_{i,j} \\
\frac{d}{dt}\boldsymbol{U}^{(3)}_i(t) &= -\frac{1}{A_i}\sum_{j\in\mathcal{N}(i)} && \Big[n^y_{i,j}\widetilde{\boldsymbol{F}}^{(2)}_{i,j} + n^x_{i,j}\widetilde{\boldsymbol{F}}^{(3)}_{i,j} \\
&&& + \tilde{n}^y_{i,j}\mathbb{1}_{\{\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j}>0\}}\widetilde{\boldsymbol{F}}^{(4)}_{i,j} \\
&&& +\tilde{n}^y_{i,j}\mathbb{1}_{\{\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j}<0\}}\widetilde{\boldsymbol{F}}^{(5)}_{i,j}\Big] w_{i,j}
\end{aligned} \tag{2.33}$$

which can be arranged in the global-in-space form

$$\begin{aligned}
\frac{d}{dt}\boldsymbol{U}^{(1)} &= B^{(1)}\widetilde{\boldsymbol{F}}^{(1)}(\boldsymbol{U}) \\
\frac{d}{dt}\boldsymbol{U}^{(2)} &= B^{(2)}\widetilde{\boldsymbol{F}}^{(2)}(\boldsymbol{U}) + B^{(3)}\widetilde{\boldsymbol{F}}^{(3)}(\boldsymbol{U}) + \\
&\quad B^{(4,x)}\widetilde{\boldsymbol{F}}^{(4)}(\boldsymbol{U}) + B^{(5,x)}\widetilde{\boldsymbol{F}}^{(5)}(\boldsymbol{U}) \\
\frac{d}{dt}\boldsymbol{U}^{(3)} &= -B^{(3)}\widetilde{\boldsymbol{F}}^{(2)}(\boldsymbol{U}) + B^{(2)}\widetilde{\boldsymbol{F}}^{(3)}(\boldsymbol{U}) + \\
&\quad B^{(4,y)}\widetilde{\boldsymbol{F}}^{(4)}(\boldsymbol{U}) + B^{(5,y)}\widetilde{\boldsymbol{F}}^{(5)}(\boldsymbol{U})
\end{aligned} \tag{2.34}$$

where $B^{(1)}, B^{(2)}, B^{(3)}, B^{(4,x)}, B^{(4,y)}, B^{(5,x)}, B^{(5,y)} \in \mathbb{R}^{M\times I}$ are sparse matrices depending only on properties of mesh (thus needing to be computed only once during a numerical simulation in the case where the mesh is constant in time) and whose component in the i-th row and l-th column is given by

$$[B^{(1)}]_{i,l} = \begin{cases} -(\boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j})\dfrac{w_{i,j}}{A_i}, & j\in\mathcal{N}(i),\ l=\psi'(i,j), \\ 0, & \text{else} \end{cases}$$

$$[B^{(2)}]_{i,l} = \begin{cases} -n^x_{i,j}\dfrac{w_{i,j}}{A_i}, & j\in\mathcal{N}(i),\ l=\psi'(i,j), \\ 0, & \text{else} \end{cases}$$

$$[B^{(3)}]_{i,l} = \begin{cases} n^y_{i,j}\dfrac{w_{i,j}}{A_i}, & j\in\mathcal{N}(i),\ l=\psi'(i,j), \\ 0, & \text{else} \end{cases}$$

$$[B^{(4,x)}]_{i,l} = \begin{cases} n^x_{i,j}\dfrac{w_{i,j}}{A_i}, & j\in\mathcal{N}(i),\ \boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j}>0,\ l=\psi'(i,j), \\ 0, & \text{else} \end{cases}$$

$$[B^{(4,y)}]_{i,l} = \begin{cases} n^y_{i,j}\dfrac{w_{i,j}}{A_i}, & j \in \mathcal{N}(i),\ \boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j} > 0,\ l = \psi'(i,j), \\ 0, & \text{else} \end{cases}$$

$$[B^{(5,x)}]_{i,l} = \begin{cases} n^x_{i,j}\dfrac{w_{i,j}}{A_i}, & j \in \mathcal{N}(i),\ \boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j} < 0,\ l = \psi'(i,j), \\ 0, & \text{else} \end{cases}$$

$$[B^{(5,y)}]_{i,l} = \begin{cases} n^y_{i,j}\dfrac{w_{i,j}}{A_i}, & j \in \mathcal{N}(i),\ \boldsymbol{n}_{i,j}\cdot\tilde{\boldsymbol{n}}_{i,j} < 0,\ l = \psi'(i,j), \\ 0, & \text{else} \end{cases}$$

In a more compact form, we write (2.34) as

$$\frac{d}{dt}\breve{\boldsymbol{U}}(t) = \breve{B}\breve{\tilde{\boldsymbol{F}}}(\breve{\boldsymbol{U}}) \tag{2.35}$$

where

$$\breve{\boldsymbol{U}} := \begin{pmatrix} \boldsymbol{U}^{(1)} \\ \boldsymbol{U}^{(2)} \\ \boldsymbol{U}^{(3)} \end{pmatrix} \in \mathbb{R}^{3M}, \qquad \breve{\tilde{\boldsymbol{F}}} := \begin{pmatrix} \tilde{\boldsymbol{F}}^{(1)} \\ \vdots \\ \tilde{\boldsymbol{F}}^{(5)} \end{pmatrix} \in \mathbb{R}^{5I},$$

$$\breve{B} := \begin{pmatrix} B^{(1)} & 0 & 0 & 0 & 0 \\ 0 & B^{(2)} & B^{(3)} & B^{(4,x)} & B^{(5,x)} \\ 0 & -B^{(3)} & B^{(2)} & B^{(4,y)} & B^{(5,y)} \end{pmatrix} \in \mathbb{R}^{3M\times 5I}$$

This global-in-space form (2.34), being written in a matricial form, with matrices that are precomputed since they depend only on constant-in-time geometric information, provides substantial reductions of the computational cost compared to the local-in-space, loop-based formulation (2.28). As said before, this reformulation has been implemented in the software LEMON-SW2D developed by Inria LEMON team. We remark that, in all results presented in this work, only the global-in-space form (2.34)-(2.35) is used. Notably, in the parareal methods developed and presented in the following chapters, both the fine and coarse models use this less expensive formulation.

Stability condition

Since (2.34) is an explicit scheme in time, it must satisfy a Courant-Friedrich-Lewy (CFL)-type stability condition, imposing a maximum permissible time step. This condition is based on the concept of domain of dependence, defined, for a given point $\boldsymbol{x} \in \Omega$, as the set of points of Ω whose solution at a previous time instant influence the solution at $\boldsymbol{x}$ at the current time, *i.e.* it is the set of departure points whose solution reaches $\boldsymbol{x}$ within a time step Δt. Soares Frazão and Guinot (2007) show that the stability of an explicit FV scheme as (2.30) is guaranteed if, for each cell $\Omega_i \subset \mathcal{T}$ of the computational mesh, the sum of the areas of the domains of dependence of each interface $\sigma_{i,j}, j \in \mathcal{N}(i)$ of Ω_i does not exceed the area A_i of Ω_i.

The solution entering Ω_i through $\sigma_{i,j}$ can be decomposed into three waves travelling with speed $\lambda_k^{\text{MODEL}}, k = 1,2,3$ (the eigenvalues associated to each shallow water model, as presented in Section 2.9). In order to avoid computations to determine which wave component enters the cell, Guinot and Soares-Frazão (2006) and Soares Frazão and Guinot (2007) propose to majorate the domain of dependence $A_{i,j}$ of $\sigma_{i,j}$ by

$$A_{i,j} \leq w_{i,j}\Delta t \tilde{\lambda}^{\text{MODEL}}_{3,(i,j)} \tag{2.36}$$

where $\lambda^{\text{MODEL}}_{3,(i,j)}$ is a majoration of the largest wave speed at interface $\sigma_{i,j}$ and reads, for each shallow water model,

$$\begin{aligned} \tilde{\lambda}^{\text{SWE}}_{3,(i,j)} = \tilde{\lambda}^{\text{SP}}_{3,(i,j)} &= \max\{\|\boldsymbol{u}_i\| + c_i, \|\boldsymbol{u}_j\| + c_j\} \\ \tilde{\lambda}^{\text{IP}}_{3,(i,j)} &= \frac{1}{\beta}\max\{\|\boldsymbol{u}_i\| + c_i, \|\boldsymbol{u}_j\| + c_j\} \\ \tilde{\lambda}^{\text{DIP}}_{3,(i,j)} &= \max\{\beta\,\|\boldsymbol{u}_i\| + \tilde{c}_i, \beta\,\|\boldsymbol{u}_j\| + \tilde{c}_j\} \end{aligned}$$

where $c_i = \sqrt{gh_i}$; h_i and $\boldsymbol{u}_i$ are respectively the water depth and the velocity vector in cell Ω_i, and $\tilde{c}_i = \beta^{1/2}[(\beta - 1)\left\|\boldsymbol{u}_i\right\| + c_i^2/\beta^2]^{1/2}$. Therefore, the domains of dependence $A_{i,j}$ are majorated by considering the largest wave velocity, which is itself majorated by using the velocity norm and the flow variables from the neighbours cells to $\sigma_{i,j}$. Note from (2.36) that $A_{i,j}$ is a rectangle of sides $w_{i,j}$ (the interface length) and $\Delta t \tilde{\lambda}_{3,(i,j)}^{\text{MODEL}}$ (the distance covered by the fastest wave along a time step Δt).

Therefore, for each cell Ω_i, it must be verified

$$\sum_{j \in \mathcal{N}(i)} A_{i,j} \leq A_i \tag{2.37}$$

and the global time step for advancing the numerical solution (2.30) is chosen such that (2.37) holds for every $\Omega_i \subset \mathcal{T}$:

$$\Delta t \leq \min_{i=1,\ldots,M} \frac{A_i}{\sum_{j \in \mathcal{N}(i)} w_{i,j} \tilde{\lambda}_{3,(i,j)}^{\text{MODEL}}} \tag{2.38}$$

2.11 Numerical examples

We present in this section some numerical simulations illustrating the results given by classical, SP and DIP shallow water models, consisting in the simulation of a flow in a fictitious "urban zone". The results of the porosity-based SWE are compared to the ones given by the classical SWE in a finer mesh with smaller time steps (considered as the reference solution), for motivating the work presented in this thesis.

Domain and discretizations

We consider a rectangular domain $\Omega_{\text{total}} = [0, L_x] \times [0, L_y]$, with $L_x = 500$ and $L_y = 45$. Denoting by $\Omega_{\text{buildings}}$ the disjoint domain occupied by the buildings, the computational domain for the classical SWE and the porosity-based models are respectively $\Omega_{\text{SWE}} = \Omega_{\text{total}} \backslash \Omega_{\text{buildings}}$ and $\Omega_{\text{porosity}} = \Omega_{\text{total}}$. Without risk of confusion, both Ω_{SWE} and Ω_{porosity} are denoted by Ω in the following.

The urban zone is defined in the region $\Omega_{\text{urban}} := [200, 295] \times [0, L_y] \subset \Omega$, consisting of a 10×4 Cartesian disposition of homogeneous square-shaped buildings with size $l_x = l_y = 5$, also homogeneously separated by a distance $w_x = w_y = w = 5$ in each direction. A schematic representation of the urban zone is presented in Figure 2.7.

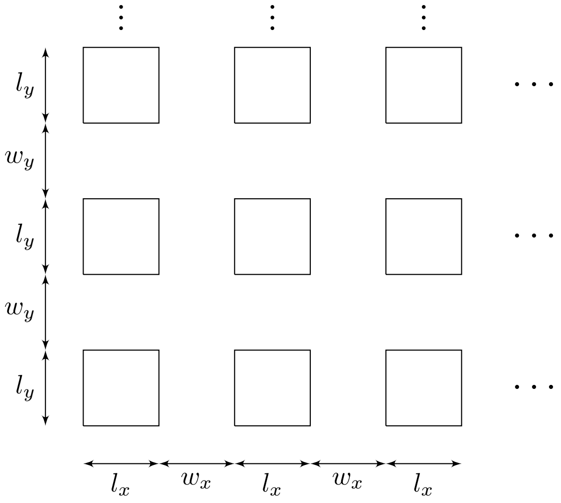


Figure 2.7: Numerical example: schematic representation of the urban zone. Squares represent buildings, with dimensions l_x and l_y respectively in the $x-$ and $y-$ direction, and separated by w_x and w_y in each direction. Only a 3×3 grid of buildings is represented.

The reference simulation of the classical shallow water equations is performed in an unstructured mesh, with mesh diameters varying from 1 (inside the urban zone) to 10 (outside the urban zone) and the buildings represented physically as holes in the mesh (see Figure 2.8). The porosity-based models are solved in a coarser Cartesian mesh with mesh sizes $\Delta x = 12.5$ and $\Delta y = 11.875$ (see Figure 2.9, in which the positions of the buildings, not physically present in the mesh, are indicated with dashed lines). The fine and coarse meshes have respectively 8472 and 160 cells. Concerning the temporal discretization, the classical SWE and the porosity-based models use homogeneous time steps respectively equal to $\delta t = 0.05$ and $\Delta t = 0.5$, chosen in order to verify the stability CFL condition (2.38).

The simulations are run from $t = 0$ to $t = T := 120$.

Porosity fields

As shown in Figure 2.10, simplified porosity fields are adopted for both the SP and DIP simulations, taking a constant value in Ω_{urban}. Outside the urban zone (in $\Omega \backslash \Omega_{\text{urban}}$), all porosities are unitary, $\phi = \phi_\Omega = \phi_\Gamma = 1$. In Ω_{urban}, both ϕ and ϕ_Ω are equal to the fraction of Ω_{urban} not occupied by buildings, *i.e.*

$$\phi(\boldsymbol{x}) = \phi_\Omega(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \in \Omega \backslash \Omega_{\text{urban}} \\ 1 - \dfrac{|\Omega_{\text{buildings}}|}{|\Omega_{\text{urban}}|} = 1 - \dfrac{40 l^2}{|\Omega_{\text{urban}}|} \approx 0.77, & \boldsymbol{x} \in \Omega_{\text{urban}} \end{cases}$$

The conveyance porosity ϕ_Γ is equal to the fraction of the $y-$section not covered by buildings:

$$\phi_\Gamma(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \in \Omega \backslash \Omega_{\text{urban}} \\ \dfrac{5w}{L_y} \approx 0.56, & \boldsymbol{x} \in \Omega_{\text{urban}} \end{cases}$$

The porosity values are attributed to each cell or interface depending if its barycenter lays on Ω_{urban} or $\Omega \backslash \Omega_{\text{urban}}$. Note that the condition $\phi_\Omega \geq \phi_\Gamma$ is satisfied in the whole domain.

Initial and boundary conditions

The initial solution is a lake-at-rest with zero initial water depth:

$$h(\boldsymbol{x}, t = 0) = 0, \qquad u_x(\boldsymbol{x}, t = 0) = u_y(\boldsymbol{x}, t = 0) = 0, \qquad \boldsymbol{x} \in \Omega \tag{2.39}$$

and, for the boundary conditions, an inward unitary flux (volumetric flux per length) is defined on the left boundary $\partial\Omega_{\text{inward}} := \{\boldsymbol{x} \in \partial\Omega | x = 0\}$, a zero water depth is imposed on the right one ($\partial\Omega_{\text{outward}} := \{\boldsymbol{x} \in \partial\Omega | x = L_x\}$) and a null flux (closed boundaries) is defined for the rest of the boundary:

$$\begin{cases} h\boldsymbol{u} \cdot \boldsymbol{n} = 1, & \boldsymbol{x} \in \partial\Omega_{\text{inward}}, t \in [0, T] \\ h = 0, & \boldsymbol{x} \in \partial\Omega_{\text{outward}}, t \in [0, T] \\ h\boldsymbol{u} \cdot \boldsymbol{n} = 0, & \boldsymbol{x} \in \partial\Omega \backslash (\partial\Omega_{\text{inward}} \cup \partial\Omega_{\text{outward}}), t \in [0, T] \end{cases} \tag{2.40}$$

where $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{x})$ is the unit normal vector to $\partial\Omega$.

Figure 2.11 compares the solution of the classical and porosity-based shallow water models at the final time of simulation in the entire spatial domain. The solutions along the slice $y = 22.5$, at $t = T/2$ and $t = T$, are shown in Figure 2.12. The computational times for each simulation are presented in Table 2.1. We observe that both porosity models provide, globally, a relative good approximation for the reference solution with a much smaller computational cost (approximately 100 times smaller). The use of a more complex porosity-based model (DIP instead of SP) does not increase the computational time since both rely on the same numerical scheme and discretization sizes. However, although this good global approximation, small scale phenomena inside the urban zone are not well represented. The better performance of the DIP model, compared to the SP, is clear in the case of the representation of the water depth. The SP model provides a better approximation of the $x-$unit discharge inside the urban zone, but overestimates it on the downstream. Concerning the $y-$unit discharge, both models provide a solution identical to zero, since the porosity fields do not depend on y, thus failing to represent the effects of the flow reflection on the buildings.

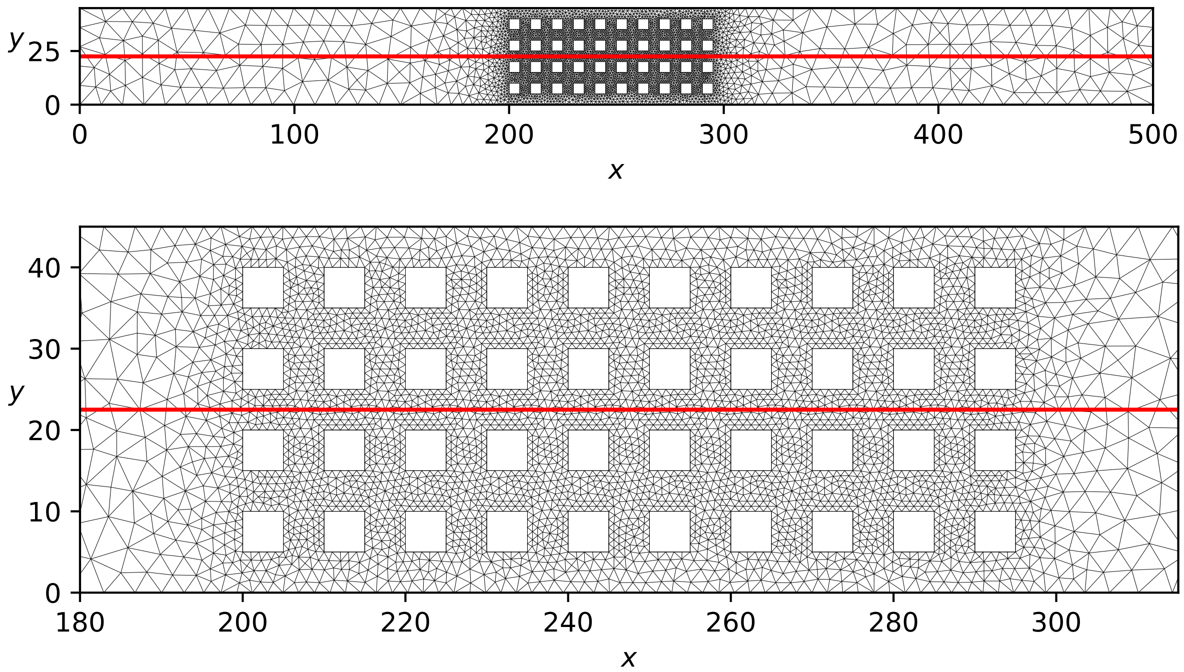


Figure 2.8: Numerical example: fine mesh used for the classical SWE. Top: full domain; bottom: zoom on the urban zone. The red line indicates the slice $y = 22.5$ along which the solutions are compared.

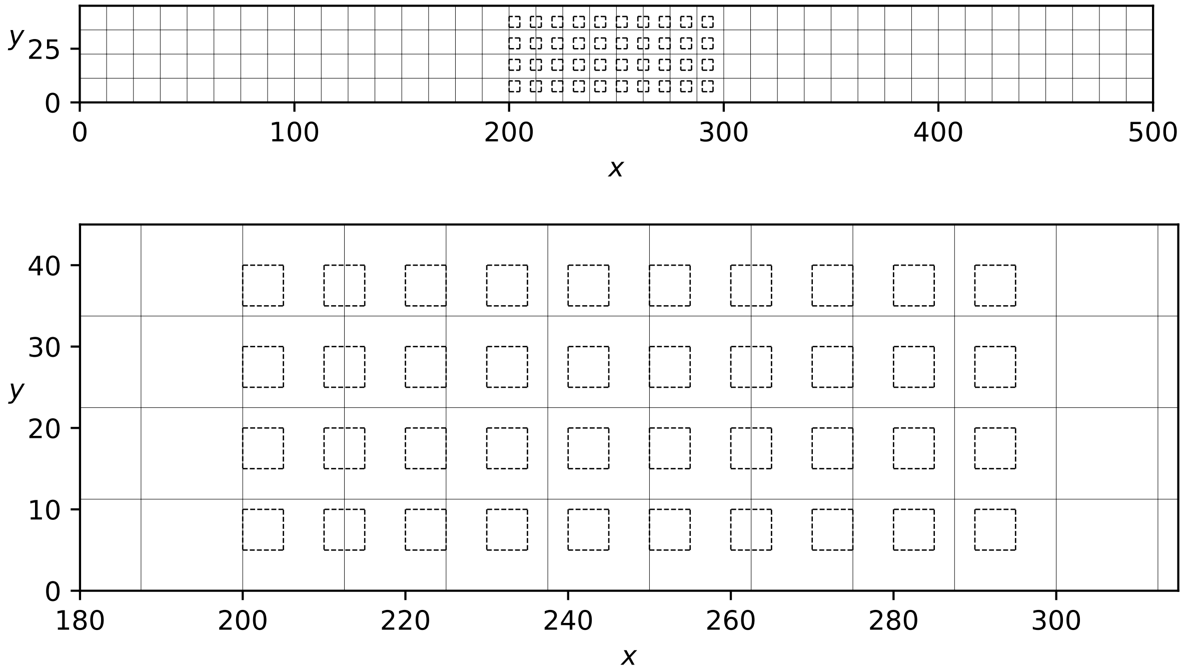


Figure 2.9: Numerical example: coarse mesh used for the porosity-based models. Top: full domain; bottom: zoom on the urban zone. Dashed lines represent the buildings' positions (not physically represented in the mesh).

Classical	SP	DIP
8.54	0.09	0.08

Table 2.1: Numerical example: computational times (in seconds) for the simulation of the classical and the porosity-based shallow water models.

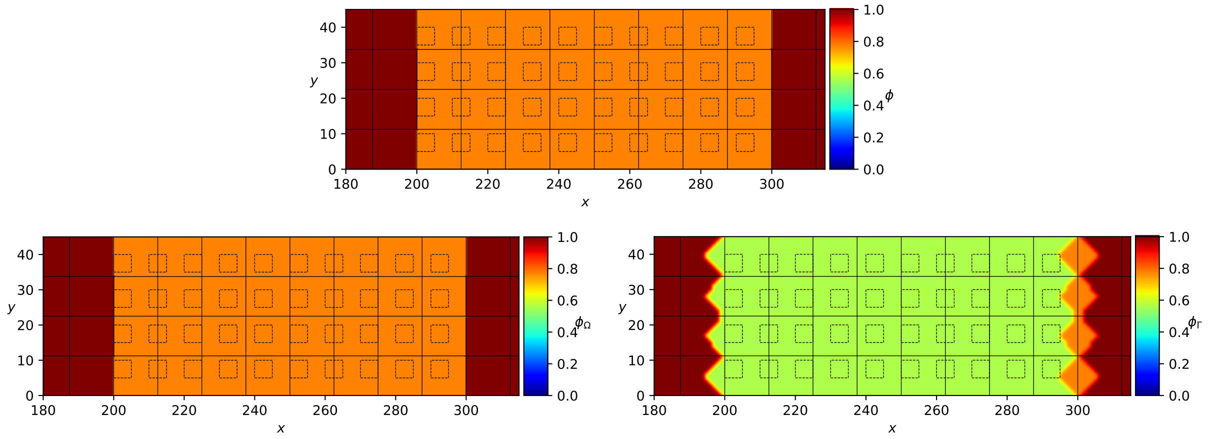


Figure 2.10: Numerical example: porosity fields used in the simulations of the porosity-based models. Top: porosity ϕ for the SP model; bottom: storage porosity ϕ_Ω (left) and conveyance porosity ϕ_Γ (right) for the DIP model. The conveyance porosity is presented as an interpolated field for an easier visualization. Zoom on the urban zone. All the porosities are equal to one in the rest of the domain.

2.12 Conclusion of the chapter

This chapter was dedicated to an overview of shallow water equations at different scales for modeling urban floods. After recalling the classical SWE, we presented a number of porosity-based shallow water models that have been proposed and studied in the past two decades as an alternative for speeding up urban flood simulations. They are volume-averaged upscaled models relying on the definition of porosity parameters for representing the urban geometry, *i.e.* the presence or absence of obstacles and the resulting space available for the flow. This average description allows to use coarser computational meshes than with the classical SWE, and, as a consequence of CFL-type stability conditions, the time-stepping can be performed using larger time steps, thus resulting in much cheaper numerical simulations. In this overview, we focused on presenting the historical evolution of some of the main porosity-based models. This evolution starts from an averaged description using a single porosity parameter, corresponding to the fraction of the space not occupied by obstacles, to models formulated in an integral form, defining two porosity parameters and a dual set of flow variables linked by closure relations, being able to better represent anisotropic phenomena and wave propagation speeds.

An explicit-in-time finite volume (FV) discretization of the presented shallow water models (the classical and the porosity-based ones) was presented, including its formulation in a global-in-space form necessary for its application in model reduction techniques, performed in the next chapter. We also showed that the classical and the porosity-based SWE are hyperbolic, which constitute a major challenge for their simulation using parallel-in-time methods.

Finally, a simple set of numerical examples, simulating a flow through an urban zone, was presented for illustrating the porosity-based models and motivating the work in the following chapters. The results show that the upscaled models provide relatively good approximations to the solution of the classical SWE (solved in a finer mesh, with smaller times steps), within a computational time smaller of two orders of magnitude. However, small scale phenomena originated *e.g.* by the reflection of flow against the obstacles are misrepresented. Therefore, in this work we aim to couple the shallow water models at different scales for taking advantage of the properties of each one, namely the more accurate solutions, provided by the classical SWE, and the reduced computational times, ensured by the porosity-based models.

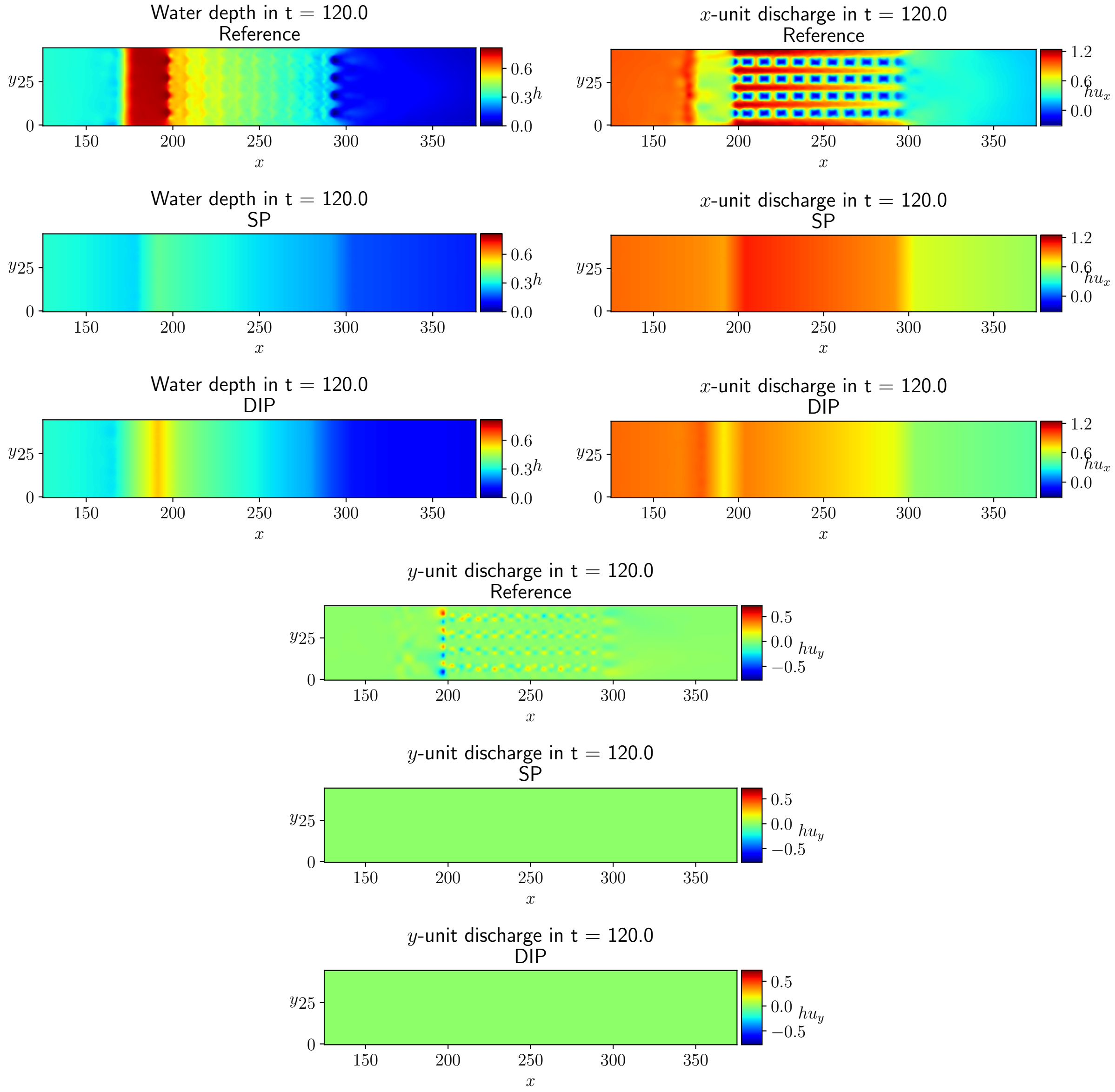
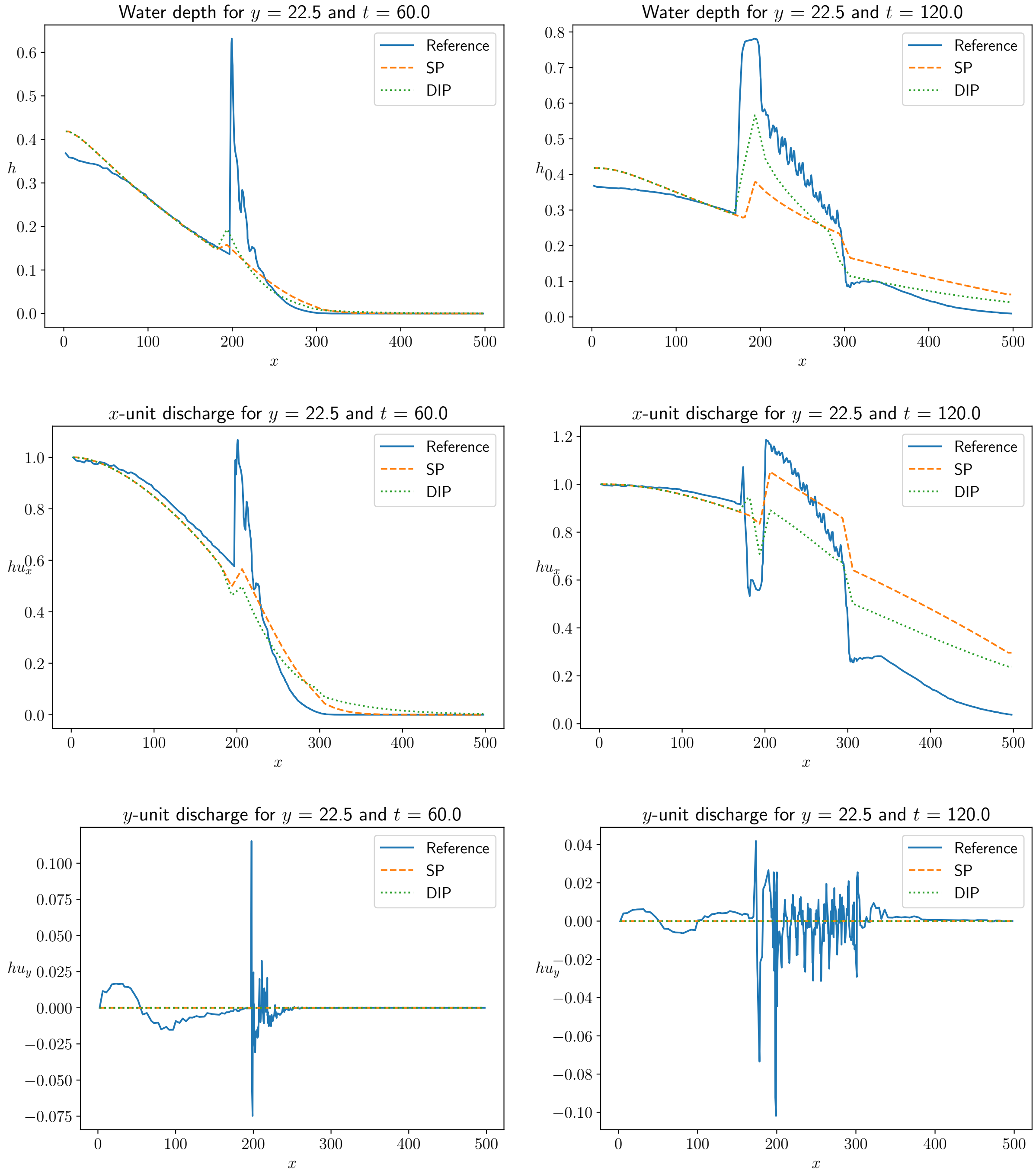


Figure 2.11: Numerical example: comparison of the solutions given by the classical SWE (reference) and the porosity-based SWE (SP and DIP) for $t = T = 120$. Top left group of figures: water depth; top right group: x-unit discharge; bottom group: y-unit discharge. In each group of three figures, the top one, the middle one and the bottom one correspond respectively to the solution of the classical SWE, the SP and the DIP models. Zoom on $[125, 375] \times [0, L_y]$. The solutions of the porosity-based models are interpolated to the fine mesh used for the classical model.


Water depth for $y = 22.5$ and $t = 60.0$

Water depth for $y = 22.5$ and $t = 120.0$

x-unit discharge for $y = 22.5$ and $t = 60.0$

x-unit discharge for $y = 22.5$ and $t = 120.0$

y-unit discharge for $y = 22.5$ and $t = 60.0$

y-unit discharge for $y = 22.5$ and $t = 120.0$

Figure 2.12: Numerical example: comparison of the solutions given by the classical SWE (reference) and the porosity-based SWE (SP and DIP) along the slice $y = 22.5$ for $t = T/2 = 60$ (left) and $t = T/2 = 120$ (right). First, second and third rows: water depth, x-unit discharge and y-unit discharge, respectively. The solutions of the porosity-based models are interpolated to the fine mesh used for the classical model.

CHAPTER 3

THE PARAREAL METHOD AND SOME ADAPTATIONS FOR HYPERBOLIC PROBLEMS

Contents

3.1 Introduction . . . 38

3.1.1 Basic notation and definitions . . . 39

3.2 The original parareal method . . . 40

3.2.1 Presentation of the algorithm . . . 40

3.2.2 Speedup estimation . . . 42

3.2.3 Exact convergence in a finite number iterations . . . 44

3.2.4 Performance of the classical parareal method . . . 45

3.2.5 Other interpretations of the parareal method . . . 47

3.3 The Krylov subspace-enhanced parareal method for linear hyperbolic problems . . . 48

3.4 The ROM-based parareal method for nonlinear hyperbolic problems . . . 49

3.4.1 The Proper Orthogonal Decomposition (POD): reduction of the linear term . . . 51

3.4.2 The empirical interpolation method (EIM): reduction of the nonlinear term . . . 52

3.4.3 Introduction of ROMs in the parareal method . . . 55

3.4.4 Speedup estimation . . . 58

3.5 Application of the ROM-based parareal method to the SWE . . . 61

3.5.1 Formulation of the POD-EIM-based reduced SWE . . . 61

3.5.2 The ROM-based parareal method for solving the SWE . . . 63

3.6 Numerical examples . . . 63

3.6.1 Definition of the test cases . . . 64

3.6.2 Simulation using the classical parareal method . . . 66

3.6.3 Simulation using the ROM-based parareal method . . . 67

3.6.4 Some indications for understanding the parareal performance . . . 75

3.6.5 A first discussion on the influence of the length time slices . . . 79

3.6.6 Influence of nonlinearities . . . 80

3.6.7 Influence of the type of computational mesh . . . 83

3.6.8 Influence of the interpolation procedure . . . 84

3.6.9 Some conclusions on the numerical simulations . . . 87

3.7 Conclusion of the chapter . . . 89

3.1 Introduction

In this chapter, we present the parareal method and some of its variants proposed for improving its performance when applied to hyperbolic problems.

As discussed in Section 1.2, Parallel-in-Time (PinT) methods have gained special attention over the past decades as alternative for overcoming the trade-off between accuracy and computational cost of numerical simulations and the saturation of spatial parallelization. Among them, the parareal, first proposed by Lions et al. (2001), is one of the most popular. This predictor-corrector iterative method, in which sequential predictions are provided by a coarse, low-expensive model and fine corrections are obtained, in parallel, using a fine, expensive and accurate one, stands out for a simple, non-intrusive and problem- and discretization- independent formulation. Indeed, since only the solutions of the fine and coarse models are required by the parareal algorithm, these models can be used in a "black-box" fashion. The popularity of the method can be explained by this simplicity and also by its successful application to a large variety of problems, specially parabolic and diffusive ones, for which a fast convergence is observed, thus allowing to obtain a numerical solution close to the reference one within a reduced computational time.

However, in the case of hyperbolic and advection-dominated problems, the parareal method (as well as other parallel-in-time methods) suffers from instability and/or convergence issues (Ruprecht, 2018). Variants of the method are proposed in the literature for overcoming these difficulties. One can cite, for example, parareal modifications using information obtained along parareal iterations, by formulating Krylov subspaces (Farhat et al., 2006; Gander and Petcu, 2008) or reduced-order models (ROMs) (Chen et al., 2014); asymptotic approximations of the solution (Haut and Wingate, 2014); semi-Lagrangian temporal discretization (Schmitt et al., 2018) and optimization techniques (De Sterck et al., 2021).

We remark that temporal parallelization has been little explored for solving the shallow water equations, which can been explained by its challenging application to hyperbolic problems. Recently, PinT has been proposed and applied for the SWE on the rotating sphere, using an asymptotic parareal method (Haut and Wingate, 2014), rational approximation of exponential integrators (Schreiber and Loft, 2019; Schreiber et al., 2019; Caliari et al., 2021), the MGRIT method combined with exponential integrators (Abel et al., 2020) and the PFASST method combined with a spatial discretization using spherical harmonics. These applications rely on specific choices of spatial and temporal discretizations and on properties of the rotating SWE, such as the presence of fast oscillations due to a linear term. For the SWE on the plane, Arbenz et al. (2012) explored a spatial and temporal parallelization when periodic boundary conditions are considered, making use of the formulation of the problem as cyclic nonlinear system of equations, in a scheme that can be classified as a method for parallelization across the method, as discussed in Section 1.2. More recently, Nielsen et al. (2018) implemented the parareal method for solving the two-dimensional nonlinear shallow water equations in the plane, discretized using explicit-in-time finite volume schemes. Contrary to the most usual practice in parareal applications, in their implementation the coarse and fine model differ only by the order of time-evolution scheme, using the same spatial meshes and adaptive time steps for attaining the maximum admissible CFL number. With this approach, only dissipative errors exist between the two models and, and stable and converging simulations are obtained. With an implementation in an HPC environment and optimized schedules of parallel tasks, large accelerations of the simulation are obtained, in the order of hundreds times faster than the simulation of the fine model.

The objective of this work being the coupling between the classical and porosity-based shallow water equations, coarsening between the fine and coarse models is a constraint, so we place ourselves in a more usual parareal scenario. We also give a special attention to the ROM-based parareal variant proposed by Chen et al. (2014), which is suitable for treating nonlinear hyperbolic problems, such as the SWE. Numerical examples presented in that work illustrate that this method is able to outperform the original parareal in some problems, in terms of stability and convergence, but are less effective in others. We explore and compare the application of the original and ROM-based parareal methods for solving the SWE and, in the following chapters, we propose modifications for improving performance of the ROM-based method and we study its behaviour in larger and more challenging problems.

We present along this chapter a historical evolution of the parareal method, from the original to the ROM-based one. The classical or original method is presented in Section 3.2, along with some numerical examples for illustrating its lack of performance when applied to hyperbolic problems. In Section 3.3, the Krylov subspace-enhanced parareal method, proposed by Farhat et al. (2006) and suitable for treating linear hyperbolic problems, is presented for introducing the idea of reusing information from previous parareal iterations. The ROM-based method proposed by Chen et al. (2014) is presented in Section 3.4, along with the model reduction techniques considered in this work, namely the combined Proper Orthogonal Decomposition (POD) and Empirical Interpolation Method (EIM). These ROM techniques are applied to the 2D nonlinear SWE in Section 3.5. The classical and ROM-based parareal algorithms are illustrated and compared in a set of relatively small numerical examples in Section 3.6. These numerical tests are performed for providing a first insight of the performance of the two methods, as well as the factors and parameters that may influence it. Concluding remarks are presented in Section 3.7.

3.1.1 Basic notation and definitions

Propagators and discretizations

We introduce some basic notations and definitions for the description of the parareal methods in this chapter. We are interested in the numerical simulation over a temporal domain $[0, T]$, with $T > 0$. This domain is divided into $N_{\Delta T} = T/\Delta T$ non-overlapping *time slices* $[t_0, t_1], [t_1, t_2], \dots, [t_{N_{\Delta T}-1}, t_{N_{\Delta T}}]$, which we assume to have the same length ΔT. Therefore, the $N_{\Delta T} + 1$ discrete time instants defining the time slices read

$$t_n := n\Delta T, \qquad n = 0, \dots, N_{\Delta T} \tag{3.1}$$

with $t_0 = 0$ and $t_{N_{\Delta T}} = T$. The instants defined in 3.1 are called hereafter as *parareal time instants*.

Let $\mathcal{E}_{\delta\tau} : \mathbb{R}^M \times \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}^M$ be a generic numerical propagator for a given problem, *i.e.* a discrete time-evolution scheme, using a constant time step $\delta\tau$. The evolution of a given quantity $\boldsymbol{y} \in \mathbb{R}^M$ by $\mathcal{E}_{\delta\tau}$ from an instant t to s, with $s - t \geq d\tau$, is denoted by $\mathcal{E}_{\delta\tau}(\boldsymbol{y}, s, t)$.

More specifically, we define a fine propagator $\mathcal{F}_{\delta t}$, associated to a constant time step $\delta t < \Delta T$, and a coarse propagator $\mathcal{G}_{\Delta t}$, associated to a constant time step $\Delta t \leq \Delta T$. The coarse propagator provides less expensive and less accurate solutions compared to $\mathcal{F}_{\delta t}$, usually with $\Delta t \gg \delta t$. The real numbers δt and Δt are called respectively the *fine* and *coarse* time steps hereafter. The reader is referred to Figure 3.1 for a schematic representation of the time steps definitions. Other similar notations are introduced along this chapter for defining modified propagators in the variants of the parareal method. We assume that the length of the time slices is a multiple integer both of Δt and δt, *i.e.* $p_{\Delta t} := \Delta T/\Delta t \in \mathbb{N}$ and $p_{\delta t} := \Delta T/\delta t \in \mathbb{N}$. Neither the homogeneity of the time steps $\Delta T, \Delta t, \delta t$ nor the assumption $p_{\Delta t}, p_{\delta t} \in \mathbb{N}$ are necessary (see *e.g.* (Carlberg et al., 2016; Nielsen et al., 2018)), but are considered here for simplification.

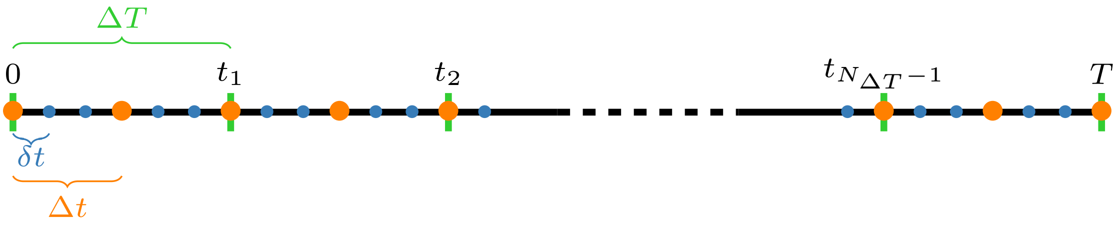


Figure 3.1: Definition of the temporal discretization of $\mathcal{F}_{\delta t}$ (small and large bullets), $\mathcal{G}_{\Delta t}$ (only orange, large bullets) and the parareal time slices (green, vertical ticks), associated respectively to homogeneous time steps δt, Δt and ΔT. The time instants $t_n, n = 0, \dots, N_{\Delta T}$, correspond to the temporal discretization using ΔT. In this illustrative example, each time slice contains $p_{\Delta t} = 2$ coarse and $p_{\delta t} = 6$ fine time steps.

The number of coarse and fine time steps in $[0, T]$ are respectively $N_{\Delta t} := T/\Delta t$ and $N_{\delta t} := T/\delta t$. The ratios $p_{\Delta t} = \Delta T/\Delta t = N_{\Delta t}/N_{\Delta T}$ and $p_{\delta t} = \Delta T/\delta t = N_{\delta t}/N_{\Delta T}$ represent respectively the number of coarse and fine time steps within each time slice. Note that a possible choice for the time steps is $\Delta t = \Delta T$, *i.e.* the coarse time steps associated to the coarse propagator $\mathcal{G}_{\Delta t}$ coincide with the time slices.

We remark that a coarser temporal discretization is not the only possible simplification of $\mathcal{G}_{\Delta t}$ w.r.t. $\mathcal{F}_{\delta t}$. Indeed, coarser spatial discretizations, lower-order numerical schemes and simplified models from the physical and/or mathematical point of view can be considered for defining $\mathcal{G}_{\Delta t}$ (Maday, 2010).

The approximation of a quantity $\boldsymbol{y}$ at time t_n is denoted by

$$\boldsymbol{y}_n \approx \boldsymbol{y}(t_n)$$

As described in the next section, the parareal method iteratively produces approximations to the solution of a given problem. We denote by $\boldsymbol{y}_n^k$ the approximation obtained in the $k-$th parareal iteration for the solution at t_n. The reference solution $\boldsymbol{y}_{\text{ref}}$ for a given problem with initial solution $\boldsymbol{y}_0$ is the one obtained by the sequential simulation of the fine propagator $\mathcal{F}_{\delta t}$ and its values at each parareal time instant t_n is denoted by

$$\boldsymbol{y}_{\text{ref},n} := \mathcal{F}_{\delta t}(\boldsymbol{y}_0, t_n, 0) \tag{3.2}$$

Concerning the spatial discretization, the number of cells in the spatial meshes associated to $\mathcal{F}_{\delta t}$ and $\mathcal{G}_{\Delta t}$ are denoted respectively by M_f and M_c.

Errors and speedup

In order to study and compare the performance of the parareal method along iterations, we define the following relative errors, computed over the mesh associated to the fine, reference propagator $\mathcal{F}_{\delta t}$:

- Error at parareal time instant t_n and iteration k:

$$e_n^k := \frac{\sum_{i=1}^{d_f M_f} |[\boldsymbol{y}_n^k]_i - [\boldsymbol{y}_{\text{ref, n}}]_i|}{\sum_{i=1}^{d_f M_f} |[\boldsymbol{y}_{\text{ref},n}]_i|} \tag{3.3}$$

- Error at iteration k:

$$\overline{e}^k := \max_{n=0,\dots,N_{\Delta T}} e_n^k \tag{3.4}$$

where $[\boldsymbol{y}]_i$ denotes the $i-$th component of the vector $\boldsymbol{y} \in \mathbb{R}^{d_f M_f}$ and d_f is the number of degrees-of-freedom for each discrete spatial point or cell, depending on the numerical scheme, such that $d_f M_f$ is the total number of components of the solution vector. In the case of the shallow water models presented in the previous chapter, $d_f = 3$ (for each computational cell in a FV scheme, the water depth and the $x-$ and $y-$unit discharges are computed).

Evidently, the evaluation of parallel-in-time methods concerns not only the quality of its solution, but also its ability in speeding up the reference numerical simulation. The speedup $s = s(k)$ of a parareal method is defined as the acceleration provided by the method w.r.t. the sequential simulation of the fine, reference propagator, at the end of the $k-$th parareal iteration. Therefore, by defining $T_{\text{par}}(k)$ and T_{ref} as the computational times for the parareal (after k iterations) and reference simulations, respectively, the speedup reads

$$s(k) = \frac{T_{\text{ref}}}{T_{\text{par}}(k)} \tag{3.5}$$

The goal in using the parareal method is to obtain $s(k) > 1$. The parareal computational time $T_{\text{par}}(k)$ and the speedup $s(k)$ depend not only on the number of iterations, but also on the number of parallel processors used in the simulations and the configurations of the fine and coarse propagators, among other factors. Along this work, estimations for the speedup are provided for the parareal method and its variants.

3.2 The original parareal method

3.2.1 Presentation of the algorithm

Following Baffico et al. (2002); Bal and Maday (2002), we present the original parareal method as a predictor-corrector iterative scheme. We consider a simple time-dependent problem

$$\begin{cases} \frac{d}{dt}\boldsymbol{y}(t) + A\boldsymbol{y}(t) = \boldsymbol{0}, \quad \text{in } [0,T] \\ \boldsymbol{y}(0) = \boldsymbol{y}_0 \end{cases} \tag{3.6}$$

where, for simplification, A is assumed independent of time.

The objective of the parareal method is to avoid the expensive computation of the approximate solutions of (3.6) using the fine propagator $\mathcal{F}_{\delta t}$ sequentially over $[0,T]$, as given by (3.2). It is done by computing an initial sequential prediction given by the low-expensive coarse propagator $\mathcal{G}_{\Delta t}$:

$$\boldsymbol{y}_{n+1}^{0} = \mathcal{G}_{\Delta t}(\boldsymbol{y}_{n}^{0}, t_{n+1}, t_n), \qquad n = 0, \dots, N_{\Delta T} - 1 \tag{3.7}$$

and by introducing the following predictor-corrector iterative scheme, in which $\mathcal{G}_{\Delta t}$ is used as predictor:

$$\boldsymbol{y}_{n+1}^{k+1} = \underbrace{\mathcal{G}_{\Delta t}(\boldsymbol{y}_{n}^{k+1}, t_{n+1}, t_n)}_{\text{Prediction}} + \underbrace{\mathcal{F}_{\delta t}(\boldsymbol{y}_{n}^{k}, t_{n+1}, t_n) - \mathcal{G}_{\Delta t}(\boldsymbol{y}_{n}^{k}, t_{n+1}, t_n)}_{\text{Correction}}, \qquad n = 0, \dots, N_{\Delta T} - 1 \tag{3.8}$$

for $k = 0, \dots, N_{\text{itermax}} - 1$, where N_{itermax} is an user-defined maximum number of iterations. Iteration (3.8) provides approximate values $\boldsymbol{y}_{n}^{k+1}, n = 1, \dots, N_{\Delta T}$, for the solution $\boldsymbol{y}$ at the end of each time slice.

The "trick" of the parareal method relies on the fact that, in iteration $k+1$ (which provides $\boldsymbol{y}_{n}^{k+1}, n = 0, \dots, N_{\Delta T}$), the correction term, (that uses the fine, expensive propagator $\mathcal{F}_{\delta t}$), depends only on $\boldsymbol{y}_{n}^{k}, n = 0, \dots, N_{\Delta T} - 1$, which are already available since these quantities have been computed in the previous iteration for all parareal time instants $t_0, \dots, t_{N_{\Delta T}}$. Therefore, the fine propagations over the time slices $[t_n, t_{n+1}], n = 0, \dots, N_{\Delta T} - 1$, are independent and can be computed in parallel.

Supposing that the parareal method is executed using as many processors as there are time slices, each parallel processor advances the expensive fine propagator over only $p_{\delta t}$ fine time steps per iteration, instead of $p_{\delta t} N_{\Delta T}$ as it would be required in a sequential simulation. The only sequential term of the parareal iteration (3.8) is the prediction given by the coarse propagator, which is supposed to be much cheaper than the fine one. Figures 3.2 and 3.3 represent respectively the sequential coarse prediction and the parallel fine correction steps of the parareal method.

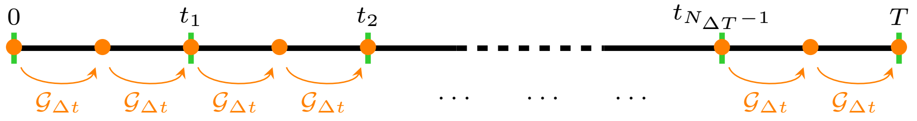


Figure 3.2: Schematic representation of the coarse prediction step in the classical parareal algorithm. The low-expensive coarse propagator is executed sequentially along the entire temporal domain $[0, T]$.

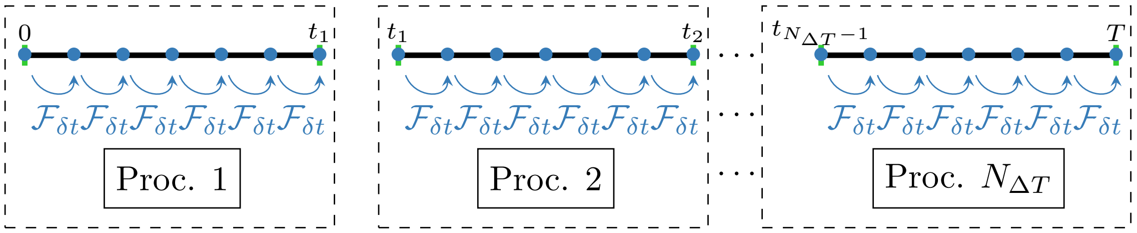

0
t_1
$\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}$
Proc. 1
t_1
t_2
$\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}$
Proc. 2
...
...
...
$t_{N_{\Delta T}-1}$
T
$\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}\mathcal{F}_{\delta t}$
Proc. $N_{\Delta T}$

Figure 3.3: Schematic representation of the fine correction step in the classical parareal algorithm. Each parallel processor performs the fine propagation along one time slice ΔT.

The parareal method (which we call hereafter as *classical* or *original parareal algorithm*) is presented in detail in Algorithm 1. Along this work, we name as "$k-$th parareal iteration" the parareal iteration that produces the approximate solution $\boldsymbol{y}_{n}^{k}, n = 0, \dots, N_{\Delta t}$. Algorithm 1 (and all parareal algorithms presented here) follow *e.g.* Ruprecht and Krause (2012); Astorino et al. (2012); Chen et al. (2014) in the sense that the $k-$th parareal iteration is indexed by $k-1$ in the iteration loop. The solution at the 0-th iteration (initial prediction given the coarse propagator, eq. (3.7)) is performed before the iteration loop.

Note that all parallel fine correction terms are computed in the beginning of each iteration, and the sequential predictor-corrector is performed later. Also note that a convergence criterion is implemented based on the fine correction and on the parareal solution. It is splitted in two steps: firstly, between the correction and the predictor-corrector steps, it is used for determining the times in which the solution needs to be updated; then, at the end of the iteration, the parareal algorithm terminates if only the solution at the last time of simulation has been updated. This implementation follows *e.g.* (Astorino et al., 2012), and the convergence criterion is detailed and justified in Section 3.2.3.

As a last remark on the formulation of the parareal algorithm, in the case where the coarse propagator is also coarser in space w.r.t. the fine one, the initial prediction (3.7) and the parareal iteration (3.8) should include interpolation operators between the coarse and the fine mesh. For the sake of clarity, these operators are not written explicitly in the parareal algorithms presented in this work.

```
Initialization: initial guess given by the coarse propagator:
$\boldsymbol{y}_0^0 = \boldsymbol{y}_0$
for $n \leftarrow 0$ to $N_{\Delta T} - 1$ do
    $\boldsymbol{y}_{n+1}^0 = \mathcal{G}_{\Delta t}(\boldsymbol{y}_n^0, t_{n+1}, t_n)$
end

$n_0 = 0$
Iterations:
for $k \leftarrow 0$ to $N_{itermax} - 1$ do
    Compute the fine term of the correction (in parallel):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\widetilde{\boldsymbol{y}}_{n+1}^k = \mathcal{F}_{\delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)$
    end

    Find the first instant $\tilde{n} \in \{1, \dots, N_{\Delta T}\}$ not satisfying a convergence criterion based on $\boldsymbol{y}_n^k, \widetilde{\boldsymbol{y}}_n^k$
    $n_0 \leftarrow \tilde{n} - 1$

    Compute the coarse predictions and correct them to obtain the final solution in the iteration
     (sequentially):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\boldsymbol{y}_{n+1}^{k+1} = \mathcal{G}_{\Delta t}(\boldsymbol{y}_n^{k+1}, t_{n+1}, t_n) + \widetilde{\boldsymbol{y}}_{n+1}^k - \mathcal{G}_{\Delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)$
    end

    if all instants converged ($\tilde{n} = N_{\Delta T}$) then
        break;
    end
end
```

Algorithm 1: Original or classical parareal algorithm.

3.2.2 Speedup estimation

We present in the following paragraphs an estimation for the numerical speedup (defined in eq. (3.5)) of the classical parareal method.

Let τ_c be the computational time for advancing one coarse time step Δt using $\mathcal{G}_{\Delta t}$. Similarly, τ_f is defined as the computational time for advancing one fine time step δt using $\mathcal{F}_{\delta t}$. Inherent costs of parallelism (initialization of the processors, communication, etc.) are neglected. We consider that $N_p \leq N_{\Delta T}$ processors are used for the parareal simulation, with $N_{\Delta T}$ as a multiple integer of N_p. The parallel tasks are homogeneously distributed to the processors, such that each of them performs the parallel fine correction step along $N_{\Delta T}/N_p$ time slices. We recall that each time slice contains $p_{\delta t} = N_{\delta t}/N_{\Delta T}$ fine time steps.

The computational time T_{ref} necessary for a sequential simulation of the fine model reads

$$T_{\text{ref}} = N_{\delta t}\tau_f \tag{3.9}$$

In the classical parareal method, the following steps are performed:

- an initial full coarse prediction (iteration $k = 0$), taking $T_c = N_{\Delta t}\tau_c$;
- within each iteration $k \geq 1$:
  - the parallel computation of the fine term of the correction, taking

$$T_{\text{corr},f} = \frac{N_{\Delta T}}{N_p} p_{\delta t}\tau_f = \frac{N_{\Delta T}}{N_p}\frac{N_{\delta t}}{N_{\Delta T}}\tau_f = \frac{N_{\delta t}}{N_p}\tau_f$$

  - the sequential computation of the predictions, taking

$$T_{\text{pred}} = N_{\Delta t}\tau_c$$

Note that, in algorithm (3.8), the coarse term of the correction has already been computed in the predictor step of the previous iteration.

Then, the computational time for completing $\hat{k}$ iterations of the classical parareal method reads

$$\begin{aligned} T_{\text{classical-parareal}}(\hat{k}) &= T_c + \hat{k}\left(T_{\text{corr},f} + T_{\text{pred}}\right) \\ &= N_{\Delta t}\tau_c + \hat{k}\left[\frac{N_{\delta t}}{N_p}\tau_f + N_{\Delta t}\tau_c\right] = (1+\hat{k})N_{\Delta t}\tau_c + \hat{k}\frac{N_{\delta t}}{N_p}\tau_f \end{aligned} \tag{3.10}$$

Then, the speedup provided by classical parareal method is estimated as

$$\begin{aligned} s_{\text{classical-parareal}}(\hat{k}) &= \frac{T_{\text{ref}}}{T_{\text{classical-parareal}}(\hat{k})} \\ &= \frac{1}{(\hat{k}+1)\dfrac{N_{\Delta t}\tau_c}{N_{\delta t}\tau_f} + \dfrac{\hat{k}}{N_p}} \end{aligned} \tag{3.11}$$

Following Ruprecht and Krause (2012), we derive and interpret some simple bounds for (3.11), in order to obtain some guidelines on the choice of the various parameters when setting up a parareal simulation. Similar bounds will be derived and interpreted for variants of the parareal method presented along this work. For that, we first introduce estimates for the computational times τ_c an τ_f. Since, for the application considered in this work, we make numerical simulations using explicit-in-time finite volume schemes (eq. (2.30)), these computational times are proportional to the spatial complexity of the discrete models, that is, $\tau_c = \mathcal{O}(M_c)$ and $\tau_f = \mathcal{O}(M_f)$, where M_c and M_f are the number of cells in the coarse and fine meshes, respectively.

Thus, we have the following bounds:

- Bound due to the complexities of the fine and the coarse propagators: from the coarse prediction term, we have
$$s_{\text{classical-parareal}}(\hat{k}) < \frac{1}{\hat{k}+1}\frac{N_{\delta t}\tau_f}{N_{\Delta t}\tau_c} = \frac{1}{\hat{k}+1}\frac{N_{\delta t}M_f}{N_{\Delta t}M_c} \tag{3.12}$$
indicating that the coarse propagator is expected to be much less expensive than the fine one, which is obtained by increasing Δt (thus reducing $N_{\Delta t}$) and/or reducing M_c. Moreover, the factor $1/(\hat{k}+1)$ indicates the limitation of the speedup by the number of iterations;
- Bound due to the number of processors and iterations: from the fine prediction term, we obtain
$$s_{\text{classical-parareal}}(\hat{k}) < \frac{N_p}{\hat{k}} \tag{3.13}$$
It indicates that the parallel efficiency of the parareal method (*i.e.* the speedup divided by the number N_p of processors) is limited by the number of iterations, decreasing by a factor $1/\hat{k}$. Therefore, (3.13) shows that the speedup is improved when more processors are used and a fast convergence (within a small number of iterations) is obtained, which can be achieved by increasing the accuracy of the coarse propagator $\mathcal{G}_{\Delta t}$.

We remark that bounds (3.12) and (3.13) translate a trade-off that needs to be established when performing a parareal simulation: the coarsening of the $\mathcal{G}_{\Delta t}$ (smaller $N_{\Delta t}$ and M_c) can improve bound (3.12), but the resulting poorer quality of $\mathcal{G}_{\Delta t}$ may require more iterations for obtaining a given accuracy for the parareal solution, thus affecting bound (3.13) negatively.

Note that, in Algorithm 1, all fine correction terms $\widetilde{\boldsymbol{y}}_n^k, n = 1, \dots, N_{\Delta T}$, are computed in parallel; then, the sequential step, computing coarse predictions and using the computed fine corrections, is performed by a single processor for obtaining $\boldsymbol{y}_n^{k+1}, n = 1, \dots, N_{\Delta T}$. However, it is possible to implement a scheduling of parallel tasks that allows the computation of the coarse prediction and fine correction steps to overlap, as proposed by Aubanel (2011), which improves the speedup estimate (3.11) and the parallel efficiency of the method. The idea of this scheduling is that the coarse propagation is also distributed among processors, and, when $\boldsymbol{y}_n^{k+1}$ is made available, it is transmitted to next processor, which can compute both $\widetilde{\boldsymbol{y}}_{n+1}^{k+1}$ (used in the next iteration) and $\boldsymbol{y}_{n+1}^{k+1}$. Nevertheless, we consider in this work the simpler formulation presented in Algorithm 1, since only small-scale parallelism is considered in the simulations presented here. Moreover, as discussed in the following chapters, this improved scheduling cannot be applied in the variant of the parareal method using reduced-order models.

3.2.3 Exact convergence in a finite number iterations

It is easy to see (Gander, 2008) that the parareal method (3.8) gives, at each iteration, exact convergence towards the reference solution for one time slice, *i.e.* $\boldsymbol{y}_n^k = \boldsymbol{y}_{\text{ref},n}$ $\forall n \leq k$. Indeed, this is true for $k = 0$, since the initial solution $\boldsymbol{y}_0$ is exact. Suppose now that it is true for the iteration $k > 0$. Therefore

$$\mathcal{G}_{\Delta t}\left(\boldsymbol{y}_k^{k+1}, t_{k+1}, t_k\right) = \mathcal{G}_{\Delta t}\left(\boldsymbol{y}_{\text{ref},k}, t_{k+1}, t_k\right) = \mathcal{G}_{\Delta t}\left(\boldsymbol{y}_k^k, t_{k+1}, t_k\right) \tag{3.14}$$

Replacing (3.14) in (3.8), we obtain

$$\boldsymbol{y}_{k+1}^{k+1} = \mathcal{F}_{\delta t}\left(\boldsymbol{y}_{\text{ref},k}, t_{k+1}, t_k\right) = \boldsymbol{y}_{\text{ref},k+1}$$

As a consequence, the parareal method always provides exact convergence w.r.t. the fine, reference solution in $[0,T]$ in a finite number of iterations; more precisely, in at most $N_{\Delta T}$ iterations. However, the algorithm is interesting only if it converges (or, at least, provides a considerable error reduction) in a much smaller number of iterations $\hat{k}$: supposing that there are as many processors as there are time slices ($N_p = N_{\Delta T}$), each iteration of the parareal algorithm costs at least $1/N_{\Delta T}$ of the sequential fine simulation; then, if $\hat{k} = N_{\Delta T}$, the parareal simulation is more expensive than the reference one (since other computational costs are enrolled in the parareal algorithm, *e.g.* the computation of the coarse predictions and parallel overhead). This property is evident in bound (3.13), since

$$s_{\text{classical-parareal}}(\hat{k} = N_{\Delta T} = N_p) < \frac{N_p}{N_p} = 1$$

The progressive convergence of the parareal method is translated in Algorithm 1 by a convergence criterion based on the relative distance between the parareal solution $\boldsymbol{y}_n^k$ and the fine correction $\widetilde{\boldsymbol{y}}_n^k$:

$$\varepsilon_n^k := \frac{\left\|\widetilde{\boldsymbol{y}}_n^k - \boldsymbol{y}_n^k\right\|}{\left\|\boldsymbol{y}_n^k\right\|} < \varepsilon_{\text{TOL}} \tag{3.15}$$

for a given tolerance ε_{TOL} and a given norm $\|\cdot\|$ in $\mathbb{R}^{M_f}$. The discrete norm $L^1(\mathbb{R}^{M_f})$ is considered here. At each iteration, the first instant t_{n_0+1} not satisfying the criterion (3.15) is found, and the solution is updated only for $n \geq n_0+1$. Therefore, the number of time steps to be computed by the parareal method is monotonically decreasing along iterations (which improves the speedup estimate (3.11)).

The convergence criterion (3.15) can be justified by

$$\begin{aligned}
\left\|\boldsymbol{y}_{\text{ref},n} - \boldsymbol{y}_n^k\right\| &\leq \left\|\boldsymbol{y}_{\text{ref},n} - \mathcal{F}_{\delta t}(\boldsymbol{y}_{n-1}^k, t_n, t_{n-1})\right\| + \left\|\mathcal{F}_{\delta t}(\boldsymbol{y}_{n-1}^k, t_n, t_{n-1}) - \boldsymbol{y}_n^k\right\| \\
&\leq \left\|\mathcal{F}_{\delta t}(\boldsymbol{y}_{\text{ref},n-1}, t_n, t_{n-1}) - \mathcal{F}_{\delta t}(\boldsymbol{y}_{n-1}^k, t_n, t_{n+1})\right\| + \left\|\mathcal{F}_{\delta t}(\boldsymbol{y}_{n-1}^k, t_n, t_{n-1}) - \boldsymbol{y}_n^k\right\| \\
&\leq L\left\|\boldsymbol{y}_{\text{ref},n-1} - \boldsymbol{y}_{n-1}^k\right\| + \left\|\widetilde{\boldsymbol{y}}_n^k - \boldsymbol{y}_n^k\right\| \\
&\leq \sum_{j=1}^{n} L^{n-j}\left\|\widetilde{\boldsymbol{y}}_j^k - \boldsymbol{y}_j^k\right\| \\
&\leq \sum_{j=k+1}^{n} L^{n-j}\left\|\widetilde{\boldsymbol{y}}_j^k - \boldsymbol{y}_j^k\right\|
\end{aligned} \tag{3.16}$$

supposing that $\mathcal{F}_{\delta t}$ is Lipschitz continuous with Lipschitz constant L. Therefore, the difference $\widetilde{\boldsymbol{y}}_n^k - \boldsymbol{y}_n^k$ between the parareal solution and the fine propagation starting from the previous parareal time instant is an indicator of the error w.r.t. the reference solution. This is quite intuitive since a small $\widetilde{\boldsymbol{y}}_n^k - \boldsymbol{y}_n^k$ means that the parareal iteration is close to the fine propagator in $[t_{n-1}, t_n]$. Then, the criterion (3.15) is used here even if the hypothesis on the Lipschitz continuity of $\mathcal{F}_{\delta t}$ is not verified. In the last line of (3.16), we used the fact that $\boldsymbol{y}_j^k = \widetilde{\boldsymbol{y}}_j^k = \boldsymbol{y}_{\text{ref},j}$ $\forall j \leq k$.

The convergence criterion (3.15) is proposed by Astorino et al. (2012), but other convergence criteria could be adopted, based *e.g.* on the difference between the solution of two consecutive parareal iterations (*e.g.* in (Ruprecht and Krause, 2012)). Also, other norms could be considered, and a detailed study could be conducted for determining an optimal criterion and tolerance ε_{TOL}. In all simulations performed in this work, we consider the convergence tolerance $\varepsilon_{\text{TOL}} = 10^{-10}$. In practice, for the problems considered here, it is verified that this tolerance leads to a convergence at the minimum rate of one time slice per iteration (even if, as said above, considerable error decreases can be obtained along iterations). Therefore, the convergence verification in Algorithm 1 could be equivalently replaced by the simpler update $n_0 \leftarrow n_0 + 1$.

3.2.4 Performance of the classical parareal method

Besides its simple and generic formulation, the parareal method stands out for its successful application to many problems, specially parabolic or diffusive ones, including very complex problems (Ruprecht, 2018). Some examples found in the literature are the heat equation (Lions et al., 2001; Astorino et al., 2012; Ruprecht et al., 2016; Zeng et al., 2019), the reaction-diffusion problem in chemical dynamics (Duarte et al., 2011), the Navier-Stokes equations (Fischer et al., 2005; Trindade and Pereira, 2004) and a splitting coupling between the heat transfer and heat flow equations (Geiser and Güttel, 2012).

However, in the case of hyperbolic problems, even the simplest ones as the one-dimensional advection equation with constant speed, the parareal method presents slow convergence and/or instabilities. Many works have shown (Mercerat et al., 2009; Dai and Maday, 2011; Steiner et al., 2015) and studied this behaviour. Bal (2005) performs an analysis for PDEs with linear operators and concludes that the parareal algorithm is unconditionally stable for most discretizations of parabolic problems, but not for hyperbolic ones. Gander and Vandewalle (2007) show that parareal presents superlinear convergence on bounded time intervals both for the heat and the advection equation, but for the latter the convergence constants are larger than the unity. A study for the advection equation using characteristics is proposed by Gander (2008), who concludes that parareal is not the ideal way to parallelize this problem. Finally, Ruprecht (2018) indicates that parareal instabilities are caused by the mismatch between the discrete phase speeds of the coarse and fine propagators for high wavenumbers, for which the convergence is also shown to be slower; thus, the different performances of the parareal method when applied to parabolic and hyperbolic problems can be explained by the fact that high wavenumbers are damped in the former but not in the latter.

We present some simple examples for illustrating these different behaviours of the parareal method when applied to hyperbolic or parabolic problems. As done by Ruprecht (2018), we consider the one-dimensional advection-diffusion equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2} \tag{3.17}$$

since the hyperbolicity and parabolicity can be easily controlled by choosing the diffusion coefficient $\nu \geq 0$. Eq. (3.17) is discretized by finite differences, using an implicit backward Euler scheme in time, and standard first-order backward and centered second-order discretizations for the first and second spatial derivatives, respectively. Using the notation proposed in Section 3.1.1, the parareal solution at iteration k and time t_n is denoted

$$\boldsymbol{y}_n^k = \begin{pmatrix} u_1 \\ \vdots \\ u_{M_f} \end{pmatrix} \in \mathbb{R}^{M_f}$$

and the reference solution $\boldsymbol{y}_{\text{ref},n} \in \mathbb{R}^{M_f}$ at t_n is given by a sequential simulation of (3.17) using the fine propagator.

For comparing the solutions between the proposed simulations, we consider the errors defined in (3.3) and (3.4), with $d_f = 1$ as the number of degrees-of-freedom for each discrete spatial point and $d_f M_f = M_f$ as the total number of components of the solution vector.

We first consider a domain $[0, 10]$ with periodic boundary conditions and the Gaussian initial condition $u(x, t = 0) = \exp\left(-5(x-5)^2\right), x \in [0, 10]$. The fine and coarse propagator $\mathcal{F}_{\delta t}$ and $\mathcal{G}_{\Delta t}$ are defined by the same spatial mesh sizes $\delta x = \Delta x = 0.2$ and by time steps $\delta t = 0.01$ and $\Delta t = 1$, respectively. The time slices coincide with the coarse time steps, *i.e.* $\Delta T = \Delta t = 1$. The simulation is run from $t = 0$ to $t = T = 10$, such that $N_{\Delta T} = 10$, *i.e.* exact convergence towards the reference solution is obtained in at most ten iterations. Four simulations are considered, from the most to the less hyperbolic: $\nu = 0$, $\nu = 0.01$, $\nu = 0.1$ and $\nu = 1$. Figure 3.4 shows the evolution of the maximum error (3.4) along iterations. We observe a much faster error decay when the problem becomes less dominated by advection phenomena. It can also be observed in Figure 3.5, which compares the reference and parareal solutions in the final time of simulation. Fewer iterations are needed in the high-diffusive simulations for obtaining solutions visually very close to the reference one.

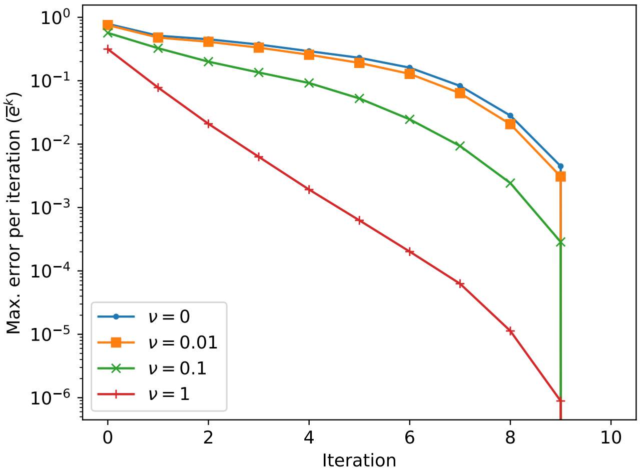


Figure 3.4: Maximum error per iteration for the one-dimensional advection-diffusion equation, with Gaussian initial condition and periodic boundary conditions, for four parareal simulations with different diffusion coefficients. Exact convergence is obtained in ten iterations.

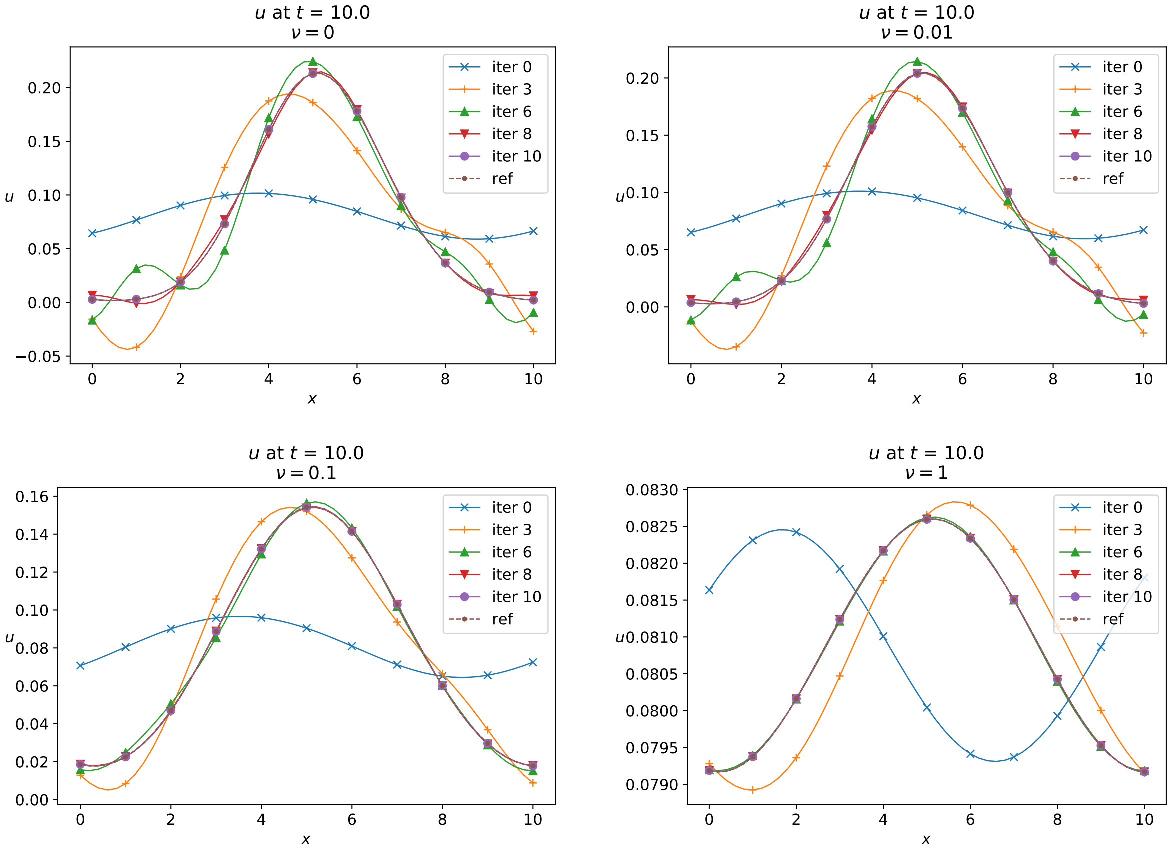


Figure 3.5: Parareal solution along iterations at $t = T = 10$ for the one-dimensional advection-diffusion equation, with Gaussian initial condition and periodic boundary conditions, for four parareal simulations with different diffusion coefficients (top left: $\nu = 0$; top right: $\nu = 0.01$; bottom left: $\nu = 0.1$; bottom right: $\nu = 1$). Dashed lines represent the reference solution. Exact convergence is obtained in ten iterations.

We then consider a second set of test cases, solved in the spatial domain $[0, 20]$, also with periodic boundary conditions and same temporal domain $[0, T]$, time slices and fine and coarse propagators as above. We consider four pure-advection simulations ($\nu = 0$), with initial solutions $u(x, t = 0) = \sin(\kappa x)$ and wavenumbers $\kappa = 8\pi/10$, $\kappa = 4\pi/10$, $\kappa = 2\pi/10$ and $\kappa = \pi/10$. Figures 3.6 and 3.7 present respectively the maximum error per iteration and the solution at $t = T$ along iterations. As stated by Ruprecht (2018), we notice a much better behaviour of the parareal method for the solutions with small wavenumber. For the simulations with high wavenumber, a slow convergence and unstable behaviours, such as non-physical amplifications, are observed.

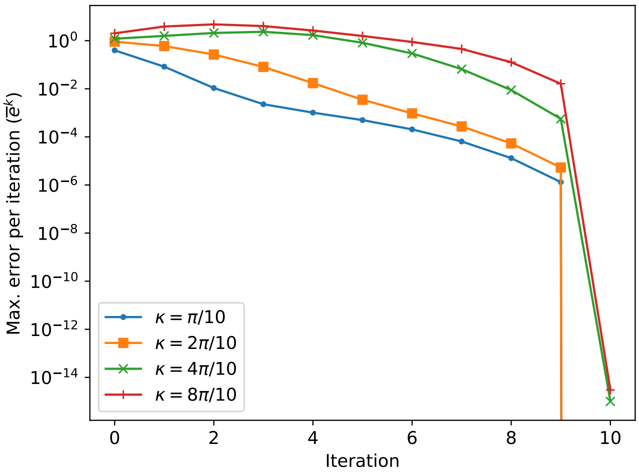


Figure 3.6: Maximum error per iteration for the one-dimensional advection-diffusion equation, with sinusoidal initial condition and periodic boundary conditions, for four parareal simulations with different wavenumbers. Exact convergence is obtained in ten iterations.

3.2.5 Other interpretations of the parareal method

The parareal method is commonly presented under the form of a predictor-corrector iterative algorithm, but other derivations and interpretations are found in the literature. Indeed, in its first appearance, by Lions et al. (2001), the method consists in computing and propagating "jumps" given by the difference between the solution of a coarse solver and a finer or exact solution over a time step, a formulation that cannot be directly applied to nonlinear problems. A similar formulation using jumps is presented by Farhat and Chandesris (2003) under the name of Parallel Implicit Time-integration Algorithm (PITA), which is equivalent to (3.8) in the case of linear problems (Gander and Petcu, 2008).

The presentation of the algorithm as a predictor-corrector scheme is introduced by Baffico et al. (2002) and Bal and Maday (2002), being generalized to both linear and nonlinear problems. We also remark that, in the predictor-corrector framework, the nomenclature "prediction" and "correction" for the algorithm's steps have different definitions in the literature. We consider here (eq. 3.8) the interpretation used *e.g.* by Fischer et al. (2005), which is the opposite from the one adopted by Ruprecht and Krause (2012) and Chen et al. (2014). A discussion on the various interpretations of the predictor-corrector scheme is proposed by Maday (2010).

A matricial formulation of the algorithm is introduced by Maday and Turinici (2002b), allowing it to be interpreted as a preconditioning procedure and extended to control problems in PDEs. Gander and Vandewalle (2007) propose a presentation under the form of a multiple shooting method, solved with a Newton's method in which the Jacobian is approximated using the coarse propagator. The same authors also propose a formulation as a nonlinear multigrid method.

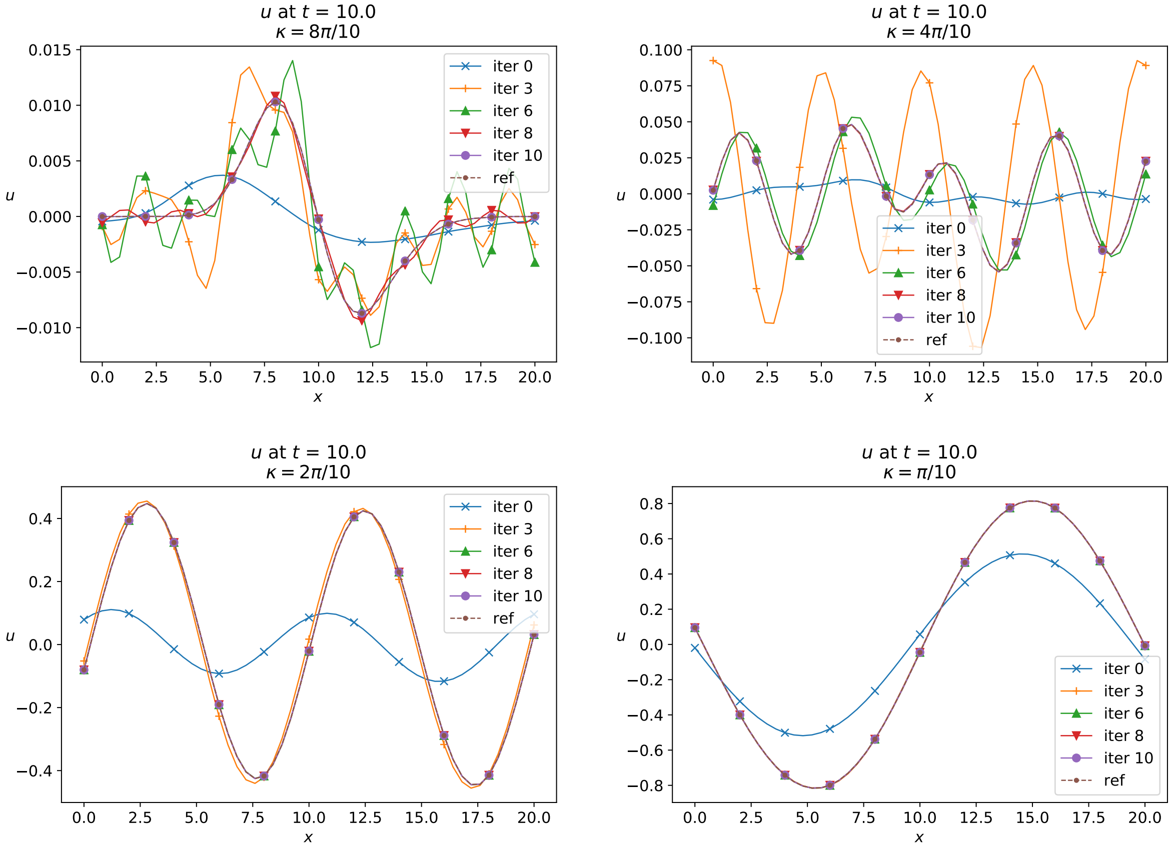


Figure 3.7: Parareal solution along iterations at $t = T = 10$ for the one-dimensional advection-diffusion equation, with sinusoidal initial condition and periodic boundary conditions, for four parareal simulations with different wavenumbers (top left: $\kappa = 8\pi/10$; top right: $\kappa = 4\pi/10$; bottom left: $\kappa = 2\pi/10$; bottom right: $\kappa = \pi/10$). Dashed lines represent the reference solution. Exact convergence is obtained in ten iterations.

3.3 The Krylov subspace-enhanced parareal method for linear hyperbolic problems

A successful approach for overcoming the convergence and stability issues of PinT methods for linear hyperbolic problems, consisting in formulating Krylov subspaces along iterations, is presented in the framework of the Parallel Implicit Time-integrator Algorithm (PITA) by Farhat et al. (2006) and extended to the nonlinear case by Cortial and Farhat (2009), with applications to structural dynamics. Gander and Petcu (2008) showed that PITA and parareal are equivalent in the linear case, and called the modified method as *Krylov subspace-enhanced parareal*. This method was further extended to hyperbolic flow problems by Ruprecht and Krause (2012), with application to a linear acoustic-advection problem.

The idea of this modified parareal approach is to improve the coarse propagations by reusing previously computed information. It is done by defining a subspace $\mathcal{S}^k \subset \mathbb{R}^{M_f}$ spanned by the parareal solution computed in all previous iterations and all parareal time instants (called *snapshots*):

$$\mathcal{S}^k := \text{span}\{\boldsymbol{y}_n^j,\ n = 0, \dots, N_{\Delta T},\ j = 0, \dots, k\} \tag{3.18}$$

The coarse propagator in the $k-$th iteration is replaced by the operator

$$\mathcal{K}_{\Delta t}^k(\boldsymbol{y}, t_{n+1}, t_n) := \mathcal{G}_{\Delta t}((\mathbb{I} - \mathbb{P}^k)\boldsymbol{y}, t_{n+1}, t_n) + \mathcal{F}_{\delta t}(\mathbb{P}^k \boldsymbol{y}, t_{n+1}, t_n) \tag{3.19}$$

where $\mathbb{I} \in \mathbb{R}^{M_f \times M_f}$ is the identity operator and $\mathbb{P}^k \in \mathbb{R}^{M_f \times M_f}$ is the projection operator onto $\mathcal{S}^k$. The modified parareal iteration reads

$$\boldsymbol{y}_{n+1}^{k+1} = \underbrace{\mathcal{K}_{\Delta t}^{k}(\boldsymbol{y}_n^{k+1}, t_{n+1}, t_n)}_{\text{Prediction}} + \underbrace{\mathcal{F}_{\delta t}(\boldsymbol{y}_n^{k}, t_{n+1}, t_n) - \mathcal{K}_{\Delta t}^{k}(\boldsymbol{y}_n^{k}, t_{n+1}, t_n)}_{\text{Correction}}, \qquad n = 0, \dots, N_{\Delta T} - 1 \tag{3.20}$$

The detailed method is presented in Algorithm 2, in which the modifications w.r.t. to the classical parareal algorithm (Algorithm 1) are highlighted. Note that the coarse correction term needs to be computed at each iteration (since it is given by the current propagator $\mathcal{K}_{\Delta t}^{k}$), contrary to the classical parareal method (in which the coarse correction is already available from the previous iteration).

The name of the method comes from the fact that the space $\mathcal{S}^0$ at iteration $k = 0$ is a Krylov subspace:

$$\mathcal{S}^0 = \operatorname{span}\{\boldsymbol{y}_0^0, \mathcal{G}_{\Delta t}\boldsymbol{y}_0^0, \mathcal{G}_{\Delta t}^2\boldsymbol{y}_0^0 \dots, \mathcal{G}_{\Delta t}^{N_{\Delta T}-1}\boldsymbol{y}_0^0\} \tag{3.21}$$

even if, for the following iterations, the spaces $\mathcal{S}^k, k \geq 1$, have a more complicated structure (Gander and Petcu, 2008). A lighter notation $\mathcal{G}_{\Delta t}^n \boldsymbol{y}_0^0 := \mathcal{G}_{\Delta t}(\boldsymbol{y}_0^0, t_n, t_0) = \boldsymbol{y}_n^0$, $n = 0, \dots, N_{\Delta T}$, is adopted in (3.21). The subspace $\mathcal{S}^0$ can be computed using a QR decomposition of the matrix whose columns are the input snapshots and, given a subspace $\mathcal{S}^k$, the subspace $\mathcal{S}^{k+1}$ can be obtained by adding the new snapshots using the Gram-Schmidt procedure, since

$$\mathcal{S}^k = \mathcal{S}^{k-1} \cup \operatorname{span}\{\boldsymbol{y}_n^k,\ n = 0, \dots N_{\Delta T}\} \tag{3.22}$$

However, it is preferable to perform a full QR decomposition in each iteration, due to unstable behaviours of the Gram-Schmidt method (Ruprecht and Krause, 2012).

Note, in (3.19)-(3.20), that the fine propagator is introduced in the sequential correction step, which would impede any speedup w.r.t. the fine, reference simulation, since each iteration would be more expensive than it. This issue can be avoided in the case of linear problems. Let us denote by $W^k := [\boldsymbol{s}_1^k, \dots, \boldsymbol{s}_r^k] \in \mathbb{R}^{M_f \times r}$, with $r = \min\{k(N_{\Delta T} + 1), M_f\}$, a matrix whose columns are the vectors of an orthonormal basis of $\mathcal{S}^k$. Thus, the projection operator reads $\mathbb{P}^k = W^k (W^k)^T$ and the coordinates of $\mathbb{P}^k\boldsymbol{y}$ w.r.t. $\mathcal{S}^k$ are $\boldsymbol{c}^k := (W^k)^T\boldsymbol{y} = [c_1^k, \dots, c_r^k]^T \in \mathbb{R}^r$, *i.e.*

$$\mathbb{P}^k\boldsymbol{y} = W^k(W^k)^T\boldsymbol{y} = W^k\boldsymbol{c}^k = \sum_{j=1}^{r} c_j^k \boldsymbol{s}_j^k \tag{3.23}$$

Replacing (3.23) into the fine term of (3.19) and using the linearity of the problem, we obtain

$$\mathcal{F}_{\delta t}(\mathbb{P}^k\boldsymbol{y}, t_{n+1}, t_n) = \mathcal{F}_{\delta t}\left(\sum_{j=1}^{r} c_j^k \boldsymbol{s}_j^k, t_{n+1}, t_n\right) = \sum_{j=1}^{r} c_j^k \mathcal{F}_{\delta t}(\boldsymbol{s}_j^k, t_{n+1}, t_n); \tag{3.24}$$

which can be efficiently computed by noticing that the basis vectors $\boldsymbol{s}_j^k$ do not depend on time. Therefore, it suffices to propagate them over a single time slice ΔT:

$$\tilde{\boldsymbol{s}}_j^k := \mathcal{F}_{\delta t}(\boldsymbol{s}_j^k, t + \Delta T, t), \qquad j = 1, \dots, r$$

which can be done in parallel, since these propagations are independent, and update the projection coefficients at each time t_n by computing $\boldsymbol{C}^{k,n} := [c_1^{k,n}, \dots, c_r^{k,n}]^T = (W^k)^T\boldsymbol{y}_n^k \in \mathbb{R}^r$. In summary, (3.24) is computed as

$$\mathcal{F}_{\delta t}(\mathbb{P}^k\boldsymbol{y}, t_{n+1}, t_n) = \sum_{j=1}^{r} c_j^{k,n} \tilde{\boldsymbol{s}}_j^k, \qquad n = 0, \dots, N_{\Delta T} - 1$$

As a last remark, note that a more efficient scheduling of parallel tasks as proposed by Aubanel (2011) cannot be applied to the Krylov-subspace-enhanced parareal algorithm, since the subspace $\mathcal{S}^k$, formulated using the solution at time slices from previous iterations, need to be computed before starting the predictor-corrector step.

3.4 The ROM-based parareal method for nonlinear hyperbolic problems

The introduction of Reduced-Order Models (ROMs) in the parareal algorithm is proposed by Chen et al. (2014) for overcoming two drawbacks of the Krylov subspace enhanced-based parareal method:

```
Initialization: initial guess given by the coarse propagator:
$\boldsymbol{y}_0^0 = \boldsymbol{y}_0$
for $n \leftarrow 0$ to $N_{\Delta T} - 1$ do
    $\boldsymbol{y}_{n+1}^0 = \mathcal{G}_{\Delta t}(\boldsymbol{y}_n^0, t_{n+1}, t_n)$
end

$n_0 = 0$
Iterations:
for $k \leftarrow 0$ to $N_{itermax} - 1$ do
    Compute the fine term of the correction (in parallel):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\widetilde{\boldsymbol{y}}_{n+1}^k = \mathcal{F}_{\delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)$
    end

    Compute the subspace
    $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \text{span}\{\boldsymbol{y}_n^k,\ n = 0, \dots N_{\Delta T}\}$

    Find the first instant $\tilde{n} \in \{1, \dots, N_{\Delta T}\}$ not satisfying a convergence criterion based on $\boldsymbol{y}_n^k, \widetilde{\boldsymbol{y}}_n^k$
    $n_0 \leftarrow \tilde{n} - 1$

    Compute the coarse term of the correction and the final correction term(in parallel):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\overline{\boldsymbol{y}}_{n+1}^k = \mathcal{K}_{\Delta t}^k(\boldsymbol{y}_n^k, t_{n+1}, t_n)$
        $\overline{\overline{\boldsymbol{y}}}_{n+1}^k = \widetilde{\boldsymbol{y}}_{n+1}^k - \overline{\boldsymbol{y}}_{n+1}^k$
    end

    Compute the coarse predictions and correct them to obtain the final solution in the iteration
     (sequentially):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\boldsymbol{y}_{n+1}^{k+1} = \mathcal{K}_{\Delta t}^k(\boldsymbol{y}_n^{k+1}, t_{n+1}, t_n) + \overline{\overline{\boldsymbol{y}}}_{n+1}^k$
    end

    if all instants converged ($\tilde{n} = N_{\Delta T}$) then
        break;
    end
end
```

Algorithm 2: Krylov subspace-enhanced parareal algorithm. Modifications w.r.t. the classical parareal method (Algorithm 1) are highlighted.

firstly, in this method the number of basis vectors grows linearly with the iterations, thus increasing the computational cost for evaluating (3.24); secondly, being based on the assumption of linearity, a parallel efficient implementation, less expensive than the sequential simulation of the fine model, is not applicable to nonlinear problems.

The term *reduced-order modeling*, or *model order reduction*, refers to a large family of numerical methods aiming to reduce the complexity of numerical simulations of mathematical models, by capturing the essential features of the problem dynamics. Being developed mainly from the 1980's, ROMs have gained popularity in several applications for overcoming computational resources limitations (Schilders, 2008). Reviews on the main ROM techniques are proposed *e.g.* by Antoulas (2004); Rozza et al. (2008); Schilders (2008).

Chen et al. (2014) propose to use Proper Orthogonal Decomposition (POD)-based (Kosambi, 1943) and Empirical Interpolation Method (EIM)-based (Barrault et al., 2004) reduced-order models in the parareal method. More specifically, the use of a particular case of the EIM that uses POD features, which we call hereafter as POD-EIM (but also found in the literature under the name *POD-Discrete Empirical Interpolation Method (POD-DEIM)* (Chaturantabut and Sorensen, 2010)), is also proposed.

Whereas the POD ROM is restricted to linear problems, the EIM and POD-EIM procedures are effective to reduce the dimension of nonlinear ones. In the following paragraphs, we briefly describe the three mentioned approaches and their application to the parareal method. In the sequel of this work, we consider the POD-EIM-based parareal method for solving the shallow water models. The computational implementation of this method is detailed in Appendix B.

We follow the presentation proposed by Chaturantabut and Sorensen (2010), by considering the system of ODEs (or a spatially discretized PDE)

$$\frac{d}{dt}\boldsymbol{y}(t) = A\boldsymbol{y}(t) + \boldsymbol{F}(\boldsymbol{y}(t)) \tag{3.25}$$

where $\boldsymbol{y} \in \mathbb{R}^{M_f}$, $A \in \mathbb{R}^{M_f \times M_f}$ is constant in time and $\boldsymbol{F} : \mathbb{R}^{M_f} \to \mathbb{R}^{M_f}$ is a nonlinear function with values on $\mathbb{R}^{M_f}$. Problem (3.25) is called hereafter as *Full-Order Model* (FOM). If M_f is large, the computational simulation of (3.25) may be very expensive, which motivates the use of model order reduction techniques to formulate an approximate problem with smaller dimension. We choose here the notation M_f, the same used for the size of the spatial discretization of the fine propagator $\mathcal{F}_{\delta t}$ in the parareal method, for highlighting that a model order reduction of this expensive propagator will be performed in the parareal framework.

3.4.1 The Proper Orthogonal Decomposition (POD): reduction of the linear term

The Proper Orthogonal Decomposition (POD) is one of the most known and popular model order reduction approaches, widely used in several applications since its first proposition by Kosambi (1943). It is also known under different names in other application domains, such as Principal Component Analyisis (PCA), Karhunen-Loève expansion and empirical orthogonal functions (Narasimha, 2011). See (Berkooz et al., 2003) for a historical review of the method. As discussed below, the POD allows to reduce the complexity of a linear problem; more precisely, it is able to reduce the dimension of the left-hand side and first right-hand side terms in eq. (3.25).

We consider here the description of the method as in (Chaturantabut and Sorensen, 2010). The objective of the POD, or *POD-Galerkin*, is to define a reduced subspace $\mathcal{S}_q \subset \mathbb{R}^{M_f}$ with small dimension $q \ll M_f$ and spanned by realizations (called *snapshots*) of the solution of (3.25) in a given number of time instants, and construct the reduced-order system via a Galerkin projection of the FOM onto $\mathcal{S}_q$.

Let $Y = [\boldsymbol{y}(t_1), \dots, \boldsymbol{y}(t_{n_s})] \in \mathbb{R}^{M_f \times n_s}$ be the *snapshots matrix*, containing in its columns the solution of (3.25) in chosen n_s time instants. The columns of Y span the subspace $\mathcal{Y}$ with dimension $r \leq \min(M_f, n_s)$ (in general, the number of degrees-of-freedom of the problem is much larger than the number of snapshots, *i.e.* $M_f \gg n_s$). The POD subspace S_q of dimension $q \leq r$ is defined as the subspace spanned by the orthonormal vectors $\{\boldsymbol{\varphi}_i\}_{i=1}^q \subset \mathbb{R}^{M_f}$, called *POD modes*, that best approximates $\mathcal{Y}$ in the sense of solving the minimization problem

$$\begin{aligned} &\min_{\{\boldsymbol{\varphi}_i\}_{i=1}^q} \sum_{j=1}^{n_s} \left\| \boldsymbol{y}(t_j) - \sum_{i=1}^{q} ((\boldsymbol{y}(t_j))^T \boldsymbol{\varphi}_i)\boldsymbol{\varphi}_i \right\|_2^2 \\ &\boldsymbol{\varphi}_i^T \boldsymbol{\varphi}_j = \delta_{ij}, \qquad i, j = 1, \dots, q \end{aligned} \tag{3.26}$$

where $\|\cdot\|_2$ is the Euclidian norm and δ_{ij} is the Kronecker delta function. In other words, we search a space $\mathcal{S}_q$ such that the sum of the differences between the snapshots and their projection onto $\mathcal{S}_q$ is minimal.

It is well-known that the solution of (3.26) is given by the q first left singular vectors of the snapshots matrix Y, *i.e.* the left singular vectors associated to the largest singular values. Therefore, the POD can be performed via the singular value decomposition (SVD) of Y:

$$Y = V\Sigma W^T \tag{3.27}$$

where $V = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_r] \in \mathbb{R}^{M_f \times r}$ and $W = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_r] \in \mathbb{R}^{n_s \times r}$ are matrices whose columns are respectively the left and right singular vectors of V, 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. The basis vectors of $\mathcal{S}_q$ are $\boldsymbol{\varphi}_i = \boldsymbol{v}_i, i = 1, \dots, q$. The truncation of the POD basis (*i.e.* the value of q) is usually determined by choosing a fraction of the “energy” to be retained (Camphouse et al., 2008; Marquez et al., 2013):

$$\frac{\sum_{i=1}^{q} \sigma_i}{\sum_{i=1}^{r} \sigma_i} > 1 - \varepsilon_{\text{sv}} \tag{3.28}$$

for a given (small) threshold ε_{sv}. In general, only few POD modes are necessary to represent more than 99% of the snapshots information (Schilders, 2008). We remark that other approaches than the SVD are possible for performing the POD, such as the so-called method of snapshots, based on the solution of an eigenvector problem of the covariance matrix of Y, and partitioned approaches, based on the partition of the snapshots set (Wang et al., 2015). The method of snapshots is more efficient for matrices $Y \in \mathbb{R}^{M_f \times n_s}$ with $M_f \gg n_s$ (which is the situation encountered in the applications in this work), since the covariance matrix $Y^T Y \in \mathbb{R}^{n_s \times n_s}$ has smaller dimension, but may affect the POD basis due to round-off errors for computing $Y^T Y$ (Chaturantabut and Sorensen, 2011). We consider here the SVD approach, efficiently performed using the *dgesvd* function from the *MKL-LAPACK* suite, as described in Appendix B.

With the basis vectors computed, the model reduction proceeds with a Galerkin projection. Let $V_q = [\boldsymbol{\varphi}_1, \dots, \boldsymbol{\varphi}_q] \in \mathbb{R}^{M_f \times q}$. Then, the approximation of $\boldsymbol{y}(t)$ in $\mathcal{S}_q$ reads

$$\boldsymbol{y}(t) \approx \sum_{i=1}^{q} \widetilde{y}_i(t) \boldsymbol{\varphi}_i = V_q \widetilde{\boldsymbol{y}}(t) \tag{3.29}$$

where $\widetilde{\boldsymbol{y}} := (\widetilde{y}_1, \ \dots, \ \widetilde{y}_q)^T$ are the coordinates of this approximation in $\mathcal{S}_q$. By replacing (3.29) in (3.25) and projecting (3.25) onto $\mathcal{S}_q$ (*i.e.* by left-multiplying it by V_q^T), we obtain the *POD-reduced-order model*

$$\begin{aligned} \frac{d}{dt}\widetilde{\boldsymbol{y}}(t) &= V_q^T A V_q \widetilde{\boldsymbol{y}}(t) + V_q^T \boldsymbol{F}(V_q \widetilde{\boldsymbol{y}}(t)) \\ &= \widehat{A} \widetilde{\boldsymbol{y}}(t) + V_q^T \boldsymbol{F}(V_q \widetilde{\boldsymbol{y}}(t)) \end{aligned} \tag{3.30}$$

where we used the property $V_q^T V_q = I$, with $I \in \mathbb{R}^{q \times q}$ being the identity matrix, since the basis $\{\boldsymbol{\varphi}_i\}_{i=1}^q$ is orthonormal.

Note that matrix $\widehat{A} := V_q^T A V_q \in \mathbb{R}^{q \times q}$ does not depend on time, thus it can be computed once at the beginning of the simulation. Also note that both the left-hand side and the linear term in (3.30) have reduced dimension $q \ll M_f$ w.r.t. the FOM (3.25). However, the nonlinear term is still expensive to compute: it needs to be evaluated in M_f points in each time step, since $V_q \widetilde{\boldsymbol{y}} \in \mathbb{R}^{M_f}$. It motivates the use of an alternative model order reduction approach for treating this term.

3.4.2 The empirical interpolation method (EIM): reduction of the nonlinear term

The reduction of the nonlinear term of eq. (3.25) can be achieved with the Empirical Interpolation Method (EIM). First proposed by Barrault et al. (2004), EIM is a reduced-basis approach consisting in a greedy determination of a low-dimensional space and a set of interpolation points and functions for computing the approximation of a parameter-dependent function in this space. The method proposed by Barrault et al. (2004) is applicable to problems with a nonaffine parameter dependency; more general frameworks and applications, including to the order reduction of PDEs models, are presented by Grepl et al. (2007) and Maday et al. (2008). A particular and simplified case of the EIM, using information provided by a POD, is presented by Chaturantabut and Sorensen (2010) under the name of POD-Discrete Empirical Interpolation Method (POD-DEIM), to which we refer indistinguishably as POD-EIM. These methods allow to reduce the complexity of nonlinear terms in reduced models, an unresolved issue in POD ROMs, as discussed above. Due to the introduction of a second stage of complexity reduction, EIM and POD-DEIM are usually referred as "hyperreduction methods" (Hernández et al., 2017), a class of model reduction approaches among which one of the best known is the Gappy-POD (Everson and Sirovich, 1995).

In the following paragraphs we briefly present the general and particular frameworks of the EIM and the POD-DEIM, respectively, pointing out the connection between them. For that, we propose as far as possible a unified notation for both presentations. In the rest of this manuscript, we consider the simplified formulation proposed by Chaturantabut and Sorensen (2010).

The empirical interpolation method

Let $f = f(\boldsymbol{x};\mu)$ be a parameter-dependent function, with $\boldsymbol{x} \in \Omega$ and $\mu \in \mathcal{D}$ being respectively the spatial variable and a set of parameters on which f depends. The objective of the EIM is to approximate f by

$$f(\boldsymbol{x};\mu) \approx f_m(\boldsymbol{x};\mu) := \sum_{i=1}^{m} c_i(\mu)\widehat{\varphi}_i(\boldsymbol{x}) \tag{3.31}$$

where $\widehat{\varphi}_i(\boldsymbol{x}), i = 1,\dots,m$, span the approximation space $\widehat{\mathcal{S}}_m$ of dimension m containing f_m, and $c_i(\mu), i = 1,\dots,m$, are coefficients determined via interpolation by solving

$$B_m\boldsymbol{c}(\mu) = \boldsymbol{f}(\mu) \tag{3.32}$$

where $B_m \in \mathbb{R}^{m\times m}$, $\boldsymbol{c}(\mu) = (c_1(\mu),\ \dots,\ c_m(\mu))^T$ and $\boldsymbol{f}(\mu) = (f(\widehat{\boldsymbol{x}}_1;\mu),\ \dots,\ f(\widehat{\boldsymbol{x}}_m;\mu)^T$, with $\widehat{\boldsymbol{x}}_i, i = 1,\dots,m$, being a set of chosen interpolation points.

The EIM consists of two greedy procedures:

1. In the first greedy procedure (Algorithm 3), a set of parameters samples $\widehat{Y}_m := \{\mu_1,\dots,\mu_m\}$ is determined and used for constructing the approximation space $\widehat{\mathcal{S}}_m = \text{span}\{\xi_i(\boldsymbol{x}) := f(\boldsymbol{x};\mu_i), i = 1,\dots,m\}$, spanned by the function f computed on these parameters. Note that, for all $l = 2,\dots,m$, the chosen parameters are those that maximize the error between f and its best approximation in the intermediate space $\widehat{\mathcal{S}}_{l-1}$; the idea is to keep the parameters that introduce more "new information" to the approximation space;

2. In the second greedy procedure (Algorithm 4), a set of interpolation points $T = \{\widehat{\boldsymbol{x}}_1,\dots,\widehat{\boldsymbol{x}}_m\}$ and interpolation functions $\widehat{\varphi}_i, i = 1,\dots,m$, constructed from $\xi_i, i = 1,\dots,m$, are determined. Each iteration $l = 2,\dots,m$, begins by solving a $(l-1)\times(l-1)$ linear system, which provides the coefficients vector $\boldsymbol{\sigma}$ for writing the new input function ξ_l, computed in the already chosen spatial points, as a linear combination of the already constructed functions $\widehat{\varphi}_1,\dots,\widehat{\varphi}_{l-1}$. That is, $\boldsymbol{\sigma}$ provides exact interpolation of ξ_l in the already chosen points. Then, a residual r_l is defined as the difference between ξ_l and a full spatial interpolation using the same coefficients vector $\boldsymbol{\sigma}$. Similarly to the previous greedy algorithm, the new chosen spatial point $\boldsymbol{x}_l$ is the one that maximizes the residual r_l, thus introducing more information to the interpolation procedure.

```
Input: The dimension $m$ of the approximation space; a finite-dimensional subset $\Xi \subset \mathcal{D}$ of the
  parameter's domain
Output: a set $\widehat{Y}_m := \{\mu_1,\dots,\mu_m\} \subset \Xi$ of parameters instances; the function $f$ computed on
  these parameters $(\xi_i(\boldsymbol{x}) := f(\boldsymbol{x};\mu_i), i = 1,\dots,m)$; the approximation space $\widehat{\mathcal{S}}_m$
$\mu_1 = \text{argmax}_{\mu\in\Xi}\{\|f(\cdot;\mu)\|_{L^\infty(\Omega)}\}$
$\xi_1(\boldsymbol{x}) = f(\boldsymbol{x},\mu_1)$
$\widehat{Y}_1 = \{\mu_1\}$
$\widehat{\mathcal{S}}_1 = \text{span}\{\xi_1(\boldsymbol{x})\}$
for $l \leftarrow 2$ to $m$ do
    Find the best approximation of $f(\cdot;\mu)$ in $\widehat{\mathcal{S}}_{l-1}$: $f^*_{l-1}(\cdot;\mu) = \text{argmin}_{z\in\widehat{\mathcal{S}}_{l-1}} \|f(\cdot,\mu) - z\|_{L^\infty(\Omega)}$
    Compute the associated error $\varepsilon^*_{l-1}(\mu) = \left\|f(\cdot;\mu) - f^*_{l-1}(\cdot;\mu)\right\|_{L^\infty(\Omega)}$
    Set $\mu_l = \text{argmax}_{\mu\in\Xi}\{\varepsilon^*_{l-1}(\mu)\}$
    $\xi_l(\boldsymbol{x}) = f(\boldsymbol{x},\mu_l)$
    $\widehat{Y}_l = \widehat{Y}_{l-1} \cup \{\mu_l\}$
    $\widehat{\mathcal{S}}_l = \text{span}\{\xi_j(\boldsymbol{x}), j = 1,\dots,l\}$
end
```

Algorithm 3: EIM: greedy algorithm for choosing the parameters instances and constructing the approximation space

```
Input: The approximation space $\widehat{\mathcal{S}}_m$; the functions $\xi_i(\boldsymbol{x}) := f(\boldsymbol{x};\mu_i), i = 1,\ldots,m$
Output: a set of interpolation points $T = \{\widehat{\boldsymbol{x}}_1,\ldots,\widehat{\boldsymbol{x}}_m\} \subset \Omega$ and interpolation functions
    $\widehat{\varphi}_i = \widehat{\varphi}_i(\boldsymbol{x}), i = 1,\ldots m$, and a matrix $B_m \in \mathbb{R}^{m\times m}$
$\widehat{\boldsymbol{x}}_1 = \text{argsup}_{\boldsymbol{x}\in\Omega}|\xi_1(\boldsymbol{x})|$
$\xi_1(\boldsymbol{x}) = f(\boldsymbol{x},\mu_1)$
$\widehat{\varphi}_1(\boldsymbol{x}) = \frac{\xi_1(\boldsymbol{x})}{\xi_1(\widehat{\boldsymbol{x}}_1)}$
Define $B_1 \in \mathbb{R}^{1\times 1}$, with $[B_1]_{1,1} = 1$
for $l \leftarrow 2$ to $m$ do
    Solve $B_{l-1}\boldsymbol{\sigma} = \boldsymbol{\xi}$ for $\boldsymbol{\sigma}$, with $\boldsymbol{\xi} = (\xi_l(\widehat{\boldsymbol{x}}_1),\ \ldots,\ \xi_l(\widehat{\boldsymbol{x}}_{l-1}))^T \in \mathbb{R}^{l-1}$
    Define the residual $r_l(\boldsymbol{x}) = \xi_l(\boldsymbol{x}) - \boldsymbol{\sigma}\cdot\widehat{\boldsymbol{\varphi}}(\boldsymbol{x})$, where $\widehat{\boldsymbol{\varphi}}(\boldsymbol{x}) = (\widehat{\varphi}_1(\boldsymbol{x}),\ \ldots,\ \widehat{\varphi}_{l-1}(\boldsymbol{x}))^T$
    Set $\widehat{\boldsymbol{x}}_l = \text{argsup}_{\boldsymbol{x}\in\Omega}|r_l(\boldsymbol{x})|$
    $\widehat{\varphi}_l(\boldsymbol{x}) = \frac{r_l(\boldsymbol{x})}{r_l(\widehat{\boldsymbol{x}}_l)}$
    Define $B_l \in \mathbb{R}^{l\times l}$, with $[B_l]_{i,j} = \widehat{\varphi}_j(\widehat{\boldsymbol{x}}_i)$
end
```

Algorithm 4: EIM: greedy algorithm for constructing the interpolation functions and points

The discrete empirical interpolation method (POD-DEIM)

The POD-DEIM approach, proposed by Chaturantabut and Sorensen (2010), is a simplified and particular case of the EIM in the sense that:

- The DEIM is formulated directly on the discrete level, *i.e.* it seeks an approximation of the vector $\boldsymbol{f}(\mu) = \big(f(\boldsymbol{x}_1;\mu),\ \ldots,\ f(\boldsymbol{x}_{M_f};\mu)\big)^T$ containing the function f computed at the M_f spatial points corresponding to the degrees-of-freedom of the full-order problem;

- The basis vectors of $\widehat{\mathcal{S}}_m$ and interpolation functions $\widehat{\boldsymbol{\varphi}}_i \in \mathbb{R}^{M_f}, i = 1,\ldots,m$, are equal and correspond to the orthonormal modes given by a POD applied to snapshots of $\boldsymbol{f}$. Therefore, the first greedy algorithm of the EIM (Algorithm 3) is replaced by a POD, and only the second one (Algorithm 4), for choosing the interpolation points, has an equivalent in the DEIM (Algorithm 5)

```
Input: vectors $\{\widehat{\boldsymbol{\varphi}}_l\}_{l=1}^m \subset \mathbb{R}^{M_f}$ of a POD reduced basis
Output: the indices $\mathcal{P} = (\mathcal{P}_1,\ldots,\mathcal{P}_m)^T \in \mathbb{R}^m$; the matrix $\widehat{P} \in \mathbb{R}^{M_f\times m}$
$\mathcal{P}_1 = \text{argmax}\{|\widehat{\boldsymbol{\varphi}}_1|\}$
$\widehat{V} = [\widehat{\boldsymbol{\varphi}}_1]$, $\widehat{P} = [\boldsymbol{e}_{\mathcal{P}_1}]$, $\mathcal{P} = [\mathcal{P}_1]$
for $l \leftarrow 2$ to $m$ do
    Solve $(\widehat{P}^T\widehat{V})\boldsymbol{\sigma} = \widehat{P}^T\widehat{\boldsymbol{\varphi}}_l$ for $\boldsymbol{\sigma}$
    Define the residual $\boldsymbol{r}_l = \widehat{\boldsymbol{\varphi}}_l - \widehat{V}\boldsymbol{\sigma}$
    Choose the new spatial point: $\mathcal{P}_l = \text{argmax}\{|\boldsymbol{r}_l|\}$
    $\widehat{V} \leftarrow [\widehat{V}\ \widehat{\boldsymbol{\varphi}}_l]$, $\widehat{P} \leftarrow [\widehat{P}\ \boldsymbol{e}_{\mathcal{P}_l}]$, $\mathcal{P} \leftarrow \begin{pmatrix} \mathcal{P} \\ \mathcal{P}_l \end{pmatrix}$
end
```

Algorithm 5: DEIM algorithm

The POD applied to the snapshots matrix $\widehat{Y} = [\boldsymbol{f}(\mu_1),\ldots,\boldsymbol{f}(\mu_{n_s})] \in \mathbb{R}^{M_f\times n_s}$ provides a set of m orthonormal vectors $\{\widehat{\boldsymbol{\varphi}}_i\}_{i=1}^m \subset \mathbb{R}^{M_f}$, which form a basis of a subspace $\widehat{\mathcal{S}}_m \subset \mathbb{R}^{M_f}$ and define the columns of the matrix $\widehat{V}_m = [\widehat{\boldsymbol{\varphi}}_1,\ldots,\widehat{\boldsymbol{\varphi}}_m] \in \mathbb{R}^{M_f\times m}$. In the application considered here, $\boldsymbol{f}$ is a time-dependent function, such that $\mu_i = t_i, i = 1,\ldots,n_s$. In the DEIM algorithm (Algorithm 5), m spatial interpolation points are chosen greedily as in the EIM (the points maximizing a residual and containing more "new information" are kept). Their indices are denoted by $\mathcal{P}_l \in \{1,\ldots,M_f\}, l = 1,\ldots,m$, and, as output, the algorithm gives a matrix $\widehat{P} \in \mathbb{R}^{M_f\times m}$ whose column l, for every $l = 1,\ldots,m$, is the $\mathcal{P}_l-$th canonical vector of $\mathbb{R}^{M_f}$, denoted by $\boldsymbol{e}_{\mathcal{P}_l}$. Note that a left-hand multiplication of a vector $\boldsymbol{v} \in \mathbb{R}^{M_f}$ by $\widehat{P}^T$ returns a vector in $\mathbb{R}^m$ containing the elements of $\boldsymbol{v}$ with indices $\mathcal{P}_1,\ldots,\mathcal{P}_m$.

Therefore, the POD-DEIM approximation of $\boldsymbol{f}$ reads

$$\boldsymbol{f}(\mu) \approx \boldsymbol{f}_m(\mu) := \widehat{V}_m \boldsymbol{c}(\mu) \tag{3.33}$$

analogously to (3.31) in the EIM. The coefficients vector $\boldsymbol{c}(\mu) \in \mathbb{R}^m$ can be uniquely determined by selecting m rows with indices $\mathcal{P}_1, \dots, \mathcal{P}_m$ (*i.e.* by left-multiplying by $\widehat{P}^T$) from the overdetermined system

$$\widehat{V}_m \boldsymbol{c}(\mu) = \boldsymbol{f}(\mu)$$

resulting in the linear system

$$\left(\widehat{P}^T \widehat{V}_m\right) \boldsymbol{c}(\mu) = \widehat{P}^T \boldsymbol{f}(\mu) \tag{3.34}$$

Note that the linear systems (3.32) and (3.34), for the EIM and the DEIM, respectively, are equivalent: their right-hand sides correspond to the function $\boldsymbol{f}$ computed at the chosen interpolation points, and, in the left-hand side, the matrix multiplying the unknown vector (respectively $B_m \in \mathbb{R}^{m \times m}$ and $\widehat{P}^T \widehat{V}_m \in \mathbb{R}^{m \times m}$) contains in each column $j = 1, \dots, m$, the interpolation function $\widehat{\varphi}_j$ computed on the m interpolation points (or, equivalently, the POD modes $\widehat{\boldsymbol{\varphi}}_j$ restricted to the spatial indices $\mathcal{P}_1, \dots, \mathcal{P}_m$).

We now proceed to the introduction of the POD-DEIM or POD-EIM for reducing (3.25). Supposing $\widehat{P}^T \widehat{V}_m$ to be nonsingular, we replace the linear system (3.34) into (3.33) for obtaining

$$\boldsymbol{f}(\mu) \approx \widehat{V}_m \left(\widehat{P}^T \widehat{V}_m\right)^{-1} \widehat{P}^T \boldsymbol{f}(\mu) \tag{3.35}$$

Taking $\boldsymbol{f}(\mu) = \boldsymbol{F}(V_q \widetilde{\boldsymbol{y}}(t))$ and replacing into (3.30) yields the *POD-EIM reduced-order model*

$$\begin{aligned} \frac{d}{dt} \widetilde{\boldsymbol{y}}(t) &= V_q^T A V_q \widetilde{\boldsymbol{y}}(t) + V_q^T \widehat{V}_m \left(\widehat{P}^T \widehat{V}_m\right)^{-1} \widehat{P}^T \boldsymbol{F}(V_q \widetilde{\boldsymbol{y}}(t)) \\ &= \widehat{A} \widetilde{\boldsymbol{y}}(t) + \widehat{B} \widehat{P}^T \boldsymbol{F}(V_q \widetilde{\boldsymbol{y}}(t)) \end{aligned} \tag{3.36}$$

Note that matrix $\widehat{B} := V_q^T \widehat{V}_m \left(\widehat{P}^T \widehat{V}_m\right)^{-1} \in \mathbb{R}^{q \times m}$, analogously to $\widehat{A}$, does not depend on time and can be precomputed. Moreover, $\widehat{P}^T \boldsymbol{F}(V_q \widetilde{\boldsymbol{y}}(t))$ is a vector of size $m \ll M_f$. In the case where $\boldsymbol{F}$ is evaluated pointwise, the last term of (3.33) can also be written as $\widehat{B} \boldsymbol{F}(\widehat{P}^T V_q \widetilde{\boldsymbol{y}}(t))$. Eq. (3.35) means that the nonlinear term is approximated by its computation on only m points, thus avoiding its evaluation on M_f points, as in the POD-based ROM (3.30). Ştefănescu et al. (2014) illustrate the advantages, in terms of computational time, of using POD-EIM instead of POD for approximating high-dimensional nonlinear systems.

3.4.3 Introduction of ROMs in the parareal method

The ROM-based parareal iteration

Inspired by the Krylov subspace-enhanced parareal method (Algorithm 2), Chen et al. (2014) propose to introduce reduced-order models formulated using the techniques described above, thus allowing an efficient application to nonlinear hyperbolic problems (and also linear ones). The idea is to replace the modified coarse propagator $\mathcal{K}_{\Delta t}$, defined in Algorithm 2, by the reduced-order model:

$$\widehat{\mathcal{K}}^k_{\Delta t}(\boldsymbol{y}, t_{n+1}, t_n) := \mathcal{F}^k_{r,\delta t}(\mathbb{P}^k \boldsymbol{y}, t_{n+1}, t_n) \tag{3.37}$$

Thus, the ROM-based parareal iteration reads

$$\boldsymbol{y}^{k+1}_{n+1} = \underbrace{\mathcal{F}^k_{r,\delta t}(\mathbb{P}^k \boldsymbol{y}^{k+1}_n, t_{n+1}, t_n)}_{\text{Prediction}} + \underbrace{\mathcal{F}_{\delta t}(\boldsymbol{y}^k_n, t_{n+1}, t_n) - \mathcal{F}^k_{r,\delta t}(\mathbb{P}^k \boldsymbol{y}^k_n, t_{n+1}, t_n)}_{\text{Correction}}, \qquad n = 0, \dots, N_{\Delta T} - 1 \tag{3.38}$$

The 0-th iteration is not changed, being provided by a sequential simulation (3.7) of the coarse propagator $\mathcal{G}_{\Delta t}$. Therefore, in the ROM-based parareal method, $\mathcal{G}_{\Delta t}$ is only used for computing the initial prediction, being replaced by reduced-order models in the following iterations. The detailed method is presented in Algorithm 6, with the modifications w.r.t. the Krylov subspace-enhanced parareal method

(Algorithm 2) being highlighted. We omit (as well as in the following of this work), the subindices indicating the dimension of the subspaces.

The advantage of $\widehat{\mathcal{K}}_{\Delta t}$ compared to $\mathcal{K}_{\Delta t}$ is that the fine propagator, present in the definition of the latter, is replaced by a reduced, thus cheaper, one. Since $\widehat{\mathcal{K}}_{\Delta t}$ must be computed sequentially in the prediction step of the parareal algorithm, a full sequential simulation of the expensive, fine model is avoided. Also note that the second term of $\mathcal{K}_{\Delta t}$ (coarse propagation of $(\mathbb{I} - \mathbb{P}^k)\boldsymbol{y}$) is not present in $\widehat{\mathcal{K}}_{\Delta t}$. It is justified by Chen et al. (2014) by supposing that the projection error vanishes asymptotically.

Simulation of the reduced-order models

The operator $\mathbb{P}^k$ in (3.37) is a projection operator for approximating $\boldsymbol{y} \in \mathbb{R}^{M_f}$ in a subspace $\mathcal{S}^k$ of dimension $q \ll M_f$ defining the reduced model and indicates that $\mathcal{F}^k_{r,\delta t}$ is solved for the coordinates of $\mathbb{P}^k\boldsymbol{y}$ in $\mathcal{S}^k$ (as in eq. (3.36)). Thus, in the simulation of (3.37), the following steps are performed:

1. Compute the reduced variables, *i.e.* the coefficients of the approximation of $\boldsymbol{y}$ in $\mathcal{S}^k$: $\widetilde{\boldsymbol{y}}(t_n) = (V^k)^T\boldsymbol{y}(t_n) \in \mathbb{R}^q$;

2. Solve the ROM for $\widetilde{\boldsymbol{y}}$ from t_n to t_{n+1} in the reduced space;

3. Return to the physical space: $\boldsymbol{y}(t_{n+1}) \approx V^k\widetilde{\boldsymbol{y}}(t_{n+1})$.

where $V^k \in \mathbb{R}^{M_f \times q}$ is a matrix whose columns are the vectors of an orthonormal basis of $\mathcal{S}^k$. Note that, in the case of a combined POD-EIM reduced model, it is also defined by a subspace $\widehat{\mathcal{S}}^k$, obtained from snapshots of the nonlinear term of the problem. Moreover, depending on the problem, the reduced model may be defined by multiple subspaces $\mathcal{S}^k$ and $\widehat{\mathcal{S}}^k$. It is the case of the SWE (2.35), whose reduced-order model is defined by three subspaces $\mathcal{S}^k$ and five subspaces $\widehat{\mathcal{S}}^k$, obtained respectively via POD and POD-EIM. A detailed formulation of the reduced SWE is presented in Section 3.5.

The notation $\mathcal{F}^k_{r,\delta t}$ in (3.37) is chosen to indicate that the reduced model is solved with the same time step δt associated to the fine propagator $\mathcal{F}_{\delta t}$. Therefore, instead of using the coarse propagator, with large time steps Δt, for computing the parareal predictions, one uses a ROM that can be seen as an approximation of $\mathcal{F}_{\delta t}$. By solving both $\mathcal{F}_{\delta t}$ and $\mathcal{F}^k_{r,\delta t}$ with the same time step, one can expect to reduce the mismatch of their discrete phase speeds, which, as discussed in Section 3.2.4, was identified by Ruprecht (2018) to be at the origin of instabilities of the parareal method when applied to hyperbolic problems.

Formulation of reduced-order models in the parareal framework

Chen et al. (2014) propose two formulations of the ROM-based parareal algorithm, for linear and nonlinear problems, the first one using POD and the second one using EIM or POD-EIM. We focus here on the ROM-based parareal method for nonlinear problems using POD-EIM, as indicated in Algorithm 6. Note that the ROMs are reformulated at each parareal iteration (which explains the superindex k in $\mathcal{F}^k_{r,\delta t}$) using snapshots collected at all parareal time instants and all previous iterations (as in the Krylov subspace-enhanced parareal method).

The subspaces $\mathcal{S}^k$ and $\widehat{\mathcal{S}}^k$ are obtained respectively from snapshots of the fine correction term (propagated from the parareal solution along each time slice) and the nonlinear function computed on them, *i.e.*

$$Y^k := \{\widetilde{\boldsymbol{y}}^j_n,\ j = 0, \dots, k;\ n = 0, \dots, N_{\Delta T}\}, \qquad \widehat{Y} := \{\boldsymbol{F}(\widetilde{\boldsymbol{y}}^j_n),\ j = 0, \dots, k;\ n = 0, \dots, N_{\Delta T}\}$$

where Y and $\widehat{Y}$ are the snapshots sets used respectively in the POD and the POD-EIM procedures. Along this work, we refer to Y and $\widehat{Y}$ indistinguishably as snapshots sets or snapshots matrices. Since the ROM is seen as an approximation for the fine, reference model, the idea is to use snapshots produced using $\mathcal{F}_{\delta t}$. Therefore, the snapshots are expected to be representative of the fine dynamics.

For expliciting the inputs for the model reduction, we define, in Algorithm 6, the POD and POD-EIM as functions of the snapshots sets and thresholds for the POD basis truncation. These thresholds may be different for the two types of subspaces and, with some abuse of nomenclature, are called respectively as "linear" and "nonlinear".

We also remark that, as in the Krylov-subspace-enhanced parareal method, it is not possible to implement a scheduling of parallel tasks, as proposed by Aubanel (2011), in the ROM-based framework. In each iteration, the reduced-order models are formulated using the fine correction terms, such that the fine and coarse propagations cannot overlap.

```
**Initialization**: initial guess given by the coarse propagator:
$\boldsymbol{y}_0^0 = \boldsymbol{y}_0$
**for** $n \leftarrow 0$ **to** $N_{\Delta T} - 1$ **do**
    $\boldsymbol{y}_{n+1}^0 = \mathcal{G}_{\Delta t}(\boldsymbol{y}_n^0, t_{n+1}, t_n)$
**end**

$n_0 = 0$
**Iterations**:
**for** $k \leftarrow 0$ **to** $N_{itermax} - 1$ **do**
    Compute the fine term of the correction (**in parallel**):
    **for** $n \leftarrow n_0$ **to** $N_{\Delta T} - 1$ **do**
        $\widetilde{\boldsymbol{y}}_{n+1}^k = \mathcal{F}_{\delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)$
    **end**

    Find the first instant $\tilde{n} \in \{1, \dots, N_{\Delta T}\}$ not satisfying a convergence criterion based on $\boldsymbol{y}_n^k, \widetilde{\boldsymbol{y}}_n^k$
    $n_0 \leftarrow \tilde{n} - 1$

    Define the snapshots sets:
    $Y^k = \{\widetilde{\boldsymbol{y}}_n^j,\ j = 0, \dots, k;\ n = 0, \dots, N_{\Delta T}\}$
    $\widehat{Y}^k = \{\boldsymbol{F}(\widetilde{\boldsymbol{y}}_n^j),\ j = 0, \dots, k;\ n = 0, \dots, N_{\Delta T}\}$

    Compute the spaces and define the reduced model $\mathcal{F}_{r,\delta t}^k$:
    $\mathcal{S}^k(Y^k, \varepsilon_{\text{sv,linear}})$ (using POD)
    $\widehat{\mathcal{S}}^k(\widehat{Y}^k, \varepsilon_{\text{sv,nonlinear}})$ (using POD-EIM)

    Compute the coarse term of the correction and the final correction term(**in parallel**):
    **for** $n \leftarrow n_0$ **to** $N_{\Delta T} - 1$ **do**
        $\overline{\boldsymbol{y}}_{n+1}^k = \mathcal{F}_{r,\delta t}^k(\mathbb{P}^k \boldsymbol{y}_n^k, t_{n+1}, t_n)$
        $\overline{\overline{\boldsymbol{y}}}_{n+1}^k = \widetilde{\boldsymbol{y}}_{n+1}^k - \overline{\boldsymbol{y}}_{n+1}^k$
    **end**

    Compute the coarse predictions and correct them to obtain the final solution in the iteration
     (**sequentially**):
    **for** $n \leftarrow n_0$ **to** $N_{\Delta T} - 1$ **do**
        $\boldsymbol{y}_{n+1}^{k+1} = \mathcal{F}_{r,\delta t}^k(\mathbb{P}^k \boldsymbol{y}_n^{k+1}, t_{n+1}, t_n) + \overline{\overline{\boldsymbol{y}}}_{n+1}^k$
    **end**

    **if** *all instants converged* ($\tilde{n} = N_{\Delta T}$) **then**
        break;
    **end**
**end**
```

Algorithm 6: ROM-based parareal algorithm. Modifications w.r.t. the Krylov subspace-enhanced parareal method (Algorithm 2) are highlighted.

An on-the-fly model reduction procedure

It should be noted that the ROM formulation in the parareal framework is quite different from classical model reduction procedures. In general, model reduction relies on an offline and an online stages (Salmoiraghi et al., 2016). In the first one, performed as a pre-processing step, the reduced model is formulated: a series of fine, reference simulations are performed, for a chosen set of parameters, in order to obtain the input snapshots sets, from which the reduced subspaces are computed. It is a very expensive stage and may rely on High Performance Computing (HPC) for processing and/or storage needs. The second, online stage consists on the simulation of the reduced model itself, for a new set of parameters. Since the reduced model has small dimension, the online stage is very fast, thus representing the advantage of the ROM w.r.t. the FOM.

Evidently, the model reduction procedure used in the parareal method is also described by an offline-online splitting. However, the ROMs are computed *on-the-fly* and reformulated at each iteration, using updated and more precise information given by the parareal iterations. Therefore, the offline stage must not be expensive, since it would inhibit the interest in using the parareal method. Then, the snapshots sets are not obtained from an expensive set of fine simulations of the reference model, but from the fine correction terms computed along parareal iterations. In comparison to the standard offline-online procedure, this on-the-fly approach has the advantage of producing snapshots sets using the same parameters set as the reference model. However, since the snapshots are obtained along parareal iterations, they can possibly be inaccurate and affect the quality of the formulated ROM, which will be discussed in more details in the next chapter.

Performance of the ROM-based parareal method

Stability and convergence results are derived by Chen et al. (2014) for the ROM-based parareal method, under assumptions on the quality of the ROM w.r.t. the FOM. Numerical simulations presented by the authors on a variety of problems show that the proposed method converges faster in situations where the classical parareal method already converges, but slower convergences are verified in some cases, indicating a dependence on the quality of the ROMs. The method has also been applied by Iizuka and Ono (2018) for studying the influence of phase accuracy in parareal convergence for solving linear mass-spring systems and a linear advection-diffusion equation (with reduced-order models formulated with only POD), and results indicate that convergence is influenced by the POD truncation and the number of input snapshots. In this work, we study the ROM-based method applied to the 2D nonlinear SWE, with ROMs formulated via a combined POD-EIM, and we focus on improving its convergence and stability by improving the ROMs formulated along the parareal iterations. Also, we evaluate the speedup performance of the method in real parallel implementations (even with a relatively small number of processors), which was not studied in the works mentioned above.

Other applications of reduced-order models to the parareal method

As a final remark, we cite other applications of reduced-order models in the framework of the parareal method. He (2010) and Grigori et al. (2021) propose respectively the use of reduced basis methods and reduced-order models obtained from asymptotic expansions for reducing the computational cost of the coarse propagator, but with a previously computed offline stage (and not on-the-fly as considered here). Carlberg et al. (2016) defines the coarse propagator as a local forecast, a technique relying on model reduction and introduced by Carlberg et al. (2015) that consists in performing the fine simulation over a small number number of time steps for predicting the solution at the end of the parareal time slices. However, this method also assumes that a low-dimensional basis is available; for example, a POD basis obtained via a standard offline-online model reduction approach.

3.4.4 Speedup estimation

We formulate in the following paragraphs an estimate for the speedup provided by the ROM-based parareal method (Algorithm 6) at a given final iteration $\hat{k}$. As done for the classical parareal method in Section 3.2.2 and inspired by Ruprecht and Krause (2012) and Chen et al. (2014), we derive and interpret some bounds for the speedup.

We recall the definitions, presented in Section 3.2.2, of τ_c and τ_f as the computational times for advancing one coarse time step Δt using $\mathcal{G}_{\Delta t}$ and one fine time step δt using $\mathcal{F}_{\delta t}$, respectively. Similarly, we define $\tau_r(k)$ as the computational time for advancing one fine time step δt using the reduced propagator $\mathcal{F}^k_{r,\delta t}$ defined at iteration k. The dependence of τ_r on k is due to the reformulation of the reduced-order model at each iteration, using an increasing set of input snapshots.

Concerning the ROM formulation, we define the computational time $\tau_{\mathcal{S}}(k)$ for obtaining, at iteration k, one subspace $\mathcal{S}^k$ (from snapshots of the solution, using POD) or $\widehat{\mathcal{S}}^k$ (from snapshots of the nonlinear term, using POD-EIM). Similarly, we define the computational time $\tau_{\widehat{B}}(\hat{k})$ for obtaining a matrix defining the reduced-order model ($\widehat{A}$ or $\widehat{B}$, see eq. (3.36)). We recall that, depending on the problem, several subspaces and matrices may be defined. Let N_{spaces} and N_{matrices} be respectively the total number of subspaces and matrices to be computed at each iteration. In the case of the two-dimensional nonlinear SWE, $N_{\text{spaces}} = 8$ and $N_{\text{matrices}} = 9$ (with only "$\widehat{B}$-type" matrices, thus the chosen notation for $\tau_{\widehat{B}}$), as detailed in Section 3.5. Moreover, since the formulations of the subspaces and matrices are independent, they can possibly be distributed to $N_{p,r}$ processors (or groups of processors), as discussed in Appendix B.2.

The computational time T_{ref} for the fine, sequential simulation is defined in (3.9). For the ROM-based parareal method, the following steps compose the total computational time $T_{\text{ROM-parareal}}(\hat{k})$:

- an initial full coarse prediction, using $\mathcal{G}_{\Delta t}$ and taking $T_c = N_{\Delta t}\tau_c$;
- within each iteration k:
  - the parallel computation of the fine term of the correction, taking
$$T_{\text{corr},f} = \frac{N_{\Delta T}}{N_p} p_{\delta t}\tau_f = \frac{N_{\Delta T}}{N_p}\frac{N_{\delta t}}{N_{\Delta T}}\tau_f = \frac{N_{\delta t}}{N_p}\tau_f$$
  - the possibly parallel formulation of the reduced-order models (subspaces and matrices), taking
$$T_{\text{ROM}} = \frac{1}{N_{p,r}}\left(N_{\text{spaces}}\tau_{\mathcal{S}}(k) + N_{\text{matrices}}\tau_{\widehat{B}}(k)\right)$$
  - the parallel computation of the coarse term of the correction, using the ROM $\mathcal{F}^k_{r,\delta t}$ and taking
$$T_{\text{corr},c} = \frac{N_{\Delta T}}{N_p} p_{\delta t}\tau_r(k) = \frac{N_{\Delta T}}{N_p}\frac{N_{\delta t}}{N_{\Delta T}}\tau_r(k) = \frac{N_{\delta t}}{N_p}\tau_r(k)$$
  - the sequential computation of the predictions, using the ROM $\mathcal{F}^k_{r,\delta t}$ and taking
$$T_{\text{pred}} = N_{\delta t}\tau_r(k)$$

Therefore,

$$\begin{aligned} T_{\text{ROM-parareal}}(\hat{k}) &= T_c + \hat{k}\left(T_{corr,f} + T_{\text{ROM}} + T_{corr,c} + T_{pred}\right) \\ &= N_{\Delta t}\tau_c + \hat{k}\left[\frac{N_{\delta t}}{N_p}\left(\tau_f + \tau_r(\hat{k})\right) + \frac{N_{\text{spaces}}\tau_{\mathcal{S}}(\hat{k}) + N_{\text{matrices}}\tau_{\widehat{B}}(\hat{k})}{N_{p,r}} + N_{\delta t}\tau_r(\hat{k})\right] \end{aligned} \tag{3.39}$$

where $\tau_{\mathcal{S}}$, τ_r and $\tau_{\widehat{B}}$ are majorated by their costs at the last iteration $\hat{k}$, which is expected due to the increasing amount of snapshots used for the ROM formulation.

Then, the speedup provided by the ROM-based parareal method is estimated as

$$\begin{aligned} s_{\text{ROM-parareal}}(\hat{k}) &= \frac{T_{\text{ref}}}{T_{\text{ROM-parareal}}(\hat{k})} \\ &= \frac{1}{\frac{N_{\Delta t}\tau_c}{N_{\delta t}\tau_f} + \frac{\hat{k}}{N_p}\frac{\tau_f + \tau_r(\hat{k})}{\tau_f} + \hat{k}\frac{\tau_r(\hat{k})}{\tau_f} + \hat{k}\frac{N_{\text{spaces}}\tau_{\mathcal{S}}(\hat{k}) + N_{\text{matrices}}\tau_{\widehat{B}}(\hat{k})}{N_{p,r}N_{\delta t}\tau_f}} \end{aligned} \tag{3.40}$$

We now derive and interpret some simple bounds for (3.40). We recall the assumptions $\tau_c = \mathcal{O}(M_c)$ and $\tau_f = \mathcal{O}(M_f)$. Similarly, we assume that the computational cost $\tau_r(\hat{k})$ associated to the reduced-order model satisfies $\tau_r(\hat{k}) = \mathcal{O}(\hat{m})$, where $\hat{m}$, as a simple majoration, is the largest dimension among all subspaces computed along the parareal simulation.

We then have the following bounds:

- Bound due to the initial prediction:

$$s_{\text{ROM-parareal}}(\hat{k}) < \frac{N_{\delta t}\tau_f}{N_{\Delta t}\tau_c} = \frac{N_{\delta t}M_f}{N_{\Delta t}M_c} \tag{3.41}$$

  which is analogous to the classical parareal bound (3.12) but reflects the fact that the coarse propagator $\mathcal{G}_{\Delta t}$ is used only in the $0-$th iteration of the ROM-based parareal method. At first sight, it indicates that the coarsening between $\mathcal{F}_{\delta t}$ and $\mathcal{G}_{\Delta t}$ is less determinant, in terms of speedup, than in the classical parareal. However, a too inaccurate initial prediction may lead to the formulation of also inaccurate snapshots, impacting the quality of the reduced-order models and slowing down the convergence.

- Bound due to the complexity of the fine and reduced propagators: the coarse prediction term (for $k \geq 1$) yields

$$s_{\text{ROM-parareal}}(\hat{k}) < \frac{\tau_f}{\hat{k}\tau_r(\hat{k})} = \frac{M_f}{\hat{k}\hat{m}} \tag{3.42}$$

  This bound plays the same role as bound (3.41) for $k \geq 1$ and points out the importance of formulating low-dimensional reduced-order models. Moreover, the factor $1/\hat{k}$ indicates the limitation of the speedup by the number of iterations;

- Bound due to the number of processors and iterations: contrary to the classical parareal, in which only the fine term is computed in the correction step (the coarse one being already available from the previous iteration), in the ROM-based method the coarse correction term must also be computed, since it is given by the ROM formulated at the current iteration. Then, the correction term yields

$$s_{\text{ROM-parareal}}(\hat{k}) < \frac{N_p}{\hat{k}}\frac{\tau_f}{\tau_f + \tau_r(\hat{k})} = \frac{N_p}{\hat{k}}\frac{M_f}{M_f + \hat{m}} \approx \frac{N_p}{\hat{k}} \tag{3.43}$$

  in which the last equality is approximated by assuming, as intended, that $\hat{m} \ll M_f$, thus $\tau_r(\hat{k}) \ll \tau_f$. Under this assumption, the approximate bound (3.42) is equal to (3.13).

- Bound due to the formulation of the reduced-order models:

$$s_{\text{ROM-parareal}}(\hat{k}) < \frac{N_{p,r}N_{\delta t}\tau_f}{\hat{k}\left(N_{\text{spaces}}\tau_{\mathcal{S}}(\hat{k}) + N_{\text{matrices}}\tau_{\widehat{\mathcal{B}}}(\hat{k})\right)} \tag{3.44}$$

  with the computational times $\tau_{\mathcal{S}}(k)$ and $\tau_{\widehat{\mathcal{B}}}(k)$ (respectively for formulating the reduced subspaces and ROM matrices) being estimated by

$$\tau_{\mathcal{S}} = \mathcal{O}\left(M_f\hat{k}^2N_{\Delta T}^2 + \frac{\hat{m}^4}{4} + \hat{m}M_f\right) \tag{3.45}$$

$$\tau_{\widehat{\mathcal{B}}} = \mathcal{O}\left(2\hat{m}^2M_f\right) \tag{3.46}$$

  A detailed derivation of estimates (3.45) and (3.46) is given in Appendix B.3.

  Thus, (3.44), can be rewritten as

$$s_{\text{ROM-parareal}}(\hat{k}) < \frac{N_{p,r}N_{\delta t}M_f}{\hat{k}\left[N_{\text{spaces}}\left(M_f\hat{k}^2N_{\Delta T}^2 + \frac{\hat{m}^4}{4} + \hat{m}M_f\right) + 2N_{\text{matrices}}\hat{m}^2M_f\right]} \tag{3.47}$$

  showing a strong dependence of the speedup on the dimension of the reduced model, and also on the number of iterations, since the number of snapshots, $k(N_{\Delta T}+1)$, increases linearly along iterations. Indeed, the term $\hat{k}^2N_{\Delta T}^2$ reflects a quadratic dependence on the number of snapshots. As a last remark, we note that bound (3.47) can be improved by computing the spaces and matrices in parallel (*i.e.* by taking $N_{p,r} > 1$). A discussion on this subject is proposed in Appendix B.2.

3.5 Application of the ROM-based parareal method to the SWE

In this section, we describe the application of the ROM-based parareal algorithm (Algorithm 6) for solving the two-dimensional nonlinear shallow water equations. We firstly describe the formulation of the reduced SWE, using the combined POD-EIM procedure described in Section 3.4.2; then, we implement it in the parareal method and, in the next section, we present some numerical tests for comparing its performance w.r.t. the classical parareal method (Algorithm 1) and also for showing its limitations, which motivates the work presented in the following chapters.

3.5.1 Formulation of the POD-EIM-based reduced SWE

We formulate here a POD-EIM-based reduced-order model for the SWE discretized using a finite volume scheme (eq. 2.34). ROMs using the same model-order reduction technique are formulated for the SWE by Ştefănescu and Navon (2013) and for the spherical SWE by Zhao et al. (2014), both in the framework of a finite difference discretization, and by Lozovskiy et al. (2016) in the case of finite elements.

Since eq. (2.34) is valid for the SWE and all porosity-based SWE presented in Chapter 2, so is the reduced model formulated in the following. However, we are only interested in the reduced-order model for the classical SWE, since it acts as fine propagator, and the model reduction is performed for approximating it.

We begin by identifying (2.34) with equation (3.25), used as starting point for the description of the model reduction in the previous section. Both are systems of time-dependent ODEs, with nonlinear terms on their right-hand sides. Note that there is no linear term in the right-hand side of (2.34), thus simplifying the model reduction. Since (2.34) contains three unknown vectors $\boldsymbol{U}^{(i)} \in \mathbb{R}^{M_f}, i = 1,2,3,$ and five different nonlinear functions $\widetilde{\boldsymbol{F}}^{(i)}(\boldsymbol{U}) \in \mathbb{R}^{I_f}, i = 1,\dots,5$ (where M_f and I_f are respectively the number of cells and interfaces in the mesh), the computation of the following subspaces are required for the model reduction:

- Three subspaces $\mathcal{S}^{(i)}, i = 1,2,3,$ with dimension $q_i \ll M_f$ and obtained respectively from snapshots of the solution $\boldsymbol{U}^{(i)}$ via a POD; their orthonormal basis vectors are the columns of the matrices $V^{(i)} \in \mathbb{R}^{M_f \times q_i}$, respectively;

- Five subspaces $\widehat{\mathcal{S}}^{(i)}, i = 1,\dots,5$ with dimensions $m_i \ll I_f$ and obtained respectively from snapshots of the nonlinear term $\widetilde{\boldsymbol{F}}^{(i)}(\boldsymbol{U})$ via a POD-EIM; their orthonormal basis vectors are the columns of the matrices $\widehat{V}^{(i)} \in \mathbb{R}^{I_f \times m_i}$, respectively, and the matrices containing the chosen spatial indexes via DEIM are denoted as $\widehat{P}^{(i)} \in \mathbb{R}^{I_f \times m_i}$;

The reduced variables are denoted by $\widetilde{\boldsymbol{U}}^{(i)} \in \mathbb{R}^{q_i}, i = 1,2,3,$ and satisfy the approximations $\boldsymbol{U}^{(i)} \approx V^{(i)}\widetilde{\boldsymbol{U}}^{(i)}$. By defining

$$\widetilde{\boldsymbol{U}} := \begin{pmatrix} \widetilde{\boldsymbol{U}}^{(1)} \\ \widetilde{\boldsymbol{U}}^{(2)} \\ \widetilde{\boldsymbol{U}}^{(3)} \end{pmatrix} \in \mathbb{R}^{q_1+q_2+q_3}, \qquad \overline{V} := \begin{pmatrix} V^{(1)} & 0 & 0 \\ 0 & V^{(2)} & 0 \\ 0 & 0 & V^{(3)} \end{pmatrix} \in \mathbb{R}^{3M_f \times (q_1+q_2+q_3)}$$

we can write

$$\overline{V}\widetilde{\boldsymbol{U}} = \begin{pmatrix} V^{(1)}\widetilde{\boldsymbol{U}}^{(1)} \\ V^{(2)}\widetilde{\boldsymbol{U}}^{(2)} \\ V^{(3)}\widetilde{\boldsymbol{U}}^{(3)} \end{pmatrix} \in \mathbb{R}^{3M_f}$$

With these definitions, the reduced-order formulation following (3.36) is straightforward: it suffices to replace $\boldsymbol{U}^{(i)}$ by $V^{(i)}\widetilde{\boldsymbol{U}}^{(i)}, i = 1,2,3,$ in (2.34); project the equations onto their respective associated subspaces $\mathcal{S}^{(i)}$ (left-hand multiplication by $(V^{(i)})^T$); and replace the nonlinear terms $\widetilde{\boldsymbol{F}}^{(j)}(\boldsymbol{U})$ by $\widehat{V}^{(j)}\left((\widehat{P}^{(j)})^T\widehat{V}^{(j)}\right)^{-1}(\widehat{P}^{(j)})^T\widetilde{\boldsymbol{F}}(\overline{V}\widetilde{\boldsymbol{U}}(t)), j = 1,\dots,5$. The obtained ROM reads

$$
\begin{aligned}
\frac{d}{dt}\widetilde{\boldsymbol{U}}^{(1)} &= \widehat{B}^{(1)}\left(\widehat{P}^{(1)}\right)^T \widetilde{\boldsymbol{F}}^{(1)}\left(\overline{V}\widetilde{\boldsymbol{U}}\right)\\
\frac{d}{dt}\widetilde{\boldsymbol{U}}^{(2)} &= \widehat{B}^{(2,1)}\left(\widehat{P}^{(2)}\right)^T \widetilde{\boldsymbol{F}}^{(2)}\left(\overline{V}\widetilde{\boldsymbol{U}}\right) + \widehat{B}^{(2,2)}\left(\widehat{P}^{(3)}\right)^T \widetilde{\boldsymbol{F}}^{(3)}\left(\overline{V}\widetilde{\boldsymbol{U}}\right) +\\
&\quad \widehat{B}^{(2,3)}\left(\widehat{P}^{(4)}\right)^T \widetilde{\boldsymbol{F}}^{(4)}\left(\overline{V}\widetilde{\boldsymbol{U}}\right) + \widehat{B}^{(2,4)}\left(\widehat{P}^{(5)}\right)^T \widetilde{\boldsymbol{F}}^{(5)}\left(\overline{V}\widetilde{\boldsymbol{U}}\right)\\
\frac{d}{dt}\widetilde{\boldsymbol{U}}^{(3)} &= \widehat{B}^{(3,1)}\left(\widehat{P}^{(2)}\right)^T \widetilde{\boldsymbol{F}}^{(2)}\left(\overline{V}\widetilde{\boldsymbol{U}}\right) + \widehat{B}^{(3,2)}\left(\widehat{P}^{(3)}\right)^T \widetilde{\boldsymbol{F}}^{(3)}\left(\overline{V}\widetilde{\boldsymbol{U}}\right) +\\
&\quad \widehat{B}^{(3,3)}\left(\widehat{P}^{(4)}\right)^T \widetilde{\boldsymbol{F}}^{(4)}\left(\overline{V}\widetilde{\boldsymbol{U}}\right) + \widehat{B}^{(3,4)}\left(\widehat{P}^{(5)}\right)^T \widetilde{\boldsymbol{F}}^{(5)}\left(\overline{V}\widetilde{\boldsymbol{U}}\right)
\end{aligned}
\tag{3.48}
$$

where the matrices $\widehat{B}^{(*)}$ do not depend on time and are defined by

$$
\begin{aligned}
\widehat{B}^{(1)} &:= \left(V^{(1)}\right)^T B^{(1)}\widehat{V}^{(1)}\left[\left(\widehat{P}^{(1)}\right)^T\widehat{V}^{(1)}\right]^{-1} \in \mathbb{R}^{q_1\times m_1}\\
\widehat{B}^{(2,1)} &:= \left(V^{(2)}\right)^T B^{(2)}\widehat{V}^{(2)}\left[\left(\widehat{P}^{(2)}\right)^T\widehat{V}^{(2)}\right]^{-1} \in \mathbb{R}^{q_2\times m_2}\\
\widehat{B}^{(2,2)} &:= \left(V^{(2)}\right)^T B^{(3)}\widehat{V}^{(3)}\left[\left(\widehat{P}^{(3)}\right)^T\widehat{V}^{(3)}\right]^{-1} \in \mathbb{R}^{q_2\times m_3}\\
\widehat{B}^{(2,3)} &:= \left(V^{(2)}\right)^T B^{(4,x)}\widehat{V}^{(4)}\left[\left(\widehat{P}^{(4)}\right)^T\widehat{V}^{(4)}\right]^{-1} \in \mathbb{R}^{q_2\times m_4}\\
\widehat{B}^{(2,4)} &:= \left(V^{(2)}\right)^T B^{(5,x)}\widehat{V}^{(5)}\left[\left(\widehat{P}^{(5)}\right)^T\widehat{V}^{(5)}\right]^{-1} \in \mathbb{R}^{q_2\times m_5}\\
\widehat{B}^{(3,1)} &:= -\left(V^{(3)}\right)^T B^{(3)}\widehat{V}^{(2)}\left[\left(\widehat{P}^{(2)}\right)^T\widehat{V}^{(2)}\right]^{-1} \in \mathbb{R}^{q_3\times m_2}\\
\widehat{B}^{(3,2)} &:= \left(V^{(3)}\right)^T B^{(2)}\widehat{V}^{(3)}\left[\left(\widehat{P}^{(3)}\right)^T\widehat{V}^{(3)}\right]^{-1} \in \mathbb{R}^{q_3\times m_3}\\
\widehat{B}^{(3,3)} &:= \left(V^{(3)}\right)^T B^{(4,y)}\widehat{V}^{(4)}\left[\left(\widehat{P}^{(4)}\right)^T\widehat{V}^{(4)}\right]^{-1} \in \mathbb{R}^{q_3\times m_4}\\
\widehat{B}^{(3,4)} &:= \left(V^{(3)}\right)^T B^{(5,y)}\widehat{V}^{(5)}\left[\left(\widehat{P}^{(5)}\right)^T\widehat{V}^{(5)}\right]^{-1} \in \mathbb{R}^{q_3\times m_5}
\end{aligned}
\tag{3.49}
$$

As in eq. (2.35), a more compact form of (3.48) reads

$$
\frac{d}{dt}\widetilde{\boldsymbol{U}}(t) = \widetilde{B}\ddot{\widetilde{\boldsymbol{F}}}(\overline{V}\widetilde{\boldsymbol{U}}) \tag{3.50}
$$

where

$$
\ddot{\widetilde{\boldsymbol{F}}} := \begin{pmatrix} \left(\widehat{P}^{(1)}\right)^T \widetilde{\boldsymbol{F}}^{(1)} \\ \vdots \\ \left(\widehat{P}^{(5)}\right)^T \widetilde{\boldsymbol{F}}^{(5)} \end{pmatrix} \in \mathbb{R}^{\sum_{i=1}^{5} m_i},
$$

$$
\widetilde{B} := \begin{pmatrix} \widehat{B}^{(1)} & 0 & 0 & 0 & 0 \\ 0 & \widehat{B}^{(2,1)} & \widehat{B}^{(2,2)} & \widehat{B}^{(2,3)} & \widehat{B}^{(2,4)} \\ 0 & \widehat{B}^{(3,1)} & \widehat{B}^{(3,2)} & \widehat{B}^{(3,3)} & \widehat{B}^{(3,4)} \end{pmatrix} \in \mathbb{R}^{\sum_{i=1}^{3} q_i \times \sum_{i=1}^{5} m_i}
$$

Therefore, the reduced-order model for the SWE, formulated in a FV framework, is a problem of dimension $\sum_{i=1}^{3} q_i$, with numerical fluxes computed on $\sum_{i=1}^{5} m_i$ chosen interfaces. The full-order model is a problem solved in M_f cells, with the fluxes computed on I_f interfaces. We thus expect in the model reduction to have $\sum_{i=1}^{3} q_i \ll M_f$ and $\sum_{i=1}^{5} m_i \ll I_f$.

3.5.2 The ROM-based parareal method for solving the SWE

The ROM-based parareal method (Algorithm 6) for solving the shallow water models uses the following propagators

- Reference propagator ($\mathcal{F}_{\delta t}$), corresponding to a fine temporal discretization (with time step δt) of (2.35) (classical SWE).

- Coarse propagator ($\mathcal{G}_{\Delta t}$), corresponding to a coarse discretization (with time step Δt) of (2.35). Either the classical or the porosity-based SWE can be used as coarse propagator.

- Reduced propagator ($\mathcal{F}^k_{r,\delta t}$), defined at each parareal iteration and corresponding to a fine temporal discretization (with time step δt) of (3.50).

Along this and the following chapters, we initially study the application of the parareal method (both the classical and the ROM-based one) for the SWE by considering $\mathcal{G}_{\Delta t}$ as a coarse discretization of the classical SWE. In other words, we first study a coupling between the classical SWE at different scales. The coupling with porosity-based models, for which additional challenges need to be dealt with, such as more complex geometries, is left to Chapter 6.

3.6 Numerical examples

We present in this section three tests cases for illustrating the application of the classical and ROM-based parareal methods to the solution of the 2D nonlinear SWE. These simulations are performed for comparing the two methods, evaluating the influence of the POD thresholds $\varepsilon_{\text{sv,linear}}$ and $\varepsilon_{\text{sv,nonlinear}}$ on the performance of the ROM-based one, as well as other simulation parameters, and identifying advantages and limitations of the ROM-based parareal method. These tests cases have an increasing complexity: the first two are "pseudo-2D" simulations (solved numerically in two dimensions, but with only one direction of propagation), respectively with and without spatial coarsening between the coarse and fine propagators; and the third test case is a real 2D one, with spatial coarsening between the propagators. To make clear the relation between the pseudo-2D test cases, they are named respectively as Test cases 1 and 1c, where "c" stands for "coarsening". The 2D test case is named for simplicity as Test case 2. In the following chapters, Test cases 1 and 2 are largely used for studying the parareal method and its proposed variants. We recall that, in all test cases, both the fine and coarse propagators are defined as discretizations of the classical SWE. The coupling with porosity-based models, which would act as coarse propagator, is left to Chapter 6.

We remark that the test cases defined in this chapter do not reflect practical choices of fine and coarse propagators. Indeed, since the models are discretized using an explicit-in-time scheme, one would choose time steps as large as possible, close to their maximum permissible value based on CFL stability conditions. By defining a test case without spatial coarsening between $\mathcal{F}_{\delta t}$ and $\mathcal{G}_{\Delta t}$ (as Test case 1), it is clear that the former, by using smaller time steps, is largely over-resolved. Moreover, the use of too small time steps (*i.e.* too small CFL numbers) may increase the numerical diffusion and lead to situations in which the coarse propagator (which uses a CFL number closer to the unity) is more accurate than the fine one w.r.t. the analytical solution. It is the case, as showed below, for Test case 1, for which the analytical solution can be easily derived. However, the objective in these first examples is to compare the performance, in terms of stability, convergence and computational time, of the classical and ROM-based parareal methods under the same (simple) configurations, for approximating a numerical solution defined as reference one (the solution given by the fine propagator). Also, by defining test cases with or without spatial coarsening, we seek to identify additional challenges that may be introduced by using spatial interpolation.

For the simulations involving spatial coarsening, a linear interpolation procedure is used, with Section 3.6.8 being dedicated to a discussion on the influence of the interpolation on the performance of the parareal methods. For evaluating and comparing the solutions, we consider the errors e^k_n (3.3), defined for each iteration and each parareal time instant, and $\overline{e}^k$ (3.4), defined for each iteration; and also the speedup $s(k)$ (3.5) computed at each iteration. We set a maximum number of iterations $N_{\text{itermax}} = 5$. All simulations are executed using 20 parallel processors, and we consider the time slices to coincide with the coarse time steps associated to the coarse propagator ($\Delta T = \Delta t$).

3.6.1 Definition of the test cases

Test case 1: pseudo-2D without spatial coarsening

In this first test case, $\mathcal{F}_{\delta t}$ and $\mathcal{G}_{\Delta t}$ use the same Cartesian, spatial mesh. The initial condition is a a lake-at-rest:

$$h(\boldsymbol{x}, t=0) = 1, \qquad u_x(\boldsymbol{x}, t=0) = u_y(\boldsymbol{x}, t=0) = 0, \qquad \boldsymbol{x} \in \Omega \tag{3.51}$$

and, concerning the boundary conditions, all the boundaries are closed (null mass flux), except for the left one ($\partial\Omega_{\text{inward}} := \{\boldsymbol{x} \in \partial\Omega | x = 0\}$), on which a unit inward flux is defined:

$$\begin{cases} h\boldsymbol{u} \cdot \boldsymbol{n} = 1, & \boldsymbol{x} \in \partial\Omega_{\text{inward}}, t \in [0, T] \\ h\boldsymbol{u} \cdot \boldsymbol{n} = 0, & \boldsymbol{x} \in \partial\Omega \backslash \partial\Omega_{\text{inward}}, t \in [0, T] \end{cases} \tag{3.52}$$

The configurations of the test case are presented in Table 3.1.

Spatial domain	$\Omega = [0, 20]^2$	
Maximum simulation time	$T = 4$	
Length of each time slice	$\Delta T = 0.2$	
Number of time slices	$N_{\Delta T} = 20$	
Number of parallel processors	$N_p = 20$	
Maximum number of iterations	$N_{\text{itermax}} = 5$	
	$\mathcal{F}_{\delta t}$	$\mathcal{G}_{\Delta t}$
Time step	$\delta t = 0.001$	$\Delta t = 0.2$
Mesh size (x-direction)	$\delta x = 1$	$\Delta x = 1$
Mesh size (y-direction)	$\delta y = 1$	$\Delta y = 1$

Table 3.1: Configurations of Test case 1 (pseudo-2D without spatial coarsening)

Test case 1c: pseudo-2D test case with spatial coarsening

In this test case, the same initial (eq. (3.51)) and boundary (eq. (3.52)) conditions of the first case are considered. Concerning the parareal configurations, the only difference w.r.t. the previous test case is the spatial mesh of the coarse propagator. We consider for $\mathcal{G}_{\Delta t}$ a spatial mesh five times coarser than the fine mesh in each direction. The simulation parameters are presented in Table 3.2. Figure 3.8 illustrates the evolution of the reference water depth in Test cases 1 and 1c.

Spatial domain	$\Omega = [0, 20]^2$	
Maximum simulation time	$T = 4$	
Length of each time slice	$\Delta T = 0.2$	
Number of time slices	$N_{\Delta T} = 20$	
Number of parallel processors	$N_p = 20$	
Maximum number of iterations	$N_{\text{itermax}} = 5$	
	$\mathcal{F}_{\delta t}$	$\mathcal{G}_{\Delta t}$
Time step	$\delta t = 0.001$	$\Delta t = 0.2$
Mesh size (x-direction)	$\delta x = 1$	$\Delta x = 5$
Mesh size (y-direction)	$\delta y = 1$	$\Delta y = 5$

Table 3.2: Configurations of Test case 1c (pseudo-2D with spatial coarsening)

Test case 2: 2D with spatial coarsening

We propose a more complex simulation with a 2D flow propagation and spatial coarsening between the fine and coarse propagators. All boundaries are closed ($h\boldsymbol{u} \cdot \boldsymbol{n} = 0, \boldsymbol{x} \in \partial\Omega, t \in [0, T]$) and the initial solution is a Gaussian water depth with null velocity:

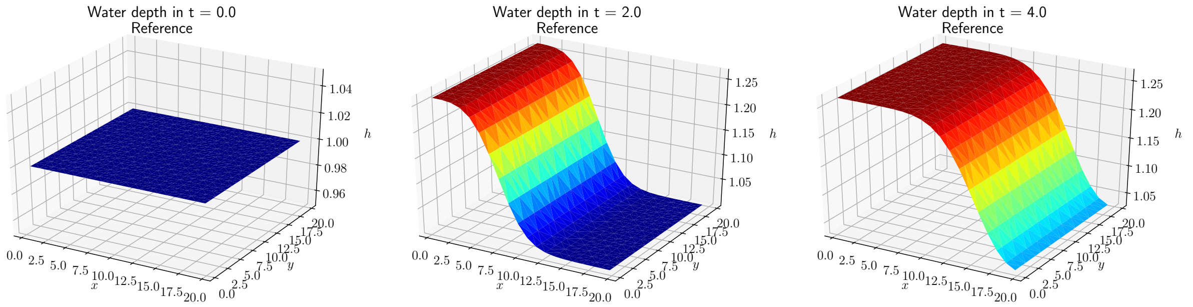


Figure 3.8: Test cases 1 and 1c (pseudo-2D): water depth provided by the reference model $\mathcal{F}_{\delta t}$ at $t = 0$ (left), $t = T/2$ (middle) and $t = T$ (right). The $x-$unit discharge profile is similar and the y-unit discharge profile is identically zero.

$$h(\boldsymbol{x}, t = 0) = h_0 + \exp\left(-\frac{(x - x_0)^2}{2\sigma_x^2} - \frac{(y - y_0)^2}{2\sigma_y^2}\right), \qquad \boldsymbol{u}(\boldsymbol{x}, t = 0) = \boldsymbol{0}, \qquad \boldsymbol{x} \in \Omega \tag{3.53}$$

with $h_0 = 1$, $x_0 = y_0 = 50$ and $\sigma_x = \sigma_y = 7.5$. The configurations for the parareal simulations are presented in Table 3.2, and the evolution of the reference water depth, presented in Figure 3.9, shows that the solution presents stronger variations in time when compared to Tests case 1 and 1c, being possibly more challenging to solve with the parareal methods.

Spatial domain	$\Omega = [0, 100]^2$	
Maximum simulation time	$T = 5$	
Length of each time slice	$\Delta T = 0.25$	
Number of time slices	$N_{\Delta T} = 20$	
Number of parallel processors	$N_p = 20$	
Maximum number of iterations	$N_{\text{itermax}} = 5$	
	$\mathcal{F}_{\delta t}$	$\mathcal{G}_{\Delta t}$
Time step	$\delta t = 0.001$	$\Delta t = 0.25$
Mesh size (x-direction)	$\delta x = 2$	$\Delta x = 5$
Mesh size (y-direction)	$\delta y = 2$	$\Delta y = 5$

Table 3.3: Configurations of Test case 2 (2D)

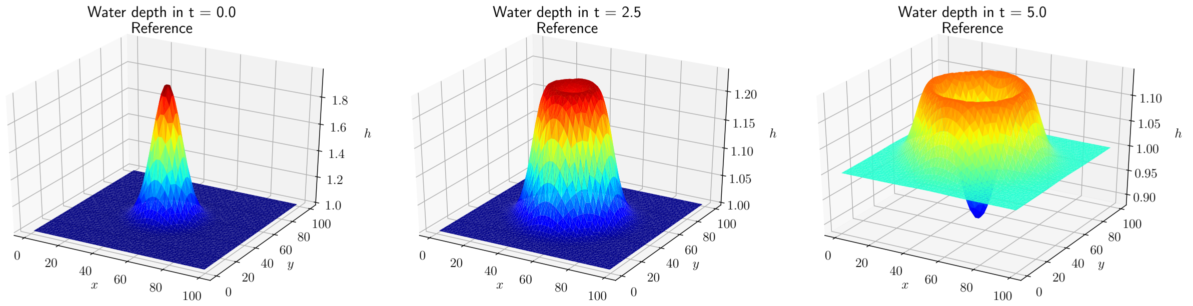


Figure 3.9: Test case 2 (2D): water depth provided by the reference model $\mathcal{F}_{\delta t}$ at $t = 0$ (left), $t = T/2$ (middle) and $t = T$ (right).

3.6.2 Simulation using the classical parareal method

Figures 3.10, 3.11 and 3.12 present the evolution of the error per iteration and time instant (e_n^k) and the maximum error per iteration ($\overline{e}^k$) using the classical parareal method, respectively for Test cases 1, 1c and 2. For Test case 1, which is relatively simple due to the one-dimensional solution and the absence of spatial coarsening between $\mathcal{F}_{\delta t}$ and $\mathcal{G}_{\Delta t}$, the classical parareal presents instabilities, leading to increasing errors across iterations. Test case 1c has a slightly better behaviour but instabilities are still observed. The best behaviour, closest to what is expected in a parareal simulation, is observed in Test case 2, with a decreasing error along iterations (even if slow), despite of a degradation of the solution in the first iteration. As discussed in Section 3.6.8, this initial degradation is due to the low-order interpolation between the spatial meshes of $\mathcal{F}_{\delta t}$ and $\mathcal{G}_{\Delta t}$.

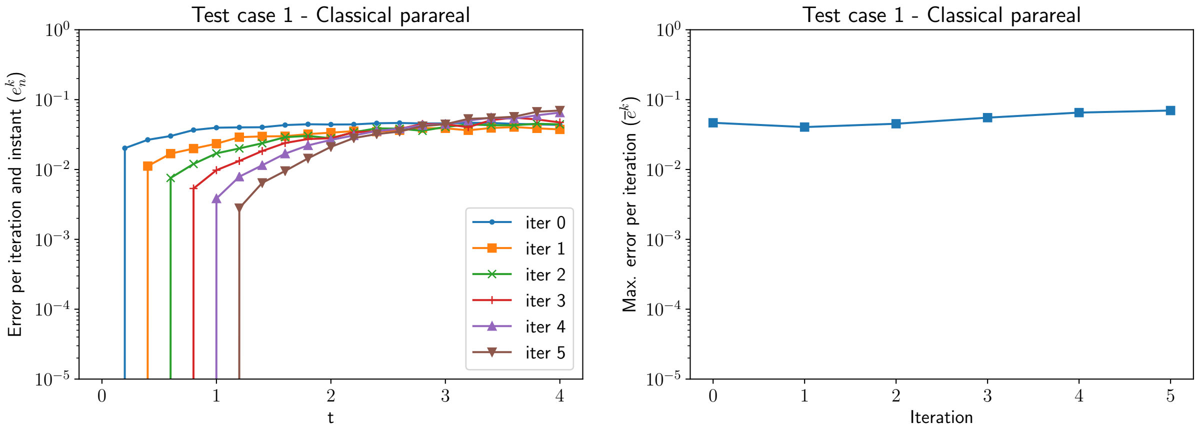


Figure 3.10: Test case 1 (pseudo-2D without spatial coarsening): relative error e_n^k per iteration and time instant (left) and maximum error $\overline{e}^k$ per iteration (right) using the classical parareal method.

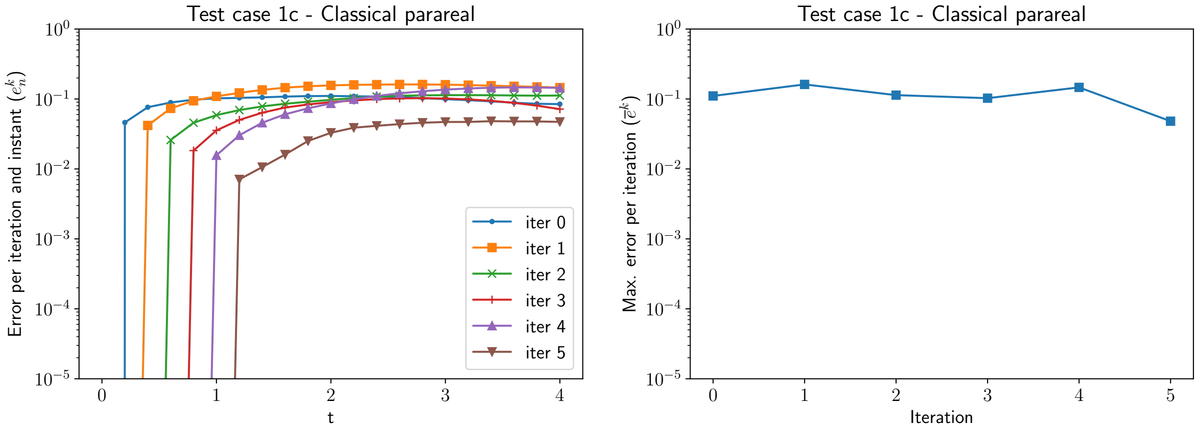


Figure 3.11: Test case 1c (pseudo-2D with spatial coarsening): relative error e_n^k per iteration and time instant (left) and maximum error $\overline{e}^k$ per iteration (right) using the classical parareal method.

The speedups obtained in each simulation are presented in Table 3.4. The increasing speedups from Test case 1 to Test case 2 illustrate that larger accelerations are obtained when there is a more important coarsening between the fine and coarse propagators, as discussed in Section 3.2.2.

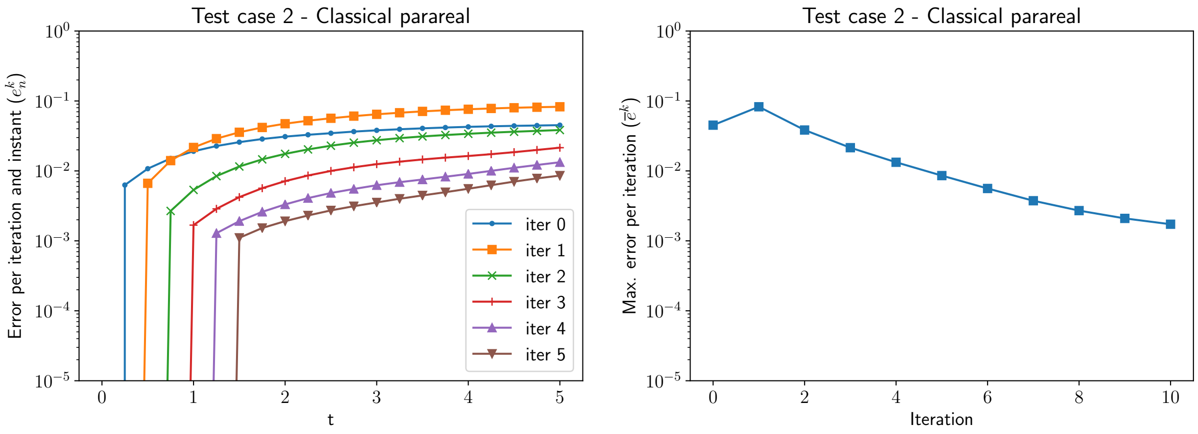


Figure 3.12: Test case 2 (2D): relative error e_n^k per iteration and time instant (left) and maximum error $\overline{e}^k$ per iteration (right) using the classical parareal method.

	Iteration				
	1	2	3	4	5
Test case 1	6.89	4.63	3.55	2.85	2.41
Test case 1c	7.25	4.93	3.79	3.06	2.58
Test case 2	14.11	7.78	5.42	4.15	3.37

Table 3.4: Test case 1, 1c and 2: speedup $s(k)$ along iterations using the classical parareal method.

3.6.3 Simulation using the ROM-based parareal method

As discussed in Section 3.4.3 and shown in Algorithm 6, the POD-EIM model reduction procedure takes as arguments two thresholds for the truncation of the POD subspaces, namely $\varepsilon_{\text{sv,linear}}$ and $\varepsilon_{\text{sv,nonlinear}}$, which are used respectively for the POD applied to snapshots of the solution and the POD-EIM applied to snapshots of the nonlinear term. We investigate the influence of these thresholds on the performance of the ROM-based parareal method, in terms of convergence and stability. In order to compare with the results of the classical parareal method, we consider the same three test cases as above.

Test case 1 (pseudo-2D without spatial coarsening)

For this test case, the ROM-based parareal method is executed considering $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) \in \{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}^2$, in a total of 16 simulations. The relative errors of each simulation are presented in Figure 3.13. The maximum error per iteration is presented in Figure 3.14 for a selected set of simulations (those with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}}$) and compared to the errors obtained by the classical parareal method. Note that the error range in the vertical axis of Figures 3.13 and 3.14 $(10^{-12} - 10^0)$ is larger than the one in Figure 3.10 $(10^{-5} - 10^0)$, since considerably smaller errors are obtained using the ROM-based parareal method, specially when smaller thresholds are considered (*i.e.* when the formulated ROMs have higher dimension). Also note that the simulations with small $\varepsilon_{\text{sv,linear}}$ and large $\varepsilon_{\text{sv,nonlinear}}$ present instabilities. Some of them do not complete $N_{\text{itermax}} = 5$ iterations, since negative water depths are computed, and only their errors before the termination of the execution are presented.

For the sake of conciseness, we present only the speedup of each simulation at the first and fifth parareal iterations in Table 3.5. Not surprisingly, the speedups are smaller than the ones observed in the classical parareal method, since the introduction of the model reduction complexifies the parareal algorithm. After $N_{\text{itermax}} = 5$ iterations, all simulations have a larger computational time than the reference one. We also observe a decrease of the speedup by increasing the dimension of the subspaces (smaller thresholds $\varepsilon_{\text{sv,linear}}$ and $\varepsilon_{\text{sv,nonlinear}}$). It is due not only to the more expensive reduced models, but also to more expensive executions of the greedy DEIM algorithm and computation of the ROM matrices, due to the increasing dimension of the subspaces $\widehat{\mathcal{S}}^{(i)}$ along iterations (see (3.47)), as discussed below. The POD cost itself is not influenced by the dimensions of the subspaces, since the truncation is a post-process of the model reduction.

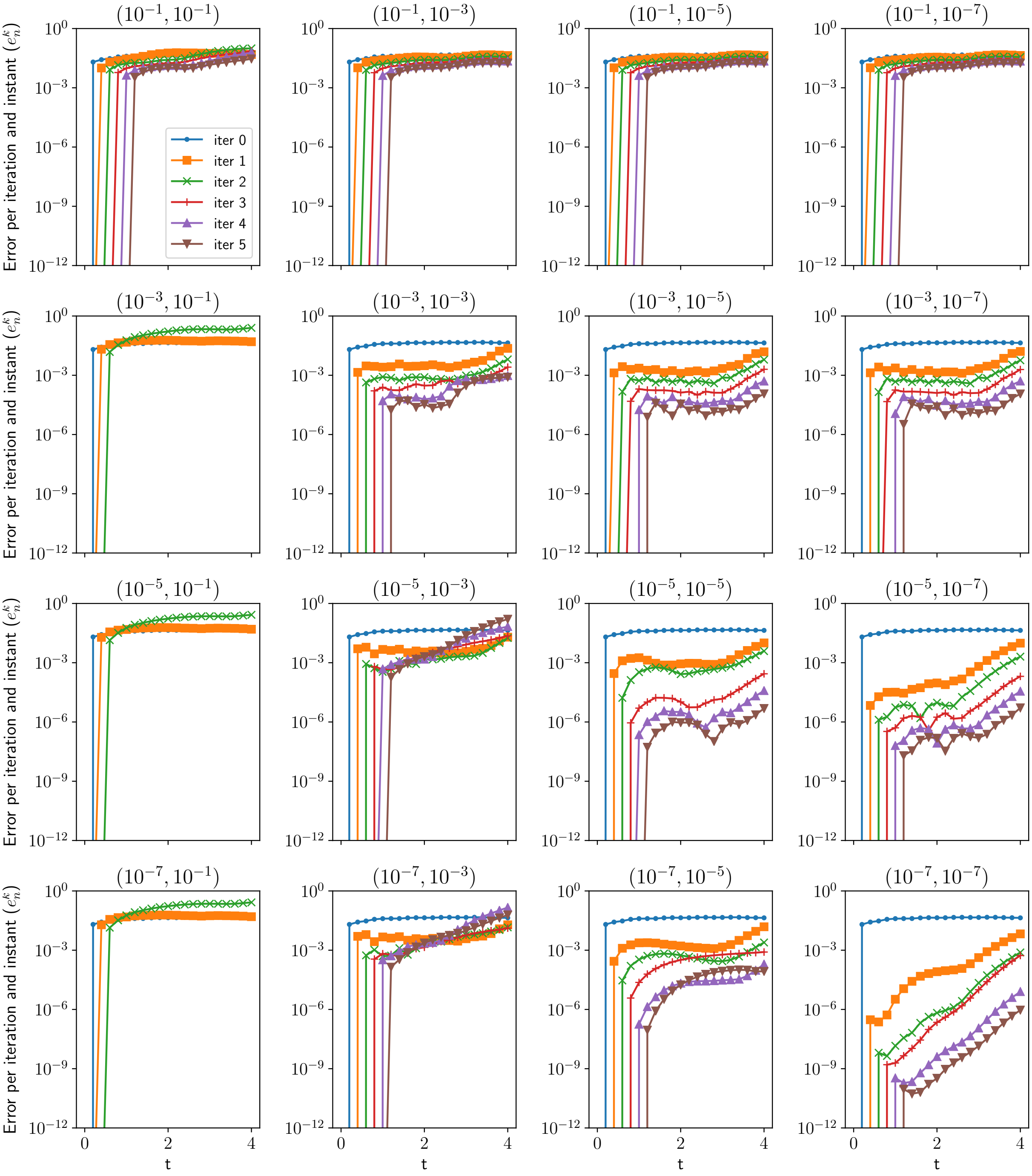


Figure 3.13: Test case 1 (pseudo-2D without spatial coarsening): relative error e_n^k using the ROM-based parareal method for various thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$. Each curve corresponds to a parareal iteration and the horizontal axis indicate the simulation time. Graphs in the same row correspond to the same $\varepsilon_{\text{sv,linear}}$ and graphs in the same column correspond to the same $\varepsilon_{\text{sv,nonlinear}}$.

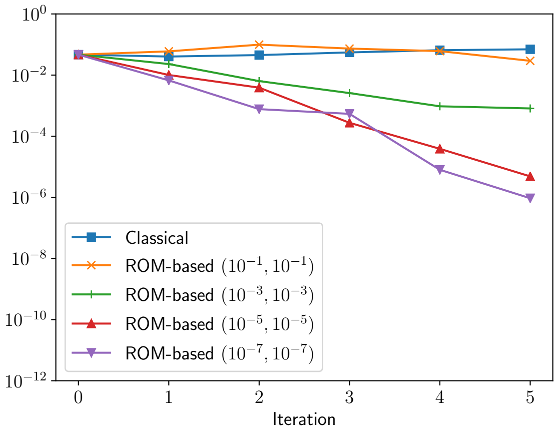


Figure 3.14: Test case 1 (pseudo-2D without spatial coarsening): relative error $\bar{e}^k$ using the ROM-based parareal method for selected thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$ and the classical parareal method.

	Iteration $k = 1$				Iteration $k = 5$			
$\varepsilon_{\text{sv,nonlinear}}$ / $\varepsilon_{\text{sv,linear}}$	10^{-1}	10^{-3}	10^{-5}	10^{-7}	10^{-1}	10^{-3}	10^{-5}	10^{-7}
10^{-1}	3.21	2.94	2.95	2.89	0.82	0.75	0.73	0.72
10^{-3}	3.11	2.97	2.96	2.83	*	0.74	0.71	0.69
10^{-5}	3.04	2.79	2.75	2.82	*	0.71	0.68	0.68
10^{-7}	2.96	2.94	2.75	2.65	*	0.72	0.68	0.66

Table 3.5: Test case 1 (pseudo-2D without spatial coarsening): speedup at the first and fifth iterations of the ROM-based parareal method, for various pairs of thresholds $\varepsilon_{\text{sv,linear}}$ (rows) and $\varepsilon_{\text{sv,nonlinear}}$ (columns). Simulations indicated with an asterisk present instabilities and do not complete five iterations.

In order to illustrate what “reduced” means when formulating reduced-order models, we present the average dimensions of the computed reduced subspaces along iterations in Table 3.6, for the subspaces computed from snapshots of the solution (using POD) and of the nonlinear terms (using POD-EIM). The subspaces $\mathcal{S}^{(3)}$ and $\widehat{\mathcal{S}}^{(3)}$ (computed respectively from snapshots of the $y-$unit discharge and of the third flux component of the SWE) are null spaces (due to the one-dimensional propagation) and are not considered in the averages. Also, the subspaces $\widehat{\mathcal{S}}^{(4)}$ and $\widehat{\mathcal{S}}^{(5)}$, accounting for the source terms, which are not considered here, are null spaces then discarded from the averages. We first observe that the number of snapshots and subspace dimensions increase along iterations, and the formulation and simulation of the ROMs become more expensive. However, we recall that, in the full-order model, the FV discretization leads to a problem of dimension $d_f M_f$ (*i.e.* with d_f degrees-of-freedom per computational cell), with nonlinear fluxes computed on I_f interfaces. On the other hand, in the reduced-order model, the dimension of the problem is given by the sum of the dimensions of the subspaces $\mathcal{S}^{(i)}, i = 1, 2, 3$, and the number of interfaces on which the nonlinear term is computed is given by the sum of the dimensions of the subspaces $\widehat{\mathcal{S}}^{(i)}, i = 1, \dots, 5$. For Test case 1, the mesh used by $\mathcal{F}_{\delta t}$ has 400 cells and nonlinear fluxes computed on 840 interfaces; whereas the dimensions of the reduced subspaces remain in the order of tenths, and, as a consequence, the ROM is much less expensive than the FOM.

Finally, we illustrate, in Figure 3.15, the physical solution (water depth) obtained in the first two iterations of the classical and ROM-based parareal methods, this last one using $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-5})$, and compared with the reference and coarse (iteration $k = 0$) solutions. We also compare with the analytical solution, which can be determined by applying the Rankine-Hugoniot condition (LeFloch and Thanh, 2007) to the shock, with travelling speed λ (unknown); left state h_L (unknown) and $h_L u_{x,L} = 1$ (given by the boundary conditions); and right state $h_R = 1$ and $h_R u_{x,R} = 0$ (given by the initial condition), yielding

	Average - $\mathcal{S}^{(i)}$					Average - $\widehat{\mathcal{S}}^{(i)}$				
Iteration k ε_{sv}	1	2	3	4	5	1	2	3	4	5
Number of snapshots	**21**	**42**	**63**	**84**	**105**	**21**	**42**	**63**	**84**	**105**
10^{-1}	2	2	2	2	2	3	3	3	3	3
10^{-3}	11	11	11	11	11	12	14	15	15	15
10^{-5}	15	16	17	17	17	17	24	25	25	25
10^{-7}	17	19	19	19	19	19	27	28	29	29

Table 3.6: Test case 1 (pseudo-2D without spatial coarsening): average dimensions $(q_1 + q_2)/2$ of the subspaces $\mathcal{S}^{(i)}, i = 1, 2$, computed from snapshots of the solutions, and $(m_1 + m_2)/2$ of the subspaces $\widehat{\mathcal{S}}^{(i)}, i = 1, 2$, computed from snapshots of the nonlinear term, for each iteration of the ROM-based parareal method, in function of $\varepsilon_{\text{sv}} = \varepsilon_{\text{sv,linear}}$ (for $\mathcal{S}^{(i)}$) and $\varepsilon_{\text{sv}} = \varepsilon_{\text{sv,nonlinear}}$ (for $\widehat{\mathcal{S}}^{(i)}$). For a given $\varepsilon_{\text{sv,linear}}$, the dimensions are nearly the same for all values of $\varepsilon_{\text{sv,nonlinear}}$, and vice versa. q_3, m_3, m_4 and m_5 are not considered in the averages since $\mathcal{S}^{(3)}$, $\widehat{\mathcal{S}}^{(3)}$, $\widehat{\mathcal{S}}^{(4)}$ and $\widehat{\mathcal{S}}^{(5)}$ are null spaces due to the one-dimensional flow and the absence of source terms. The number of snapshots indicated in the second row is the upper limit for each dimension.

$$\begin{cases} (h_R u_{x,R}) - (h_L u_{x,L}) = \lambda(h_R - h_L) \\ \left[\left(\frac{g h_R^2}{2} + h_R u_{x,R}^2\right) - \left(\frac{g h_L^2}{2} + h_L u_{x,L}^2\right)\right] = \lambda(h_R u_{x,R} - h_L u_{x,L}) \end{cases} \tag{3.54}$$

which is a nonlinear system on h_L and λ with solution $h_L \approx 1.2665$ and $\lambda \approx 3.75232$. Therefore, the shock position at time t is $x = \lambda t$.

As discussed before, under the chosen configurations for the fine and coarse propagators, the latter is more accurate w.r.t. the analytical solution, due to the use of a CFL number closer to the unity. We compare, however, the performance of the classical and ROM-based parareal methods in approximating a numerical solution defined as reference. We observe in Figure 3.15 that the classical method presents a clear unstable behaviour, whereas the solution of the ROM-based solution almost visually coincides with the reference one from the second iteration.

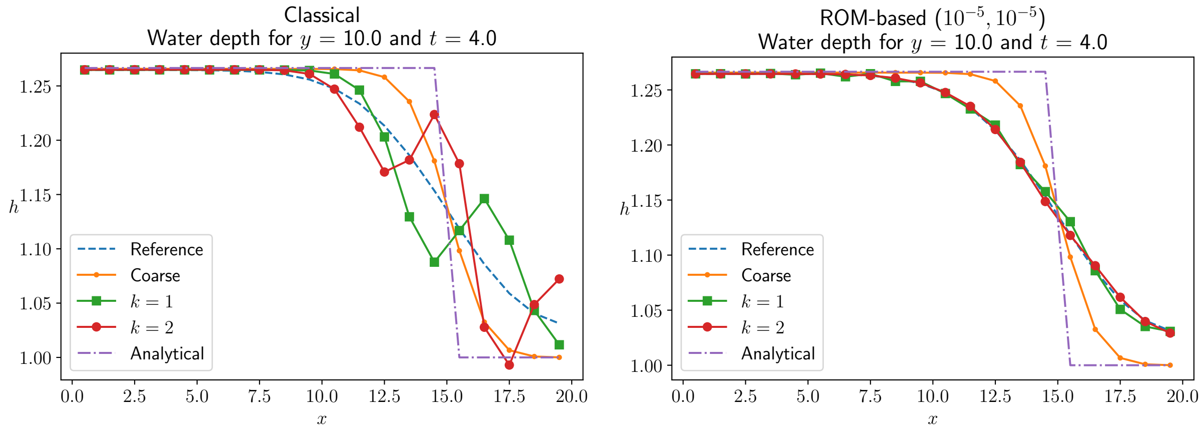


Figure 3.15: Test case 1 (pseudo-2D without spatial coarsening): final water depth ($t = T = 4$) along $y = 10$ at the two first parareal iterations. Dashed and dot-dashed curves represent respectively the reference (provided by $\mathcal{F}_{\delta t}$) and analytical solutions. Small bullets represent the coarse solution (given by $\mathcal{G}_{\Delta t}$, at the 0-th parareal iteration). Left: classical parareal method. Right: ROM-based parareal method with $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-5})$.

Test case 1c (pseudo-2D test case with spatial coarsening)

For the simulation of Test case 1c using the ROM-based parareal method, we consider the same 16 pairs of thresholds used for Test case 1. The maximum relative error per iteration for selected thresholds and the errors per iteration and time for all simulations are presented respectively in Figures 3.16 and 3.18. As in Test case 1, a large range $10^{-12} - 10^{0}$ is considered, compared to Figure 3.11. The behaviour of the method in function of $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$ is similar to the one observed for Test case 1, with better convergence for simultaneously small $\varepsilon_{\text{sv,linear}}$ and $\varepsilon_{\text{sv,nonlinear}}$, and unstable behaviours for small $\varepsilon_{\text{sv,linear}}$ combined with large $\varepsilon_{\text{sv,nonlinear}}$. However, for the simulations presenting a good convergence behaviour, the error decrease is less remarkable than in Test case 1, which is clear by comparing Figures 3.14 and 3.16. The speedups and dimensions of the subspaces are similar to Test case 1 and omitted.

Figure 3.17 presents the water depth provided by the classical and ROM-based methods at $t = T$. We observe that, in this test case, the very low spatial resolution of the coarse propagator strongly degrades its quality w.r.t. the analytical solution. The classical parareal method presents a less unstable behaviour, when compared to Test case 1, but still with a very slow convergence towards the reference solution. For the ROM-based method, we choose the same model reduction thresholds as in Test case 1, namely $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-5})$. Under these configurations, an unstable behaviour is observed for advanced times of the simulation (which is clear by comparing Figures 3.13 and 3.18), and more iterations are required for well approximating the reference solution. Therefore, in this case better choices of model reduction thresholds would be $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-7}), (10^{-7}, 10^{-7})$.

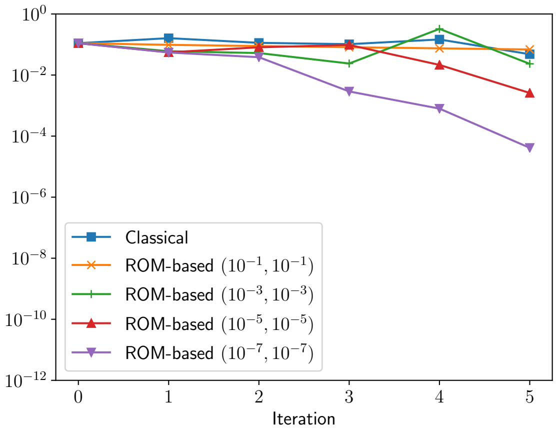


Figure 3.16: Test case 1c (pseudo-2D with spatial coarsening): relative error $\bar{\varepsilon}^k$ using the ROM-based parareal method for selected thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$ and the classical parareal method.

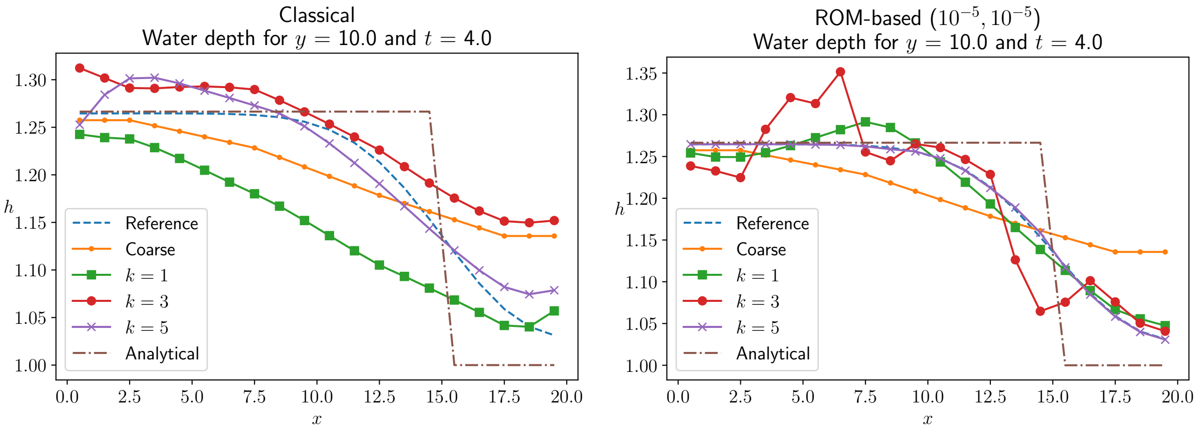


Figure 3.17: Test case 1c (pseudo-2D with spatial coarsening): final water depth ($t = T = 4$) along $y = 10$ at the first, third and fifth first parareal iterations. Dashed and dot-dashed curves represent respectively the reference (provided by $\mathcal{F}_{\delta t}$) and analytical solutions. Small bullets represent the coarse solution (given by $\mathcal{G}_{\Delta t}$, at the 0-th parareal iteration). Left: classical parareal method. Right: ROM-based parareal method with $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-5})$.

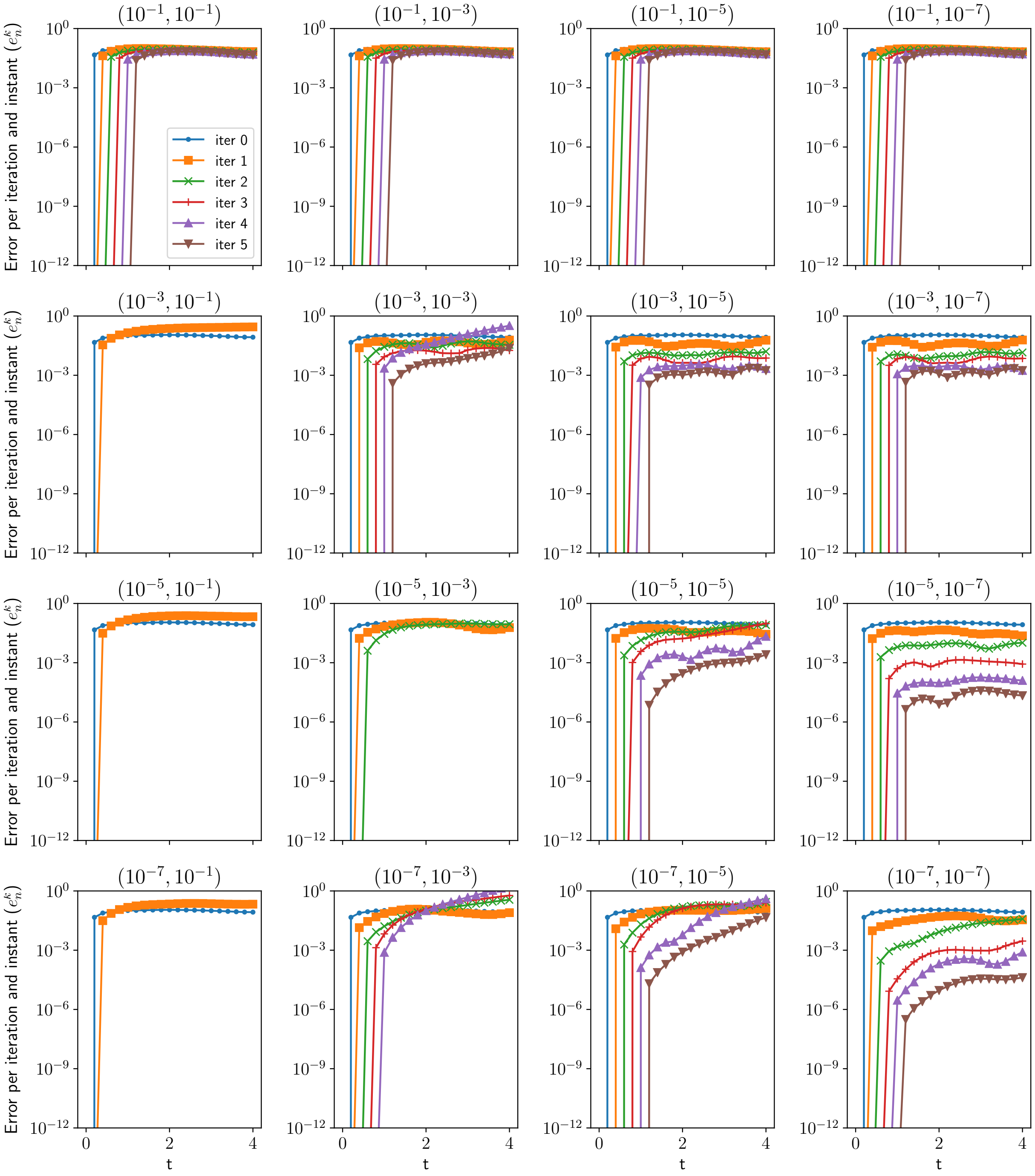


Figure 3.18: Test case 1c (pseudo-2D with spatial coarsening): relative error e_n^k using the ROM-based parareal method for various thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$. Each curve corresponds to a parareal iteration and the horizontal axis indicate the simulation time. Graphs in the same row correspond to the same $\varepsilon_{\text{sv,linear}}$ and graphs in the same column correspond to the same $\varepsilon_{\text{sv,nonlinear}}$.

Test case 2 (2D)

For this test case, we perform 16 simulations, but with $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}^2$. These thresholds are larger than the ones considered for the pseudo-2D tests cases due to a more critical stable behaviour for small $\varepsilon_{\text{sv,linear}}$ in this more complex simulation. Indeed, the evolution of the errors per iteration and time, presented in Figure 3.19, reveals unstable behaviours, to a greater or lesser extent, for all simulations with $\varepsilon_{\text{sv,linear}} \leq 10^{-3}$. Note that the error range $10^{-5} - 10^{0}$ in the vertical axis of the graphs is the same of Figure 3.12, since no remarkable error reduction using the ROM-based parareal method is observed. It is clear in Figure 3.20: although a faster reduction of the maximum error in the first iteration for $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-3}$ and $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$, there is no further improvement in the following iterations, such that the classical parareal method performs better.

The speedups at the first and fifth iteration are presented in Table 3.7. As observed for the classical parareal method, these speedups are considerably larger than in the simpler Test cases 1 and 1c and, after five iterations, they remain larger than the unity. However, all simulations are considerably more expensive than the classical parareal method.

	Iteration $k=1$				Iteration $k=5$			
$\varepsilon_{\text{sv,nonlinear}}$ / $\varepsilon_{\text{sv,linear}}$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-1}	10^{-2}	10^{-3}	10^{-4}
10^{-1}	7.39	7.17	7.29	6.24	1.64	1.60	1.59	1.44
10^{-2}	7.21	7.12	7.18	7.15	1.56	1.57	1.52	1.44
10^{-3}	6.15	6.93	6.82	6.68	1.41	1.44	1.39	1.33
10^{-4}	6.55	6.59	6.70	5.07	1.31	1.27	1.28	1.09

Table 3.7: Test case 2 (2D): speedup at the first and fifth iterations of the ROM-based parareal method, for various pairs of thresholds $\varepsilon_{\text{sv,linear}}$ (rows) and $\varepsilon_{\text{sv,nonlinear}}$ (columns).

The average dimension of the subspaces computed with POD and POD-EIM are presented in Table 3.8. Since Test case 2 is a two-dimensional problem, only $\widehat{\mathcal{S}}^{(4)}$ and $\widehat{\mathcal{S}}^{(5)}$ are not considered in the averages (due to the absence of source terms). As in Test case 1, the reduced spaces have small dimension, in the order of tenths for all iterations and threshold values, but here the reduction of the computational complexity provided by the ROM is much more important, since the computational mesh for $\mathcal{F}_{\delta t}$ in Test case 2 has 2500 cells and 5100 interfaces.

	Average - $\mathcal{S}^{(i)}$					Average - $\widehat{\mathcal{S}}^{(i)}$				
Iteration k / ε_{sv}	1	2	3	4	5	1	2	3	4	5
Number of snapshots	**21**	**42**	**63**	**84**	**105**	**21**	**42**	**63**	**84**	**105**
10^{-1}	1	1	1	1	1	1	1	1	1	2
10^{-2}	3	3	4	4	5	3	4	5	5	5
10^{-3}	5	7	9	11	13	6	9	11	13	17
10^{-4}	7	12	16	20	30	8	14	19	23	27

Table 3.8: Test case 2 (2D): average dimension $(q_1+q_2+q_3)/3$ of the subspaces $\mathcal{S}^{(i)}, i=1,2,3$ computed from snapshots of the solutions, and $(m_1+m_2+m_3)/3$ of the subspaces $\widehat{\mathcal{S}}^{(i)}, i=1,2,3$ computed from snapshots of the nonlinear term, for each iteration of the ROM-based parareal method, in function of $\varepsilon_{\text{sv}} = \varepsilon_{\text{sv,linear}}$ (for $\mathcal{S}^{(i)}$) and $\varepsilon_{\text{sv}} = \varepsilon_{\text{sv,nonlinear}}$ (for $\widehat{\mathcal{S}}^{(i)}$). For a given $\varepsilon_{\text{sv,linear}}$, the dimensions are nearly the same for all values of $\varepsilon_{\text{sv,nonlinear}}$, and vice versa. m_4 and m_5 are not considered in the average since $\widehat{\mathcal{S}}^{(4)}$ and $\widehat{\mathcal{S}}^{(5)}$ are null spaces due to the absence of source terms. The number of snapshots indicated in the second row is the upper limit for each dimension.

The final water depth along $y=50$ is presented in Figure 3.21. The classical parareal method degrades the quality of the solution in the first iteration, but is able to slowly converge in the following ones. In the case of the ROM-based method, using $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-4}, 10^{-4})$, the general profile of the reference solution is well approximated from the first iteration, but instabilities arise near the peaks and the valley in the following iterations. These different behaviours are clearer by comparing the solution in the entire spatial domain (Figure 3.22).

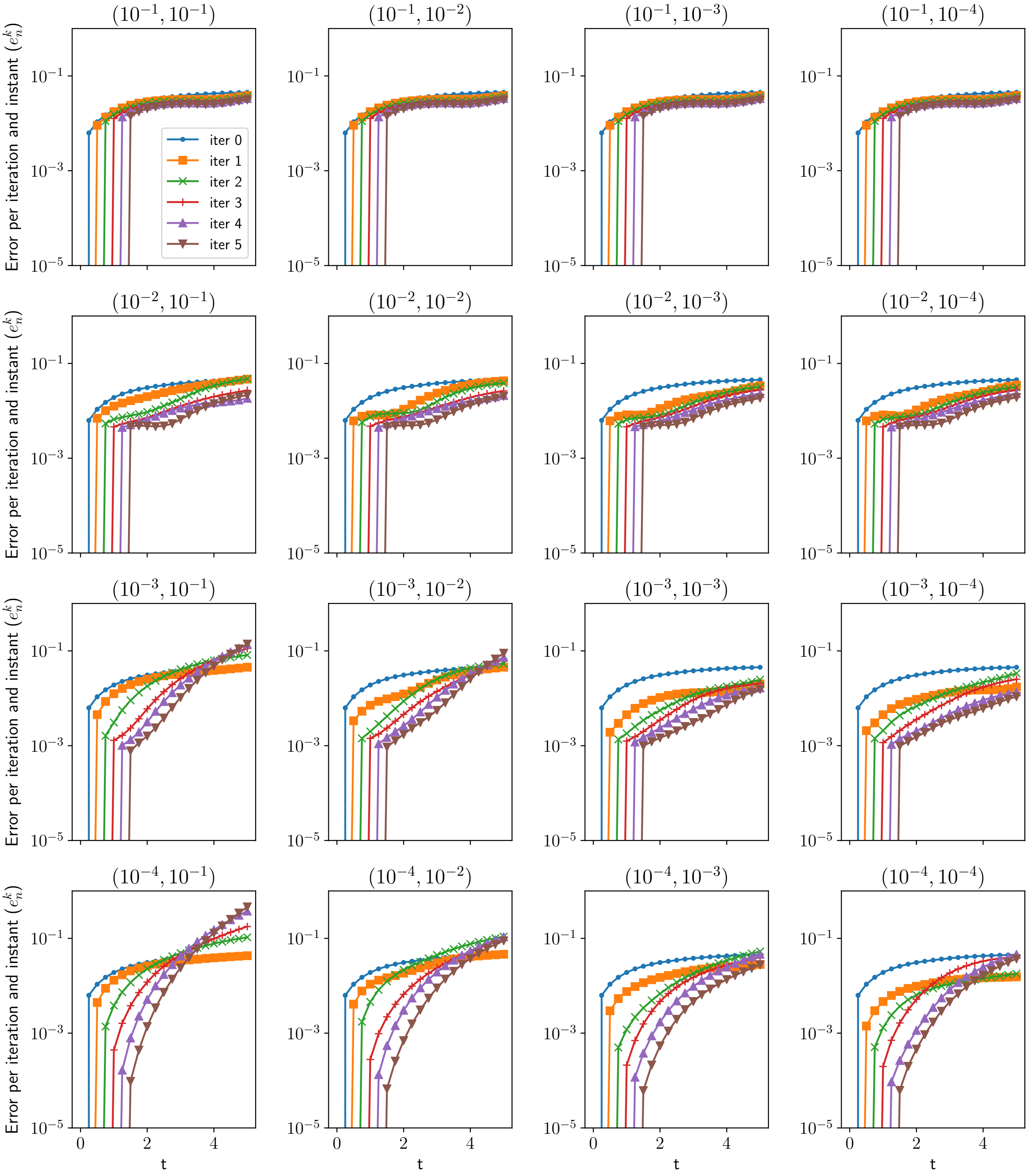


Figure 3.19: Test case 2 (2D): relative error e_n^k using the ROM-based parareal method for various thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$. Each curve corresponds to a parareal iteration and the horizontal axis indicate the simulation time. Graphs in the same row correspond to the same $\varepsilon_{\text{sv,linear}}$ and graphs in the same column correspond to the same $\varepsilon_{\text{sv,nonlinear}}$.

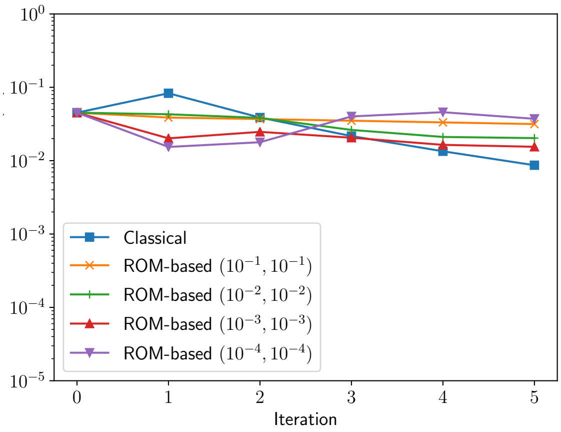


Figure 3.20: Test case 2 (2D): relative error $\overline{e}^k$ using the ROM-based parareal method for selected thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$ and the classical parareal method.

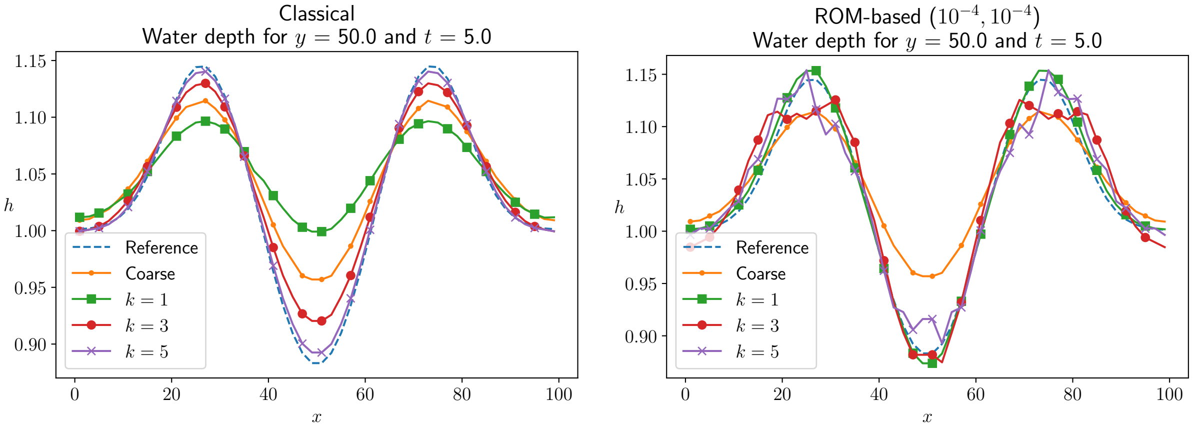


Figure 3.21: Test case 2 (2D): final water depth ($t = T = 5$) along $y = 50$ at the first, third and fifth first parareal iterations. Dashed curves represent the reference (provided by $\mathcal{F}_{\delta t}$). Small bullet represents the coarse solution (given by $\mathcal{G}_{\Delta t}$, at the 0-th parareal iteration). Left: classical parareal method. Right: ROM-based parareal method with $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-4}, 10^{-4})$.

3.6.4 Some indications for understanding the parareal performance

As shown by (Ruprecht, 2018) and discussed in Section 3.2.4, the performance of the parareal method is related to its behaviour on high wavenumbers of the solution. The parareal convergence is slower near the end of the spectrum and instabilities appear for high wavenumbers due to the difference of discrete phase speeds between the fine and coarse propagators.

Therefore, the wavenumber spectrum of the solution can provide useful information for understanding the parareal behaviour. We make use of it in order to obtain some clues on why the classical parareal method is stable in Test case 2 and unstable in the simpler Test case 1, and also on the behaviour of the ROM-based method for different model reduction thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$. For that, we consider the "one-dimensional version" of Test cases 1 and 2, solved respectively in $\Omega = [0, 20]$ and $\Omega = [0, 100]$, with same initial and boundary conditions and parareal, and same propagators and configurations as above, but restricted to $x-$direction. Since Test case 1, solved in a Cartesian mesh, has only one direction of propagation, it is equivalent to its one-dimensional version; for Test case 2, the solution of the 2D problem along the $x-$direction is not identical to the solution of the 1D problem, but quite similar behaviors are observed, *e.g.* the convergence of the classical parareal method and the performance of the ROM-based one in function of $\varepsilon_{\text{sv,linear}}$ and $\varepsilon_{\text{sv,nonlinear}}$, even if, in general, a faster convergence is obtained in the one-dimensional case.

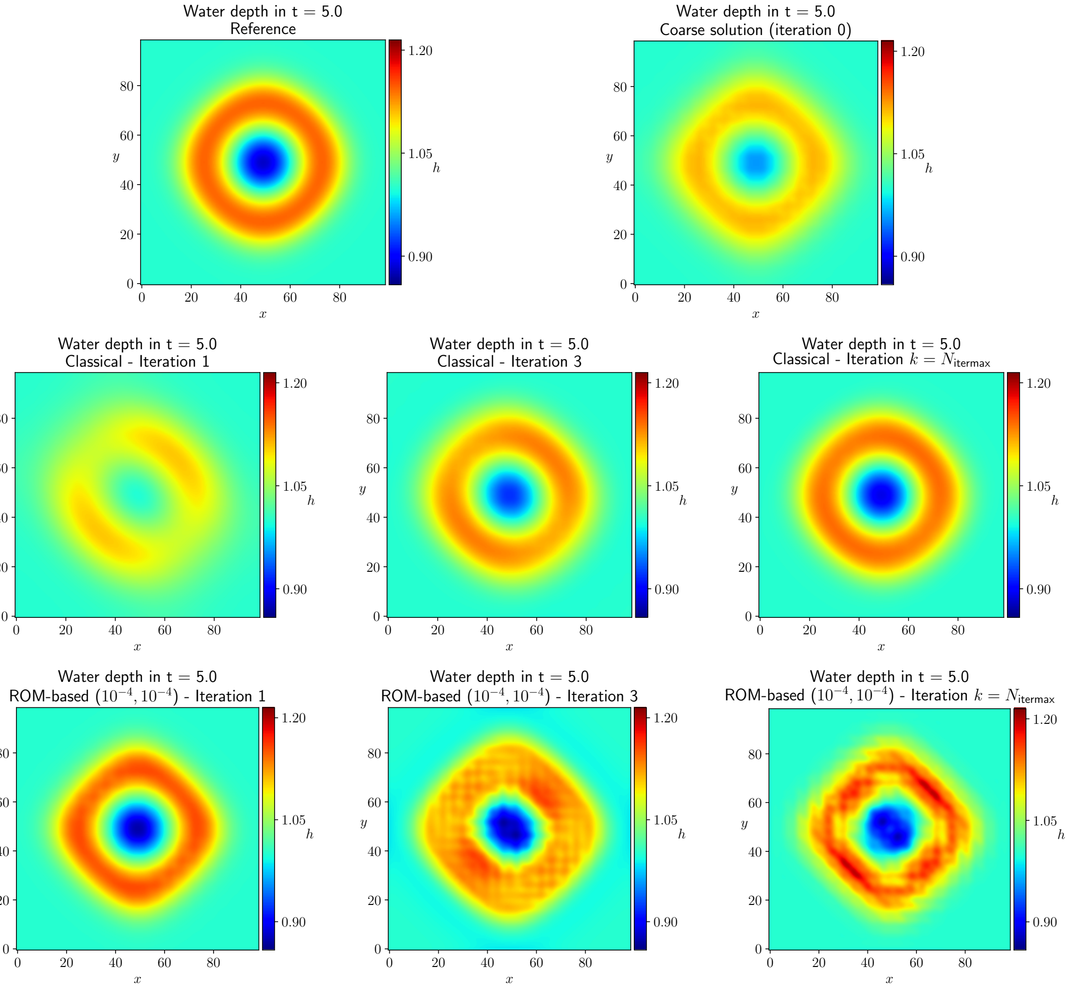


Figure 3.22: Test case 2 (2D) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$: final water depth ($t = T = 5$). Top left: reference solution; top right: coarse solution (0-th solution of the parareal methods). Second row: classical parareal method. Third row: ROM-based parareal method with $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-4}, 10^{-4})$. First, second and third columns: parareal iterations $k = 1, 3, 5$, respectively.

We perform a Fourier analysis by computing the Discrete Fourier Transform (DFT) of a given function ψ defined in Ω (Sundararajan, 2001):

$$\hat{\psi}(\kappa) = \sum_{j=0}^{M_f-1} \psi(x_j) e^{-i2\pi \frac{j}{M_f}\overline{\kappa}} = \sum_{j=0}^{M_f-1} \psi(x_j) e^{-i2\pi j\kappa}$$

where $\overline{\kappa} \in \{0, 1, \dots, M_f - 1\}$, is the wavenumber, expressed in cycles per unit spatial distance, $\kappa := \overline{\kappa}/M_f \in [0, 1[$ is a normalized wavenumber, $\hat{\psi}(\kappa) \in \mathbb{C}$ is the Fourier mode of ψ at κ, i is the imaginary unit and $x_j, j = 0, \dots, M_f$ are the equally M_f spatial points discretizing Ω. We denote by $|\hat{\psi}(k)|$ the amplitude of $\hat{\psi}(k)$. The DFT is performed using the Python's library Numpy.

In the following paragraphs, we study how the amplitude of the wavenumber spectra behaves along parareal iterations for the one-dimensional versions of Test cases 1 and 2. However, as said above, the parareal performance is rather linked to the mismatch between discrete phase speeds of the coarse and fine propagators, and a more comprehensive study should rely on the study of the phase speed of the numerical schemes and discretizations considered here. Also, even if it is observed that similar behaviours are obtained in one and two-dimensional problems, a more complete and complex investigation should be conducted in the two-dimensional context.

Behaviour of the classical and ROM-based parareal methods in Test cases 1 and 2

Figures 3.23 and 3.24 present, respectively for Test cases 1 and 2, the wavenumber spectra $\hat{h}(k)$ of the water depth computed by the fine and coarse propagators and by the classical and ROM-based parareal methods, at the final time of simulation ($t = T$). By comparing both test cases, we observe that the Fourier modes at the end of the spectrum have much larger amplitudes in Test case 1 than in Test case 2, which can give indications for explaining the poor performance of the classical parareal method for solving the former.

Interestingly, the amplitude errors between the fine and coarse propagators are much smaller in Test case 1 than in Test case 2. In the latter case, we notice that the coarse propagator largely overestimates the amplitude of high wavenumber modes. This, however, does not contradict the stability behaviours of the simulations presented above. Indeed, as pointed out by (Ruprecht, 2018), phase speed errors, and not amplitude ones, are at the origin of parareal instabilities. The author observes that, when the fine and coarse propagators have the same phase speed, amplitude errors are quickly corrected through iterations. Therefore, although a good amplitude representation in Test case 1, phase mismatches may exist between the fine and coarse propagators, leading to unstable behaviours. In Test case 2, peaks and valleys of the fine and coarse solutions are in the same spatial position (see Figure 3.21), suggesting a good phase representation, although an amplitude mismatch.

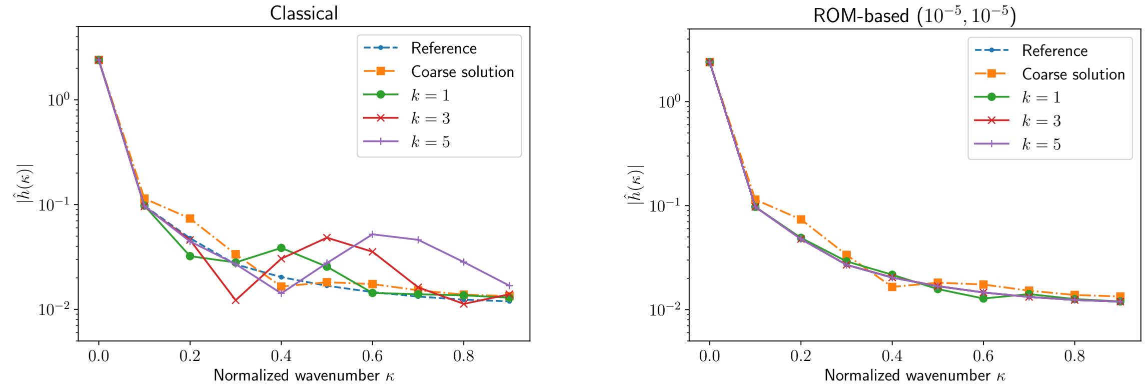


Figure 3.23: One dimensional version of Test case 1: wavenumber spectrum (amplitude of modes in function of the wavenumber) of the water depth computed by the fine (reference) and coarse propagators, and parareal solutions at $t = T$ for given iterations. Left: classical parareal method. Right: ROM-based parareal method with $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-5})$.

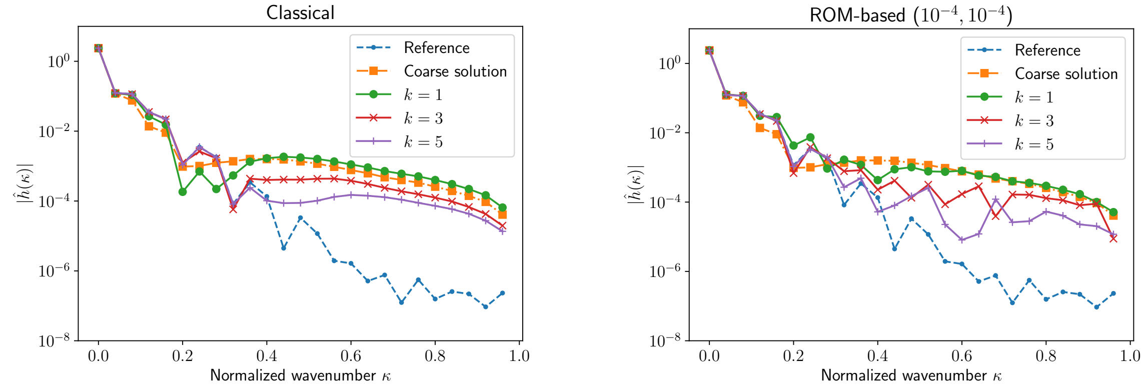


Figure 3.24: One dimensional version of Test case 2: wavenumber spectrum (amplitude of modes in function of the wavenumber) of the water depth computed by the fine (reference) and coarse propagators, and parareal solutions at $t = T$ for given iterations. Left: classical parareal method. Right: ROM-based parareal method with $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-4}, 10^{-4})$.

Concerning the convergence of the spectra in the parareal simulations, we observe, in Test case 1, that an amplification of the modes propagates towards high wavenumbers along iterations of the classical parareal method, whereas the ROM-based one is able to quickly approximate, from the first iteration, the amplitudes in the entire spectrum. It suggests that the formulated ROMs are able to minimize the phase speed mismatch w.r.t. fine propagator. In Test case 2, the classical and the ROM-based methods have a similar behaviour in terms of convergence of the spectrum. After few iterations, small wavenumbers in the parareal solutions are well approximated, but a much slower convergence is observed for high wavenumbers.

Behaviour of the ROM-based parareal method for different model reduction thresholds

Figures 3.13, 3.18 and 3.19 indicate that the ROM-based parareal method converges faster when the POD reduced bases (formulated from snapshots of the solution) are enough high-dimensional (*i.e.* for small values of the threshold $\varepsilon_{\text{sv,linear}}$), but instabilities may arise when the POD-EIM bases (formulated from snapshots of the nonlinear term) are not high-dimensional as well (*i.e.* if $\varepsilon_{\text{sv,nonlinear}}$ is too large). This behaviour is observed systematically in the application of the ROM-based parareal method.

Then, we fix, for Test cases 1 and 2, the smallest threshold $\varepsilon_{\text{sv,linear}}$ considered above (respectively 10^{-7} and 10^{-4}) and we vary $\varepsilon_{\text{sv,nonlinear}}$. The spectra of the water depth provided at the first and fifth parareal iterations are presented in Figure 3.25. In both test cases, a slow or no convergence of high wavenumbers is observed when $\varepsilon_{\text{sv,nonlinear}}$ is not sufficiently small. For Test case 1, the spectrum at the fifth iteration visually coincides with the reference one for $\varepsilon_{\text{sv,nonlinear}} \leq 10^{-5}$; for Test case 2, $\varepsilon_{\text{sv,nonlinear}} = 10^{-4}$ clearly outperforms larger values of $\varepsilon_{\text{sv,nonlinear}}$.

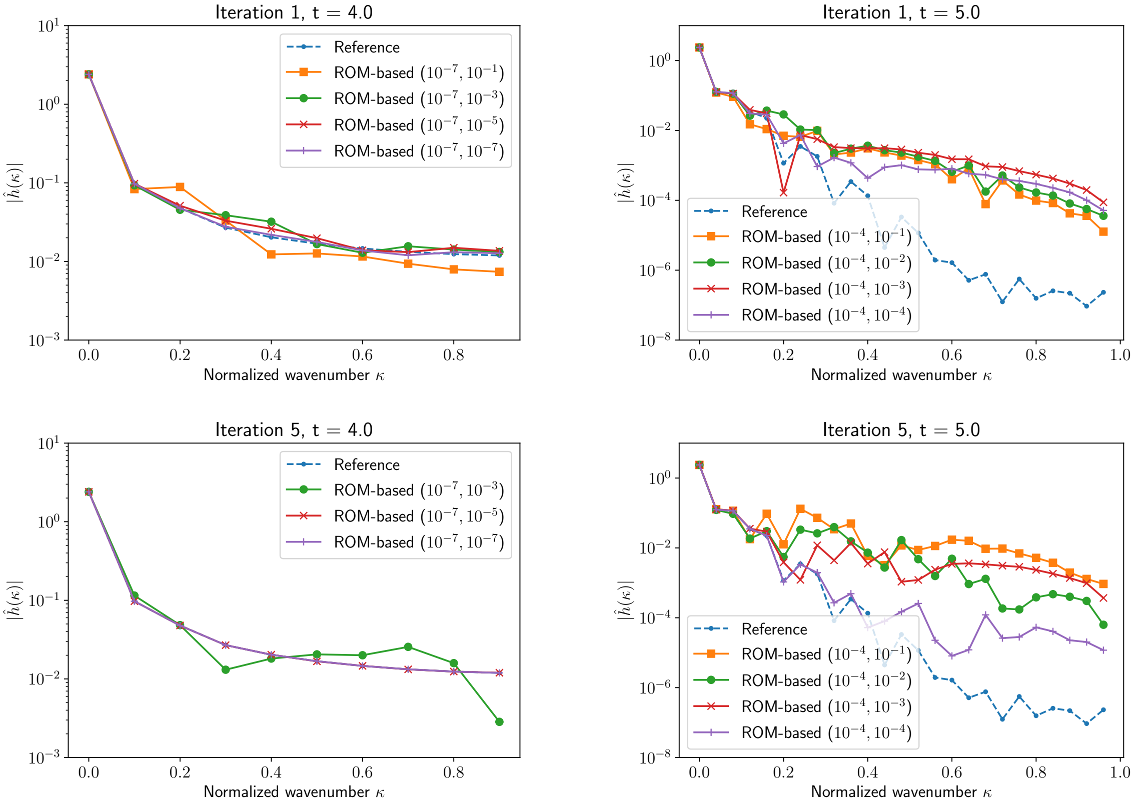


Figure 3.25: One dimensional version of Test cases 1 (left) and 2 (right): wavenumber spectrum (amplitude of modes in function of the wavenumber) of the water depth computed by the ROM-based parareal methods at $t = T$, for fixed thresholds $\varepsilon_{\text{sv,linear}} = 10^{-7}$ (left) and $\varepsilon_{\text{sv,linear}} = 10^{-4}$ (right), and various values of $\varepsilon_{\text{sv,nonlinear}}$. First and second row: first and fifth parareal iterations, respectively.

Also interesting results are illustrated in Figure 3.26, showing the spectra of the water depth provided by the reduced model at the first and fifth iterations, *i.e.* the spectra of $\mathcal{F}^{k}_{r,\delta t}(\mathbb{P}^k \boldsymbol{y}_0, t_n, 0)$, $n = 0, \dots, N_{\Delta t}$; $k = 1, 5$. In the fifth iteration of Test case 1, the spectrum for all $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) =$

$(10^{-7}, \varepsilon_{\text{sv,nonlinear}})$ visually coincides with the reference one, even for $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-7}, 10^{-3})$, whose parareal solution presents strong instabilities and fails in approximating the reference spectrum (see Figures 3.13 and 3.25) An exception is made for $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-7}, 10^{-1})$, which does not reach five iterations due to instabilities in the second parareal iteration. Results are less remarkable in Test case 2, but we notice a clear improvement of the quality of the amplitude spectrum of the reduced model by reducing $\varepsilon_{\text{sv,nonlinear}}$. Therefore, a possible explication for the ROM-based parareal performance is that, for small $\varepsilon_{\text{sv,linear}}$, decreasing $\varepsilon_{\text{sv,nonlinear}}$ improves the reduced model in terms of spectrum amplitude, but if $\varepsilon_{\text{sv,nonlinear}}$ is not sufficiently small, phase speed errors w.r.t. the reference model may still be present and lead to instabilities of the ROM-based parareal method.

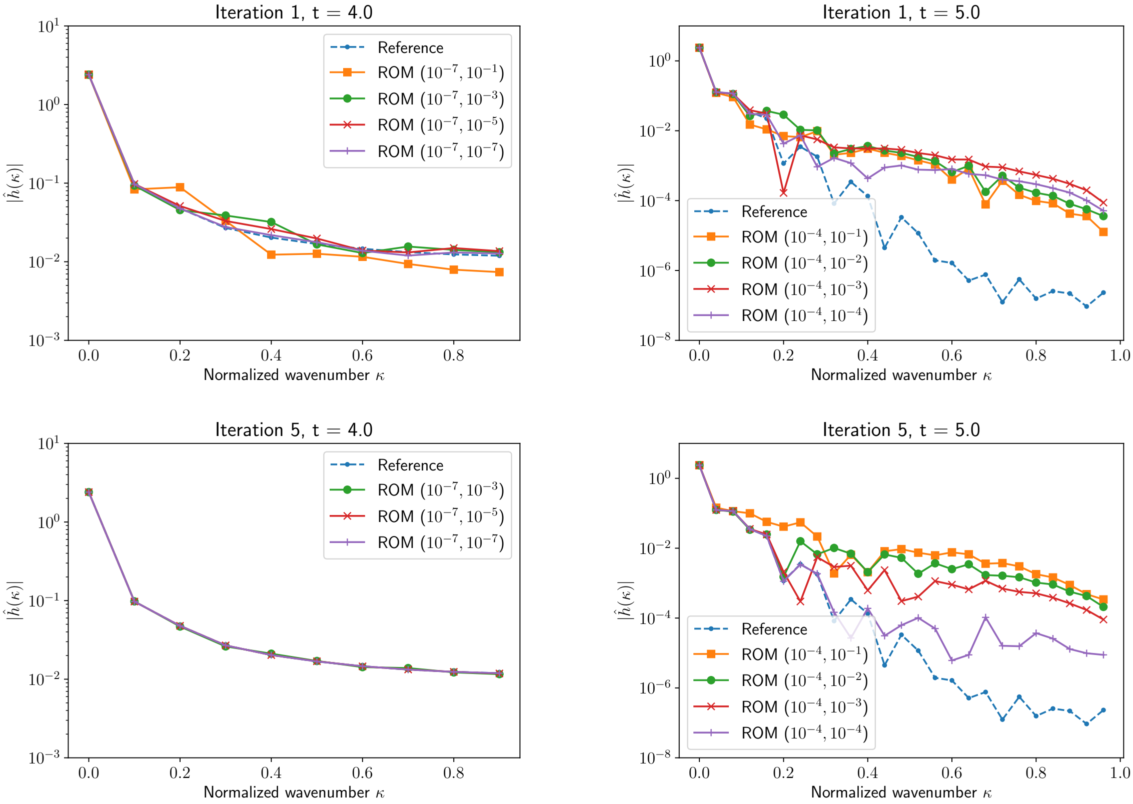


Figure 3.26: One dimensional version of Test cases 1 (left) and 2 (right): wavenumber spectrum (amplitude of modes in function of the wavenumber) of the water depth at $t = T$ computed by the reduced-order models formulated at the first (first row) and fifth (second row) parareal iterations, for fixed thresholds $\varepsilon_{\text{sv,linear}} = 10^{-7}$ (left) and $\varepsilon_{\text{sv,linear}} = 10^{-4}$ (right), and various values of $\varepsilon_{\text{sv,nonlinear}}$.

3.6.5 A first discussion on the influence of the length time slices

In all numerical examples presented above, it was considered that the parareal time slices ΔT coincide with the coarse time steps Δt associated to the coarse propagator $\mathcal{G}_{\Delta t}$. Since the temporal domain is relatively small, containing only $N_{\Delta t} = 20$ coarse time steps, it is a quite natural choice allowing to make use of the maximum possible parallelization, since the fine correction step of the parareal algorithm is distributed among $N_{\Delta T}$ parallel processors.

It is possible, evidently, to choose $\Delta T > \Delta t$, *i.e.* $N_{\Delta T} < N_{\Delta t}$, which may lead both to positive and negative influence on the performance of the parareal method. By increasing ΔT, we reduced the number $N_{\Delta T}$ of time slices, and the convergence is expected to be faster, since the parareal solution converges exactly towards the reference one in at most $N_{\Delta T}$ iterations, as discussed in Section 3.2.4. Moreover, as observed by Ruprecht (2018), the parareal convergence is slower when the number of time slices increase, specially for high wavenumbers. On the other hand, the reduction of $N_{\Delta T}$ also reduces the expected speedup. More precisely, bound (3.13), formulated in the framework of the classical parareal method, should be rewritten as

$$s_{\text{classical-parareal}}(\hat{k}) < \frac{\min(N_p, N_{\Delta T})}{\hat{k}} \tag{3.55}$$

i.e. if $N_{\Delta T} < N_p$, the fine correction step is distributed among only $N_{\Delta T}$ parallel processors, meaning that all available N_p processors cannot be used for parallelizing the parareal simulation.

We present in the following paragraphs a first study of the influence of the time slice length on the convergence of the classical and ROM-based parareal methods. For the small problems considered here, only few lengths of time slices can be chosen. A more detailed study covering both convergence and speedup and considering problems defined in larger temporal domains (thus allowing larger ranges for ΔT) is proposed in Chapter 5.

We consider Test cases 1 and 2. For each of them, we perform simulations both in the classical and ROM-based frameworks, with $N_{\Delta T} \in \{5, 10, 20\}$, corresponding respectively to $\Delta T \in \{0.8, 0.4, 0.2\}$ for Test case 1 and $\Delta T \in \{1, 0.5, 0.25\}$ for Test case 2. We still consider $N_{\text{itermax}} = 5$. Note that, in the cases with $N_{\Delta T} = 5$, exact convergence is obtained after $N_{\text{itermax}} = 5$ parareal iterations. In the ROM-based simulations, we fix the model reduction thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-5})$ and $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-4}, 10^{-4})$ respectively for Test cases 1 and 2.

Figure 3.27 compares the evolution of the relative error $\overline{e}^k$ for each simulation. We observe, both for Test cases 1 and 2, that the increase of the time slice length has a positive influence on the convergence of the classical parareal method. In Test case 1, instabilities are controlled and the error decreases monotonically for large enough time slices. In Test case 2, which already converges for small ΔT, the convergence speed increases with larger ΔT. A less clear influence is observed in the ROM-based simulations. In Test case 1, the convergence slows down by increasing ΔT, contrary to what is observed in the classical method. In Test case 2, a slightly faster convergence is observed by increasing ΔT, but still relatively slow. As will be discussed in details in the next chapter, the performance of the ROM-based method also depends on the number of snapshots used in the model reduction. Since the snapshots are taken on the parareal time instants, their number decreases when ΔT increases, which can degrade the quality of the ROMs w.r.t. the reference solution since they are formulated using a smaller amount of input information.

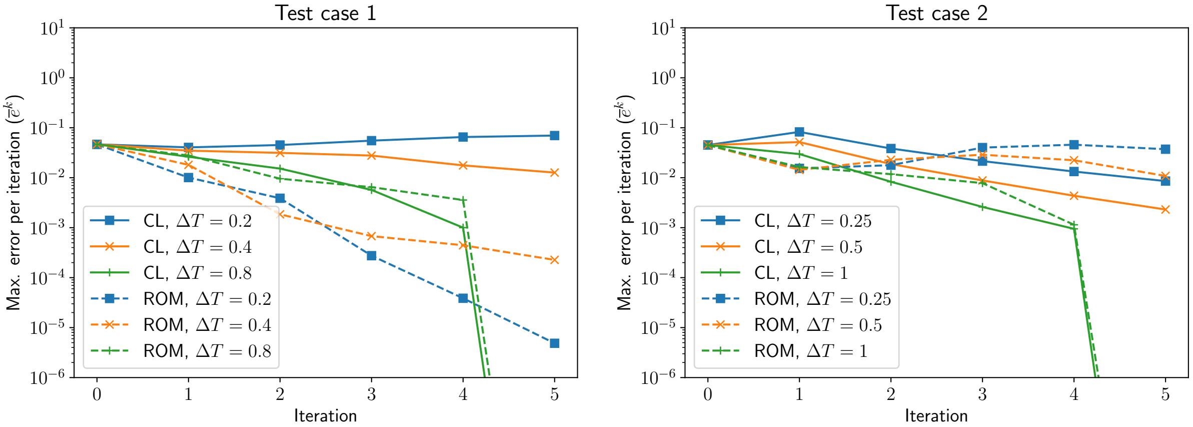


Figure 3.27: Test cases 1 (left) and 2 (right): evolution of the relative error $\overline{e}^k$ for the classical ("CL", full lines) and ROM-based parareal ("ROM", dashed lines) methods using various lengths of time slices. The ROM-based simulations use the thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-5})$ (Test case 1) and $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-4}, 10^{-4})$ (Test case 2).

As said above, a more detailed discussion on the influence of ΔT will be performed in Chapter 5 by using larger problems in time. In this and the next chapter, we still work on relatively small problems and consider $\Delta T = \Delta t$, unless explicitly specified.

3.6.6 Influence of nonlinearities

We consider Test case 2 for investigating how the intensity of nonlinear effects influence the convergence of the classical and ROM-based parareal methods. The importance of nonlinear effects in wave propagation models can be described by the ratio between the wave amplitude and the mean water depth (Filippini et al., 2015). Therefore, the nonlinearities in Test case 2 can be controlled by modifying its initial condition (3.12), *e.g.* by keeping a fixed Gaussian amplitude and varying the mean water depth h_0. We consider $h_0 \in \{1/16, 1/4, 1, 4\}$, with $h_0 = 1$ as the default case presented above. We do not consider $h_0 > 4$ since the solution quickly reaches the domain boundaries, having its profile strongly modified.

Note that the reference solution is different for each value of h_0, and the relative error (3.3) between the coarse and fine propagators (*i.e.* at iteration $k = 0$) is naturally smaller for larger h_0. Therefore, for properly comparing the error evolution in each case, we present, in Figure 3.28, the evolution of $\overline{e}^k/\overline{e}^0$, *i.e.* the fraction represented by the maximum error per iteration relative to the initial error (at the 0-th iteration). In the classical parareal method, after an initial degradation of the solution (which is due to the linear interpolation procedure, as will be discussed in Section 3.6.8), the error decreases in the following iterations for all values of h_0, with a slightly faster convergence for larger h_0 (*i.e.* for less important nonlinear effects). In the ROM-based framework, an opposite behaviour is observed: after an initial important error decrease in all simulations, the quality of the solution degrades in the following iterations, specially for larger values of h_0, indicating a bad quality of the formulated ROMs. The different behaviours of the classical and ROM-based methods are clearer when we analyse the evolution in time of $e_n^{N_\text{itermax}}/e_n^0$, *i.e.* the fraction represented by the error $e_n^{N_\text{itermax}}$, computed at the last iteration, w.r.t. e_n^0, computed at $k = 0$. These errors are also presented in Figure 3.28. In the classical parareal method, smaller errors are obtained for larger values of h_0, and in all cases we observe a relatively small degradation of the solution in time. On the other hand, this degradation is much more important in the ROM-based parareal method for large h_0, expliciting the poor quality of the model reduction for advanced time steps.

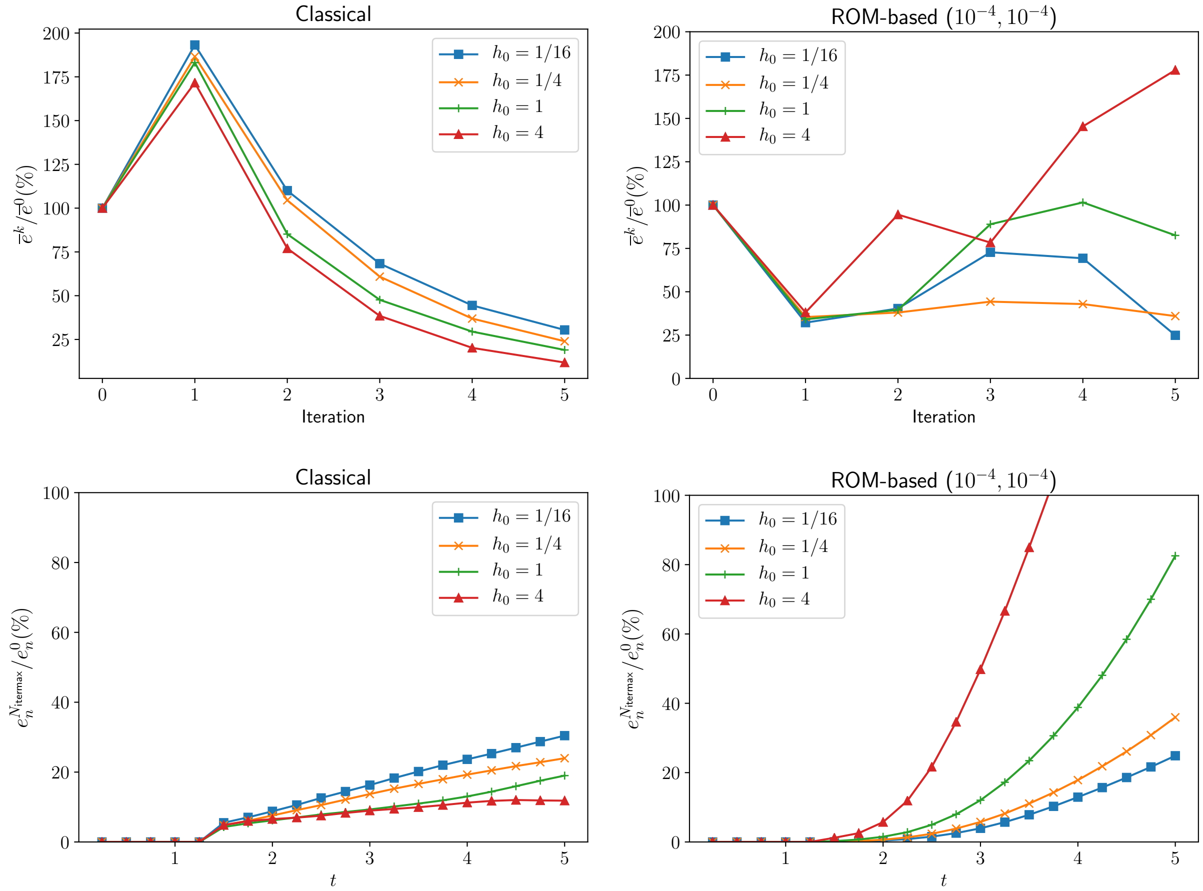


Figure 3.28: Test case 2 (2D) with various initial water depths h_0: fraction represented by the maximum relative error per iteration (top) and error per time instant at the final iteration (bottom) relative to the initial error, *i.e.* $\overline{e}^k/\overline{e}^0$ and $e_n^{N_\text{itermax}}/e_n^0$. Values expressed in percentage. Left; classical parareal method. Right: ROM-based parareal method with $(\varepsilon_\text{sv,linear}, \varepsilon_\text{sv,nonlinear}) = (10^{-4}, 10^{-4})$.

Some hypotheses can be formulated for understanding these different behaviours of the classical and ROM-based methods. Figure 3.29 illustrates the final water depth, at $t = T$, provided by $\mathcal{F}_{\delta t}$ with $h_0 \in \{1/16, 1, 4\}$. For small initial depth, the larger nonlinearity of the problem leads to the formation of propagating shocks, *i.e.* the profile of the solution is closer to the one of Test case 1. However, due to the numerical discretization, this profile is relatively smooth and simple, when compared to the profiles obtained with larger initial water depths, for which we observe the formation of peaks and valleys. For $h_0 = 4$, the propagating wave reaches the domain boundaries near the end of the simulation, increasing the complexity of the solution profile. As will be discussed in the next chapter, POD-based reduced-order models are known to perform better when approximating smooth and simple solutions. It can explain the poor performance of the ROM-based method when h_0 is larger, since the model reduction becomes more challenging.

On the other hand, if we look to the wavenumber spectra of the reference solutions for each value of h_0, computed using the one-dimensional version of Test case 2 (Figure 3.30), we notice that the amplitudes of the Fourier modes increase for smaller h_0, mainly for medium to high wavenumbers (thus looking more similar to the spectrum in Test case 1), which suggests a possible explication to the observed performance of the classical parareal method, with a slightly faster convergence for larger h_0. In the end of the simulation ($t = T$), we notice an amplification of high wavenumbers for $h_0 = 4$, which may be due to the reflection of the solution on the domain boundaries; however, this reflection only occurs near the end of the temporal domain, and no impact on the overall convergence behaviour is observed.

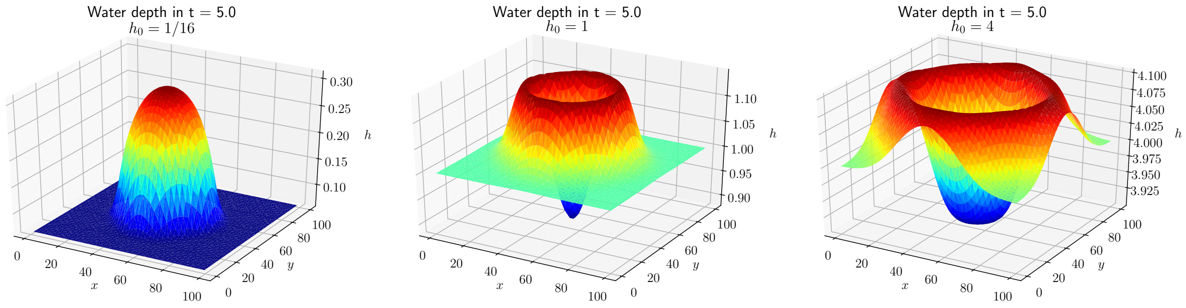


Figure 3.29: Test case 2 (2D): water depth provided by the reference model $\mathcal{F}_{\delta t}$ at $t = T$ with $h_0 = 1/16$ (left), $h_0 = 1$ (middle) and $h_0 = 4$.

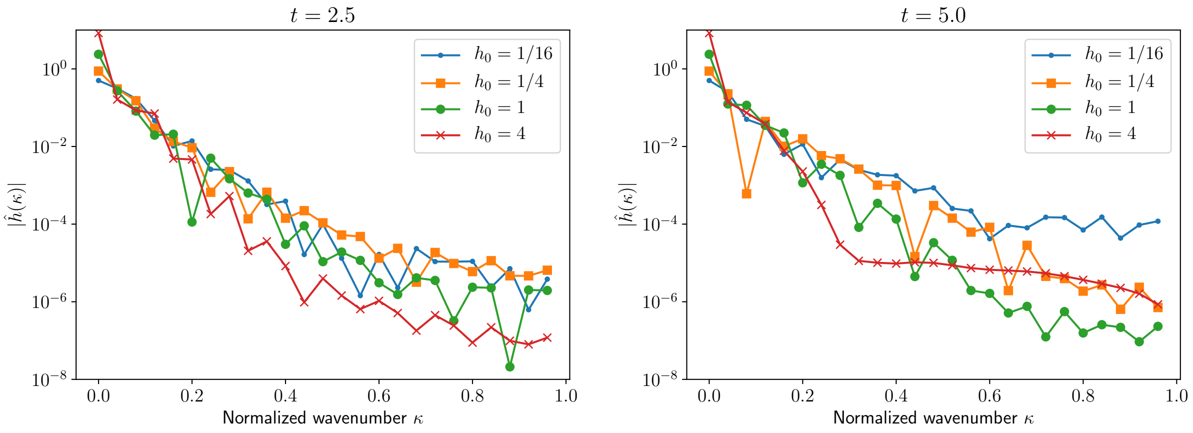


Figure 3.30: One dimensional versions of Test case 2 (right): wavenumber spectrum (amplitude of modes in function of the wavenumber) of the water depth at $t = T/2$ (left) and $t = T$ (right) computed by $\mathcal{F}_{\delta t}$ with various initial water depths h_0.

3.6.7 Influence of the type of computational mesh

All presented simulations were performed considering Cartesian meshes, both for the fine and coarse propagators. We investigate here if the computational mesh is relevant for the performance of the classical and ROM-based parareal methods. For that, we consider Test case 1 (pseudo-2D without spatial coarsening) and Test case 2 (2D) with the same parameters as described respectively in Tables 3.1 and 3.3, but using structured triangular meshes instead of Cartesian ones. The mesh sizes are chosen such as to respect the CFL stability condition both by the coarse and fine propagators, and we compare the solutions for some chosen pairs $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$ of model reduction thresholds.

Test case 1 (pseudo-2D without spatial coarsening)

The structured triangular mesh for this test case has 128 cells, against 400 in the quadrilateral, Cartesian mesh. Under this configuration, the classical parareal method presents even more important instabilities and is not able to complete all iterations, as shown in Figure 3.31. For the ROM-based parareal method, we perform simulations with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} \in \{10^{-3}, 10^{-4}, 10^{-5}\}$. The maximum error per iteration, compared to the ones obtained with the same thresholds but in the case of the Cartesian mesh, are also presented in Figure 3.31. We notice that the use of a triangular mesh clearly degrades the performance of the ROM-based parareal method, with strong instabilities for smaller thresholds and slow convergence for the largest ones. For $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-7}$, the simulation does not complete four parareal iterations.

This behaviour can be explained by the solution profile induced by the computational mesh. Even if the analytical flow is purely one-dimensional, the numerical solution is not, since the triangular mesh contains interfaces not aligned with the $x-$ and $y-$directions. Indeed, as shown in Figure 3.32, the $y-$unit discharge presents strong discontinuities, induced by the orientation of the diagonal interfaces (we recall that, with the quadrilateral mesh, the $y-$unit discharge is strictly equal to zero in this case). As a consequence, oscillations are also observed in the profile of the $x-$ unit discharge. Then, the poor convergence and stability of the ROM-based parareal method are due, firstly, to the more complex problem (2D instead of 1D), and, secondly, to the introduction of discontinuities of the solution, which represents a challenge for the model reduction, as will be discussed in Section 4.2. Therefore, if an unstructured mesh were used, we could expect an even worse behaviour.

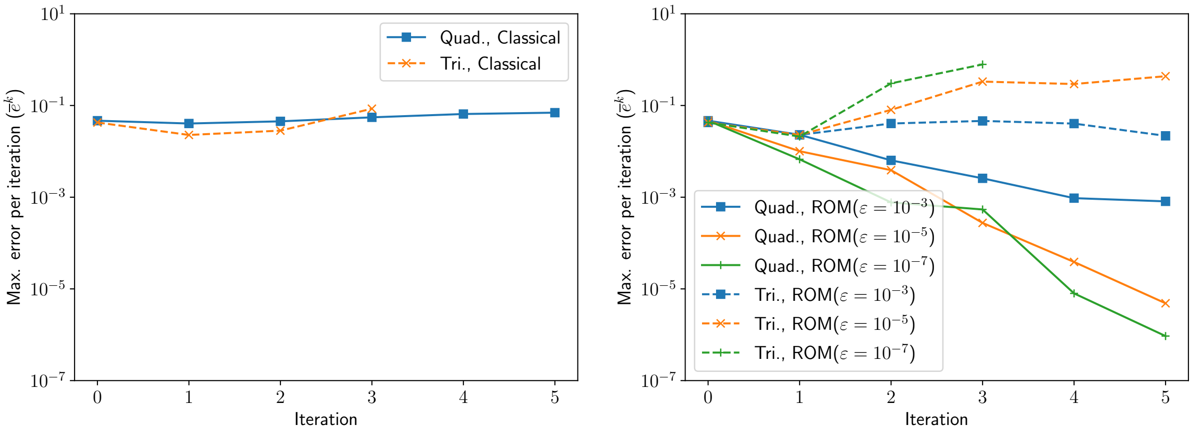


Figure 3.31: Test case 1 (pseudo-2D without spatial coarsening): evolution of the relative error $\bar{e}^k$ for the classical (left) and ROM-based (right) parareal methods using $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} \in \{10^{-3}, 10^{-4}, 10^{-5}\}$, considering Cartesian ("Quad.", full lines) and structured triangular ("Tri.", dashed lines) computational meshes. Some of the simulations do not complete all iterations due to instabilities.

Test case 2 (2D)

For the two-dimensional test case, we consider structured triangular meshes both for the coarse and fine propagators, respectively with 392 and 2312 cells (respectively against 400 and 2500 in the quadrilateral meshes). The maximum error per iteration for the classical and ROM-based parareal method, the last

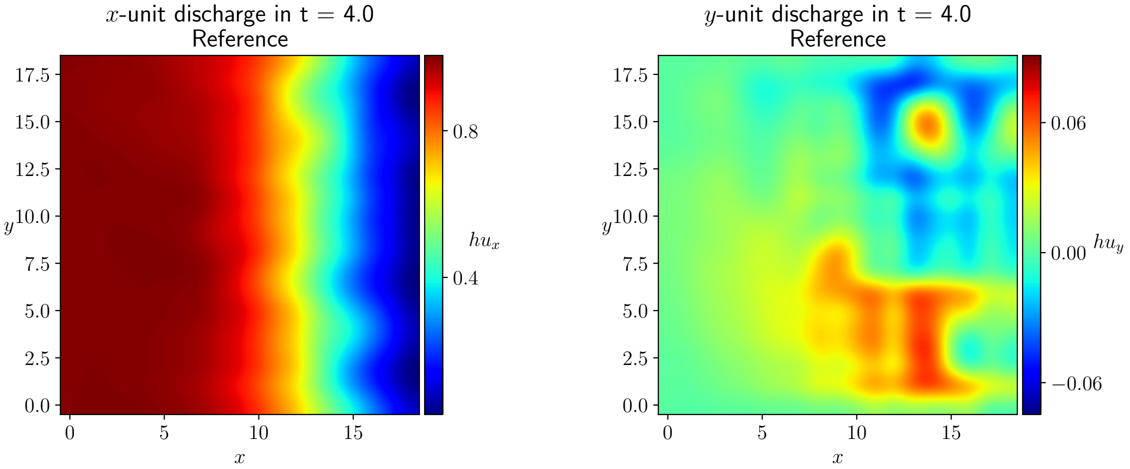


Figure 3.32: Test case 1 (pseudo-2D without spatial coarsening) with a structured triangular mesh: $x-$ (left) and $y-$unit discharge at $t = T$. Solution provided by the fine, reference propagator $\mathcal{F}_{\delta t}$.

one considering the thresholds $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, is presented in Figure 3.33. Note that a slightly better performance is obtained using Cartesian meshes, but much less notably than in Test case 1. For the classical method, the simulations using Cartesian and triangular meshes present nearly the same convergence rate, and for the ROM-based method we observe similar unstable and slow converging behaviours. Indeed, since Test case 2 is already two-dimensional, the effects of introducing a computational mesh not aligned with the axes is less important than in Test case 1.

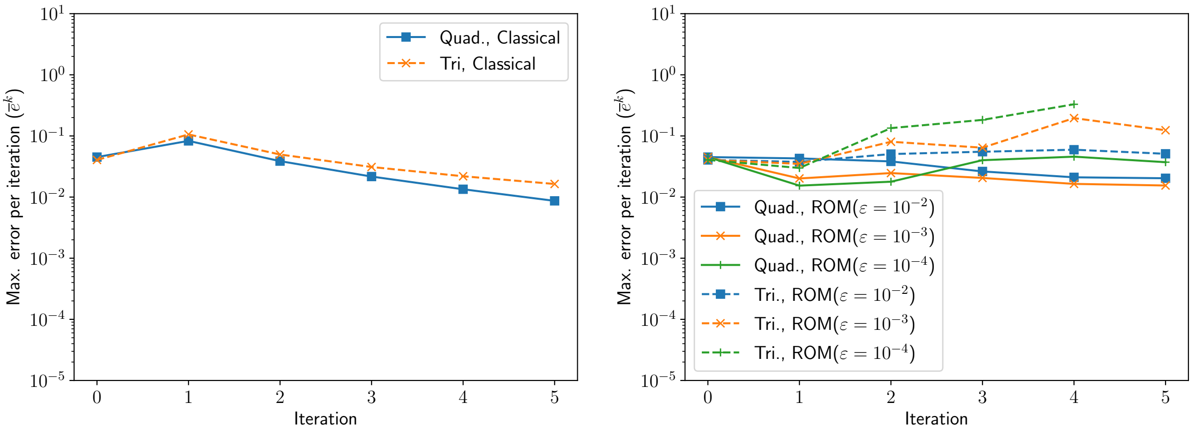


Figure 3.33: Test case 2 (pseudo-2D without spatial coarsening): evolution of the relative error $\bar{e}^k$ for the classical (left) and the ROM-based (right) parareal methods using using $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, considering Cartesian ("Quad.", full lines) and structured triangular ("Tri.", dashed lines) computational meshes. Some of the simulations do not complete all iterations due to instabilities.

3.6.8 Influence of the interpolation procedure

Spatial coarsening between the fine and coarse propagators represents an additional challenge for parareal simulations. In the framework of the one-dimensional advection-diffusion equation, Ruprecht (2014) noticed that a faster convergence of the parareal method is obtained when higher-order interpolation schemes are used, specially when the problem is discretized with high spatial resolutions. Using a Fourier analysis applied to the one-dimensional advection equation, Lunet (2018) identified that the parareal method with spatial coarsening leads to the formation of wavenumbers not present in the initial solution and to the amplification of high wavenumbers, these effects being less important when higher-order interpolation is used. These benefits are confirmed by Lunet (2018) in the simulation of the three-dimensional Navier-Stokes equations using Cartesian meshes.

In the following paragraphs, we consider theses aspects for evaluating the influence of the interpolation scheme on the presented simulations and to decide the main scheme to be used along this work. This choice should rely on the types of mesh commonly used in the envisaged applications of this work. As stated by Kim et al. (2014), the optimal mesh for flooding simulation depends on the problem and the computational domain. For simulations in rectangular channels, Cartesian meshes are more efficient (with efficiency defined by the authors as accuracy per computational effort), but unstructured meshes perform better with uneven topographies and when obstacles are present in the domain. Local mesh refinements and the use of mixed meshes combining triangular and quadrilateral elements are also shown to improve the accuracy. Therefore, in the context of urban floods, unstructured meshes are more likely to be used (see *e.g.* the meshes shown in Figures 1.2 and 2.8). Moreover, even if structured meshes are used, the presence of obstacles (*i.e.* "holes" in the mesh) may impede the application of high-order interpolation schemes, since the necessary points for the interpolation stencil may not be available.

Thus, we consider a more flexible interpolation scheme, being able to interpolate between unstructured grids. It is also applied for the test cases presented in this chapter, even if interpolation procedures based on regular grids could be used for these examples. Namely, we use interpolation procedures based on Delaunay triangulations of the cell barycenters, as detailed in Appendix C. We consider linear and cubic interpolations: for the linear one, we use the open-source C++ library *delaunay_linterp*[1], which was incorporated to the code developed in this work; for the cubic one, we use the function *CloughTocher2DInterpolator* of the Python Scipy library[2], which produces cubic, piecewise continuously differentiable interpolants. In both cases, the interpolation weights are precomputed and reused for all simulations using the same computational meshes.

All simulations with spatial coarsening presented along this chapter used linear interpolation. We choose some of them for comparing the results obtained with the cubic one, namely the simulations using the classical parareal method (both for Test cases 1c and 2), the ROM-based one with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-3}$ and $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-7}$ (for Test case 1c), and the ROM-based one with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-2}$ and $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$ (for Test case 2).

The evolution of the errors per iteration and time for Test cases 1c and 2 are presented respectively in Figures 3.34 and 3.35. The maximum error per iteration is presented in Figure 3.36 for both test cases.

For the classical parareal method, there is almost no difference in using linear or cubic interpolation in Test case 1c, with the error curving presenting the same (unsatisfactory) behaviour. On the other hand, notable improvements are observed in Test case 2, specially in the first parareal iteration. The initial degradation of the solution with linear interpolation is no longer observed with the cubic one.

On the other hand, the observed behaviours are quite different in the ROM-based parareal method. We recall that, in this case, spatial interpolation is performed only in the $0-$th parareal iteration, since the coarse propagator is not used in the following ones. With cubic interpolation, we notice an important loss of quality of the solution of Test case 1c, specially in the case $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-7}$, which presented a fast convergence with linear interpolation, but is unstable with the cubic one, not completing two parareal iterations. The reasons for this behaviour are not clear. A possible explication for this result is that, in this problem, the cubic interpolation introduces behaviours of the solution not corresponding to the dynamics of the problem, thus leading to unstable reduced models. A discussion on the quality of the reduced order models formulated in the parareal framework is addressed in the next chapter. We can also suppose that, the computational meshes being relatively small, there are not enough cells for properly defining stencils for the cubic interpolation. For Test case 2, the ROM-based parareal method with linear or cubic interpolation behaves similarly.

A major drawback in increasing the interpolation order is the larger computational cost. As discussed in Appendix C, the stencils for the cubic interpolation are much larger than in the nonlinear case, since global approximations of the solution derivatives are performed, and important costs are observed, mainly when the fine and coarse meshes contain a large number of cells (which is the case of Test case 2 compared to Test case 1). It is clear in Table 3.9, showing the speedup after five iterations for Test cases 1c and 2 using linear and cubic interpolation, with severe reductions of the computational cost in the two-dimensional case. This reduction is more important in the classical parareal framework, since interpolations in both senses between the fine and coarse meshes are performed at each iteration. In the ROM-based method, interpolation is only performed in the 0-th iteration, from the coarse to the fine

[1] https://github.com/rncarpio/delaunay_linterp

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.CloughTocher2DInterpolator.html

mesh, and the impacts of using a higher-order interpolation are less important.

Based on these observations, we consider the linear interpolation procedure in the following chapters, which focus on improving the ROM-based parareal method and studying the behaviour of both parareal methods for solving problems defined in larger spatial and temporal domains. Therefore, the proposed studies are performed considering the same interpolation scheme for all simulations. In Chapter 6, for the application of parareal methods for coupling the classical and porosity-based shallow water models, we briefly come back to the discussion on the influence of the spatial interpolation order.

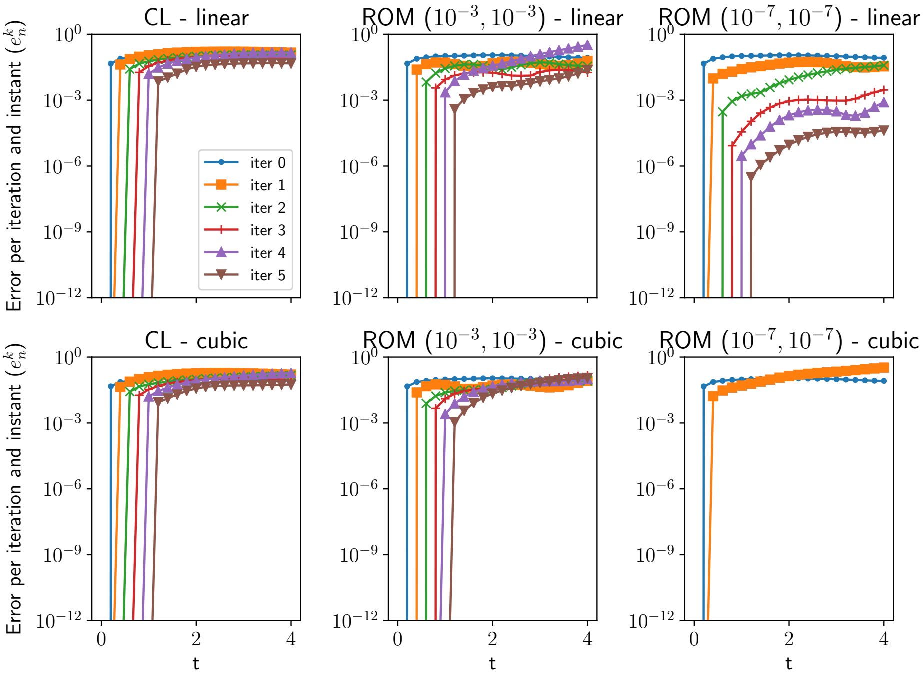


Figure 3.34: Test case 1c (pseudo-2D with spatial coarsening): relative error e_n^k with linear (first row) and cubic (second row) spatial interpolation. “CL” stands for the classical parareal method and “ROM $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$” stands for the ROM-based parareal method using the thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$. The simulation ROM $(10^{-7}, 10^{-7})$ is unstable and does not complete two parareal iterations.

	Test case 1c			Test case 2		
Interpolation	CL	ROM $(10^{-3}, 10^{-3})$	ROM $(10^{-7}, 10^{-7})$	CL	ROM $(10^{-2}, 10^{-2})$	ROM $(10^{-4}, 10^{-4})$
Linear	2.58	0.77	0.68	3.37	1.57	1.09
Cubic	2.47	0.70	*	0.70	1.31	1.05

Table 3.9: Test cases 1c and 2: speedup $s(5)$ at the fifth iteration with linear (first row) and cubic (second row) spatial interpolation. “CL” stands for the classical parareal method and “ROM $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$” stands for the ROM-based parareal method using the thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$. The simulation indicated with an asterisk presents instabilities and does not complete five iterations.

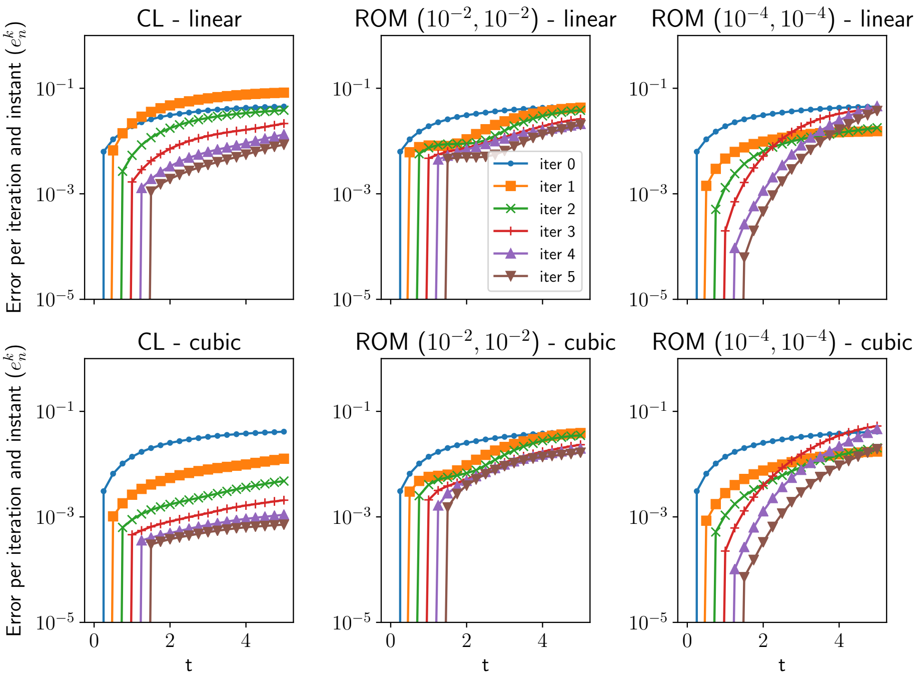


Figure 3.35: Test case 2 (2D): relative error e_n^k with linear (first row) and cubic (second row) spatial interpolation. "CL" stands for the classical parareal method and "ROM $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$" stands for the ROM-based parareal method using the thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$.

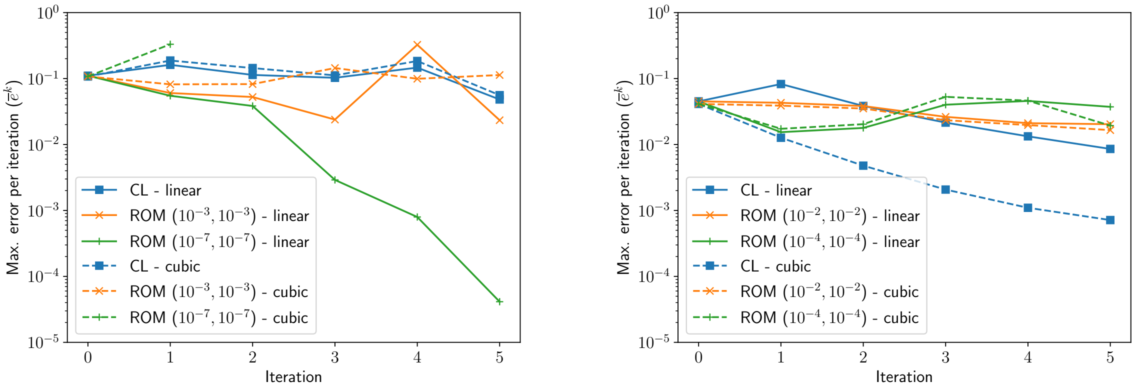


Figure 3.36: Test cases 1c (left) and 2 (right): relative error $\bar{e}^k$ using the ROM-based parareal method with linear (full lines) and cubic (dashed lines) spatial interpolation. "CL" stands for the classical parareal method and "ROM $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$" stands for the ROM-based parareal method using the thresholds $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$.

3.6.9 Some conclusions on the numerical simulations

The following conclusions can be made from the three test cases presented in this section:

- The deficiencies of the classical parareal method when applied to hyperbolic problems is well illustrated in the relatively simple Test cases 1 and 1c. A better convergence behaviour is observed in Test case 2, but it is still far below the performance verified for parabolic problems (see *e.g.* Figure 3.4). Under the same configurations, and depending on the considered model reduction parameters, the ROM-based method is able to provide a much faster convergence and also more stable behaviours, specially in Test case 1 and 1c, at the cost of a considerably more expensive simulation, when compared to the classical method;

- The performance of the ROM-based parareal method is strongly determined by the reduced-order models formulated along the iterations. Both the subspaces computed from snapshots of the solution and from snapshots of the nonlinear terms have an influence, but with different behaviours;

- Low-dimensional POD spaces computed from snapshots of the solution (large values of $\varepsilon_{\text{sv,linear}}$) lead to a very slow convergence of the parareal method, with a very small error decrease along iterations. We can conclude that the ROMs do not contain enough information for properly approximating the reference model. In the three test cases, when the largest $\varepsilon_{\text{sv,linear}}$ is used ($\varepsilon_{\text{sv,linear}} = 10^{-1}$), no improvement is observed by decreasing $\varepsilon_{\text{sv,nonlinear}}$ (compare the graphs in the first row of Figures 3.13, 3.18 and 3.19);

- On the other hand, high-dimensional POD spaces computed from snapshots of the solution (small values of $\varepsilon_{\text{sv,linear}}$) are able to accelerate the parareal convergence, but may lead to unstable behaviours. Several simulations stop before N_{itermax} due to the computation of negative water depths.

- These instabilities are reduced by increasing the dimension of the POD-EIM spaces computed from snapshots of the nonlinear terms (*i.e.* by reducing $\varepsilon_{\text{sv,nonlinear}}$). We conjecture that this behaviour is due to the improvement of the approximating reduced space and of the interpolation procedure;

- Even if the cost of the POD does not depend on the thresholds $\varepsilon_{\text{sv,linear}}$ and $\varepsilon_{\text{sv,nonlinear}}$ (since the truncation is a post-process of the model reduction), the improvements obtained by increasing the dimensions of the subspaces has as drawback more expensive parareal simulations, due to the more expensive greedy algorithm for choosing the interpolation points, the larger ROM matrices to be computed and the more expensive simulation of the larger-dimensional ROMs;

- Note however that, even when small thresholds $\varepsilon_{\text{sv,linear}}$ and $\varepsilon_{\text{sv,nonlinear}}$ are considered, the ROMs remain effectively low-dimensional problems when compared to the FOM, specially in the two-dimensional test case, in which the dimensions of the reduced spaces remain at the other of tenths, against the thousands of cells and interfaces of the fine computational mesh.

- The three test cases were presented in an increasing order of complexity, and we observe a influence of this factor on the ROM-based parareal performance. Firstly, by comparing the one-dimensional test cases (Test cases 1 and 1c) with the two-dimensional one (Test case 2), much smaller errors are obtained in the former case. Secondly, the errors are smaller in Test case 1 than in Test case 1c, with remarkable error reductions in the first parareal iteration: since there is no spatial coarsening between the fine and the coarse propagator in the former, the snapshots and the ROM are a better representation of the reduced model. The instabilities are also influenced by the complexity of the problem, being less remarkable in the one-dimensional simulations. Indeed, in Test case 2, the chosen range of $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$ is smaller than in Test cases 1 and 1c due to more critical unstable behaviours of the method.

- Finally, simulations were performed for identifying other factors that may influence the performance of the classical and ROM-based parareal methods. We considered different degrees of nonlinearity of the problem (by choosing various initial water depths for a given wave amplitude), the use of Cartesian or triangular meshes, the spatial interpolation order, and the length ΔT of the parareal time slices. The classical parareal method showed to be most influenced by the length of time slices and by the order of spatial interpolation. For the ROM-based method, the simulations indicate that its performance strongly depends on the quality of the ROMs and on its ability in approximating the reference solution. Notably, a better convergence behaviour is observed when the reference solution has simpler profiles, which can be obtained by using Cartesian meshes when there is a preferential flow direction, or by avoiding the formation of peaks and valleys in the solution profile via an increment of the nonlinearity of the problem. These factors had smaller or even opposite effects on the performance of the classical parareal method. The influence of ΔT is less clear for the ROM-based method, since it also affects the number of input snapshots for the model reduction, which will be considered in detail in the next chapter. Concerning the spatial interpolation, no or even negative effects are observed in the ROM-based method, contrary to the classical one, and more detailed investigations should be performed for identifying the causes. In the rest of this work, we consider linear interpolation, unless explicitly specified.

3.7 Conclusion of the chapter

In this chapter, we made an overview of the parareal method and some of its variants specially conceived for treating hyperbolic problems. The classical or original parareal was presented and we discussed and illustrated its weaknesses when applied to hyperbolic or advection-dominated problems by using some simple numerical examples. Then, we presented the evolution of variants of the parareal method that aim to overcome these issues by reusing information from the previous parareal iterations. An introduction to this idea was presented with the Krylov-subspace-enhanced parareal method, designed for improving the parareal performance when solving linear hyperbolic problems, and, with more details, we presented the introduction of reduced-order models formulated on-the-fly in the parareal algorithm. By formulating ROMs using the combined POD-EIM procedure, this ROM-based parareal method can be efficiently implemented for treating nonlinear hyperbolic problems. Speedup estimations for the classical and the ROM-based parareal method were derived and interpreted, providing guidelines for their implementation.

After this presentation of the methods, we proposed their application to the parallel-in-time simulation of the two-dimensional nonlinear SWE. We considered both the classical and ROM-based parareal methods. For implementing the latter, we firstly formulated a POD-EIM reduced-order model for the SWE discretized with a finite volume scheme.

The numerical tests presented in this chapter performed a coupling between the classical SWE at different scales. Even if relatively small, the tests allowed to obtain a first overview of the behaviour and performance of the parareal methods applied to this problem. By comparing the results of the classical and ROM-based methods using the same configuration, we noticed that the latter is able to provide faster convergence and more stable behaviour, but with a larger computational cost. Moreover, the performance of the ROM-based parareal method strongly depends on the truncation of the POD basis computed with POD or POD-EIM, indicating that the quality of the formulated ROMs is a major determinant factor. This subject is treated in details in the next chapter. Other factors influencing the performance of the classical and ROM-based parareal methods were identified. As a general conclusion, the ROM-based method showed to behave better when the problem's solution has simpler profiles, that can be easily captured by the model reduction procedure, which depends on factors such that the computational mesh and the nonlinearity of the problem. The classical parareal method showed to be most influenced by other factors, such as the length of parareal time slices and the order of spatial interpolation.

CHAPTER 4

IMPROVEMENTS OF THE ROM-BASED PARAREAL METHOD

Contents

4.1 Introduction . . . 91

4.2 Discussion on the quality of the reduced models . . . 92

4.3 Enrichment of the snapshots sets . . . 95

4.3.1 Proposed modification . . . 95

4.3.2 Speedup estimation . . . 96

4.3.3 Numerical examples . . . 96

4.4 A principal interval decomposition (PID) approach . . . 104

4.4.1 The principal interval decomposition and extension to the POD-EIM . . . 104

4.4.2 A PID-ROM-based parareal method . . . 105

4.4.3 Speedup estimation . . . 105

4.4.4 Numerical examples . . . 107

4.5 An adaptive approach . . . 110

4.5.1 An adaptive ROM-based parareal method . . . 111

4.5.2 Speedup estimation . . . 113

4.5.3 Numerical examples . . . 116

4.6 Combination of the proposed improvements . . . 121

4.7 Conclusion of the chapter . . . 123

4.1 Introduction

In this chapter, we propose a number of modifications to the ROM-based parareal method in order to improve its stability and convergence behaviour.

The numerical tests presented in Section 3.5 show that, even if able to outperform the classical parareal method when applied to the SWE, the ROM-based one still presents instabilities, specially for advanced times of the numerical simulation and when the problem complexity increases. This unstable behaviour can be avoided by considering very small-dimensional ROMs (*i.e.* by considering large thresholds $\varepsilon_{\mathrm{sv,linear}}$ for the POD), but with sacrifice of the parareal convergence. Better behaviours are obtained by reducing both thresholds $\varepsilon_{\mathrm{sv,linear}}$ and $\varepsilon_{\mathrm{sv,nonlinear}}$, but unsatisfactory results are still observed in the end of the temporal domain. These results indicate that the ROMs formulated on-the-fly have not enough quality for providing good results when used in the parareal iterations.

Therefore, we propose here three modifications of the ROM-based parareal method for overcoming these issues. The first one consists in enriching the snapshots sets used as input for the model reduction procedures, with extra snapshots whose computation does not require any additional computational cost, since they are naturally computed along the parareal iterations. This approach has been studied in a submitted paper (Caldas Steinstraesser et al., 2020a). The second modification consists in performing local-in-time model reductions, *i.e.* within time windows in the temporal domain. Known as principal interval decomposition (PID) (Ijzerman, 2000), this approach was formulated in the POD framework, and we extend it to the combined POD-EIM and apply it to the ROM-based parareal method. Finally, the last modification is an adaptive approach, based on the method proposed by Maday and Mula (2020) in the classical parareal framework, and consisting in using progressively refined fine propagators along parareal iterations, instead of a fixed and expensive one. Here, we also extend this approach to the ROM-based method.

For formulating and testing these improvements, we still place ourselves in the coupling between fine and coarse discretizations of the classical SWE. This study will provide insights and guidelines for the application of the methods to the coupling with porosity-based models, presented later in this work. In order to keep a consistent comparison and with different degrees of complexity, the methods proposed here are evaluated using Test cases 1 and 2 presented in Section 3.5, the former being an example of test with stable behaviour and fast convergence but smaller speedup, and the latter a test with larger speedup but slower convergence and more critical stability behaviour. We consider a fixed pair $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}})$ of model reduction thresholds for each case, namely $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-5})$ for Test case 1 and $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-4}, 10^{-4})$ for Test case 2. In a first moment, each proposed modification is tested and studied alone, without the influence of the others, and after that we consider their combined application.

This chapter is organized as follows: in Section 4.2, we propose and illustrate a discussion on the quality of the reduced models formulated in the ROM-based parareal method; the proposed modifications (enrichment of the snapshots sets, PID and adaptive approach) are presented respectively in Sections 4.3, 4.4 and 4.5. Their combined application is tested in Section 4.6 and a conclusion is presented in Section 4.7.

4.2 Discussion on the quality of the reduced models

POD reduced-order models are constructed as the best representation of a given dataset of snapshots: by definition, the POD basis is optimal, in the sense of solving the minimization problem (3.26) (Chaturantabut and Sorensen, 2010). However, if the snapshots themselves are not a good representation of the reference problem, then one can not expect the reduced model to be a good approximation of the reference one. Moreover, even if the POD is the best approximation of a given dataset, it is not necessarily a proper representation of the dynamics of the problem that produced the snapshots, since it may be induced by low-energy features, which are neglected in the POD (Rowley, 2005; Azaïez et al., 2021). Therefore, keeping more POD modes, *i.e.* choosing smaller thresholds ε_{sv} for the truncation (3.28), can degrade the quality of the reduced-order model and lead to non-physical behaviours (Rowley et al., 2004).

Several approaches are proposed in the literature for overcoming these difficulties in the formulation of POD reduced-order models. Some examples are the mean subtraction of snapshots (Müller, 2008) (a common but not consensual (Chen et al., 2012c; El Adawy et al., 2018) procedure in POD); the enrichment of the snapshots, using *e.g.* finite difference quotients (Müller, 2008) or the gradient of the solution (Akkari et al., 2017); greedy selection of snapshots (Chen et al., 2018); greedy basis construction (Haasdonk, Bernard and Ohlberger, Mario, 2008); the so-called balanced POD method (Rowley, 2005) (which requires the formulation and simulation of a dual problem); and the introduction of stabilization terms to the governing equations (Azaïez et al., 2021).

An illustrative example

Let us present a simple example to illustrate these difficulties on the formulation of ROMs. We propose to verify the influence of the quantity and quality of the input snapshots w.r.t. the reference problem and the number of kept POD modes. We consider an one-dimensional test case for the classical SWE, solved with a finite volume discretization in spatial and temporal domains $\Omega = [0, 100]$ and $[0, T]$, respectively, with $T = 20$. Closed boundary conditions (null mass fluxes) are imposed on both boundaries and the initial solution $\boldsymbol{y}_0$ is given by

$$h(x, t=0) = h_0 + \exp\left(-\frac{(x - x_0)^2}{2\sigma_x^2}\right), \qquad hu_x(x, t=0) = 0, \qquad x \in \Omega$$

with $h_0 = 1$, $x_0 = 50$ and $\sigma_x = 7.5$. Note that this test case corresponds to the one-dimensional version of Test case 2, but solved in a larger temporal domain.

We consider as reference solution the one provided by the simulation of the classical SWE (2.35) using a propagator $\mathcal{F}_{\delta t}$ defined by an homogeneous time step $\delta t = 0.005$ and an homogeneous mesh size $\delta x = 2$, resulting on $N_{\delta t} = 4000$ time steps and $M_f = 50$ spatial cells. The formulated reduced model is denoted as $\mathcal{F}_{\delta t,r}$ and solved with the same time step δt. This ROM is formulated using the combined POD and EIM techniques, with the POD applied to a snapshots set Y of the solution, with a threshold $\varepsilon_{\text{sv,linear}}$ for the truncation of the reduced basis, and the POD-EIM applied to a snapshots set $\widehat{Y}$ of the nonlinear term and threshold $\varepsilon_{\text{sv,nonlinear}}$, as described in the previous chapter. By defining the time instants $\tilde{t}_n = n\delta t, n = 0, \dots, T/\delta t = N_{\delta t}$, (the tilde being used for avoiding any confusion with the times $t_n = n\Delta T$ defined in the previous chapter in the parareal framework), the reference and reduced solutions at time $\tilde{t}_n$, are respectively denoted by

$$\boldsymbol{y}_{\text{ref},n} = \mathcal{F}_{\delta t}(\boldsymbol{y}_0, \tilde{t}_n, 0) \in \mathbb{R}^{d_f M_f}, \qquad \boldsymbol{y}_{r,n} = \mathcal{F}_{\delta t,r}(\mathbb{P}\boldsymbol{y}_0, \tilde{t}_n, 0) \in \mathbb{R}^{d_f M_f}, \qquad n = 0, \dots, N_{\delta t} - 1$$

with $d_f = 2$ for the 1D SWE.

In order to compare the quality of the reduced order model under different configurations, we define, for each time $\tilde{t}_n$, the error

$$e_n := \frac{\sum_{i=1}^{d_f M_f} |[\boldsymbol{y}_{r,n}]_i - [\boldsymbol{y}_{\text{ref, n}}]_i|}{\sum_{i=1}^{d_f M_f} |[\boldsymbol{y}_{\text{ref},n}]_i|} \tag{4.1}$$

analogous to the parareal error (3.3).

We begin by illustrating the influence of the number of snapshots on the quality of the ROM. The model reduction is performed using snapshots provided by the fine, reference solution, taken at every $\widehat{\Delta T} = \alpha_{\text{s}} \Delta T$, where $\Delta T = 2$ and α_{s} takes value in $\{1/8, 1/4, 1/2, 1\}$. The "s" in α_{s} stands for "snapshots". We define $N_{\Delta T} := T/\Delta T = 10$. Smaller values of α_{s} mean that more snapshots are used for the model reduction. Therefore, the snapshots sets used for the POD and POD-EIM procedures read respectively

$$\begin{aligned} Y &= \{\boldsymbol{y}_{\text{ref},n}, 0 \leq n \leq N_{\delta t} \mid t_n = l\widehat{\Delta T} = l\alpha_{\text{s}}\Delta T, l \in \mathbb{N}\} \\ \widehat{Y} &= \{\boldsymbol{F}(\boldsymbol{y}_{\text{ref},n}), 0 \leq n \leq N_{\delta t} \mid t_n = l\widehat{\Delta T} = l\alpha_{\text{s}}\Delta T, l \in \mathbb{N}\} \end{aligned} \tag{4.2}$$

where $\boldsymbol{F}$ is the nonlinear term of the governing equations. As in the previous chapter, (4.2) is a generic notation equivalent to the one used in the definition of the ROM-based parareal method (Algorithm 6). It is implicit that, in the model reduction of the SWE, a set of snapshots is defined for each solution component, flux term and source term (see Section 3.5).

For truncating the POD basis, we consider $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} =: \varepsilon_{\text{sv}}$, with $\varepsilon_{\text{sv}} \in \{10^{-\gamma}, \gamma = 1, \dots, 5\}$. Figure 4.1 shows the evolution of e_n for these thresholds and the chosen values for α_{s}. We observe that, for a fixed truncation threshold ε_{sv}, the quality of the ROM increases by reducing α_{s}, *i.e.* by using more snapshots, mainly for smaller values of ε_{sv}. Moreover, the quality of the solution increases when more POD modes are kept (smaller thresholds ε_{sv}). An exception is made for $\alpha_{\text{s}} = 1/4$ and $\varepsilon_{\text{sv}} = 10^{-5}$. For this case, an unstable behaviour is observed, possibly indicating that non-physical features are captured by the POD, as discussed in the beginning of this section.

We now perform a second set of test cases, with the same configurations as above, but with a different procedure for generating the snapshots, closer to the one performed in the ROM-based parareal method. We define a coarse model $\mathcal{G}_{\Delta t}$, using a time step $\Delta t = \Delta T = 2$ and the same mesh size $\Delta x = \delta x = 2$ of $\mathcal{F}_{\delta t}$. The time instants associated to the temporal discretization using ΔT are denoted, as in the parareal method, by $t_n, n = 0, \dots, N_{\Delta T}$. The numerical solution given by $\mathcal{G}_{\Delta t}$ is denoted by

$$\widehat{\boldsymbol{y}}_n = \mathcal{G}_{\Delta t}(\boldsymbol{y}_0, t_n, 0) \in \mathbb{R}^{d_f M_f}, \qquad n = 0, \dots, N_{\Delta T} - 1$$

The snapshots are generated by propagating the coarse solution $\widehat{\boldsymbol{y}}_n$ using the fine model $\mathcal{F}_{\delta t}$ along each time slice $[t_n, t_{n+1}], n = 0, \dots N_{\Delta T} - 1$. That is, we define

$$\Psi_n := \{\mathcal{F}_{\delta t}(\widehat{\boldsymbol{y}}_n, t_n + l\alpha_{\text{s}}\Delta T, t_n), l \in \mathbb{N}^* \mid t_n + l\alpha_{\text{s}}\Delta T \in]t_n, t_{n+1}]\}, \qquad n = 0, \dots, N_{\Delta T} - 1 \tag{4.3}$$

as the set of fine solutions propagated from $\widehat{\boldsymbol{y}}_n$ and defined at times in $]t_n, t_{n+1}]$ separated by $\widehat{\Delta T} = \alpha_{\text{s}}\Delta T$. Then, the snapshots sets are defined by

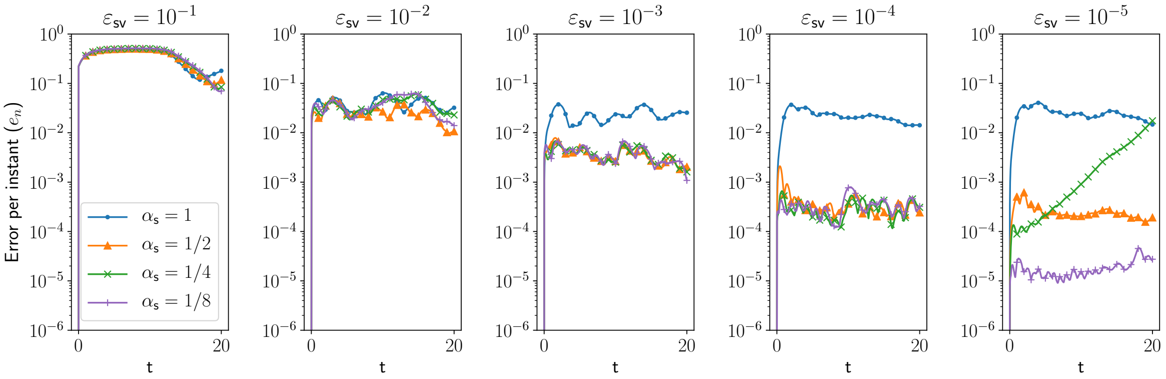


Figure 4.1: Relative error between the reduced and reference models, for various POD truncation thresholds and rates α_s for taking snapshots. The snapshots are obtained from the reference solution.

$$\begin{aligned} Y &= \{\boldsymbol{y}_0\} \cup \left(\bigcup_{n=0}^{N_{\Delta T}-1} \Psi_n \right) \\ \widehat{Y} &= \{\boldsymbol{F}(\boldsymbol{y}_0)\} \cup \left(\bigcup_{n=0}^{N_{\Delta T}-1} \boldsymbol{F}(\Psi_n) \right) \end{aligned} \tag{4.4}$$

where $\boldsymbol{F}(\Psi_n)$ is a set containing the nonlinear term $\boldsymbol{F}$ of the SWE computed on each element in Ψ_n, *i.e.* $\boldsymbol{F}(\Psi_n) := \{\boldsymbol{F}(\boldsymbol{v}), \boldsymbol{v} \in \Psi_n\}$.

Note that, in the case $\alpha_s = 1$, the ROM constructed using the snapshots sets defined by (4.3)-(4.4) coincides with the ROM constructed in the first iteration of the ROM-based parareal method using $\mathcal{G}_{\Delta t}$ and $\mathcal{F}_{\delta t}$ as coarse and fine propagators, respectively (see Algorithm 6), *i.e.* the ROM constructed from the snapshots sets Y^0 and $\widehat{Y}^0$. Indeed, these sets contain the solution of $\mathcal{G}_{\Delta t}$ propagated by $\mathcal{F}_{\delta t}$ along each time slice.

The relative errors for this new set of reduced models are presented in Figure 4.2. As before (Figure 4.1), we notice that larger numbers of kept POD modes (smaller ε_{sv}) increase the quality of the ROM solution, but may lead to instabilities (in this case, instabilities are observed for $\alpha_s = 1/2$ and $\varepsilon_{sv} = 10^{-4}$). However, the improvements in function of ε_{sv} are less important than in the previous case. We also observe an improvement of the ROM by increasing the number of snapshots (smaller α_s), but mainly with large numbers of POD modes (smaller ε_{sv}). Moreover, by comparing the order of magnitude of the errors in Figures 4.1 and 4.2, we conclude that the ROM formulated from more precise snapshots (obtained from the reference solution) is more accurate than the one formulated using coarser solutions.

Therefore, summarizing the results presented above, we can list three factors influencing the quality of the reduced model, which will be used in this chapter for proposing modifications to the ROM-based parareal method:

- Number of snapshots: higher-quality ROMs seems be obtained by increasing the number of input snapshots, *i.e.* if $\widehat{\Delta T} = \alpha_s \Delta T \gg \delta t$, then there may be not enough snapshots for representing the dynamics of the fine, reference model;

- Accuracy of the snapshots: snapshots constructed from more accurate simulations, closer to the reference model, gives a better representation of the fine dynamics and can produce more accurate ROMs;

- Number of kept POD modes: increasing the number of POD basis vector can improve the ROMs quality, but may lead to instabilities and non-physical behaviours, as discussed by Rowley et al. (2004).

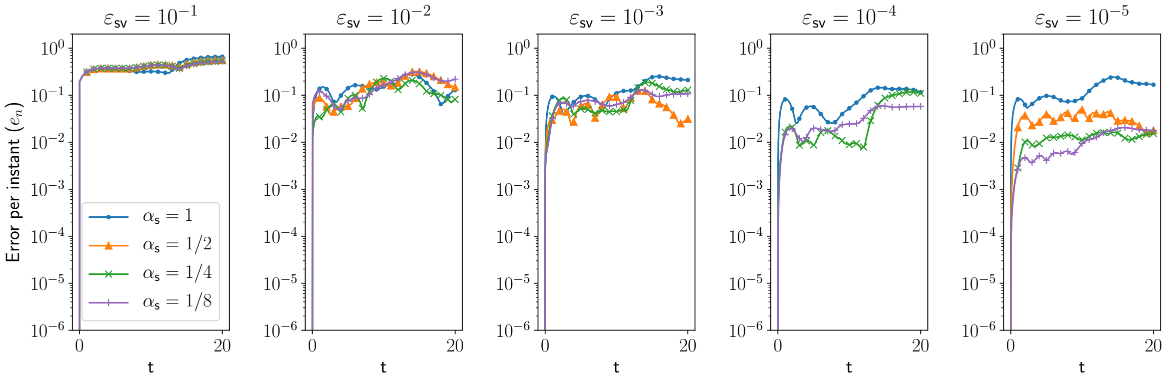


Figure 4.2: Relative error between the reduced and reference models, for various POD truncation thresholds and rates α_s for taking snapshots. The snapshots are obtained from the fine propagation of coarse solutions. For $\varepsilon_{sv} = 10^{-4}$ and $\alpha_s = 1/2$, the solution is unstable, not completing the simulation, and its error is not presented.

4.3 Enrichment of the snapshots sets

4.3.1 Proposed modification

As suggested by the results presented in Section 4.2, the quality of the reduced-order models and the parareal method using them strongly depend on the snapshots used in the model reduction procedures. Therefore, an obvious improvement of the method consists in improving the input snapshots sets. As identified above, it could be done by increasing the number of snapshots and/or constructing them using more accurate simulations. The POD improvement techniques listed above (greedy procedures, balanced POD, etc.) could also be considered. We recall, however, that the ROMs are formulated on-the-fly at each parareal iteration, as discussed in Section 3.4.3. Therefore, the offline stage of the model reduction procedure should not be too expensive. The listed improvements techniques may require an important computational time, and are more adequate to a classical ROM procedure, in which an expensive offline stage is a less important constraint. Furthermore, increasing the quality of the snapshots would also make the parareal method more expensive, since a more accurate coarser model would be needed for producing them.

Therefore, we focus here in improving the ROMs by increasing the number of input snapshots. As detailed in Algorithm 6, the snapshots at each iteration k are the fine correction terms of the parareal solution, *i.e.* $\widetilde{\boldsymbol{y}}_n^j = \mathcal{F}_{\delta t}(\boldsymbol{y}_n^j, t_{n+1}, t_n), j = 0, \dots k, n = 0, \dots, N_{\Delta T}$. Similarly, the snapshots for reducing the nonlinear terms of the equations are obtained by computing the nonlinear function on these quantities. Note that the snapshots are taken exclusively at the parareal time instants $t_n, n = 0, \dots, N_{\Delta T}$ defining the time slices. However, each fine propagation over $[t_n, t_{n+1}]$ computes intermediate solutions on the time steps of the fine discretization comprised between t_n and t_{n+1}. This intermediate information is not used neither in the classical nor in the ROM-based parareal methods.

We then propose to enrich the snapshots sets with a certain number of these intermediate solutions obtained in the fine correction step of the parareal algorithm. Note that this modification does not require any additional computational cost to compute the extra snapshots, since they are already available from the fine prediction step of the algorithm. However, as discussed in Section 4.3.2, this enrichment leads to more expensive model reduction procedures. We remark that an influence of the number of snapshots has been observed by Iizuka and Ono (2018) in an application of the ROM-based parareal method, using POD-only reduced-order models for solving linear problems. Results presented by the authors indicate that convergence is accelerated when sufficiently large numbers of input snapshots are used for the model reduction, but no discussion on this influence is proposed.

By reusing the notation introduced in the numerical examples of Section 4.2, we define a time step $\widehat{\Delta T} = \alpha_s \Delta T$ for taking the extra snapshots, with the enrichment rate $\alpha_s \leq 1$ defined such that $\delta t \leq \widehat{\Delta T} \leq \Delta T$ and $\widehat{\Delta T}$ is an integer multiple of δt (see Figure 4.3).

Analogously to eq. (4.3) and (4.4), we define the sets

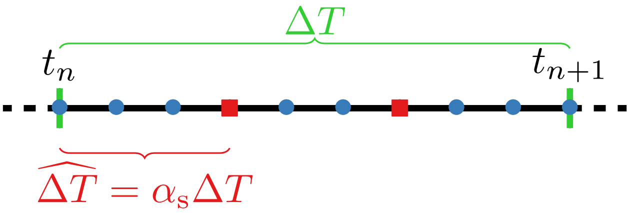


Figure 4.3: Definition of the time step $\widehat{\Delta T}$ for taking the extra snapshots in the enriched ROM-based parareal method (red squares). Zoom on a time slice $[t_n, t_{n+1}]$. Blue dots and red squares indicate time instants of the fine temporal discretization. For simplification, time instants of the coarse temporal discretization are not represented.

$$\Upsilon_n^k := \{\mathcal{F}_{\delta t}(\boldsymbol{y}_n, t_n + l\alpha_s \Delta T, t_n), l \in \mathbb{N}^* \mid t_n + l\alpha_s\Delta T \in \,]t_n, t_{n+1}[\}, \qquad n = 0, \ldots, N_{\Delta T} - 1$$

containing the extra snapshots computed at parareal iteration k and defined in each time slice $[t_n, t_{n+1}]$. Then, the enriched snapshots sets, used respectively for the POD and the POD-EIM procedures, read

$$Y^k := \{\widetilde{\boldsymbol{y}}_n^j,\ j = 0, \ldots, k;\ n = 0, \ldots, N_{\Delta T}\} \bigcup_{n=0}^{N_{\Delta T}-1} \bigcup_{j=0}^{k} \Upsilon_n^k$$
$$\widehat{Y}^k := \{\boldsymbol{F}(\widetilde{\boldsymbol{y}}_n^j),\ j = 0, \ldots, k;\ n = 0, \ldots, N_{\Delta T}\} \bigcup_{n=0}^{N_{\Delta T}-1} \bigcup_{j=0}^{k} \boldsymbol{F}(\Upsilon_n^k)$$

where $\boldsymbol{F}(\Upsilon_n^k) := \{\boldsymbol{F}(\boldsymbol{v}), v \in \Upsilon_n^k\}$.

The modified ROM-based parareal algorithm is presented in Algorithm 7, in which the modifications w.r.t. Algorithm 6 are highlighted.

4.3.2 Speedup estimation

Even if the proposed modification does not imply additional costs for obtaining the extra snapshots, it increases the computational cost $\tau_S(\hat{k})$ for the model reduction (defined in Subsection 3.2.2), since larger input datasets are used. More precisely, the bound (3.47) now reads

$$s_{\text{ROM-enriched-parareal}}(\hat{k}) < \frac{N_{p,s} N_{\delta t} M_f}{\hat{k}\left[N_{\text{spaces}}\left(M_f \hat{k}^2 \alpha_s^2 N_{\Delta T}^2 + \frac{\hat{m}^4}{4} + \hat{m} M_f\right) + 2N_{\text{matrices}} \hat{m}^2 M_f\right]} \tag{4.5}$$

The dependence of bound (4.5) on α_s^2 is a direct consequence of the quadratic dependence of the POD on the number of snapshots. Moreover, the enrichment of the snapshots set can introduce indirect additional costs for the ROM-based parareal method, since the dimension $\hat{m}$ of the ROM, and consequently the computational time $\tau_r(\hat{k})$ for solving it, are likely to be larger, as well as the time $\tau_{\widehat{B}}(\hat{k})$ for computing the ROM matrices. Then, it is evident that one should keep the number of snapshots as small as possible, *i.e.* α_s should not be too small. A natural first choice is $\alpha_s = 1/2$, which means that only one extra snapshot is taken at each time slice. In the next paragraphs, we present some numerical tests for studying the influence of α_s on the quality of the parareal solution and on the computational time.

4.3.3 Numerical examples

We illustrate and study the enriched ROM-based parareal method by considering Test cases 1 (pseudo-2D without spatial coarsening) and 2 (2D) presented in Section 3.6 and whose configurations are presented respectively in Tables 3.1 and 3.3. For each of them, we perform simulations using the ROM-based parareal method with enrichment of the snapshots sets (Algorithm 7) for several values of $\alpha_s \leq 1$. The objective is to study the influence of the number of the snapshots on the quality of the solution and on the computational time. We recall that $\alpha_s = 1$ corresponds to the non-enriched ROM-based parareal method (Algorithm 6).

```
Initialization: initial guess given by the coarse propagator:
$\boldsymbol{y}_0^0 = \boldsymbol{y}_0$
for $n \leftarrow 0$ to $N_{\Delta T} - 1$ do
    $\boldsymbol{y}_{n+1}^0 = \mathcal{G}_{\Delta t}(\boldsymbol{y}_n^0, t_{n+1}, t_n)$
end

$n_0 = 0$
Iterations:
for $k \leftarrow 0$ to $N_{itermax} - 1$ do
    Compute the fine term of the correction (in parallel):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\widetilde{\boldsymbol{y}}_{n+1}^k = \mathcal{F}_{\delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)$
        Store intermediary solutions defined at each $\widehat{\Delta T} = \alpha_{\mathrm{s}}\Delta t,\ \delta t \leq \widehat{\Delta T} \leq \Delta T$ :
          $\Upsilon_n^k := \{\mathcal{F}_{\delta t}(\boldsymbol{y}_n, t_n + l\alpha_{\mathrm{s}}\Delta T, t_n), l \in \mathbb{N}^* \mid t_n + l\alpha_{\mathrm{s}}\Delta T \in ]t_n, t_{n+1}[\}$
    end

    Find the first instant $\tilde{n} \in \{1, \dots, N_{\Delta T}\}$ not satisfying a convergence criterion based on $\boldsymbol{y}_n^k, \widetilde{\boldsymbol{y}}_n^k$
    $n_0 \leftarrow \tilde{n} - 1$

    Define the snapshots sets:
    $Y^k = \{\widetilde{\boldsymbol{y}}_n^j,\ j = 0, \dots, k;\ n = 0, \dots, N_{\Delta T}\} \bigcup_{n=0}^{N_{\Delta T}-1} \bigcup_{j=0}^{k} \Upsilon_n^k$
    $\widehat{Y}^k = \{\boldsymbol{F}(\widetilde{\boldsymbol{y}}_n^j),\ j = 0, \dots, k;\ n = 0, \dots, N_{\Delta T}\} \bigcup_{n=0}^{N_{\Delta T}-1} \bigcup_{j=0}^{k} \boldsymbol{F}\left(\Upsilon_n^k\right)$

    Compute the spaces and define the reduced model $\mathcal{F}_{r,\delta t}^k$:
    $\mathcal{S}^k(Y^k, \varepsilon_{\mathrm{sv,linear}})$ (using POD)
    $\widehat{\mathcal{S}}^k(\widehat{Y}^k, \varepsilon_{\mathrm{sv,nonlinear}})$ (using POD-EIM)

    Compute the coarse term of the correction and the final correction term(in parallel):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\overline{\boldsymbol{y}}_{n+1}^k = \mathcal{F}_{r,\delta t}^k(\mathbb{P}^k \boldsymbol{y}_n^k, t_{n+1}, t_n)$
        $\overline{\overline{\boldsymbol{y}}}_{n+1}^k = \widetilde{\boldsymbol{y}}_{n+1}^k - \overline{\boldsymbol{y}}_{n+1}^k$
    end

    Compute the coarse predictions and correct them to obtain the final solution in the iteration
      (sequentially):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\boldsymbol{y}_{n+1}^{k+1} = \mathcal{F}_{r,\delta t}^k(\mathbb{P}^k \boldsymbol{y}_n^{k+1}, t_{n+1}, t_n) + \overline{\overline{\boldsymbol{y}}}_{n+1}^k$
    end

    if all instants converged ($\tilde{n} = N_{\Delta T}$) then
        break;
    end
end
```

Algorithm 7: ROM-based parareal algorithm with enrichment of the snapshots sets used for the model reduction. Modifications w.r.t. the ROM-based parareal method (Algorithm 6) are highlighted.

Test case 1 (pseudo-2D without spatial coarsening)

As shown in Figure 3.13, the ROM-based parareal method with $\varepsilon_{\mathrm{sv,linear}} = \varepsilon_{\mathrm{sv,nonlinear}} = 10^{-5}$ provides a good convergence behaviour for Test case 1, being able to overcome the instabilities verified in the classical method (Figure 3.10). Thus, we expect a faster convergence by enriching the snapshot sets. We perform simulations with α_{s} taking value in $\{1, 1/2, 1/4, 1/10, 1/200\}$. We remark that $1/200$ is the smallest possible value for α_{s}, since $p_{\delta t} = \Delta T/\delta t = 200$, which means that snapshots are taken at every time step of the fine discretization.

The evolution of the error per time instant and iteration (e_n^k) and the maximum error per iteration ($\bar{e}^k$) are presented respectively in Figures 4.4 and 4.5. By using the simplest enrichment possible ($\alpha_{\mathrm{s}} = 1/2$, *i.e.* one extra snapshot per time slice), an important improvement of the solution is obtained in the first two iterations, mainly in the first half of the temporal domain, for which the average error decreases from 10^{-3} (in the non-enriched case) to 10^{-5}, approximately. We also observe a reduction of the maximum error (which corresponds to the error at $t = T$ in all simulations), of approximately one order of magnitude in the second iteration. Less or no improvements are observed by taking smaller values of α_{s}.

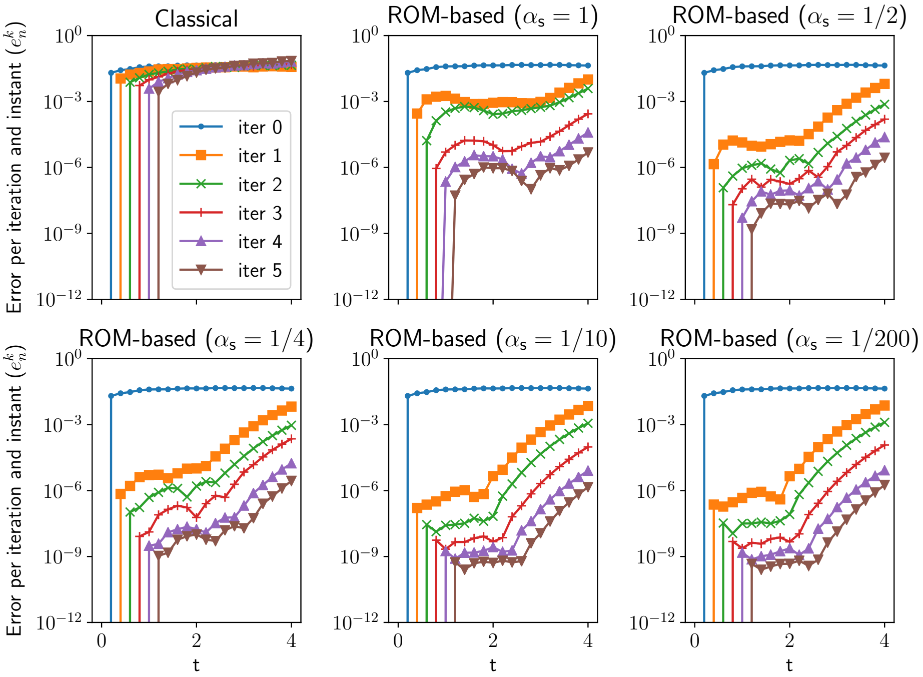


Figure 4.4: Test case 1 (pseudo-2D without spatial coarsening) with $\varepsilon_{\mathrm{sv,linear}} = \varepsilon_{\mathrm{sv,nonlinear}} = 10^{-5}$: relative error e_n^k using the classical parareal method (top left) and the ROM-based one without ($\alpha_{\mathrm{s}} = 1$, top middle) and with enrichment of the snapshots sets for various values of $\alpha_{\mathrm{s}} < 1$ (remaining figures).

As an illustration of the physical solution provided by the parareal methods, Figure 4.6 shows the water depth at the final time of simulation along $y = 10$. The solution of the non-enriched ROM-based parareal method is already very close to the reference one, and even better approximations are obtained with the simplest enrichment ($\alpha_{\mathrm{s}} = 1/2$).

Figure 4.7 shows the computational time and the speedup along iterations for each parareal simulation. Note that the classical parareal method is much cheaper than the ROM-based methods, due to its simpler formulation. Also, the drawback of using too small values of α_{s} is clear, specially for $\alpha_{\mathrm{s}} = 1/10$ and $\alpha_{\mathrm{s}} = 1/200$; in this last case, the parareal method is slower than the reference simulation from the first iteration. It is confirmed in Table 4.1, showing the fraction of the computational time spent in the model reduction procedures. With no enrichment ($\alpha_{\mathrm{s}} = 1$), this fraction remains small in the first iterations, whereas for $\alpha_{\mathrm{s}} \leq 1/10$, the model reduction rapidly corresponds to more than a half of the total iteration time. Note that apparently inconsistent results are observed when comparing the first iteration for $\alpha_{\mathrm{s}} = 1$ and $\alpha_{\mathrm{s}} = 1/2, 1/4$. The reason is that, for $\alpha_{\mathrm{s}} < 1$, a computational structure is created in the first iteration for storing the additional snapshots, thus increasing the total iteration time and leading to a smaller fraction represented by the model reduction itself.

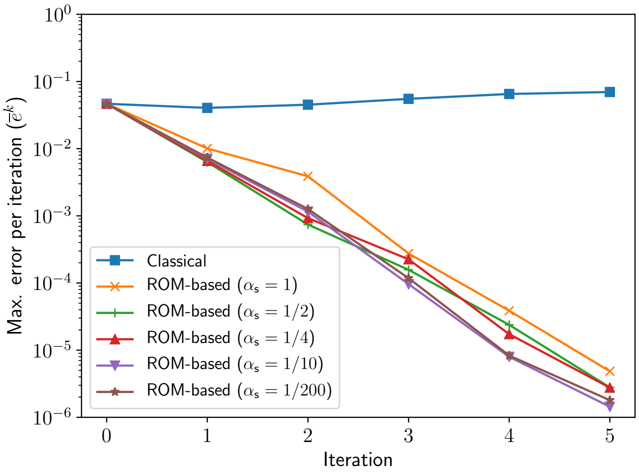


Figure 4.5: Test case 1 (pseudo-2D without spatial coarsening) with $\varepsilon_{\mathrm{sv,linear}} = \varepsilon_{\mathrm{sv,nonlinear}} = 10^{-5}$: maximum error per iteration ($\bar{e}^k$) using the classical parareal method and the ROM-based one without ($\alpha_s = 1$) and with enrichment of the snapshots sets for various values of $\alpha_s < 1$.

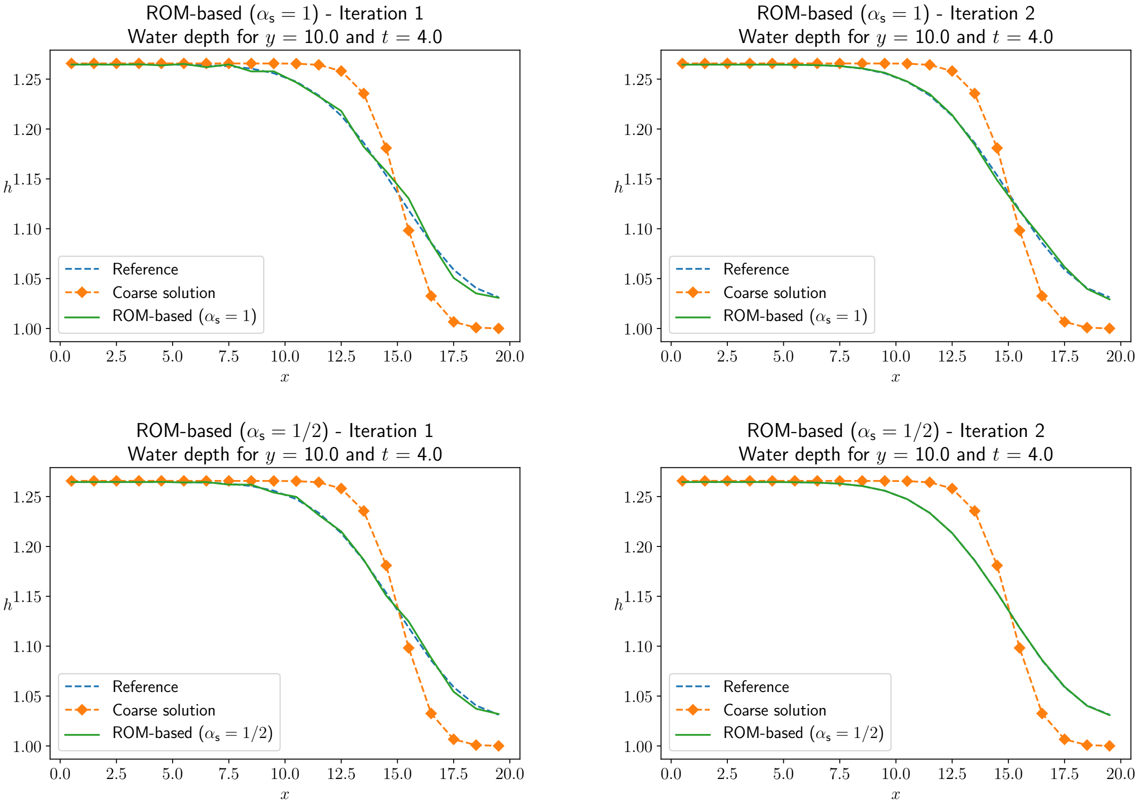


Figure 4.6: Test case 1 (pseudo-2D without spatial coarsening) with $\varepsilon_{\mathrm{sv,linear}} = \varepsilon_{\mathrm{sv,nonlinear}} = 10^{-5}$: final water depth ($t = T = 4$) along $y = 10$. Blue, dashed curves represent the reference solution. Orange, dashed curves represent the coarse solution (0-th parareal iteration). First and second rows: ROM-based method with $\alpha_s = 1$ and $\alpha_s = 1/2$, respectively. First and second columns: first and second parareal iterations, respectively. Iterations are plotted separately for a better visualization.

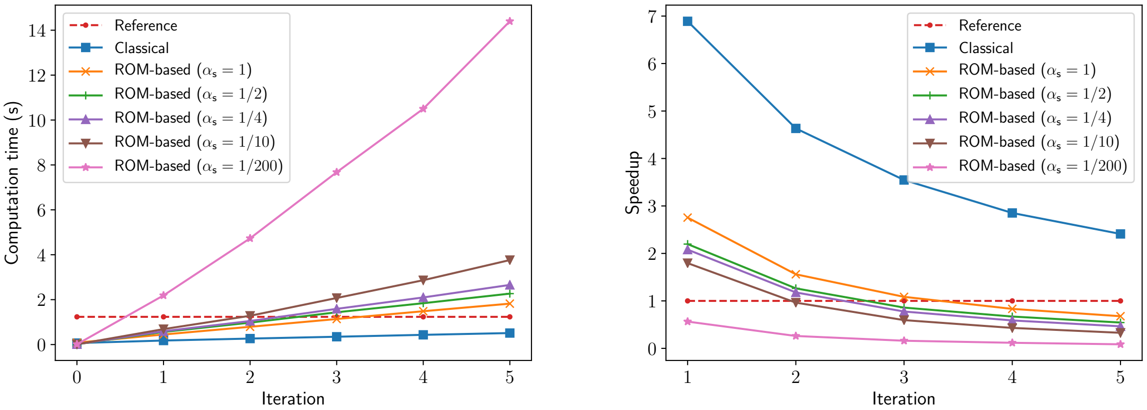


Figure 4.7: Test case 1 (pseudo-2D without spatial coarsening) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-5}$: computational time $T_{\text{par}}(k)$ (left) and speedup $s(k)$ (right) along iterations using the classical parareal method and the ROM-based one without ($\alpha_s = 1$) and with enrichment of the snapshots sets for various values of $\alpha_s < 1$. The horizontal, dashed lines correspond to the computational time of the reference simulation (left) and a unitary speedup (right).

By joining the observations on the convergence and speedup behaviours of the performed simulations, we can say that the best compromise is verified for the ROM-based method without enrichment ($\alpha_s = 1$), if one looks to the maximum error per iteration, or with few extra snapshots per time slice ($\alpha_s = 1/2$ and $\alpha_s = 1/4$), by considering the error evolution along the entire temporal domain. In these three cases, the parareal simulation is slower than the reference simulation after few iterations (four for $\alpha_s = 1$ and three for $\alpha_s = 1/2$ and $\alpha_s = 1/4$) , but, as shown in Figures 4.4, 4.5 and 4.6, a high-quality approximation is obtained in the first iteration, corresponding to a speedup of approximately 3 for the non-enriched case and 2 for the enriched ones. These speedups are considerably smaller than the one obtained by the classical parareal method, which, however, is not able to improve the quality of the solution along iterations.

The results presented and discussed above are obtained for a fixed pair of model reduction thresholds, namely $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-5}$. We remark that, in this relatively simple and one-dimensional problem, in which the model reduction is able to well represent the reference solution, an even faster convergence can be obtained by increasing the ROM dimension, *i.e.* by keeping a larger number of POD basis functions. Figure 4.8 illustrates it, presenting the errors obtained by the ROM-based parareal method with $\alpha_s \in \{1, 1/2, 1/4\}$, for $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 0$ (*i.e.* without truncation of the reduced bases). We observe that for $\alpha_s = 1/2$, exact convergence (up to the chosen parareal convergence threshold $\varepsilon_{\text{TOL}} = 10^{-10}$) is obtained after one parareal iteration in the first half of the temporal domain; for $\alpha_s = 1/4$, the first iteration provides almost exact convergence in the entire temporal domain. However, this result cannot be generalized to more complex problems, *e.g.* Test case 2 and even Test case 1 solved in larger temporal and spatial domains (as discussed in chapter 5), for which instabilities are observed when relatively high-dimensional ROMs are considered.

	Iteration k				
α_s	1	2	3	4	5
1	6.5%	4.1%	6.3%	7.9%	10.8%
1/2	3.4%	8.1%	12.6%	14.7%	22.2%
1/4	5.9%	15.1%	27.6%	31.0%	39.6%
1/10	15.6%	35.7%	52.8%	54.9%	70.0%
1/200	67.4%	76.4%	78.5%	82.5%	80.1%

Table 4.1: Test case 1 (pseudo-2D without spatial coarsening): fraction of the computational time spent on the model reduction procedures w.r.t. the total simulation time in the ROM-based parareal method, for each iteration and various values of α_s.

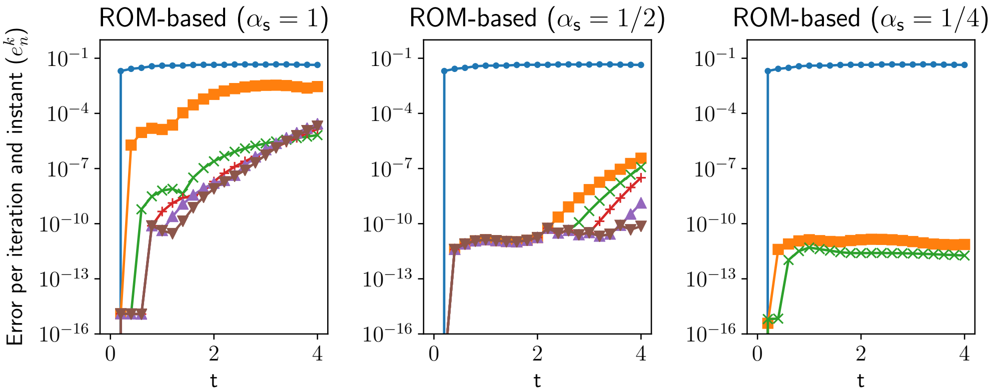


Figure 4.8: Test case 1 (pseudo-2D without spatial coarsening) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 0$: relative error e_n^k using the ROM-based parareal method without ($\alpha_s = 1$, left) and with enrichment of the snapshots sets, for $\alpha_s = 1/2$ (middle) and $\alpha_s = 1/4$ (right). The legend is omitted for the sake of clarity and is the same as in Figure 4.4. Exact convergence is obtained in the case $\alpha_s = 1/4$ after two iterations.

Test case 2 (2D)

As a last illustration of the ROM-based parareal method with enrichment of the snapshots sets, we consider the two-dimensional test case presented in the previous chapter, with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$. As shown in Figure 3.19, the ROM-based parareal method is unstable under these configurations, whereas the classical method (Figure 3.12) presents a more stable behaviour. We run the enriched method with α_s taking value in $\{1, 1/2, 1/5, 1/10, 1/250\}$, with the last one being the smallest possible value for α_s, since $p_{\delta t} = \Delta T/\delta t = 250$.

The errors e_n^k and $\bar{e}^k$ for each parareal simulation are presented respectively in Figure 4.9 and 4.10. We observe that the enrichment of the snapshots sets is able to stabilize the parareal solution, such that the increasing error within iterations is no longer observed for $\alpha_s \leq 1/2$, and mainly for $\alpha_s \leq 1/5$. For $\alpha_s = 1/2$, a less important unstable behaviour is still present in the very last time slices in the third iteration, but it is further controlled. No improvement is observed for $\alpha_s < 1/5$, with a slightly lower performance for $\alpha_s \geq 1/10$, possibly indicating that the additional information captured by the ROM does not improve its quality. We also notice that all ROM-based simulations, including for $\alpha_s = 1$, provide much smaller errors, compared to the classical method, in the first half of the simulation. Since the initial water depth is a Gaussian curve, the solution in the beginning of the simulation is relatively smooth and is well captured by the ROMs. Larger errors are observed in the end of the simulation (when the solution presents a more complex profile) and are relatively close to the errors of the classical method; this is evident in Figure 4.10, since the maximum error per iteration corresponds to the error at the final instant $t = T$ in all simulations.

The physical results are illustrated in Figure 4.11 with the final water depth along iterations for the non-enriched ROM-based method and the one with $\alpha_s = 1/5$. In the former case, a slightly better approximation is obtained in the first iteration, but instabilities arise in the following ones, confirming the error behavior presented in Figures 4.9. These instabilities are not present in the enriched simulation, with a very close approximation to the reference solution.

The speedup for each simulation along iterations is presented in Figure 4.12. We first notice that the speedups are globally larger than in Test case 1 (Figure 4.7), which, accordingly to bounds (3.42) and (3.41), is due to the more expensive fine propagator used in this two-dimensional test case. As before, the classical parareal method is much cheaper than the ROM-based one, and the latter only provides limited speedups. A good but still relatively rough approximation of the reference solution is provided after one iteration of the non-enriched method, with a speedup close to 5. More accurate solutions require more iterations and the use of $\alpha_s < 1$. For example, three iterations with $\alpha_s = 1/5$ (whose solution is illustrated in Figure 4.11) provide a speedup no larger than 2. Since the classical parareal method is stable in this test case, it is more interesting than the ROM-based one: for example, its maximum error $\bar{e}^5$ at the fifth iteration is close to the error $\bar{e}^3$ at the third iteration of the ROM-based method with $\alpha_s = 1/5$ (see Figure 4.10), but with a speedup factor of approximately 4, *i.e.* nearly two times faster.

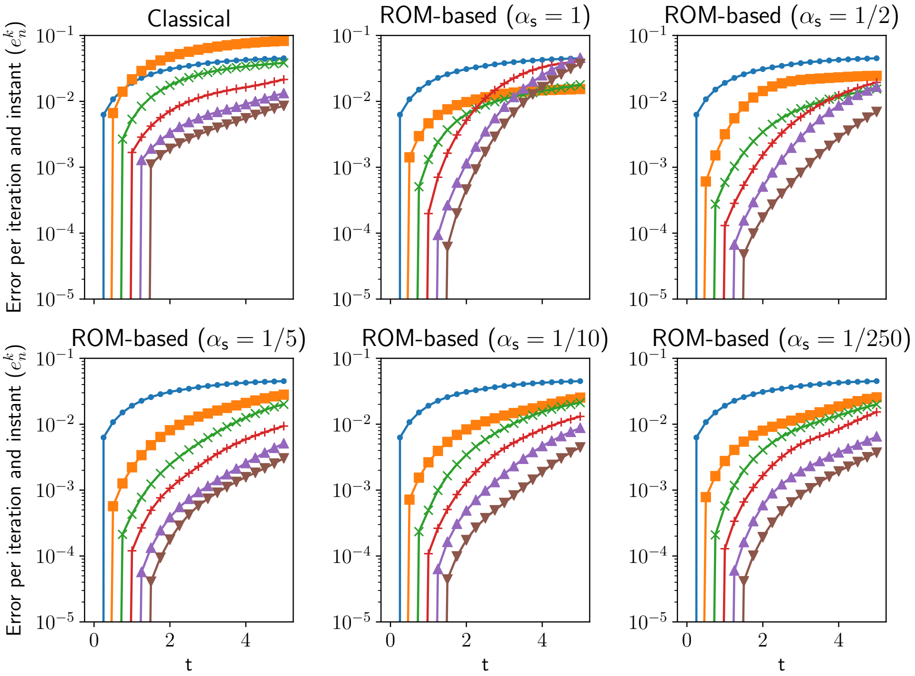


Figure 4.9: Test case 2 (2D) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$: relative error e_n^k using the classical parareal method (top left) and the ROM-based one without ($\alpha_s = 1$, top middle) and with enrichment of the snapshots sets for various values of $\alpha_s < 1$ (remaining figures). The legend is omitted for the sake of clarity and is the same as in Figure 4.4.

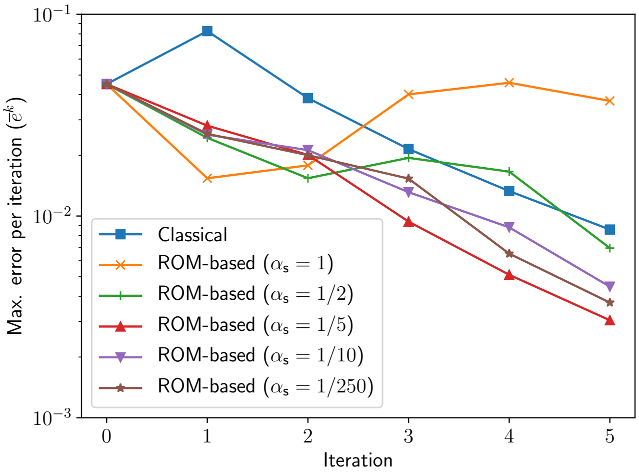


Figure 4.10: Test case 2 (2D) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$: maximum error per iteration ($\bar{e}^k$) using the classical parareal method and the ROM-based one without ($\alpha_s = 1$) and with enrichment of the snapshots sets for various values of $\alpha_s < 1$.

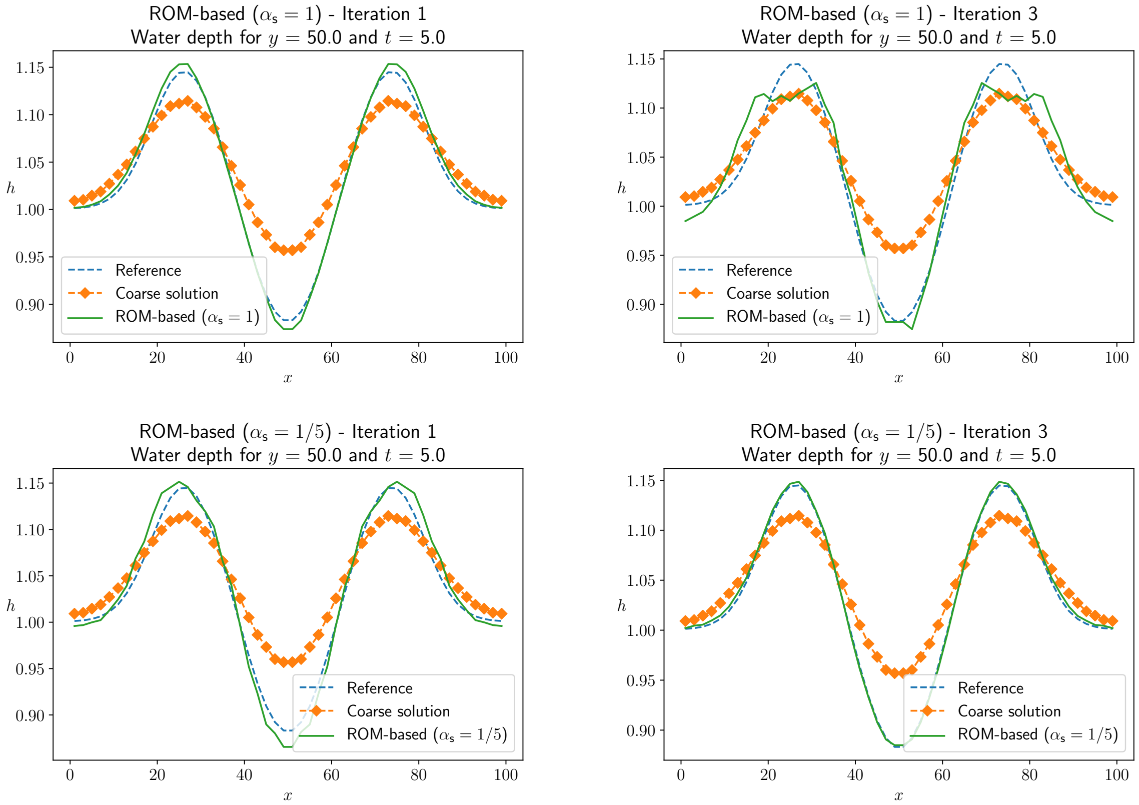


Figure 4.11: Test case 2 (2D) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$: final water depth ($t = T = 5$) along $y = 50$. Blue, dashed curves represent the reference solution. Orange, dashed curves represent the coarse solution (0-th parareal iteration). First and second rows: ROM-based method with $\alpha_s = 1$ and $\alpha_s = 1/5$, respectively. First and second columns: first and third parareal iterations, respectively. Iterations are plotted separately for a better visualization.

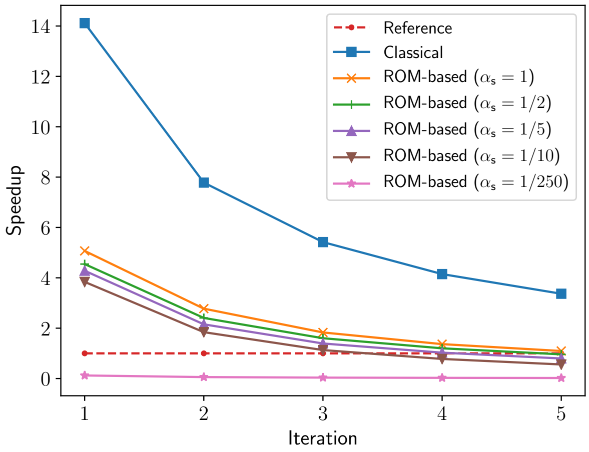


Figure 4.12: Test case 2 (2D) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$: speedup $s(k)$ along iterations using the classical parareal method and the ROM-based one without ($\alpha_s = 1$) and with enrichment of the snapshots sets for various values of $\alpha_s < 1$.

4.4 A principal interval decomposition (PID) approach

In this section, we propose the introduction of the method known as Principal Interval Decomposition (PID) in the ROM-based parareal method. Whereas the enrichment approach proposed in the previous section aims to construct input snapshots sets more representative of the fine dynamics problem, this PID approach focuses in constructing ROMs more capable of capturing local phenomena in space and time.

As pointed out by Borggaard et al. (2016), good ROM representations of smooth flows can be obtained using very few POD modes; however, this model-order reduction technique can be ineffective for approximating highly non-stationary, nonlinear problems, since the POD modes are a global representation of the solution and are associated to an energy distributed over the entire domain. Thus, local in time and space features of the flow can be misrepresented by the POD ROM. We remark that these difficulties may arise in the applications envisaged in this thesis (simulations of urban floods), specially when complex urban geometries are considered. Small scale phenomena such as discontinuities of the flow velocity due to the reflection on obstacles may constitute a challenge for the model reduction.

4.4.1 The principal interval decomposition and extension to the POD-EIM

An alternative for obtaining better representation of these local phenomena is proposed by Ijzerman (2000) under the name of principal interval decomposition (PID). It consists in subdividing the temporal domain in a given number of non-overlapping time windows, and a POD ROM is formulated in each one of them, using local-in-time snapshots. In its original formulation, a single POD basis function (the dominant mode) is defined per time interval and the lengths of the intervals are chosen adaptively for attaining a prescribed accuracy. Extensions allowing to use more POD modes per interval and using modes from neighbours intervals are proposed by Borggaard et al. (2007). A simpler and more computationally efficient approach, using a predetermined number of homogeneous time windows, is explored by San and Borggaard (2015). This last approach is also implemented by Zokagoa and Soulaimani (2018) for solving the nonlinear SWE, with the nonlinearities being approximated using statistical mean values.

We describe the principal interval decomposition following its presentation by San and Borggaard (2015), and we extend its main idea to the POD-EIM framework for treating nonlinear problems. We call this extension hereafter as *PID-EIM ROM*. We divide the temporal domain $[0, T]$ into N_{PID} non-overlapping windows $\mathbb{T}^{(i)} := [\tilde{t}_{i-1}, \tilde{t}_i]$, $i = 1, \dots, N_{\text{PID}}$, to which we refer as *PID windows*, with $\tilde{t}_0 = 0$ and $\tilde{t}_{N_{\text{PID}}} = T$. For simplicity and already aiming the application of PID-EIM to the parareal method, we suppose that the extremities of the windows are parareal time instants, associated to the time step ΔT, *i.e.*

$$\{\tilde{t}_i, i = 0 \dots, N_{\text{PID}}\} \subset \{t_i, i = 0, \dots, N_{\Delta T}\} \tag{4.6}$$

Figure 4.13 illustrates the definition of the PID windows.

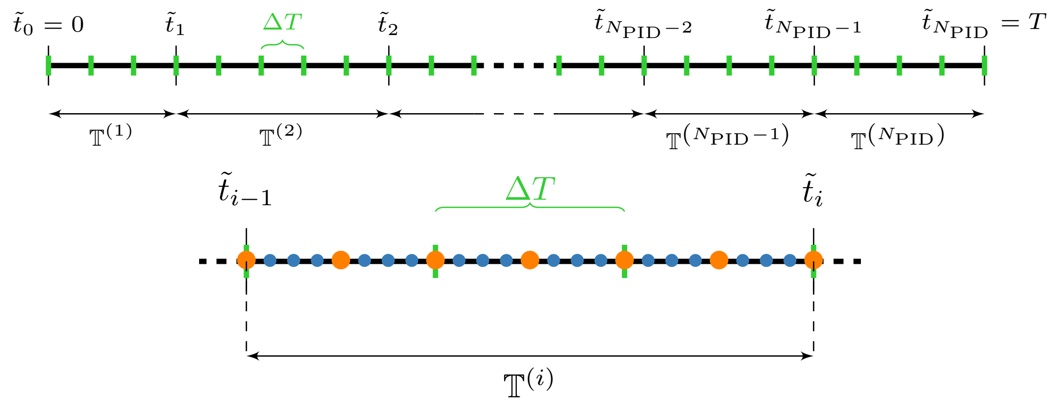

$\tilde{t}_0 = 0$
$\tilde{t}_1$
ΔT
$\tilde{t}_2$
$\tilde{t}_{N_{\text{PID}}-2}$
$\tilde{t}_{N_{\text{PID}}-1}$
$\tilde{t}_{N_{\text{PID}}} = T$
$\mathbb{T}^{(1)}$
$\mathbb{T}^{(2)}$
$\mathbb{T}^{(N_{\text{PID}}-1)}$
$\mathbb{T}^{(N_{\text{PID}})}$
$\tilde{t}_{i-1}$
ΔT
$\tilde{t}_i$
$\mathbb{T}^{(i)}$

Figure 4.13: Definition of the PID time windows. Top: representation of several windows, with only the parareal time instants (green, vertical ticks, defining the parareal time slices) represented. Bottom: zoom over a time window containing three time slices, also showing the fine (blue and orange bullets) and coarse (large, orange bullets) temporal discretizations.

Let $Y, \widehat{Y} \in \mathbb{R}^{M_f \times n_s}$ be the snapshots matrices (respectively of the solution and of the nonlinear term of the problem) containing in their columns n_s snapshots defined along the entire temporal domain, at times $\hat{t}_i \in [0,T], i = 1, \dots, n_s$. For each time interval $\mathbb{T}^{(i)}$ we define the snapshot submatrices $Y^{(i)}, \widehat{Y}^{(i)} \in \mathbb{R}^{M_f \times n_s^{(i)}}$ defined by keeping the $n_s^{(i)}$ columns respectively of Y and $\widehat{Y}$ that correspond to times $\hat{t}_i \in \mathbb{T}^{(i)}$. The POD procedure is then applied for each $Y^{(i)}$ for obtaining the reduced subspaces $\mathcal{S}^{(i)}$ spanned by $V_{q_i}^{(i)} := [\boldsymbol{\phi}_1^{(i)}, \dots, \boldsymbol{\phi}_{q_i}^{(i)}] \in \mathbb{R}^{M_f \times q_i}$ with dimension $q_i \ll M_f$. Similarly, the POD-EIM applied to $\widehat{Y}^{(i)}$ produces a subspace $\widehat{\mathcal{S}}^{(i)}$ spanned by $\widehat{V}_{m_i}^{(i)} := [\widehat{\boldsymbol{\phi}}_1^{(i)}, \dots, \widehat{\boldsymbol{\phi}}_{m_i}^{(i)}] \in \mathbb{R}^{M_f \times m_i}$ with dimension $m_i \ll M_f$, and a matrix $\mathcal{P}^{(i)} \in \mathbb{R}^{M_f \times m_i}$ whose columns are the canonical vectors of $\mathbb{R}^{M_f}$ corresponding to the chosen spatial interpolation points. Then, analogously to (3.36), the local ROMs for (3.25), defined in each window $\mathbb{T}^{(i)}$, read

$$\frac{d}{dt}\widetilde{\boldsymbol{y}}(t) = (V_{q_i}^{(i)})^T A V_{q_i}^{(i)} \widetilde{\boldsymbol{y}}(t) + (V_{q_i}^{(i)})^T \widehat{V}_{m_i}^{(i)} \left((\widehat{P}^{(i)})^T \widehat{V}_{m_i}^{(i)} \right)^{-1} (\widehat{P}^{(i)})^T \boldsymbol{F}(V_{q_i}^{(i)} \widetilde{\boldsymbol{y}}(t)), \qquad i = 1, \dots, N_{\mathrm{LTP}} \tag{4.7}$$

We denote by $\mathcal{F}_{r,\delta t}^{(i)}$ the reduced propagator defined by (4.7). The propagation of a vector $\boldsymbol{y} \in \mathbb{R}^{M_f}$ by $\mathcal{F}_{r,\delta t}^{(i)}$ between two times t and s, with $s > t$, is defined if $[t,s] \subset \mathbb{T}^{(i)}$ and the propagated solution is denoted by

$$\mathcal{F}_{r,\delta t}^{(i)}(\mathbb{P}^{(i)}\boldsymbol{y}, s, t) \tag{4.8}$$

where $\mathbb{P}^{(i)}$ is a projection operator onto the subspace $\mathcal{S}^{(i)}$ defined in $\mathbb{T}^{(i)}$. As in (3.37), it is implicit, in notation (4.8), that the propagation consists in projecting $\boldsymbol{y}$ onto $\mathcal{S}^{(i)}$ for obtaining $\widetilde{\boldsymbol{y}}$, propagating $\widetilde{\boldsymbol{y}}$ using (4.7) and then coming back to the physical space.

Therefore, the PID-EIM defined in the entire temporal domain $[0,T]$ reads

$$\mathcal{F}_{r,\delta t}(\boldsymbol{y}, s, t) = \sum_{i=0}^{N_{\mathrm{PID}}} \mathbb{1}_{\{[t,s] \subset \mathbb{T}^{(i)}\}} \mathcal{F}_{r,\delta t}^{(i)}(\boldsymbol{y}, s, t) \tag{4.9}$$

if $[t,s]$ overlaps only one PID window. If it is not the case, then the propagation along each subinterval of $[t,s]$ is performed using the reduced propagator associated to the time window containing the subinterval. Special care must be taken at the intersecting times between the windows. If $\widetilde{\boldsymbol{y}}_i$ denotes the reduced solution defined at $\tilde{t}_i = \mathbb{T}^{(i)} \bigcup \mathbb{T}^{(i+1)}$ and provided by a propagation in $\mathbb{T}^{(i)}$ using $\mathcal{F}_{r,\delta t}^{(i)}$, then the initial solution for the propagation in $\mathbb{T}^{(i+1)}$ using $\mathcal{F}_{r,\delta t}^{(i+1)}$ reads $(V^{(i+1)})^T V^{(i)} \widetilde{\boldsymbol{y}}_i$ (which corresponds to a projection from $\mathcal{S}^{(i)}$ onto the physical space, followed by a projection onto $\mathcal{S}^{(i+1)}$).

4.4.2 A PID-ROM-based parareal method

We investigate here the use of the PID-EIM approach in the parareal method. At each iteration, we define, for each $i = 1, \dots, N_{\mathrm{PID}}$, the snapshot sets $Y^{(i),k}$ and $\widehat{Y}^{(i),k}$ as submatrices of Y^k and $\widehat{Y}^k$, respectively, which are used for computing the local reduced spaces $\mathcal{S}^{(i),k}$ and $\widehat{\mathcal{S}}^{(i),k}$ and defining the local ROM $\mathcal{F}_{r,\delta t}^{(i),k}$. Under assumption (4.6), the time slices $[t_n, t_{n+1}], n = 0, \dots, N_{\Delta T}$, intersect only one PID time window each and the ROM at iteration k and in $[0,T]$ is defined, following (4.9), as

$$\mathcal{F}_{r,\delta t}^{k}(\boldsymbol{y}, t_{n+1}, t_n) = \sum_{i=0}^{N_{\mathrm{PID}}} \mathbb{1}_{\{[t_{n+1}, t_n] \subset \mathbb{T}^{(i)}\}} \mathcal{F}_{r,\delta t}^{(i),k}(\boldsymbol{y}, t_{n+1}, t_n)$$

The proposed method, to which refer as *PID-ROM parareal method*, is presented in details in Algorithm 8, in which the modifications w.r.t. Algorithm 6 are highlighted.

4.4.3 Speedup estimation

As a major feature, the formulation of the subspaces $\mathcal{S}^{(i)}$ in the PID model reduction procedure has a smaller computational cost compared to the full POD applied over the entire temporal domain $[0,T]$, due to the superlinear cost of the SVD w.r.t. the number of snapshots, as discussed in Section 3.2.2. Indeed, in the case of the *MKL-LAPACK* function *dgesvd*, considered in this work and whose application to a matrix in $\mathbb{R}^{q \times n}$ has a complexity $\mathcal{O}(qn^2 + n^3)$ (see Appendix B.3), the cost of the full POD applied to a

```
Initialization: initial guess given by the coarse propagator:
$\boldsymbol{y}_0^0 = \boldsymbol{y}_0$
for $n \leftarrow 0$ to $N_{\Delta T} - 1$ do
    $\boldsymbol{y}_{n+1}^0 = \mathcal{G}_{\Delta t}(\boldsymbol{y}_n^0, t_{n+1}, t_n)$
end

$n_0 = 0$
Iterations:
for $k \leftarrow 0$ to $N_{itermax} - 1$ do
    Compute the fine term of the correction (in parallel):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\widetilde{\boldsymbol{y}}_{n+1}^k = \mathcal{F}_{\delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)$
    end

    Find the first instant $\tilde{n} \in \{1, \ldots, N_{\Delta T}\}$ not satisfying a convergence criterion based on $\boldsymbol{y}_n^k, \widetilde{\boldsymbol{y}}_n^k$
    $n_0 \leftarrow \tilde{n} - 1$

    for $i \leftarrow 1$ to $N_{PID}$ do
        Define the snapshots sets:
        $Y^{(i),k} = \{\widetilde{\boldsymbol{y}}_n^j,\ j = 0, \ldots, k;\ n \mid t_n \in \mathbb{T}^{(i)}\}$
        $\widehat{Y}^{(i),k} = \{\boldsymbol{F}(\widetilde{\boldsymbol{y}}_n^j),\ j = 0, \ldots, k;\ n \mid t_n \in \mathbb{T}^{(i)}\}$

        Compute the local spaces and define the local reduced model $\mathcal{F}_{r,\delta t}^{(i),k}$:
        $\mathcal{S}^{(i),k}(Y^{(i),k}, \varepsilon_{\text{sv,linear}})$ (using POD)
        $\widehat{\mathcal{S}}^{(i),k}(\widehat{Y}^{(i),k}, \varepsilon_{\text{sv,nonlinear}})$ (using POD-EIM)

    end
    Compute the coarse term of the correction and the final correction term(in parallel):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\overline{\boldsymbol{y}}_{n+1}^k = \mathcal{F}_{r,\delta t}^k(\mathbb{P}^k \boldsymbol{y}_n^k, t_{n+1}, t_n)$
        $\overline{\overline{\boldsymbol{y}}}_{n+1}^k = \widetilde{\boldsymbol{y}}_{n+1}^k - \overline{\boldsymbol{y}}_{n+1}^k$
    end

    Compute the coarse predictions and correct them to obtain the final solution in the iteration
      (sequentially):
    for $n \leftarrow n_0$ to $N_{\Delta T} - 1$ do
        $\boldsymbol{y}_{n+1}^{k+1} = \mathcal{F}_{r,\delta t}^k(\mathbb{P}^k \boldsymbol{y}_n^{k+1}, t_{n+1}, t_n) + \overline{\overline{\boldsymbol{y}}}_{n+1}^k$
    end

    if all instants converged ($\tilde{n} = N_{\Delta T}$) then
        break;
    end
end
```

Algorithm 8: PID-ROM-based parareal algorithm. Modifications w.r.t. the ROM-based parareal method (Algorithm 6) are highlighted.

snapshot matrix in $\mathbb{R}^{M_f \times n_s}$ is proportional to $M_f n_s^2 + n_s^3$. By supposing that all N_{PID} time windows have the same length, then the PID applied to N_{PID} matrices in $\mathbb{R}^{M_f \times (n_s/N_{\text{PID}})}$ has a complexity proportional to $N_{\text{PID}}\left[M_f(n_s/N_{\text{PID}})^2 + (n_s/N_{\text{PID}})^3\right] = M_f n_s^2/N_{\text{PID}} + n_s^3/N_{\text{PID}}^2 < M_f n_s^2 + n_s^3$.

Therefore, in the PID-EIM approach, the computational time $\tau_{\mathcal{S}}(\hat{k})$ for computing N_{PID} subspaces using the POD-EIM approach can be estimated, under the same hypotheses considered for deriving (3.45), by

$$\begin{aligned} \tau_{\mathcal{S}}(\hat{k}) &= N_{\text{PID}}\mathcal{O}\left(M_f\hat{k}^2\left(\frac{N_{\Delta T}+1}{N_{\text{PID}}}\right)^2 + \hat{k}^3(N_{\Delta T}+1)^3 + \sum_{i=2}^{\hat{m}} i^3 + \hat{m}M_f\right) \\ &= \mathcal{O}\left(\frac{M_f\hat{k}^2 N_{\Delta T}^2}{N_{\text{PID}}} + N_{\text{PID}}\left(\frac{\hat{m}^4}{4} + \hat{m}M_f\right)\right) \end{aligned} \tag{4.10}$$

where no assumption is made on the dimension $\hat{m}$ of the subspaces in function of the number and size of the N_{PID} windows. Therefore, the computation of the subspaces in the PID-EIM is likely to be less expensive due to the POD step (the first term in (4.10)), but more expensive due to the DEIM step (the second term in (4.10)).

Moreover, in each PID time window and each parareal iteration, a full set of N_{matrices} needs to be computed, thus totalizing $N_{\text{PID}}N_{\text{matrices}}$ and leading to a larger computational cost. Therefore, bound (3.47), estimated in Section 3.4.4 in the ROM-based parareal framework, is rewritten under the PID-EIM approach as

$$s_{\text{ROM-PID-parareal}}(\hat{k}) < \frac{N_{p,s}N_{\delta t}M_f}{\hat{k}\left[N_{\text{spaces}}\left(\frac{M_f\hat{k}^2 N_{\Delta T}^2}{N_{\text{PID}}} + N_{\text{PID}}\left(\frac{\hat{m}^4}{4} + \hat{m}M_f\right)\right) + 2N_{\text{PID}}N_{\text{matrices}}\hat{m}^2 M_f\right]} \tag{4.11}$$

The influence of the formulation of several subspaces and ROM matrices on the computational time of the PID-ROM parareal is assessed in the following paragraphs using some numerical examples.

4.4.4 Numerical examples

We illustrate the PID-ROM-based parareal method by considering the PID-approach as the only improvement for the model reduction (*i.e.* we consider $\alpha_s = 1$ for not enriching the input snapshot sets). For each test case, we perform simulations with various numbers of time windows.

Test case 1 (pseudo-2D without spatial coarsening)

For Test case 1, we perform simulations with N_{PID} taking value in $\{1, 2, 4, 10, 20\}$, the first one of the these values being equivalent to the ROM-based method and the last one being the largest value possible for N_{PID}, since each PID window has the length of one time slice ΔT. For each value of N_{PID}, all time windows have the same length.

The evolution of the errors e_n^k and $\bar{e}^k$ is presented respectively in Figures 4.14 and 4.15. It is clear that the PID approach, with $N_{\text{PID}} > 1$, degrades the performance of the ROM-based parareal method. Indeed, the method with $N_{\text{PID}} = 1$ behaves well in this test case because the model reduction is able to properly represent the dynamics of the global, fine solution, which consists of a relatively simple profile. By formulating local-in-time ROMs to be used for updating the global solution at each parareal iteration, this good global representation is no longer observed. Indeed, we observe in Figure 4.14 that the first time window in all simulations presents a similar behaviour, with larger errors near the end of the window (which is also observed for $N_{\text{PID}} = 1$, within the entire temporal domain). Therefore, less accurate solutions are used as initial conditions for the model reduction in the next window, such that different error behaviours are observed along the temporal domain.

Test case 2 (2D)

We now consider Test case 2 with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$, which has an unstable behaviour in the ROM-based parareal method, as illustrated in Figure 3.19. As for Test case 1, we consider $N_{\text{PID}} \in \{1, 2, 4, 10, 20\}$, with homogeneous time windows in each case. After a deterioration of the quality of the solution (compared to $N_{\text{PID}} = 1$) in the first iteration, the instabilities in the end of the

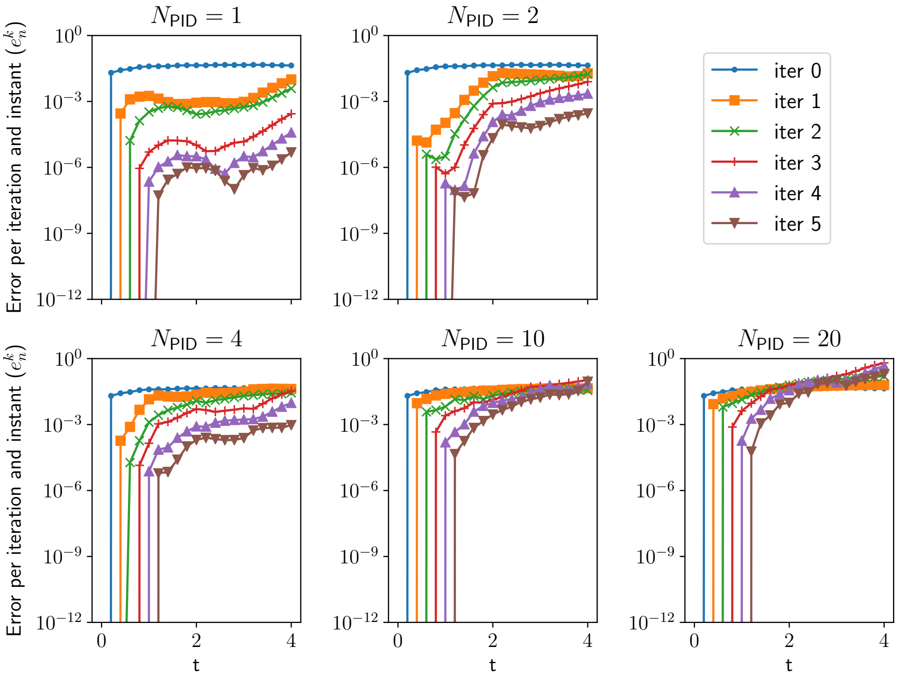


Figure 4.14: Test case 1 (pseudo-2D without spatial coarsening) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-5}$: relative error e_n^k using the PID-ROM-based parareal method for various numbers of PID windows.

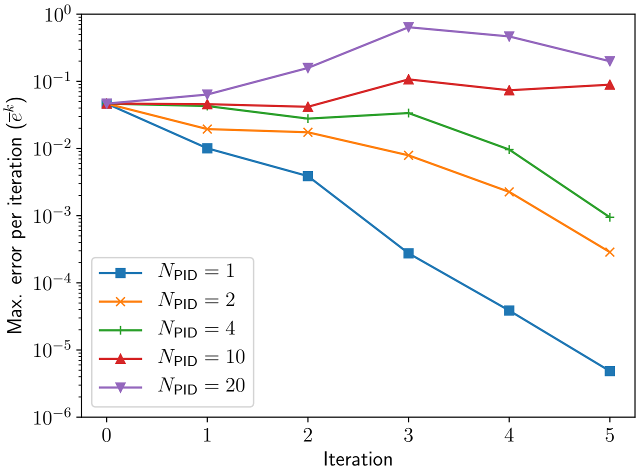


Figure 4.15: Test case 1 (pseudo-2D without spatial coarsening) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-5}$: maximum error per iteration $(\bar{e}^k)$ using the PID-ROM-based parareal method for various numbers of PID windows.

temporal domain are less important, and smaller errors are obtained, as presented in Figure 4.16. The solution of Test case 2 presents stronger variations when compared to Test case 1, being more challenging for the model reduction, and some benefits are observed by formulation local-in-time ROMs. However, as shown in Figure 4.17, the error decrease is still small for all simulations, and this proposed PID approach is considerably less performing than the enrichment of the input snapshots sets (compare with Figure 4.10).

We use this second test case for debriefing and evaluating the PID-ROM parareal method in terms of computational time. The speedup for each simulation is presented in Figure 4.18. The accelerations for $N_{\text{PID}} = 1, 2, 4$ are quite close, but important slowdowns are observed for larger number of PID windows. This behaviour can be explained by the results shown in Table 4.2, which details, for the first and fifth iterations of each simulation, the fraction of the total iteration times spent in computing the ROM spaces and ROM matrices. The computational time for obtaining the reduced spaces presents only a slight increasing in function of N_{PID}, indicating that the compromise between the positive and negative effects of the PID approach respectively on the costs of the POD and the DEIM (see eq. (4.11)) remains quite balanced. On the other hand, we observe an important increase of the computational time for obtaining the ROM matrices, exceeding 20% and 30% of the iteration time for $N_{\text{PID}} = 10$ and $N_{\text{PID}} = 20$, respectively. We recall that a full set of ROM matrices need to be computed for each PID window and each parareal iteration.

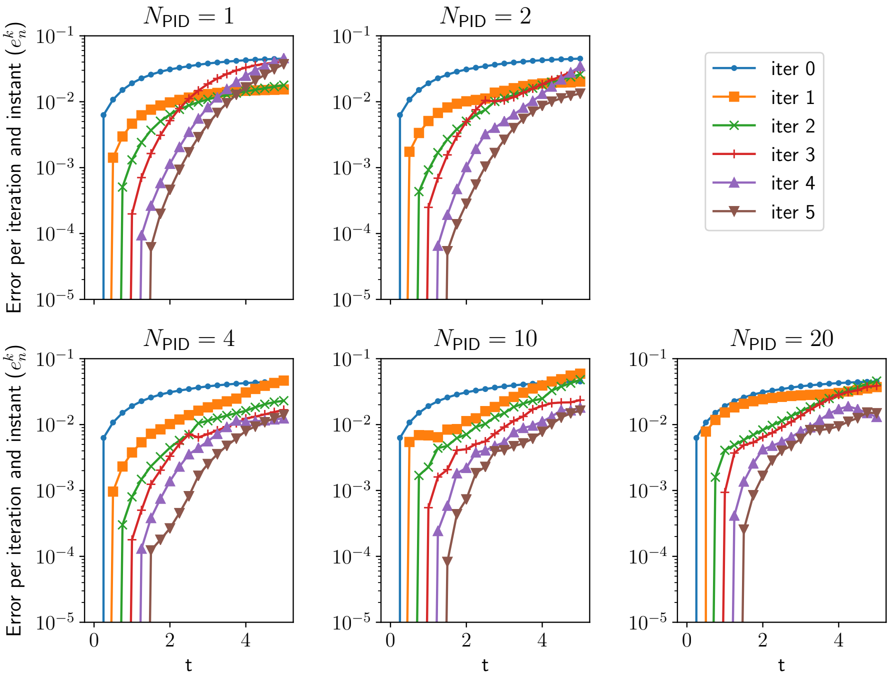


Figure 4.16: Test case 2 (2D) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$: relative error e_n^k using the PID-ROM-based parareal method for various numbers of PID windows.

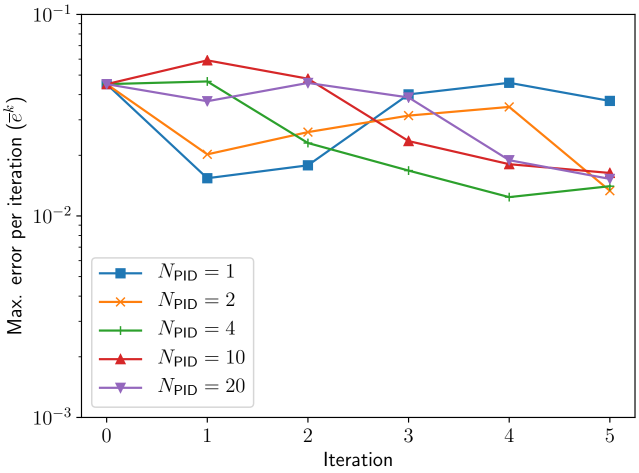


Figure 4.17: Test case 2 (2D) with $\varepsilon_{\text{sv,linear}} = \varepsilon_{\text{sv,nonlinear}} = 10^{-4}$: maximum error per iteration ($\bar{e}^k$) using the PID-ROM-based parareal method for various numbers of PID windows.

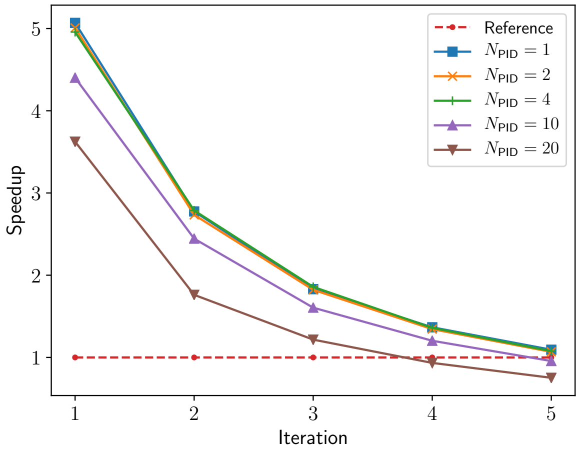


Figure 4.18: Test case 2 (2D) with $\varepsilon_{\mathrm{sv,linear}} = \varepsilon_{\mathrm{sv,nonlinear}} = 10^{-4}$: speedup $s(k)$ along iterations using the PID-ROM-based parareal method for various numbers of PID windows. The red, dashed line corresponds to an unitary speedup.

	Iteration $k = 1$		Iteration $k = 5$	
N_{PID}	Spaces	Matrices	Spaces	Matrices
1	1.5%	1.8%	10.1%	3.4%
2	1.9%	6.0%	12.2%	7.8%
4	1.7%	10.1%	10.6%	13.7%
10	2.1%	20.7%	12.2%	24.8%
20	2.2%	32.9%	14.0%	37.4%

Table 4.2: Test case 2 (2D) with $\varepsilon_{\mathrm{sv,linear}} = \varepsilon_{\mathrm{sv,nonlinear}} = 10^{-4}$: fraction of the computational time spent for computing the ROM spaces and ROM matrices w.r.t. the total iteration time at the first and fifth iterations, using the PID-ROM-based parareal method for various numbers of PID windows.

4.5 An adaptive approach

In this section, we consider an adaptive approach recently introduced by Maday and Mula (2020) in the framework of the classical parareal method and we explore its application to the ROM-based algorithm. This approach is proposed by the authors to overcome the major limitation, in terms of parallel efficiency, of the parareal algorithm: the cost of the fine solver. Indeed, since the fine propagator used along parareal iterations has the same accuracy as the reference one, the fine computation at each iteration using N_p processors has, in an ideal case, a cost $1/N_p$ times smaller than the cost of the reference solution. This issue is evident in bound (3.43), which indicates that the numerical speedup after k parareal iterations decreases by a factor $1/k$. Therefore, the adaptive algorithm consists in using an adaptive fine propagator, with an increasing accuracy (and thus an increasing cost) across iterations, instead of the expensive, reference propagator with a fixed accuracy. In this section, we briefly describe this approach and its properties, following its presentation by Maday and Mula (2020), before proposing its application to the ROM-based parareal algorithm.

The adaptive parareal method

Let $[\mathcal{E}(\boldsymbol{y}, t_{n+1}, t_n); \zeta]$ be a generic propagator for solving a time-dependent problem, say (3.6), and defined by an accuracy ζ, in the sense that

$$\|\mathcal{E}_{\mathrm{exact}}(\boldsymbol{y}, t_{n+1}, t_n) - [\mathcal{E}(\boldsymbol{y}, t_{n+1}, t_n); \zeta]\| \leq \zeta(t_{n+1} - t_n)(1 + \|\boldsymbol{y}\|) \tag{4.12}$$

where ζ is the time step in the case of an Euler scheme and $\mathcal{E}_{\mathrm{exact}}(\boldsymbol{y}, t_{n+1}, t_n)$ is the exact propagator for (3.6). By defining the accuracies of the fine and the coarse propagators respectively as $\varepsilon_{\mathcal{F}_{\delta t}}$ and $\varepsilon_{\mathcal{G}_{\Delta t}}$, definition (4.12) yields $\mathcal{F}_{\delta t}(\boldsymbol{y}, t_{n+1}, t_n) = [\mathcal{E}(\boldsymbol{y}, t_{n+1}, t_n); \varepsilon_{\mathcal{F}_{\delta t}}]$ and $\mathcal{G}_{\Delta t}(\boldsymbol{y}, t_{n+1}, t_n) = [\mathcal{E}(\boldsymbol{y}, t_{n+1}, t_n); \varepsilon_{\mathcal{G}_{\Delta t}}]$. The idea of the adaptive parareal method is to replace (3.8) by the predictor-corrector iteration

$$\boldsymbol{y}_{n+1}^{k+1} = \underbrace{\mathcal{G}_{\Delta t}(\boldsymbol{y}_n^{k+1}, t_{n+1}, t_n)}_{\text{Prediction}} + \underbrace{[\mathcal{E}(\boldsymbol{y}, t_{n+1}, t_n); \zeta_n^k] - \mathcal{G}_{\Delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)}_{\text{Correction}}, \qquad n = 0, \dots, N_{\Delta T} - 1 \tag{4.13}$$

i.e. the fine, reference model is replaced by a propagator with an accuracy ζ_n^k to be chosen adaptively and depending on the iteration and time of simulation. If $\zeta_n^k \geq \varepsilon_{\mathcal{F}_{\delta t}}$, then this new propagator is less expensive than the reference one. By using propagators with an increasing accuracy across iterations (*i.e.* with a decreasing ζ_n^k), a substantial reduction of the computational cost is obtained in early iterations, for which a too precise simulation may not be necessary since the parareal solution itself is relatively inaccurate. Maday and Mula (2020) show that the adaptive approach, in an ideal framework, has always a larger parallel efficiency (defined as the ratio between the speedup and the number of parallel processors) than the classical one and independent of the number of iterations. Thus, the main obstacle for the parallel efficiency is no longer the cost of the fine, reference propagator, but that of the coarse one.

A question that naturally arises when using (4.13) is how to choose the accuracies ζ_n^k. Maday and Mula (2020) derive an expression for the maximum ζ_n^k that delivers at iteration k a solution whose accuracy is close to the one delivered by an idealized parareal method that uses the exact propagator $\mathcal{E}_{\text{exact}}$ as fine one. However, since the derived expression is based on theoretical assumptions on the propagators, it was verified that it does not give good results when used in practice. As pointed out by the authors, the choice of the adaptive accuracies should rely on a posteriori error estimators. However, they tested, with satisfactory results, a simpler and predetermined formulation

$$\zeta_n^k = \begin{cases} \varepsilon_{\mathcal{G}}^{1-\frac{k+1}{\hat{k}}} \varepsilon_{\mathcal{F}_{\delta t}}^{\frac{k+1}{\hat{k}}}, & k < \hat{k} \\ \varepsilon_{\mathcal{F}_{\delta t}}, & k \geq \hat{k} \end{cases} \tag{4.14}$$

where $\hat{k}$ is the number of iterations for the convergence, to a given target accuracy $\varepsilon_{\mathcal{F}_{\delta t}}$, of the classical parareal method using $\mathcal{F}_{\delta t}$ as fine propagator in all iterations.

4.5.1 An adaptive ROM-based parareal method

We propose to use a similar approach in the framework of the ROM-based parareal method. However, in contrast to (4.14), in which a new fine propagator is used at each iteration, we consider a simplified scenario where there exists a fixed number N_{ad} of fine propagators, each one defined by a time and space discretizations. We propose this approach having in mind the application envisaged in this work: this scenario is likely to be met in the simulation of urban floods, where the mesh size and the time step are usually linked by stability conditions and the mesh generation may be laborious, such that few different propagators may be available.

Let $\widehat{\mathcal{F}}^{N_{\text{ad}}-1}_{\delta t_{N_{\text{ad}}-1}}, \widehat{\mathcal{F}}^{N_{\text{ad}}-2}_{\delta t_{N_{\text{ad}}-2}}, \dots, \widehat{\mathcal{F}}^{0}_{\delta t_0}$ be the N_{ad} fine propagators, defined with increasing accuracy such that $\widehat{\mathcal{F}}^{0}_{\delta t_0} = \mathcal{F}_{\delta t}$. The propagator $\widehat{\mathcal{F}}^{j}_{\delta t_j}$ has an homogeneous time step δt_j and a mesh size h_j, $j = 0, \dots N_{\text{ad}} - 1$. Since N_{ad} may be smaller than the number N_{itermax} of parareal iterations, each propagator may be used in more than one iteration. Then, we define a function $J : \{0, \dots, N_{\text{itermax}} - 1\} \rightarrow \{0, \dots, N_{\text{ad}} - 1\}$ indicating the fine propagator to be used in each iteration. Thus, the iteration of the proposed approach (which we call hereafter as *adaptive ROM-based parareal method*) reads

$$\boldsymbol{y}_{n+1}^{k+1} = \underbrace{\widehat{\mathcal{F}}^{k,J(k)}_{r,\delta t_{J(k)}}(\mathbb{P}^{J(k)} \boldsymbol{y}_n^{k+1}, t_{n+1}, t_n)}_{\text{Prediction}} + \underbrace{\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}(\boldsymbol{y}_{n-1}^k) - \widehat{\mathcal{F}}^{k,J(k)}_{r,\delta t_{J(k)}}(\mathbb{P}^{J(k)} \boldsymbol{y}_n^k, t_{n+1}, t_n)}_{\text{Correction}}, \qquad n = 0, \dots, N_{\Delta T} - 1 \tag{4.15}$$

where $\widehat{\mathcal{F}}^{k,J(k)}_{r,\delta t}$ is a ROM approximating $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ and $\mathbb{P}^{J(k)}$ is the projection operator into the reduced subspace defining $\widehat{\mathcal{F}}^{k,J(k)}_{r,\delta t}$.

The ROMs are formulated as in Algorithm (6), using the POD and POD-EIM procedures applied respectively to snapshots of the solution and snapshots of the nonlinear term. The only difference concerns the number of snapshots in each iteration. As discussed in Section 3.4.3, the idea behind the definition of the snapshot sets Y^k and $\widehat{Y}^k$, which contain respectively the fine evolution of the parareal solutions along each time slice ($\widetilde{\boldsymbol{y}}^j_n, j = 0, \dots, k;\; n = 0, \dots, N_{\Delta T}$, with $\widetilde{\boldsymbol{y}}^j_n = \mathcal{F}_{\delta t}(\boldsymbol{y}^j_{n-1})$), and the nonlinear term computed on these quantities ($\boldsymbol{F}(\widetilde{\boldsymbol{y}}^j_n), j = 0, \dots, k; n = 0, \dots, N_{\Delta T}$), is to represent the dynamics of the fine, reference model. In the adaptive approach, the snapshots obtained in previous iterations using coarser intermediate models may not be a good representation of the dynamics of the current model $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ used in iteration k. Therefore, the ROM $\widehat{\mathcal{F}}^{k,J(k)}_{r,\delta t_{J(k)}}$ that approximates $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ is formulated using only the snapshots obtained in the iterations in which $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ is used. Then, we define the snapshots sets for each iteration k as:

$$\begin{aligned} Y^k_{\text{ad}} &= \{\widetilde{\boldsymbol{y}}^j_n,\; j \mid 0 \le j \le k \text{ and } J(j) = J(k);\; n = 0, \dots, N_{\Delta T}\} \\ \widehat{Y}^k_{\text{ad}} &= \{\boldsymbol{F}(\widetilde{\boldsymbol{y}}^j_n),\; j \mid 0 \le j \le k \text{ and } J(j) = J(k);\; n = 0, \dots, N_{\Delta T}\} \end{aligned} \tag{4.16}$$

Therefore, since the number of snapshots used for the model reduction is smaller than in the non-adaptive ROM-based parareal method, we can also expect an improvement of the speedup due to the less expensive model reduction procedures.

As a last remark in the definition of the adaptive algorithm, note that the property $\boldsymbol{y}^k_n = \boldsymbol{y}_{\text{ref},n}\ \forall n \le k$ is no longer valid, *i.e.* the adaptive parareal method (both in the classical and ROM-based frameworks) does not ensure exact convergence for one time slice per iteration, since the reference model is not used in all iterations. Indeed, this property is verified only in the iterations using $\widehat{\mathcal{F}}^0_{\delta t_0} = \mathcal{F}_{\delta t}$:

$$\boldsymbol{y}^k_n = \boldsymbol{y}_{\text{ref},n}\ \forall n \le k - \tilde{k}$$

where $\tilde{k} := \min\{0 \le k \le N_{\text{itermax}} \mid J(k) = 0\}$ is the first iteration using $\widehat{\mathcal{F}}^0_{\delta t_0} = \mathcal{F}_{\delta t}$.

Definition of the fine models used in each iteration

A criterion must be chosen for defining the function $J(k)$, *i.e.* for deciding whether to move from one fine propagator to the next (a more refined one). In the same spirit of (3.16) and under the same assumptions, we write

$$\begin{aligned} \left\|\boldsymbol{y}_{\text{ref},n} - \boldsymbol{y}^k_n\right\| &\le \left\|\boldsymbol{y}_{\text{ref},n} - \mathcal{F}_{\delta t}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1})\right\| + \left\|\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1}) - \boldsymbol{y}^k_n\right\| \\ &\quad + \left\|\mathcal{F}_{\delta t}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1}) - \widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1})\right\| \\ &\le \left\|\mathcal{F}_{\delta t}(\boldsymbol{y}_{\text{ref},n-1}, t_n, t_{n-1}) - \mathcal{F}_{\delta t}(\boldsymbol{y}^k_{n-1}, t_n, t_{n+1})\right\| + \left\|\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1}) - \boldsymbol{y}^k_n\right\| \\ &\quad + \left\|\mathcal{F}_{\delta t}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1}) - \widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1})\right\| \\ &\le L\left\|\boldsymbol{y}_{\text{ref},n-1} - \boldsymbol{y}^k_{n-1}\right\| + \left\|\widetilde{\boldsymbol{y}}^k_n - \boldsymbol{y}^k_n\right\| + (\delta t + \delta t_{J(k)})\Delta T(1 + \left\|\boldsymbol{y}^k_{n-1}\right\|) \\ &\le \sum_{j=1}^{n} L^{n-j}\left\|\widetilde{\boldsymbol{y}}^k_j - \boldsymbol{y}^k_j\right\| + (\delta t + \delta t_{J(k)})\Delta T \sum_{j=0}^{n} L^{n-j}(1 + \left\|\boldsymbol{y}^k_{j-1}\right\|) \end{aligned} \tag{4.17}$$

where we used (4.12) for estimating

$$\begin{aligned} \left\|\mathcal{F}_{\delta t}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1}) - \widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1})\right\| &\le \left\|\mathcal{F}_{\delta t}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1}) - \mathcal{E}_{\text{exact}}(\boldsymbol{y}^k_{n-1}, t_{n+1}, t_n)\right\| \\ &\quad + \left\|\mathcal{E}_{\text{exact}}(\boldsymbol{y}^k_{n-1}, t_{n+1}, t_n) - \widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}(\boldsymbol{y}^k_{n-1}, t_n, t_{n-1})\right\| \\ &\le \delta t \Delta T(1 + \left\|\boldsymbol{y}^k_{n-1}\right\|) + \delta t \Delta T(1 + \left\|\boldsymbol{y}^k_{n-1}\right\|) \\ &= (\delta t + \delta t_{J(k)})\Delta T(1 + \left\|\boldsymbol{y}^k_{n-1}\right\|) \end{aligned}$$

In the last line of (4.17), the first term is a summation of the differences between the parareal solutions $\boldsymbol{y}^k_j$ and the fine propagation along the previous time slice using the intermediate fine model $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ ($\widetilde{\boldsymbol{y}}^k_j = \widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}(\boldsymbol{y}^k_{j-1}, t_j, t_{j-1})$). The second term is an estimation of the error between $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ and the reference propagator $\mathcal{F}_{\delta t}$. In other words, the error in the adaptive parareal method w.r.t. the reference solution can be decomposed in a term accounting for the error in approximating the solution of $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ and a second term intrinsic to the use of $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ instead of $\mathcal{F}_{\delta t}$.

Therefore, as in (3.15), we propose an adaptive criterion based on the residual $\left\| \widetilde{\boldsymbol{y}}_n^k - \boldsymbol{y}_n^k \right\|$ between the parareal solution and the fine propagation using the intermediate fine propagator $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$:

$$r^k := \max_{0<n\leq N_{\Delta t}} \frac{\left\| \widetilde{\boldsymbol{y}}_n^k - \boldsymbol{y}_n^k \right\|}{\left\| \boldsymbol{y}_n^k \right\|} \leq \varepsilon_{\mathrm{ad}} \tag{4.18}$$

for a small $\varepsilon_{\mathrm{ad}}$, which means that the transition between one fine propagator to the next (finer) one is performed when a good approximation to the former is obtained. Then, the function $J(k)$ can be defined as

$$\begin{aligned} &J(0) = N_{\mathrm{ad}} - 1; \\ &J(k) = \max\{0, J(k-1) - \theta^{k-1}\}, \qquad k = 1, \dots, N_{\mathrm{itermax}-1} \end{aligned} \tag{4.19}$$

where

$$\theta^k := \begin{cases} 1, & \text{if } r^k \leq \varepsilon_{\mathrm{ad}} \text{ or } k > N^{J(k)}_{\mathrm{itermax}} - 1 \\ 0, & \text{else} \end{cases}$$

and $N^{J(k)}_{\mathrm{itermax}} - 1$ is a maximum iteration index in which $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ should be used, being defined in situations in which the residual r^k decreases slowly.

The resulting adaptive ROM-based parareal method is presented in details in Algorithm 9. It can be straightforwardly simplified to the framework of the classical parareal method.

4.5.2 Speedup estimation

Contrary to the snapshot enrichment and PID approaches, presented respectively in Sections 4.3 and 4.4 and whose impact on the speedup concerns mainly the model reduction procedure (thus modifying bound (3.44)), the adaptive approach has influence on several terms of the computational time of the parareal iteration (eq. (3.39)).

In order to make the speedup estimation more tangible, we formulate some hypothesis on the fine models $\widehat{\mathcal{F}}^i_{\delta t_i}, i = 0, \dots, N_{\mathrm{ad}} - 1$. We suppose that a constant factor $0 < \beta < 1$ links the temporal discretizations of two consecutive fine models, *i.e.* $\delta t_i = \beta \delta t_{i+1}, i = 0, \dots, N_{\mathrm{ad}} - 2$, such that their respective numbers of time steps satisfy $N_{\delta t i} + 1 = (N_{\delta t i+1} + 1)/\beta$, thus $N_{\delta t i} \approx N_{\delta t i+1}/\beta$. Similarly, the spatial mesh sizes are linked by a constant factor $0 < \gamma \leq 1$: $h_i = \gamma h_{i+1}, i = 0, \dots, N_{\mathrm{ad}} - 2$ (in a Cartesian mesh, one would have $\delta x_i = \gamma \delta x_{i+1}$ and $\delta y = \gamma \delta y_{i+1}$). In a two-dimensional mesh, it implies the approximate relation $M_{f_i} \approx M_{f_{i+1}}/\gamma^2$ between the numbers of mesh cells. Therefore, the dimensions of each intermediate fine propagator are linked to the reference one by

$$N_{\delta t i} \approx \beta^i N_{\delta t 0} = \beta^i N_{\delta t}, \qquad \tau_{f_i} \approx \gamma^{2i} \tau_{f_0} = \gamma^{2i} \tau_f, \qquad i = 0, \dots, N_{\mathrm{ad}} - 1 \tag{4.20}$$

where $\tau_{f_i} = \mathcal{O}(M_{f_i})$ is the computational time necessary for advancing one time step δt_i using $\widehat{\mathcal{F}}^i_{\delta t_i}$. Note that keeping the same CFL number along the fine models means that $\beta = \gamma$. Finally, we suppose that each fine model $\widehat{\mathcal{F}}^i_{\delta t_i}$ is used along $\hat{k}_i$ iterations of the parareal simulation, with $\sum_{i=0}^{N_{\mathrm{ad}}-1} \hat{k}_i =: \hat{k}$.

Under these assumptions, the major improvement of the parareal speedup concerns bound (3.43), *i.e.* the one related to the parallel computation of the fine correction term. Indeed, the fine parallel step at the $k-$th parareal iteration reads, in the adaptive approach,

$$T^k_{\mathrm{corr,f}} = \frac{N_{\Delta T}}{N_p} \frac{N_{\delta t J(k)}}{N_{\Delta T}} \tau_{f_{J(k)}} = \beta^{J(k)} \gamma^{2J(k)} \frac{N_{\delta t}}{N_p} \tau_f < \frac{N_{\delta t}}{N_p} \tau_f \tag{4.21}$$

Similarly, the parallel computation of the coarse correction term (given by the reduced model) in iteration k takes the computational time

$$T^k_{\mathrm{corr,c}} = \frac{N_{\Delta T}}{N_p} \frac{N_{\delta t J(k)}}{N_{\Delta T}} \tau_r(k) = \beta^{J(k)} \frac{N_{\delta t}}{N_p} \tau_r(k) < \frac{N_{\delta t}}{N_p} \tau_r(k) \tag{4.22}$$

Using (4.21) and (4.22) in a similar reasoning for deriving (3.43), and supposing that $\beta^{J(k)} \tau_r(k) \ll \beta^{J(k)} \gamma^{2J(k)} \tau_f$ (thus $T^k_{\mathrm{corr,c}} \ll T^k_{\mathrm{corr,f}}$), we obtain the new bound associated to the correction step of the parareal algorithm:

```
**Initialization**: initial guess given by the coarse propagator:
$\boldsymbol{y}^0_0 = \boldsymbol{y}_0$
**for** $n \leftarrow 0$ **to** $NN_{\Delta T} - 1$ **do**
    $\boldsymbol{y}^0_{n+1} = \mathcal{G}_{\Delta t}(\boldsymbol{y}^0_n, t_{n+1}, t_n)$
**end**

$n_0 = 0$
$J(0) = N_{\text{ad}} - 1$
**Iterations**:
**for** $k \leftarrow 0$ **to** $N_{itermax} - 1$ **do**
    Compute the fine term of the correction (**in parallel**):
    **for** $n \leftarrow n_0$ **to** $N_{\Delta T} - 1$ **do**
        $\widetilde{\boldsymbol{y}}^k_{n+1} = \widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}(\boldsymbol{y}^k_n, t_{n+1}, t_n)$
    **end**

    Find the first instant $\tilde{n} \in \{1, \dots, N_{\Delta t}\}$ not satisfying a convergence criterion based on $\boldsymbol{y}^k_n, \widetilde{\boldsymbol{y}}^k_n$
    $n_0 \leftarrow \tilde{n} - 1$

    Define the snapshots sets:
    $\boldsymbol{Y}^k_{\text{ad}} = \{\widetilde{\boldsymbol{y}}^j_n,\ j \text{ s.t. } 0 \leq j \leq k \text{ and } J(j) = J(k);\ n = 0, \dots, N_{\Delta T}\}$
    $\widehat{\boldsymbol{Y}}^k_{\text{ad}} = \{\boldsymbol{F}(\widetilde{\boldsymbol{y}}^j_n),\ j \text{ s.t. } 0 \leq j \leq k \text{ and } J(j) = J(k);\ n = 0, \dots, N_{\Delta T}\}$

    Compute the spaces and define the reduced model $\widehat{\mathcal{F}}^{k,J(k)}_{r,\delta t}$:
    $\mathcal{S}^k(\boldsymbol{Y}^k_{\text{ad}}, \varepsilon_{\text{sv,linear}})$ (using POD)
    $\widehat{\mathcal{S}}^k(\widehat{\boldsymbol{Y}}^k_{\text{ad}}, \varepsilon_{\text{sv,nonlinear}})$ (using POD-DEIM)

    Compute the coarse term of the correction and the final correction term(**in parallel**):
    **for** $n \leftarrow n_0$ **to** $N_{\Delta T} - 1$ **do**
        $\overline{\boldsymbol{y}}^k_{n+1} = \widehat{\mathcal{F}}^{k,J(k)}_{r,\delta t}(\mathbb{P}^k \boldsymbol{y}^k_n, t_{n+1}, t_n)$
        $\overline{\overline{\boldsymbol{y}}}^k_{n+1} = \widetilde{\boldsymbol{y}}^k_{n+1} - \overline{\boldsymbol{y}}^k_{n+1}$
    **end**

    Compute the coarse predictions and correct them to obtain the final solution in the iteration
     (**sequentially**):
    **for** $n \leftarrow n_0$ **to** $N_{\Delta T} - 1$ **do**
        $\boldsymbol{y}^{k+1}_{n+1} = \widehat{\mathcal{F}}^{k,J(k)}_{r,\delta t}(\mathbb{P}^k \boldsymbol{y}^{k+1}_n, t_{n+1}, t_n) + \overline{\overline{\boldsymbol{y}}}^k_{n+1}$
    **end**

    Decide whether the transition to the next fine propagator will be made
    **if** $J(k) > 0$ ***and*** *(* $r^k \leq \varepsilon_{ad}$ ***or*** $k + 1 > N^{J(k)}_{itermax}$ *)* **then**
        $J(k+1) = J(k) - 1$
        $n_0 \leftarrow 0$
        Interpolate solutions to finer mesh;
    **else**
        $J(k+1) = J(k)$;
    **end**
**end**
```

Algorithm 9: Adaptive ROM-based parareal algorithm. Modifications w.r.t. the ROM-based parareal method (Algorithm 6) are highlighted.

$$\begin{aligned} s_{\text{ROM-adaptive-parareal}}(\hat{k}) &< \frac{T_{\text{ref}}}{\sum_{k=1}^{\hat{k}} (T_{\text{corr,f}}^k + T_{\text{corr,c}}^k)} \approx \frac{T_{\text{ref}}}{\sum_{k=1}^{\hat{k}} T_{\text{corr,f}}^k} \\ &= \frac{N_{\delta t}\tau_f}{\sum_{k=1}^{\hat{k}} \beta^{J(k)}\gamma^{2J(k)} \frac{N_{\delta t}}{N_p}\tau_f} = \frac{N_p}{\sum_{i=0}^{N_{\text{ad}}-1} \hat{k}_i \beta^i \gamma^{2i}} \\ &= \frac{N_p}{\hat{k}_0 + \sum_{i=1}^{N_{\text{ad}}-1} \hat{k}_i \beta^i \gamma^{2i}} \end{aligned} \tag{4.23}$$

We remark that this estimate is also valid for the adaptive approach applied to the classical parareal method. In the last line of (4.23), we make explicit the influence of the $\hat{k}_0$ iterations using $\widehat{\mathcal{F}}_{\delta t_0}^0$ and the iterations using the intermediate propagators. Note that $\hat{k}_0 + \sum_{i=1}^{N_{\text{ad}}-1} \hat{k}_i \beta^i \gamma^{2i} < \hat{k} = \sum_{i=0}^{N_{\text{ad}}-1} \hat{k}_i$. Therefore, by only comparing bounds (3.43) and (4.23), one easily concludes that $s_{\text{ROM-adaptive-parareal}}(\hat{k}) > s_{\text{ROM-parareal}}(\hat{k}) = N_p/\hat{k}$, *i.e.* for the same number of iterations, the adaptive parareal approach is always faster than the non-adaptive one, since in the latter we have $\hat{k}_0 = \hat{k}$. Therefore, the major parallel limitation of the parareal algorithm is overcame. If β and γ are small enough, one concludes from (4.23) that the computational time for the fine corrections is largely dominated by the (few) $\hat{k}_0$ iterations in which $\mathcal{F}_{\delta t}$ is used (instead of $\hat{k}$ iterations as in the non-adaptive approach). Moreover, in the particular case where $\beta = \gamma$ and $N_{\text{ad}} = \hat{k}$, with each fine propagator being used in one iteration ($\hat{k}_i = 1, i = 0, \dots N_{\text{ad}} - 1$), eq. (4.23) yields

$$s_{\text{ROM-adaptive-parareal}}(\hat{k}) < \frac{N_p}{\sum_{i=0}^{N_{\text{ad}}-1} \beta^{3i}} = \frac{(1-\beta^3)}{1-\beta^{3N_{\text{ad}}}} N_p \approx (1-\beta^3) N_p$$

by supposing that $\beta^{3N_{\text{ad}}} \ll 1$. It means that the parallel efficiency of the parareal method is no longer limited by the number of iterations, which is one of the main conclusions of the work of Maday and Mula (2020).

Similarly but less remarkably, bound (3.42), linked to the prediction term (given by the reduced model), is rewritten as

$$s_{\text{ROM-adaptive-parareal}}(\hat{k}) < \frac{M_f}{\hat{m}(\hat{k}_0 + \sum_{i=1}^{N_{\text{ad}}-1} \hat{k}_i \beta^i)} \tag{4.24}$$

where the factor $1/\hat{k}$ is replaced by the larger factor $1/(\hat{k}_0 + \sum_{i=1}^{N_{\text{ad}}-1} \hat{k}_i \beta^i)$. Note that we do not make any additional assumption on the dimension $\hat{m}$ of the ROM and the cost $\tau_r(k)$ for solving it in the adaptive approach, and we still majorate this cost by $\tau_r(\hat{k}) = \mathcal{O}(\hat{m})$.

Finally, concerning the formulation of the reduced-order models, we also expect a smaller computational cost for two reasons: firstly, the input snapshot matrices for the POD have a smaller number of rows, due to the smaller spatial discretization of the intermediate fine models (in iteration k, each snapshot matrix has $\gamma^{2J(k)} M_f$ rows, instead of M_f as in the non-adaptive approach); secondly, as already discussed, the snapshot matrices in the adaptive approach contain only snapshots produced by the current intermediate model, thus resulting in a smaller number of columns (at most $\hat{k}_{J(k)}(N_{\Delta T}+1)$ columns in iteration k, instead of $k(N_{\Delta T}+1)$ in the non-adaptive method). Then, by assuming, for simplification, that all the fine models are used in the same number of parareal iterations (*i.e.* $\hat{k}_i = \hat{k}/N_{\text{ad}}, i = 0, \dots, N_{\text{ad}} - 1$), bound (3.44) is improved to

$$s_{\text{ROM-adaptive-parareal}}(\hat{k}) < \frac{N_{p,s} N_{\delta t} M_f}{\hat{k} N_{\text{spaces}} \frac{\hat{m}^4}{4} + \tilde{k} M_f \left[N_{\text{spaces}} \left(\frac{\hat{k}^2}{N_{\text{ad}}^2} N_{\Delta T}^2 + \hat{m} \right) + 2 N_{\text{matrices}} \hat{m}^2 \right]} \tag{4.25}$$

where the second term in the denominator is multiplied by $\tilde{k} := \sum_{i=0}^{N_{\text{ad}}-1} \hat{k}_i \gamma^{2i}$ instead of the larger value $\hat{k}$, and the quadratic term on $\hat{k}$ is divided by N_{ad}^2.

As a drawback of the adaptive approach concerning the speedup, a spatial interpolation must be performed at each transition between two consecutive fine models $\widehat{\mathcal{F}}_{\delta t_i}^i$ and $\widehat{\mathcal{F}}_{\delta t_{i-1}}^{i-1}$ (in the case where they use different spatial meshes). Evidently, this additional cost is larger for transitions between the finest propagators (*i.e.* for smaller pairs $(i, i-1) \in \{0, \dots, N_{\text{ad}} - 1\}^2$).

4.5.3 Numerical examples

We consider here Test cases 1 and 2 for illustrating the adaptive parareal method, both in the frameworks of the classical and ROM-based methods (respectively eq. (4.13) and (4.15)). As described above, we consider a small number N_{ad} of predefined fine propagators, with $\widehat{\mathcal{F}}^{i}_{\delta t_i}$ refined in space and/or in time w.r.t. $\widehat{\mathcal{F}}^{i+1}_{\delta t_{i+1}}$, $i = 0, \ldots, N_{\text{ad}} - 2$. We also consider two different thresholds ε_{ad} for the transition between the fine models.

Test case 1 (pseudo-2D without spatial coarsening)

Test case 1 does not involve spatial coarsening (see Table 3.1). Therefore, the intermediate adaptive fine propagators are defined exclusively by decreasing time steps, chosen so as to be integer multiples of each other. We consider $N_{\text{ad}} = 5$ fine propagators (the reference and four intermediate ones), whose configuration are summarized in Table 4.3

Propagator	Time step	Mesh size
$\widehat{\mathcal{F}}^{4}_{\delta t_4}$	$\delta t_4 = 0.1$	$\delta x_4 = \delta x = \delta y_4 = \delta y = 1$
$\widehat{\mathcal{F}}^{3}_{\delta t_3}$	$\delta t_3 = 0.05$	$\delta x_3 = \delta x = \delta y_3 = \delta y = 1$
$\widehat{\mathcal{F}}^{2}_{\delta t_2}$	$\delta t_2 = 0.01$	$\delta x_2 = \delta x = \delta y_2 = \delta y = 1$
$\widehat{\mathcal{F}}^{1}_{\delta t_1}$	$\delta t_1 = 0.005$	$\delta x_1 = \delta x = \delta y_1 = \delta y = 1$
$\widehat{\mathcal{F}}^{0}_{\delta t_0} = \mathcal{F}_{\delta t}$	$\delta t_0 = \delta t = 0.001$	$\delta x_0 = \delta x = \delta y_0 = \delta y = 1$

Table 4.3: Test case 1 (1D without spatial coarsening): definition of the fine propagators for the adaptive parareal method.

Concerning the use of these propagators, we consider two adaptive configurations, named for simplicity as AD_{A} and AD_{B}, for which we consider respectively the thresholds $\varepsilon_{\text{ad}} = 10^{-2}$ and $\varepsilon_{\text{ad}} = 10^{-3}$ for deciding whether a transition between fine propagators is made. In both configurations, we perform ten parareal iterations ($N_{\text{itermax}} = 10$), with each fine propagator used at most in two iterations. These configurations are summarized in Table 4.4 and are also considered for the simulation of Test case 2.

Configuration	N_{itermax}	ε_{ad}	N^4_{itermax}	N^3_{itermax}	N^2_{itermax}	N^1_{itermax}	N^0_{itermax}
AD_{A}	10	10^{-2}	2	4	6	8	10
AD_{B}	10	10^{-3}	2	4	6	8	10

Table 4.4: Test cases 1 and 2: configuration of the adaptive parareal method.

We perform the adaptive parareal simulations both in the framework of the classical and the ROM-based parareal methods. Figures 4.19 and 4.20 present respectively the relative errors e^k_n and $\bar{e}^k$ for each simulation. We observe a small improvement of the classical parareal method by using the adaptive approach, but globally the error reduction is still unsatisfactory. For the ROM-based method, a good convergence is observed, but needing more iterations than in the non-adaptive approach, since coarser fine propagators are used along the simulation. We notice that, in all adaptive simulations, the residual r^k defined in (4.18) remains between 10^{-3} and 10^{-2} in all iterations. As a consequence, the intermediate fine propagators $\widehat{\mathcal{F}}^{i}_{\delta t_i}, i = 1, \ldots, 4$ are used in only one iteration each in configuration AD_{A} (the transition between the propagators being determined by $r^k < \varepsilon_{\text{ad}} = 10^{-2}$), and $\widehat{\mathcal{F}}^{0}_{\delta t_0} = \mathcal{F}_{\delta t}$ is used from the fifth to the tenth iterations. On the other hand, in configuration AD_{B}, each fine propagator is used in two iterations (the transition being ruled by $N^i_{\text{itermax}}, i = 0, \ldots, N_{\text{ad}} - 1$). It explains the faster convergence, in terms of number of iterations, of configuration AD_{A} compared to AD_{B}.

The computational times presented in Figure 4.21 clearly illustrates the interest in using the adaptive parareal approach. In the adaptive simulations, the computational times are largely dominated by the iterations using the reference propagators $\widehat{\mathcal{F}}^{0}_{\delta t_0} = \mathcal{F}_{\delta t}$ (iterations 5-10 in configuration AD_{A} and iterations 9-10 in configuration AD_{B}). It is particularly clear in the ROM-based simulations, since the adaptive approach also reduces the cost of the model reduction procedures. Therefore, even if the convergence is slower (in terms of number of iterations) in the adaptive simulations, a faster simulation (in terms of speedup) can be obtained. It is illustrated in Figure 4.20 by comparing the number of iterations required by each simulation for reaching the accuracy of the finest intermediate propagator ($\widehat{\mathcal{F}}^1_{\delta t_1}$) w.r.t. $\widehat{\mathcal{F}}^0_{\delta t_0}$, indicated by an horizontal line: three iterations are needed in the non-adaptive ROM-based simulation, whereas adaptive configurations $\mathrm{AD_A}$ and $\mathrm{AD_B}$ require respectively five and eight iterations. As shown in Figure 4.21, three non-adaptive iterations takes almost the reference computational time; on the other hand, five and eight iterations of configurations $\mathrm{AD_A}$ and $\mathrm{AD_B}$ are completed within less than half of this cost.

It is more objectively illustrated in Table 4.5, showing the number of iterations and respective speedup for obtaining given target accuracies, corresponding to the maximum error per iteration (the results for the classical parareal method are not shown due to its poor convergence behaviour, as illustrated in Figure 4.20). The non-adaptive approach converges in fewer iterations, but the same errors can be obtained within smaller computational times by using adaptive configurations. This evaluation of the number of iterations and speedup for reaching target accuracies is defined more precisely in Section 4.6 and used for objectively comparing several configurations of the parareal method.

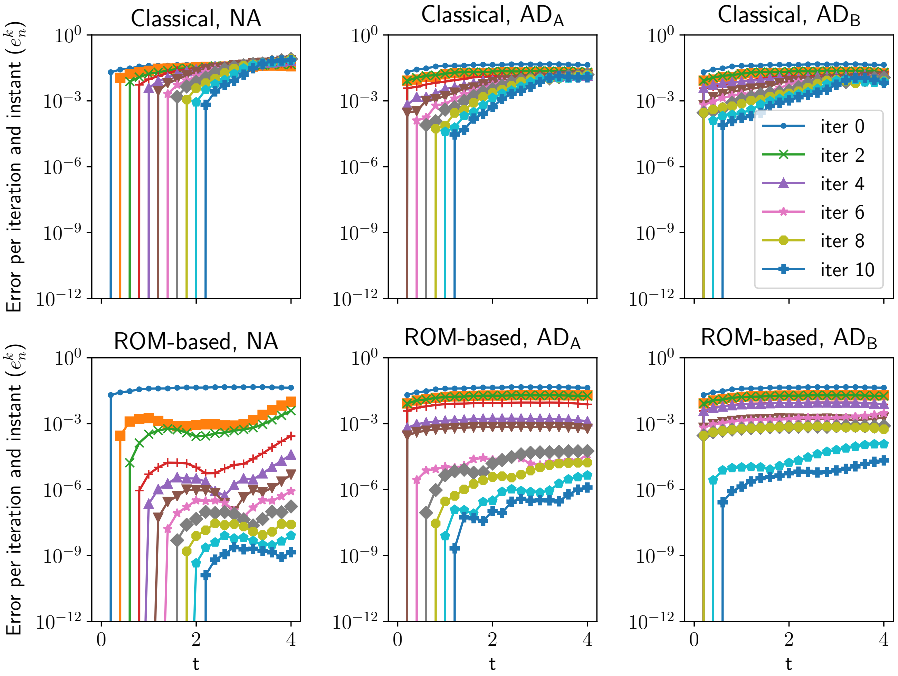


Figure 4.19: Test case 1 (pseudo-2D without spatial coarsening): relative error e_n^k using the classical (top) and ROM-based (bottom) parareal methods, without (NA) and with the adaptive approach, for configurations $\mathrm{AD_A}$ and $\mathrm{AD_B}$. For the sake of conciseness, only some iterations are indicated in legend.

	ROM, NA		ROM, $\mathrm{AD_A}$		ROM, $\mathrm{AD_B}$	
Target accuracy ($\bar{e}^k$)	Iteration	Speedup	Iteration	Speedup	Iteration	Speedup
10^{-2}	2	1.56	3	5.89	3	6.02
10^{-3}	3	1.09	5	3.31	7	2.65
10^{-4}	4	0.83	6	1.77	10	1.01

Table 4.5: Test case 1 (pseudo-2D without spatial coarsening): number of iterations and respective speedup for obtaining given target accuracies (maximum error per iteration) w.r.t. $\mathcal{F}_{\delta t} = \widehat{\mathcal{F}}^0_{\delta t_0}$, for the ROM-based parareal method without (NA) and with the adaptive approach (configurations $\mathrm{AD_A}$ and $\mathrm{AD_B}$). The error $\bar{e}^0$ in the 0-th iteration (corresponding to the error between $\mathcal{G}_{\Delta t}$ and $\mathcal{F}_{\delta t}$) is 4.66×10^{-2}.

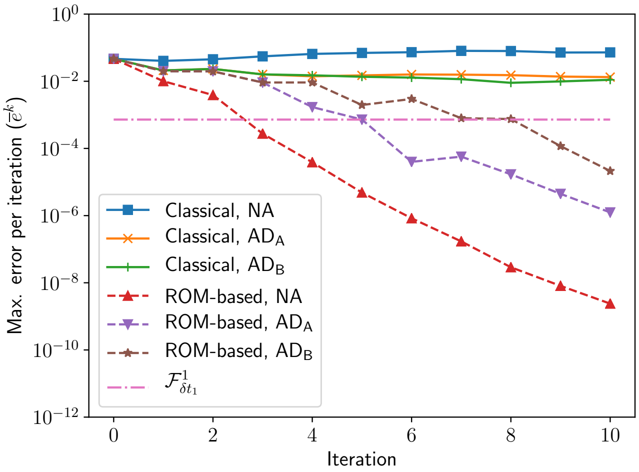


Figure 4.20: Test case 1 (pseudo-2D without spatial coarsening): relative error $\bar{e}^k$ using the classical (full lines) and ROM-based (dashed lines) parareal methods, without (NA) and with the adaptive approach, for configurations AD_A and AD_B. The dash-dotted, horizontal line indicates the maximum error in time between $\widehat{\mathcal{F}}^1_{\delta t_1}$ (the finest intermediate propagator) and $\mathcal{F}_{\delta t}$.

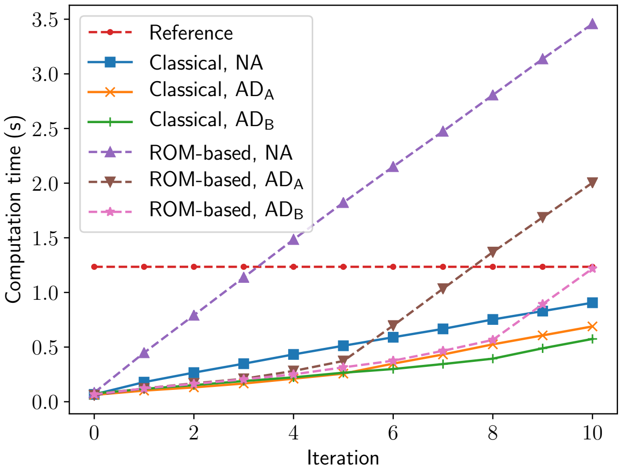


Figure 4.21: Test case 1 (pseudo-2D without spatial coarsening): computational time along iterations using the classical (full lines) and ROM-based (dashed lines) parareal methods, without (NA) and with the adaptive approach, for configurations AD_A and AD_B. The horizontal, dashed line indicates the computational time of the reference simulation.

Test case 2 (2D)

Contrary to Test case 1, in Test case 2 there is spatial coarsening between the fine and coarse propagators (Table 3.3), which is taken into account for defining the adaptive propagators. We consider five fine propagators (the reference and four intermediate ones), *i.e.* $N_{\text{ad}} = 5$, all of them characterized by homogeneous time steps and Cartesian spatial meshes, as described in Table 4.6. Each fine propagator is refined w.r.t. the previous one in time and/or space. Note that they are not formulated following strictly the notation introduced for the speedup estimation (Section 4.5.2), since we do not consider constant ratios β and γ for defining the time step and mesh sizes. However, as presented in the following, the conclusions on the speedup are observed in practice. Concerning the use of the propagators along iterations, the same configurations AD_A and AD_B presented in Table 4.4 are considered. As in Test case 1, the residuals r^k remains between 10^{-2} and 10^{-3} in all iterations, and the number of iterations in which each fine propagator is used is the same as above.

Propagator	Time step	Mesh size
$\widehat{\mathcal{F}}^4_{\delta t_4}$	$\delta t_4 = 0.05$	$\delta x_4 = \delta y_4 \approx 3.85$
$\widehat{\mathcal{F}}^3_{\delta t_3}$	$\delta t_3 = 0.05$	$\delta x_3 = \delta y_3 \approx 2.94$
$\widehat{\mathcal{F}}^2_{\delta t_2}$	$\delta t_2 = 0.005$	$\delta x_2 = \delta y_2 \approx 2.94$
$\widehat{\mathcal{F}}^1_{\delta t_1}$	$\delta t_1 = 0.005$	$\delta x_0 = \delta x = \delta y_0 = \delta y = 2$
$\widehat{\mathcal{F}}^0_{\delta t_0} = \mathcal{F}_{\delta t}$	$\delta t_0 = \delta t = 0.001$	$\delta x_0 = \delta x = \delta y_0 = \delta y = 2$

Table 4.6: Test case 2 (2D): definition of the fine propagators for the adaptive parareal method.

The relative errors e_n^k and $\overline{e}^k$ for this test case are presented respectively in Figures 4.22 and 4.23. For the classical parareal simulations, we observe a similar convergence behaviour, with slightly larger errors in the adaptive approach due to the use of less refined models. For the ROM-based simulations, the adaptive approach leads to a slower convergence in the beginning of the temporal domain, but the instabilities observed in the non-adaptive simulation for advanced time steps are less important. It indicates that a gradual formulation of the reduced-order models in the parareal framework may improve its stability: instead of using the relatively inaccurate snapshots, obtained along parareal iterations, for constructing a ROM approximating the very fine model $\widehat{\mathcal{F}}^0_{\delta t_0} = \mathcal{F}_{\delta t}$, they are used for constructing a ROM that approximates the less fine propagators $\widehat{\mathcal{F}}^{J(k)}_{\delta t_{J(k)}}$ at each iteration k.

The computational times presented in Figure 4.24 behave similar than in Test case 1, but the advantages of using the adaptive approach are more remarkable, since Test case 2 is a larger problem and the intermediate fine propagators are coarsen both in time and space w.r.t. the reference one. The computational times of the adaptive simulation are largely dominated by the iterations using $\widehat{\mathcal{F}}^0_{\delta t_0} = \mathcal{F}_{\delta t}$. Table 4.7, presenting the number of iterations and respective speedup for reaching given target accuracies, reveals important advantages in using the adaptive approach in the classical parareal framework. For the ROM-based method, less remarkable advantages are observed since the maximum error per iteration decreases slowly. We consider larger target accuracies than in Test case 1 since the parareal methods are less performing in Test case 2.

However, even if the adaptive approach allows to reach given errors faster than the non-adaptive method, it still provides quite unsatisfactory results in terms of convergence for Test case 2. Indeed, as shown in Figure 4.23 the accuracy of $\widehat{\mathcal{F}}^1_{\delta t_1}$ is not reached by any of the performed simulation within N_{itermax} iterations. Therefore, in order to obtain a good approximation for the reference solution, it would be more interesting to perform a serial simulation of $\widehat{\mathcal{F}}^1_{\delta t_1}$, which would provide a speedup roughly equal to 5, since $\widehat{\mathcal{F}}^1_{\delta t_1}$ uses the same spatial mesh as $\mathcal{F}_{\delta t}$ but a time step five times larger. It indicates that the configurations of the intermediate propagators, which were chosen arbitrarily, are not well suited, and more efficient choices should be made. In any case, as discussed above, interesting conclusions can be made from this study of the adaptive approach, specially concerning the improved stability of the ROM-based parareal method.

	CL, NA		CL, AD$_\text{A}$		CL, AD$_\text{B}$	
Target accuracy ($\overline{e}^k$)	Iteration	Speedup	Iteration	Speedup	Iteration	Speedup
10^{-2}	5	3.37	7	4.23	9	6.23
5×10^{-3}	7	2.46	8	3.38	10	4.54
	ROM, NA		ROM, AD$_\text{A}$		ROM, AD$_\text{B}$	
Target accuracy ($\overline{e}^k$)	Iteration	Speedup	Iteration	Speedup	Iteration	Speedup
10^{-2}	8	0.69	10	0.80	10	1.69
5×10^{-3}	9	0.62	#	#	#	#

Table 4.7: Test case 2 (2D): number of iterations and respective speedup for obtaining given target accuracies (maximum error per iteration), for the classical (CL) the ROM-based parareal methods without (NA) and with the adaptive approach (configuration AD$_\text{A}$ and AD$_\text{B}$). "#" indicates that the target accuracy is not attained within N_{itermax} iterations. The error $\overline{e}^0$ in the 0-th iteration (corresponding to the error between $\mathcal{G}_{\Delta t}$ and $\mathcal{F}_{\delta t}$) is 4.51×10^{-2}.

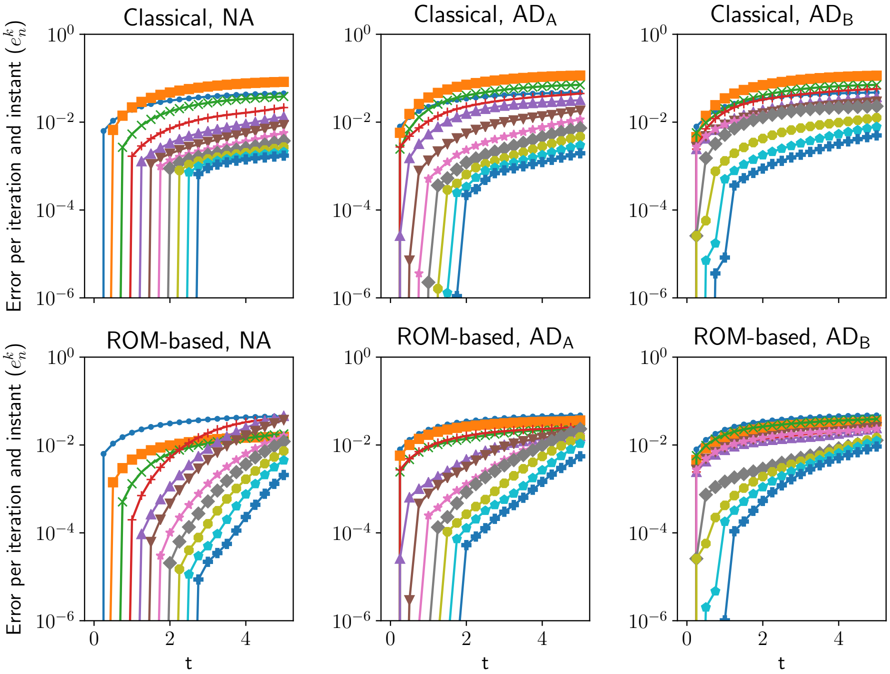


Figure 4.22: Test case 2 (2D): relative error e_n^k using the classical (top) and ROM-based (bottom) parareal methods, without (NA) and with the adaptive approaches, for configurations AD_A and AD_B. For the sake of conciseness, the legend is omitted and coincides with the legend in Figure 4.20.

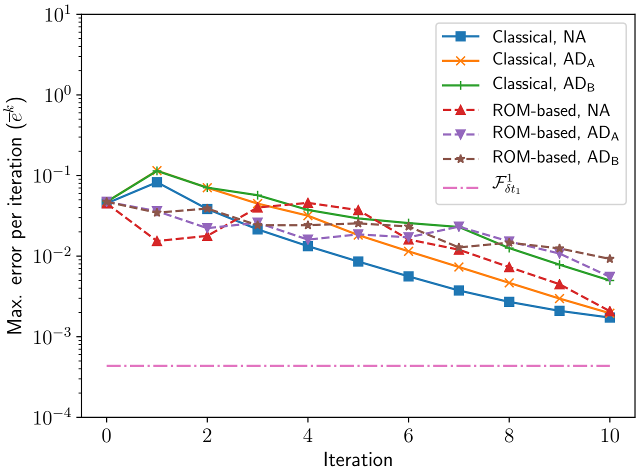


Figure 4.23: Test case 2 (2D): relative error e_n^k using the classical (full lines) and ROM-based (dashed lines) parareal methods, without (NA) and with the adaptive approach, for configurations AD_A and AD_B. The dash-dotted, horizontal line indicates the maximum error in time between $\widehat{\mathcal{F}}^1_{\delta t_1}$ (the finest intermediate propagator) and $\mathcal{F}_{\delta t}$.

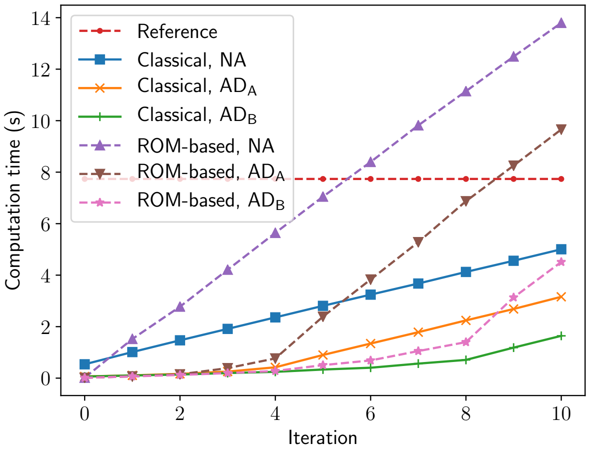


Figure 4.24: Test case 2 (2D): computational time along iterations using the classical (full lines) and ROM-based (dashed lines) parareal methods, without (NA) and with the adaptive approaches, for configurations AD_A and AD_B. The horizontal, dashed line indicates the computational time of the reference simulation.

4.6 Combination of the proposed improvements

Along this chapter, the three proposed modifications to the ROM-based parareal method were tested and evaluated alone. These approaches were able to improve, to a greater or lesser extent, the convergence and stability of the method, with different influences on the computational cost and, as a consequence, on the numerical speedup w.r.t. the reference simulation. In this section, we test and study the behaviour of the combined implementation of these modifications.

Since each proposed approach depends on user-defined parameters (the snapshot enrichment factor α_s; the number of PID windows N_PID and their sizes; and the number N_ad of adaptive fine models and their configuration), a proper study of their combination may lead to a large number of parareal simulations. Therefore, an objective criterion must be chosen for comparing them. We then define

$$k_{\varepsilon,n} := \min K_{\varepsilon,n} \tag{4.26}$$

with

$$K_{\varepsilon,n} = \{0 \leq k \leq N_\text{itermax} \mid e_n^k \leq \varepsilon\} \tag{4.27}$$

and

$$s_{\varepsilon,n} := s(k_{\varepsilon,n})\frac{t_n}{T}, \qquad K_{\varepsilon,n} \neq \emptyset \tag{4.28}$$

That is, $k_{\varepsilon,n}$ is the first parareal iteration such that the relative error e_n^k at t_n is no larger than a given threshold ε; and $s_{\varepsilon,n}$ is the parareal speedup at this iteration, scaled by t_n/T (since the reference simulation from 0 to t_n would take approximately t_n/T of the simulation from 0 to T). In the case where the relative error does not reach ε within N_itermax iterations (*i.e.* $K_{\varepsilon,n} = \emptyset$), the speedup $s_{\varepsilon,n}$ is not defined. Both $k_{\varepsilon,n}$ and $s_{\varepsilon,n}$ are used for comparing the simulations presented in this section.

Finally, all the simulations are performed with $N_\text{itermax} = 10$.

Test case 1 (1D without spatial coarsening)

The following configurations are considered for each modification of the ROM-based parareal method:

- Enrichment of the input snapshot sets: α_s takes value in $\{1, 1/2, 1/4\}$ (3 possible configurations);
- PID approach: N_PID takes value in $\{1, 2, 4\}$, with homogeneous time windows (3 possible configurations);

- Adaptive approach: we consider the same $N_{\mathrm{ad}} = 5$ fine propagators presented in Table 4.3 and configurations $\mathrm{AD_A}$ and $\mathrm{AD_B}$ presented in Table 4.4, in addition to the non-adaptive approach (3 possible configurations).

Additionally, the classical parareal method is also run in the non-adaptive and adaptive frameworks ($\mathrm{AD_A}$ and $\mathrm{AD_B}$). Therefore, a total of 27 ROM-based and 3 classical parareal simulations are performed. The values of $k_{\varepsilon,n}$ and $s_{\varepsilon,n}$ are presented respectively in Tables 4.8 4.9, for $\varepsilon \in \{10^{-2}, 10^{-3}, 10^{-4}\}$ and $n \in \{N_{\Delta T}/2, N_{\Delta T}\}$

Test case 2 (2D)

We consider the following configurations for each modification of the ROM-based parareal method:

- Enrichment of the input snapshot sets: α_{s} takes value in $\{1, 1/2, 1/5\}$ (3 possible configurations);
- PID approach: N_{PID} takes value in $\{1, 2, 4\}$, with homogeneous time windows (3 possible configurations);
- Adaptive approach: we consider the non-adaptive configuration and the configurations $\mathrm{AD_A}$ and $\mathrm{AD_B}$ described in Table 4.4 for the adaptive approach, with the fine propagators as in Table 4.6 (3 possible configurations).

As in the previous test case, the classical parareal method is also run in the non-adaptive and adaptive frameworks (configurations $\mathrm{AD_A}$ and $\mathrm{AD_B}$), Thus, 27 ROM-based and 3 classical parareal simulations are performed. The number of iterations $k_{\varepsilon,n}$ and their respective speedups $s_{\varepsilon,n}$, also for $n \in \{N_{\Delta T}/2, N_{\Delta T}\}$ but for $\varepsilon \in \{10^{-2}, 5 \times 10^{-3}, 10^{-3}\}$, are presented respectively in Table 4.10 and 4.11.

Discussion

The results presented in Tables 4.8- 4.11 reveal that, among the three proposed approaches, the adaptive one provides the most notable improvements in terms of speedup. Even if more iterations are required to reach a given relative error (specially under configuration $\mathrm{AD_B}$, in which the intermediate fine propagators are used in more than one iteration each), the reduced costs along parareal iterations provide interesting accelerations of the simulation. In Test case 1, the error $e^k_{N_{\Delta t}}$ in the final instant of simulation decreases from 4.35×10^{-2} to 10^{-2}, 10^{-3} and 10^{-4} with approximate speedups up to 6, 3 and 2, respectively, in the adaptive configurations. In Test case 2, the obtained solutions are less accurate and the speedups smaller, but good accelerations of the numerical simulation are observed. In both Test cases 1 and 2, the adaptive configuration $\mathrm{AD_B}$ requires more iterations to reach the target accuracies, but, while in Test case 1 the speedups for $\mathrm{AD_A}$ and $\mathrm{AD_B}$ are quite similar, in Test case 2 $\mathrm{AD_B}$ performs better, since the coarsening between the intermediate and reference propagators is more important. Also, as already discussed, the adaptive approach in the framework of the classical parareal method provides good results in Test case 2 but not in Test case 1.

We also observe some positive effects by combining the proposed approaches, specially in Test case 2, in which the non-modified ROM-based parareal method presents instabilities and slow convergence (see Figure 3.19). A decrease of the number of iterations and large speedups for reaching the target accuracies at $t = T$ are obtained by taking an enrichment factor $\alpha_{\mathrm{s}} < 1$, both in the non-adaptive and adaptive approaches, indicating that this enrichment improves the quality of the formulated ROMs at advanced times of simulation. The PID approach also improves the performance of the method, showing that local-in-time ROMs is beneficial for ensuring the quality of the solution in this test case, as discussed previously in this chapter. For example, the simulation with $\alpha_{\mathrm{s}} = 1/2$, $N_{\mathrm{PID}} = 1/2$ and configuration $\mathrm{AD_B}$ presents good results in terms of number of iterations and speedup, for reaching both the accuracies 10^{-2} and 5×10^{-3} at $t = T$. However, the accuracy 10^{-3} is not reached within the performed iterations.

These combined improvements are less remarkable in Test case 1, in which the non-modified ROM-based parareal method already presents a good convergence (see Figure 3.13). For a given low target accuracy ($\varepsilon = 10^{-2}$), the enrichment of snapshots sets ($\alpha_{\mathrm{s}} < 1$) is able, in some simulations, to accelerate the initial convergence and reduce the number of iterations $k_{\varepsilon,n}$, thus increase the speedup $s_{\varepsilon,n}$. In some adaptive cases, slightly larger speedups are observed with $\alpha_{\mathrm{s}} < 1$ than with $\alpha_{\mathrm{s}} = 1$ for the same $k_{\varepsilon,n}$, due to the very low-expensive first iterations (in which coarser fine models are used) and small variabilities of the computational time among executions. However, best speedups for attaining higher accuracies (smaller ε) are obtained in adaptive simulations with $\alpha_{\mathrm{s}} = 1$ and $N_{\mathrm{PID}} = 1$. Indeed, for a given adaptive configuration and a given target accuracy ε, the number of iterations $k_{\varepsilon,n}$ is quite similar for most of the simulations; therefore the enrichment and the PID approaches are more likely to increase the computational time, specially at advanced iterations. Good results in terms of number of iterations and speedup are observed mainly with $N_{\mathrm{PID}} = 1$ and $\alpha_{\mathrm{s}} = 1, 1/2$, in both adaptive configuration AD_{A} and AD_{B}.

It is also interesting to note that, in general, in Test case 1 the same number of iterations is necessary for reaching the target accuracies at $t = T/2$ and $t = T$, whereas in Test case 2 more iterations are necessary for $t = T$. It reflects the behaviour of the error in time in these two simulations, as shown in the error curves presented along this chapter. For Test case 1, the error for a given iteration is more uniform in time, whereas a more important degradation of the solution for advanced time steps is observed in Test case 2.

4.7 Conclusion of the chapter

In this chapter, we proposed and evaluated three modifications for improving the ROM-based parareal method, in terms of stability, convergence and speedup. They were motivated by verifying that the performance of the parareal method strongly depends on the quality of the reduced-order models formulated along parareal iterations. It is a major issue in this context, because the ROMs are constructed using solutions obtained from the parareal iterations; therefore, they may not be a good enough representation of the dynamics of the fine, reference model, mainly in the first iterations. It is important to note that the ROMs are formulated on the fly along the parareal iterations, thus the proposed modifications should remain low-expensive.

The first modification is the enrichment of the input snapshots sets for the model reduction procedure by collecting extra snapshots. As a major feature, these extra snapshots do not require any extra additional cost to be computed, since they are naturally provided by the fine correction step of the parareal method. In the ROM-based parareal method proposed by (Chen et al., 2014), the snapshots are taken only on the parareal time instants (defining the time slices), and we proposed to collect extra snapshots at intermediate time steps. However, even if the snapshots are computed with no extra cost, they should not be too numerous, since the POD model reduction has a quadratic cost on the number of snapshots. Numerical tests showed that this approach provides important improvements to the convergence of the ROM-based parareal method, and good compromises between quality of the solution and numerical speedup were obtained by enriching the snapshots sets with few extra snapshots per time slice.

The second proposed modification is the formulation of local-in-time reduced models within a given number of time windows, inspired by the principal interval decomposition (PID) proposed by Ijzerman (2000). This approach was extended to the POD-EIM framework and incorporated to the parareal method. Numerical experiments showed that the PID-ROM-based parareal method has a more stable behaviour when the POD-based parareal method presents instabilities; however, when the latter already presents a good convergence, the PID approach may degrade the parareal solution. In terms of computational time, the PID does not significantly increase the cost for computing the reduced subspaces; however, a full set of ROM matrices needs to be computed in each time window, and its cost becomes important for a large number of windows. As a more general conclusion, among the three approaches proposed in this chapter, the PID one showed to be the less efficient in improving the ROM-based method.

Finally, the last modification consists in an adaptive parareal approach, as proposed by Maday and Mula (2020). We extended it to ROM-based parareal method. Contrary to the snapshots enrichment and the PID approaches, the adaptive parareal is applicable both in the classical and ROM-based frameworks. It consists in using progressively refined fine propagators along parareal iterations, instead of a fixed one (the reference propagator) in all iterations, with the objective of reducing the computational cost. In the ROM-based framework, numerical tests showed that this approach is not only able to improve the speedup, reaching given target accuracies within smaller computational times (even if more parareal iterations are required), but also to improve the stability, indicating that gradual formulations of the ROMs can be beneficial (instead of formulating ROMs, using relatively inaccurate snapshots, to approximate very refined propagators).

We closed the chapter by testing the combined application of the proposed approaches. Advantages in terms of stability and speedup were observed mainly for Test case 2, which does not behave well in the non-modified ROM-based parareal method.

				$t = T/2$			$t = T$		
	Error at iteration 0			$e^0_{N_{\Delta T}/2} = 4.40\text{E-}2$			$e^0_{N_{\Delta T}} = 4.35\text{E-}2$		
	ε			1E-2	1E-3	1E-4	1E-2	1E-3	1E-4
	Adaptive	N_{PID}	$1/\alpha_{\text{s}}$						
ROM	NA	1	1	1	1	4	2	3	4
			2	1	1	1	1	2	4
			4	1	1	1	1	2	4
		2	1	2	3	5	3	5	7
			2	1	3	4	2	3	4
			4	1	3	4	2	3	4
		4	1	3	4	6	5	4	8
			2	2	3	4	2	4	5
			4	2	2	4	2	3	5
	AD_{A}	1	1	3	5	6	3	5	6
			2	3	5	6	3	5	6
			4	3	5	6	3	5	7
		2	1	3	5	7	3	7	9
			2	3	5	6	3	6	8
			4	3	5	6	3	7	8
		4	1	5	8	10	8	10	#
			2	4	6	7	3	8	9
			4	3	6	7	3	8	9
	AD_{B}	1	1	3	7	9	3	7	10
			2	3	7	8	3	7	8
			4	3	7	8	3	7	8
		2	1	3	7	9	3	9	#
			2	3	7	8	3	7	9
			4	3	7	8	3	7	9
		4	1	5	8	10	8	10	#
			2	4	7	9	3	8	10
			4	3	7	8	3	7	9
CL	NA	−−	−−	7	9	10	#	#	#
	AD_{A}	−−	−−	4	10	#	#	#	#
	AD_{B}	−−	−−	5	#	#	8	#	#

Color codes (ranges of $k_{\varepsilon,n}$)				
#	7 − 10	5 − 6	3 − 4	1 − 2

Table 4.8: Test case 1 (1D without spatial coarsening): number of iterations $k_{\varepsilon,n}$ for reaching a given relative error ε, for $n = N_{\Delta T}/2$ and $n = N_{\Delta T}$, and $\varepsilon \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, for the parareal simulations combining the snapshot enrichment, PID and adaptive approaches. "ROM" and "CL" stand respectively for the ROM-based and the classical parareal method, and "NA" stands for "non-adaptive". The PID and enrichment configurations do not apply to the classical parareal. The relative errors at iteration 0 (given by the coarse propagator $\mathcal{G}_{\Delta t}$) are indicated in the top of the table. A color code corresponding to ranges of $k_{\varepsilon,n}$ is defined. "#" indicates that the error ε is not reached within the performed iterations.

				$t = T/2$			$t = T$		
	Error at iteration 0			$e^0_{N_{\Delta T}/2} = 4.40$E-2			$e^0_{N_{\Delta T}} = 4.35$E-2		
	ε			1E-2	1E-3	1E-4	1E-2	1E-3	1E-4
	Adaptive	N_{PID}	$1/\alpha_{\text{s}}$						
ROM	NA	1	1	1.38	1.38	0.45	1.37	0.90	0.70
			2	1.10	1.10	1.10	2.20	1.26	0.67
			4	1.04	1.04	1.04	2.08	1.18	0.59
		2	1	0.80	0.55	0.34	1.10	0.67	0.49
			2	1.12	0.44	0.33	1.28	0.88	0.66
			4	1.05	0.39	0.29	1.17	0.77	0.58
		4	1	0.55	0.41	0.28	0.83	0.67	0.42
			2	0.63	0.42	0.32	1.27	0.65	0.52
			4	0.57	0.57	0.28	1.13	0.75	0.45
	AD$_{\text{A}}$	1	1	2.94	1.66	0.89	5.89	3.31	1.77
			2	3.29	1.50	0.66	6.58	3.00	1.33
			4	3.05	1.33	0.61	6.11	2.66	0.85
		2	1	2.54	1.44	0.57	5.08	1.13	0.69
			2	2.89	1.35	0.63	5.78	1.26	0.66
			4	2.67	1.17	0.57	5.33	0.79	0.59
		4	1	1.45	0.82	0.43	1.65	0.87	#
			2	1.75	0.59	0.42	5.24	0.65	0.53
			4	2.40	0.53	0.38	4.80	0.59	0.47
	AD$_{\text{B}}$	1	1	3.01	1.33	0.69	6.02	2.65	1.01
			2	3.27	1.12	0.52	6.53	2.25	1.05
			4	3.12	0.90	0.47	6.24	1.80	0.95
		2	1	2.82	1.20	0.64	5.64	1.28	#
			2	2.89	1.00	0.50	5.78	1.99	0.76
			4	2.63	0.8	0.45	5.27	1.60	0.68
		4	1	1.46	0.82	0.43	1.64	0.86	#
			2	1.88	0.87	0.43	5.07	1.44	0.67
			4	2.41	0.75	0.42	4.82	1.51	0.65
CL	NA	$--$	$--$	0.93	0.74	0.68	#	#	#
	AD$_{\text{A}}$	$--$	$--$	2.92	0.90	#	#	#	#
	AD$_{\text{B}}$	$--$	$--$	2.32	#	#	3.13	#	#

Color codes (ranges of $s_{\varepsilon,n}$)				
#	$0-1$	$1-2$	$2-4$	$4-$

Table 4.9: Test case 1 (1D without spatial coarsening): speedup $s_{\varepsilon,n}$ for reaching a given relative error ε, for $n = N_{\Delta T}/2$ and $n = N_{\Delta T}$, and $\varepsilon \in \{10^{-2}, 10^{-3}, 10^{-4}\}$, for the parareal simulations combining the snapshot enrichment, PID and adaptive approaches. "ROM" and "CL" stand respectively for the ROM-based and the classical parareal method, and "NA" stands for "non-adaptive". The PID and enrichment configurations do not apply to the classical parareal. The relative errors at iteration 0 (given by the coarse propagator $\mathcal{G}_{\Delta t}$) are indicated in the top of the table. A color code corresponding to ranges of speedup $s_{\varepsilon,n}$ is defined. "#" indicates that the error ε is not reached within the performed iterations.

				$t = T/2$			$t = T$		
	Error at iteration 0			$e^0_{N_{\Delta T}/2} = 3.79\text{E-}2$			$e^0_{N_{\Delta T}} = 4.72\text{E-}2$		
	ε			1E-2	5E-3	1E-3	1E-2	5E-3	1E-3
	Adaptive	N_{PID}	$1/\alpha_{\text{s}}$						
ROM	NA	1	1	1	4	6	7	8	#
			2	1	2	4	3	4	8
			5	1	2	4	3	4	8
		2	1	1	3	5	6	8	10
			2	1	2	4	3	5	7
			5	1	2	3	3	4	6
		4	1	2	3	5	6	8	#
			2	1	2	4	3	5	7
			5	1	2	4	3	4	6
	AD$_{\text{A}}$	1	1	4	6	8	10	#	#
			2	5	5	7	6	7	#
			5	4	5	7	6	8	10
		2	1	4	6	8	9	#	#
			2	4	4	6	6	7	10
			5	4	4	7	6	7	#
		4	1	4	6	8	9	#	#
			2	4	5	7	6	7	10
			5	4	4	7	5	7	10
	AD$_{\text{B}}$	1	1	7	7	#	10	#	#
			2	7	8	10	8	10	#
			5	7	8	10	8	10	#
		2	1	7	7	10	7	#	#
			2	7	7	10	7	9	#
			5	7	7	9	7	10	#
		4	1	7	7	#	7	10	#
			2	7	7	10	7	9	#
			5	7	7	9	7	9	#
CL	NA	−−	−−	4	4	9	5	7	#
	AD$_{\text{A}}$	−−	−−	5	6	9	7	8	#
	AD$_{\text{B}}$	−−	−−	8	9	#	9	10	#

Color codes (ranges of $k_{\varepsilon,n}$)				
#	$7-10$	$5-6$	$3-4$	$1-2$

Table 4.10: Test case 2 (2D): number of iterations $k_{\varepsilon,n}$ for reaching a given relative error ε, for $n = N_{\Delta T}/2$ and $n = N_{\Delta T}$, and $\varepsilon \in \{10^{-2}, 5 \times 10^{-3}, 10^{-3}\}$, for the parareal simulations combining the snapshot enrichment, PID and adaptive approaches. "ROM" and "CL" stand respectively for the ROM-based and the classical parareal method, and "NA" stands for "non-adaptive". The PID and enrichment configurations do not apply to the classical parareal. The relative errors at iteration 0 (given by the coarse propagator $\mathcal{G}_{\Delta t}$) are indicated in the top of the table. A color code corresponding to ranges of $k_{\varepsilon,n}$ is defined. "#" indicates that the error ε is not reached within the performed iterations.

				$t = T/2$			$t = T$		
	Error at iteration 0			$e^0_{N_{\Delta T}/2} = 3.79\text{E-}2$			$e^0_{N_{\Delta T}} = 4.72\text{E-}2$		
	ε			1E-2	5E-3	1E-3	1E-2	5E-3	1E-3
	Adaptive	N_{PID}	$1/\alpha_{\text{s}}$						
ROM	NA	1	1	1.39	0.68	0.46	0.69	0.62	#
			2	1.2	0.80	0.48	0.96	0.79	0.51
			5	1.08	1.08	0.51	1.39	0.80	0.44
		2	1	1.37	0.67	0.44	0.89	0.65	#
			2	1.23	1.23	0.58	1.17	0.92	0.48
			5	2.16	1.11	0.38	1.00	0.76	0.42
		4	1	1.39	0.68	0.54	0.75	0.75	0.51
			2	2.34	0.81	0.59	1.18	0.92	0.54
			5	2.16	0.68	0.49	1.37	0.75	0.43
	AD$_{\text{A}}$	1	1	4.88	1.00	0.56	0.80	#	#
			2	1.51	1.51	0.68	1.90	1.36	#
			5	3.33	1.27	0.58	1.61	0.89	0.58
		2	1	4.32	1.01	0.57	0.93	#	#
			2	3.81	3.81	0.89	1.79	1.28	0.68
			5	2.92	2.92	0.53	1.49	1.06	#
		4	1	3.65	0.93	0.54	0.88	#	#
			2	3.44	1.34	0.63	1.72	1.26	0.66
			5	2.88	2.88	0.56	2.44	1.13	0.56
	AD$_{\text{B}}$	1	1	3.55	3.55	#	1.69	#	#
			2	3.02	2.25	0.75	4.49	1.50	#
			5	1.99	1.47	0.60	2.94	1.19	#
		2	1	2.96	2.96	0.82	5.93	#	#
			2	2.68	2.68	0.75	5.35	2.10	#
			5	1.81	1.81	0.80	3.62	1.17	#
		4	1	2.55	2.55	#	5.10	1.54	#
			2	2.29	2.29	0.68	4.59	1.84	#
			5	1.71	1.71	0.77	3.41	1.54	#
CL	NA	−−	−−	2.71	2.07	0.97	3.37	2.46	#
	AD$_{\text{A}}$	−−	−−	4.13	2.79	1.41	4.23	3.38	#
	AD$_{\text{B}}$	−−	−−	5.09	3.11	#	6.23	4.54	#

Color codes (ranges of $s_{\varepsilon,n}$)				
#	$0-1$	$1-2$	$2-4$	$4-$

Table 4.11: Test case 2 (2D): speedup $s_{\varepsilon,n}$ for reaching a given relative error ε, for $n = N_{\Delta T}/2$ and $n = N_{\Delta T}$, and $\varepsilon \in \{10^{-2}, 5\times10^{-3}, 10^{-3}\}$, for the parareal simulations combining the snapshot enrichment, PID and adaptive approaches. "ROM" and "CL" stand respectively for the ROM-based and the classical parareal method, and "NA" stands for "non-adaptive". The PID and enrichment configurations do not apply to the classical parareal. The relative errors at iteration 0 (given by the coarse propagator $\mathcal{G}_{\Delta t}$) are indicated in the top of the table. A color code corresponding to ranges of speedup $s_{\varepsilon,n}$ is defined. "#" indicates that the error ε is not reached within the performed iterations.

CHAPTER 5

STRATEGIES FOR LONGER PARAREAL SIMULATIONS

Contents

5.1 Introduction . . . 129

5.2 Motivating numerical examples . . . 130

5.3 Choice of the length of time slices . . . 132

5.4 Local-in-time parareal . . . 136

5.4.1 Proposed approach . . . 136

5.4.2 Speedup estimation . . . 137

5.4.3 Numerical examples . . . 139

5.5 Conclusion of the chapter . . . 144

5.1 Introduction

In Chapter 3, an initial comparison of the performances of the classical and ROM-based parareal was performed; in Chapter 4, modifications of the ROM-based method were presented, and, to a greater or lesser extent, they showed to be able to improve the stability and convergence of the method. In both chapters, the study was performed considering relatively simple test cases, defined in small spatial and temporal domains. All simulations considered $N_{\Delta T} = N_{\Delta t} = 20$ time slices, *i.e.* each parallel processor was responsible for performing the parallel fine correction step along one coarse time step Δt, which was convenient since $N_p = 20$ parallel processors were considered for the simulations.

In this chapter, we will see that, in the case of more challenging simulations, performed within more time steps and in larger spatial domains, convergence and stability are harder to obtain. Notably, the proposed improvements to the ROM-based parareal method may not be enough to obtain good results. Indeed, most of the error figures presented in the previous chapter (*e.g.* Figures 4.4, 4.9, 4.16 and 4.22) indicate a degradation of the parareal solution near the end of the temporal domain. Motivated by these observations, we investigate in this chapter two strategies for performing larger parareal simulations in time, and we compare their impact on the performance of the classical and ROM-based parareal methods.

The first strategy is to consider larger time slices, with each slice containing several coarse time steps, *i.e.* $\Delta T > \Delta t$. By increasing ΔT, we reduced the number $N_{\Delta T}$ of time slices, and the convergence is expected to be faster, since the parareal solution converges exactly towards the reference one in at most $N_{\Delta T}$ iterations, as discussed in Section 3.2.4. Moreover, as observed by Ruprecht (2018), the parareal convergence is faster when the number of time slices decreases, mainly for high wavenumbers.

The second strategy is to perform local-in-time applications of the parareal method, *i.e.* instead of applying the method for solving the problem in the entire temporal domain, we solve a sequence of parareal simulations defined in smaller time windows, which we expect to present a more stable behaviour.

In this approach, the solution of each "local parareal" is used as initial condition for the next ones. We remark that a similar approach is proposed by Eghbal et al. (2017) and referred as *windowing technique*, formulated in the framework of the Navier-Stokes for controlling parareal instabilities observed with high Reynolds number (which corresponds to an advection-dominated problem). This approach is also used by Nielsen et al. (2018) for solving the two-dimensional nonlinear shallow water equations with the parareal method in an HPC environment. In their implementation, faster convergence is obtained by dividing the simulation in time windows, and the speedup is improved by estimating their optimal length and by proposing a scheduling of parallel tasks allowing the concurrent computation of more than one local parareal simulation.

If one does not consider the possible beneficial effects on the parareal convergence, both the use of large time slices and parareal simulations defined in time windows present obvious drawbacks concerning the possible parallelization that can be achieved. In the first approach, the number of time slices to be distributed among parallel processors is smaller. If we consider a fixed number N_p of available processors, this is not an issue while $N_{\Delta T} \geq N_p$, but otherwise, there will be an underuse of these available parallel resources. In the local-in-time approach, the expected parallelism efficiency decreases, since the algorithm is decomposed into sequential parts, each one containing a smaller number of time slices to be parallelized. Moreover, additional computational costs due to the initialization of each parareal simulation can possibly be important. Note that this second strategy also poses a convergence limitation to the method: if a local parareal receives the solution of a previous one that has not converged exactly to the fine, reference solution, it will not be able to converge neither, since the initial conditions are not exact. However, if each local parareal is able to reasonably reduce the error w.r.t. the reference model, one could a expect a good global parareal solution, with a good speedup provided by parallelism.

This chapter is organized as follows: in Section 5.2, we present some motivating numerical examples, which are the same of the previous chapters, but using larger spatial and temporal domains, and we show that convergence and stability become more challenging. In Sections 5.3 and 5.4, we study respectively the influence of the time slice lengths and the local-in-time approach, both in the classical and ROM-based frameworks. Detailed considerations on the speedup impact of defining local-in-time parareal simulations are presented. A conclusion is made in Section 5.5.

5.2 Motivating numerical examples

In order to motivate the work presented in this chapter, we illustrate the limitations of the proposed modifications to the ROM-based parareal method by simulating Test case 1 (pseudo-2D without spatial coarsening) and Test case 2 (2D) presented previously in this work, but in larger temporal and spatial domains. For evidencing the relation to the previous simulations, these modified tests are named here as Test cases 1.1 and 2.1, respectively. We make the simulations considering parareal configurations that presented relatively good convergence and speedup behaviour, as shown in Tables 4.8-4.11, in order to evidence that less satisfactory results are obtained in these modified test cases.

Test case 1.1 (pseudo-2D without spatial coarsening in larger spatial and temporal domains)

We modify Test case 1 by considering a 5 times larger spatial domain in the $x-$direction (along which the flow propagates). The temporal domain is also taken 5 times larger than before. The mesh sizes and time steps for the fine and coarse propagators are not modified. These and other basic parareal configurations are summarised in Table 5.1.

Note that, under these configurations, the solution profiles in Test cases 1 and 1.1 are similar, differing only for their position in time and along the $x-$direction. We consider the ROM-based parareal method with $\alpha_\mathrm{s} = 1/2$, $N_\mathrm{PID} = 1$ and configuration $\mathrm{AD_A}$ for the adaptive approach (see Tables 4.4 and 4.3). As shown in Tables 4.8-4.9, this combined configuration presented a good behaviour in terms of convergence and speedup. This is no longer the case when the same configurations are used for solving Test case 1.1, with instabilities arising in the very last time steps of the first parareal iterations, as shown in Figure 5.1. This unstable behaviour rapidly evolves and produces negative water depths that terminate the execution at the second parareal iteration.

Spatial domain	$\Omega = [0, 100] \times [0, 20]$	
Maximum simulation time	$T = 20$	
Length of each time slice	$\Delta T = 0.2$	
Number of time slices	$N_{\Delta T} = 100$	
Number of parallel processors	$N_p = 20$	
	$\mathcal{F}_{\delta t}$	$\mathcal{G}_{\Delta t}$
Time step	$\delta t = 0.001$	$\Delta t = 0.2$
Mesh size (x-direction)	$\delta x = 1$	$\Delta x = 1$
Mesh size (y-direction)	$\delta y = 1$	$\Delta y = 1$

Table 5.1: Configurations of Test case 1.1 (pseudo-2D without spatial coarsening in larger spatial and temporal domains)

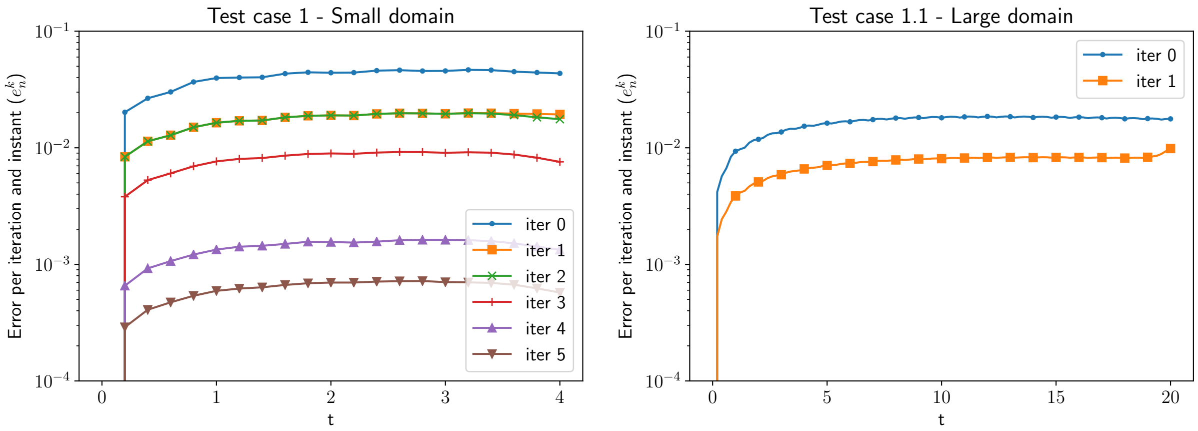


Figure 5.1: Test cases 1 (left) and 1.1 (right): evolution of the relative error e_n^k. Both tests use the same parareal configurations ($\alpha = 1/2$, $N_{\text{PID}} = 1$, adaptive configuration AD_A). Test case 1.1 is performed in larger spatial and temporal domains and does not complete two iterations due to instabilities.

Test case 2.1 (2D in larger spatial and temporal domains)

This second motivating test case is related to Test case 2, also studied in details for various parareal configurations in Chapter 4. We modify it by doubling the spatial domain size in both directions, as well as the total simulation time. The initial and boundary conditions, mesh sizes and time steps of the fine and coarse propagators are maintained. Table 5.2 presents the basic parareal configurations. We consider the ROM-based parareal method with $\alpha_\text{s} = 1/2$, $N_{\text{PID}} = 2$ and adaptive configuration AD_B (see Tables Tables 4.6 and 4.4), which provided relatively good convergence and speedup for Test case 2, as shown in Tables 4.10 and 4.11.

Spatial domain	$\Omega = [-50, 150]^2$	
Maximum simulation time	$T = 10$	
Length of each time slice	$\Delta T = 0.25$	
Number of time slices	$N_{\Delta T} = 40$	
Number of parallel processors	$N_p = 20$	
	$\mathcal{F}_{\delta t}$	$\mathcal{G}_{\Delta t}$
Time step	$\delta t = 0.001$	$\Delta t = 0.25$
Mesh size (x-direction)	$\delta x = 2$	$\Delta x = 5$
Mesh size (y-direction)	$\delta y = 2$	$\Delta y = 5$

Table 5.2: Configurations of Test case 2.1 (2D in larger spatial and temporal domains)

Figures 5.2 and 5.3 compare respectively the evolution of the relative errors e_n^k and $\overline{e}^k$, for Test cases 2 and 2.1 under the same parareal configurations. The conclusions are the same as for the one-dimensional test case. Even if in Test case 2.1 the parareal simulation is able to complete all iterations, instabilities are observed for advanced time steps and the error presents a slower decrease across iterations, compared to Test case 2.

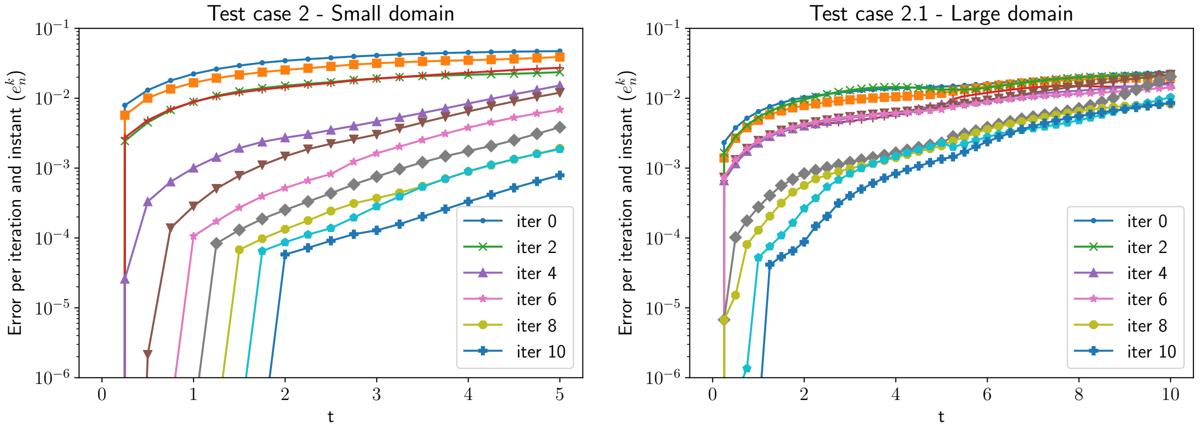


Figure 5.2: Test cases 2 (left) and 2.1 (right): evolution of the relative error e_n^k. Both tests use the same parareal configurations ($\alpha_\mathrm{s} = 1/2$, $N_\mathrm{PID} = 2$, adaptive configuration $\mathrm{AD_B}$). Test case 2.1 is performed in larger spatial and temporal domains.

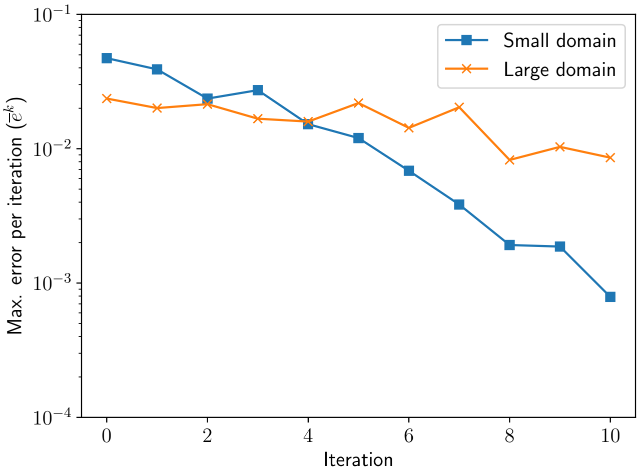


Figure 5.3: Test cases 2 and 2.1: evolution of the relative error $\overline{e}^k$. Both tests use the same parareal configurations ($\alpha_\mathrm{s} = 1/2$, $N_\mathrm{PID} = 2$, adaptive configuration $\mathrm{AD_B}$). Test case 2.1 is performed in larger spatial and temporal domains.

5.3 Choice of the length of time slices

We begin by studying the influence of the length of the time slices on the performance of the classical and ROM-based parareal methods. It is a more natural approach, since it is simply a choice of parameters for the parareal simulation. An initial study of the influence of the time slice length was already performed in Chapter 3, and showed that larger time slices presented a positive influence mainly for the classical parareal method. This study was quite limited in that case, since it was performed on small simulations, containing few coarse time steps, thus limiting the range of time slice lengths. As discussed in that chapter, the increase of ΔT may reduce the expected possible parallelism and lead to an underuse of the available parallel capacity (see eq. (3.55)). Therefore, the use of an adaptive approach should be envisaged for ensuring reasonable speedups. It is considered in the examples present below. For both Test cases 1.1 and 2.1, we present two sets of simulations, one for the classical and another for the ROM-based parareal method, using different lengths of time slices. In all simulations, we set $N_{\text{itermax}} = 10$.

Test case 1.1 (pseudo-2D without spatial coarsening in larger spatial and temporal domains)

For the one dimensional test case, we consider the following parareal configurations:

- Classical parareal method with adaptive configuration AD_A;
- ROM-based parareal method with $\alpha_\text{s} = 1/2$, $N_{\text{PID}} = 1$ and adaptive configuration AD_A.

Note that the configuration for the ROM-based method is the same used in the motivating example presented in Section 5.2, which quickly lead to an unstable behaviour (Figure 5.1).

In each case, we perform simulations with $\Delta T \in \{0.2, 0.4, 1, 2\}$, corresponding respectively to $N_{\Delta T} \in \{100, 50, 20, 10\}$. We remark that $\Delta T = 0.2$ is the smallest possible value for $N_{\Delta T}$, since $\Delta t = 0.2$. Also, since we consider $N_p = 20$ parallel processors, only half of the available parallel capacity is used in the case $\Delta T = 2$ ($N_{\Delta T} = 10$).

Figure 5.4 presents the evolution of the maximum error per iteration for each simulation. We observe, in the framework of the classical parareal method, a clear improvement of the convergence in function of the length of the time slices. For smaller ΔT, instabilities are present and lead to increasing errors, a similar behaviour to the one observed in Test case 1, presented in Chapter 3. For the smallest ΔT, the simulation is not able to complete $N_{\text{itermax}} = 10$ iterations. On the other hand, for larger ΔT, the error decreases monotonically, and a relatively fast convergence is obtained for $\Delta T = 2$.

In the ROM-based simulations, we observe similar results but with a less clear influence of the number of time slices. For $\Delta T = 0.2$ and $\Delta T = 0.4$, the simulations are unstable and only complete few parareal iterations. For larger time slices, a better convergence is observed, but the error behaviour still indicates the presence of instabilities. For advanced iterations, the convergence stagnates for $\Delta T = 2$, and better results, with errors close to the one observed in the best classical parareal simulation, are obtained for $\Delta T = 1$.

Figure 5.5 presents the evolution of the computational time for each simulation, and the speedups at iterations $k = 5$ and $k = N_{\text{itermax}} = 10$ are indicated in Table 5.3. Note that the adaptive approach ensures relatively large speedups, at least in the beginning of the simulations. Indeed, in the chosen adaptive configuration AD_A, each intermediate fine propagator is in general used along one iteration (due to the large threshold $\varepsilon_{\text{ad}} = 10^{-2}$), and the reference model is used only from $k = 5$. Thus, for the classical parareal method, all simulations are faster than the reference one after ten iterations, even in the case $\Delta T = 2$, for which ten non-adaptive iterations would necessarily lead to exact convergence and a speedup smaller than the unity. In the ROM-based case, the adaptive approach also allows to obtain larger speedups, but the computational cost remains a major issue, with speedups smaller than the unity at the last iteration.

	Classical		ROM-based	
	$s(5)$	$s(10)$	$s(5)$	$s(10)$
$\Delta T = 0.2$	5.64	*	*	*
$\Delta T = 0.4$	7.12	2.01	*	*
$\Delta T = 1$	9.94	2.66	2.80	0.79
$\Delta T = 2$	6.32	1.53	2.48	0.68

Table 5.3: Test case 1.1: speedup at the fifth and tenth iterations for the classical and ROM-based parareal methods, for various time slice lengths. Asterisks indicate that the simulation is unstable and does not reach the respective iteration.

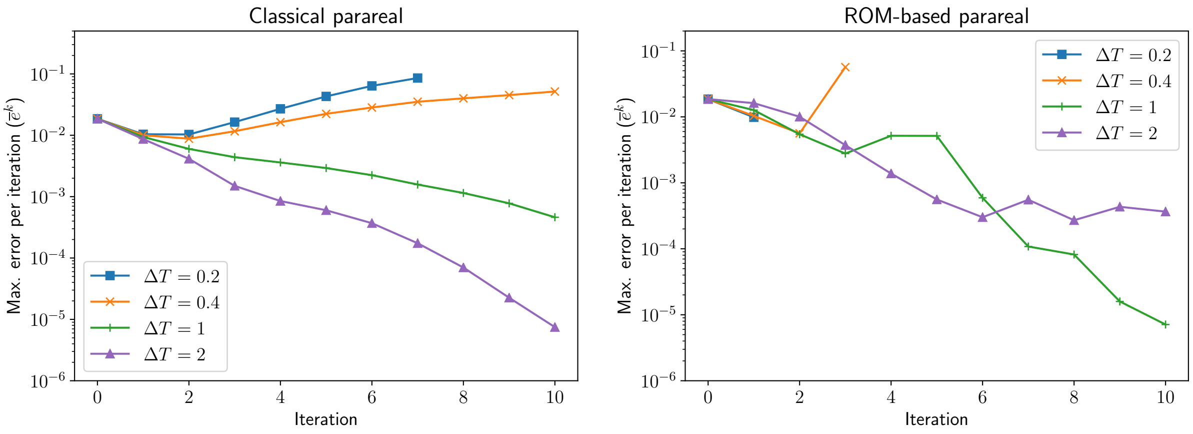


Figure 5.4: Test case 1.1: evolution of the relative error $\bar{e}^k$ using the classical (left) and ROM-based (right) parareal methods for various time slice lengths. Some simulations do not complete $N_{\text{itermax}} = 10$ iterations due to instabilities.

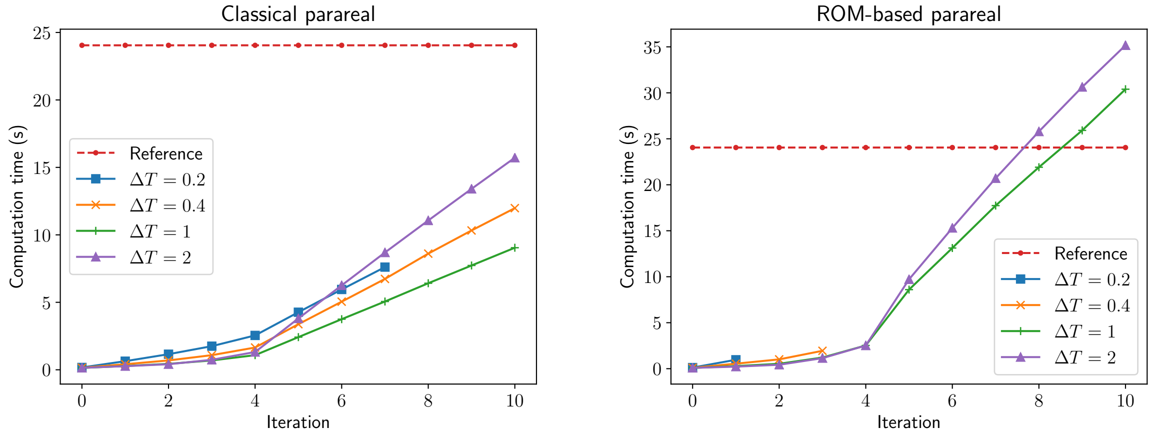


Figure 5.5: Test case 1.1: evolution of the computational times (in seconds) using the classical (left) and ROM-based (right) parareal methods for various time slice lengths. The dashed, horizontal line indicates the computational time for the reference simulation. Some simulations do not complete $N_{\text{itermax}} = 10$ iterations due to instabilities.

Test case 2.1 (2D in larger spatial and temporal domains)

As in the previous test case, we perform, for Test case 2.1, simulations using both the classical and the ROM-based framework. We choose configurations that performed relatively well in the smaller Test case 2, as shown in Tables 4.10-4.11:

- Classical parareal method with adaptive configuration AD_B;
- ROM-based parareal with $\alpha_\text{s} = 1/2$, $N_\text{PID} = 2$ and adaptive configuration AD_B.

where the second configuration is the same used in the motivating example presented in Section 5.2. The lengths of the time slices are set to $\Delta T \in \{0.25, 0.5, 1, 2.5\}$, corresponding respectively to $N_{\Delta T} \in \{40, 20, 10, 4\}$, with $\Delta T = 0.25$ as the smallest possible value. The available parallel resources ($N_p = 20$) are underused for $\Delta T = 1$ and $\Delta T = 2.5$.

Figure 5.6 presents the evolution of the maximum error per iteration for each simulation. The conclusions are the same as in the one-dimensional test case. A clearly faster convergence is observed in the classical parareal method when the time slices are larger; improvements are also observed in the ROM-based case, but less clearly and remarkably. In both cases, smaller errors are obtained for $\Delta T = 2.5$, for which $N_{\Delta T} = 4$, *i.e.* exact convergence should be expected in at most four iterations, and the speedup should decrease following $4/k$. The use of an adaptive approach with the intermediate fine propagators used in more than one iteration (configuration $\mathrm{AD_B}$, with a relatively small threshold $\varepsilon_{\mathrm{ad}} = 10^{-3}$), provide a relatively good performance in terms of speedup, as shown in Figure 5.7 and Table 5.4.

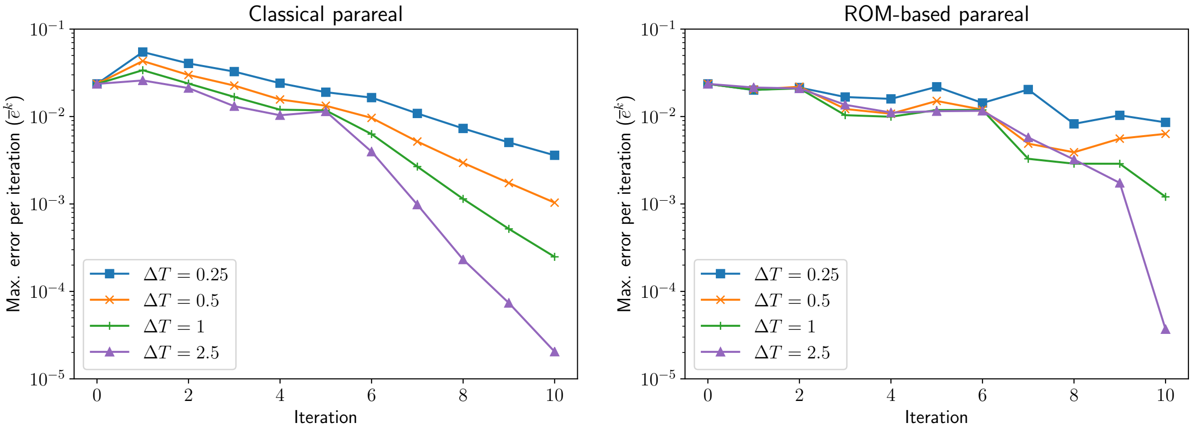


Figure 5.6: Test case 2.1: evolution of the relative error $\bar{e}^k$ using the classical (left) and ROM-based (right) parareal methods for various time slice lengths.

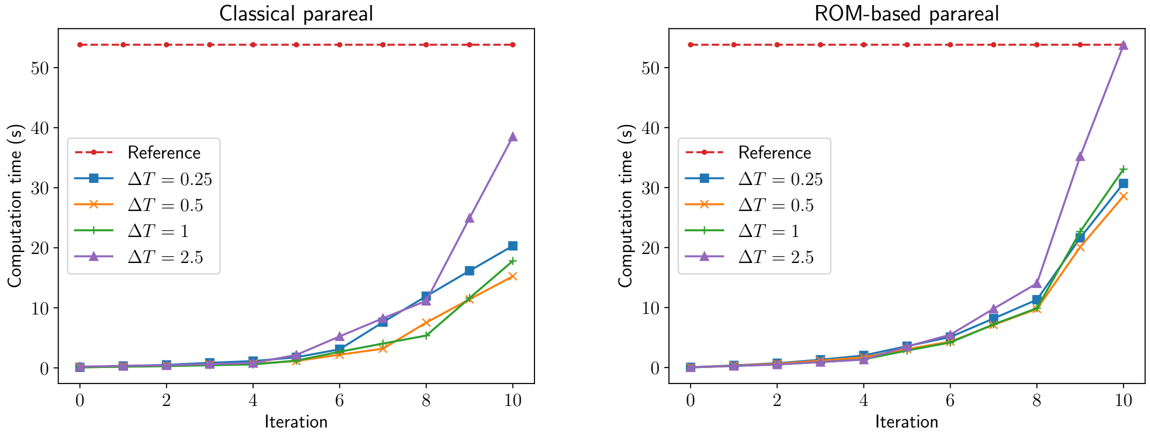


Figure 5.7: Test case 2.1: evolution of the computational times (in seconds) using the classical (left) and ROM-based (right) parareal methods for various time slice lengths. The dashed, horizontal line indicates the computational time for the reference simulation.

	Classical		ROM-based	
	$s(5)$	$s(10)$	$s(5)$	$s(10)$
$\Delta T = 0.25$	30.57	2.65	15.08	1.75
$\Delta T = 0.5$	47.21	3.53	17.48	1.88
$\Delta T = 1$	44.23	3.02	19.01	1.63
$\Delta T = 2.5$	24.92	1.40	15.33	1.00

Table 5.4: Test case 2.1: speedup at the fifth and tenth iterations for the classical and ROM-based parareal methods, for various time slice lengths.

5.4 Local-in-time parareal

In this second approach, we investigate the sequential execution of parareal simulations defined in smaller time windows, *i.e.* we propose to divide the temporal domain into a given number of non-overlapping time windows and execute a full parareal simulation in each of them, providing the initial solution to the next time window. We refer to this approach as “Local-in-Time Parareal”, or, more concisely, “LTP”. Analogously, the parareal method executed in the entire temporal domain is referred to as “global-in-time” parareal. In the following paragraphs, we briefly describe the proposed strategy (whose definitions are closely related to the PID, described in Section 4.4). We also define the global solution provide by this local-in-time approach, and we propose a more detailed discussion of its impacts on the expected speedup.

5.4.1 Proposed approach

We define N_{LTP} time windows $\breve{\mathbb{T}}^{(i)} := [\breve{t}_{i-1}, \breve{t}_i]$, $i = 1, \dots, N_{\text{LTP}}$. The times $\breve{t}_i, i = 0, \dots, N_{\text{LTP}}$ are parareal time instants, *i.e.* $\{\breve{t}_i, i = 0, \dots, N_{\text{LTP}}\} \subset \{t_i, i = 0, \dots, N_{\Delta T}\}$, with $\breve{t}_0 = 0$ and $\breve{t}_{N_{\text{LTP}}} = T$. We also define $\breve{n}_i, i = 0, \dots, N_{\text{LTP}}$, as the global indices (relative to the parareal time slices) of the times defining the LTP windows, *i.e.* $\breve{t}_i = t_{\breve{n}_i}$ with $\breve{n}_0 = 0$ and $\breve{n}_{N_{\text{LTP}}} = N_{\Delta T}$. Figure 5.8 illustrates these definitions. Note that the time windows can have different lengths.

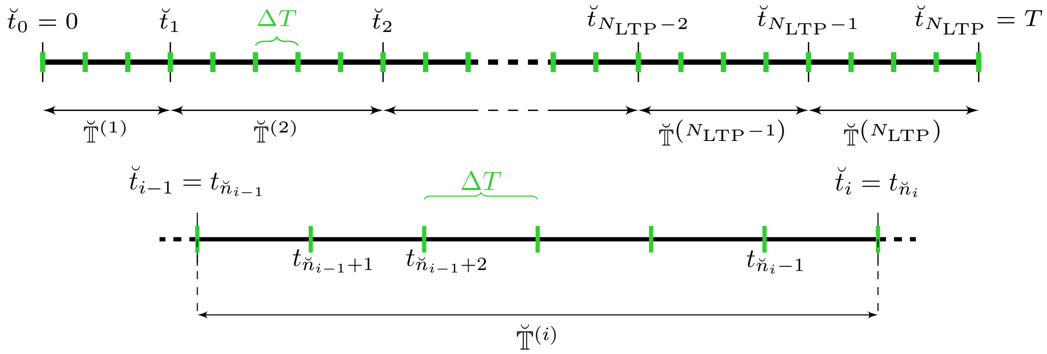

$\breve{t}_0 = 0$
$\breve{t}_1$
ΔT
$\breve{t}_2$
$\breve{t}_{N_{\text{LTP}}-2}$
$\breve{t}_{N_{\text{LTP}}-1}$
$\breve{t}_{N_{\text{LTP}}} = T$
$\breve{\mathbb{T}}^{(1)}$
$\breve{\mathbb{T}}^{(2)}$
$\breve{\mathbb{T}}^{(N_{\text{LTP}}-1)}$
$\breve{\mathbb{T}}^{(N_{\text{LTP}})}$
$\breve{t}_{i-1} = t_{\breve{n}_{i-1}}$
ΔT
$\breve{t}_i = t_{\breve{n}_i}$
$t_{\breve{n}_{i-1}+1}$
$t_{\breve{n}_{i-1}+2}$
$t_{\breve{n}_i-1}$
$\breve{\mathbb{T}}^{(i)}$

Figure 5.8: Definition of the LTP time windows. This definition is analogous to the PID windows (see Figure 4.13). Top: representation of several windows, with only the parareal times instants (green, vertical ticks, defining the parareal time slices) represented. Bottom: zoom over a time window containing six parareal time slices. Instants of the fine and coarse temporal discretizations are not represented for simplification.

We denote by

$$\begin{aligned} \mathcal{P}^{(i)} : \mathbb{R}^{d_f M} &\longrightarrow \mathbb{R}^{d_f M} \\ \boldsymbol{y} &\longmapsto \breve{\boldsymbol{y}}^{(i)} := \mathcal{P}^{(i)}(\boldsymbol{y}) \end{aligned} \tag{5.1}$$

a parareal instance defined in $\breve{\mathbb{T}}^{(i)} = [\breve{t}_{i-1}, \breve{t}_i]$ that starts with an initial solution $\boldsymbol{y} \in \mathbb{R}^{d_f M}$ defined at $\breve{t}_{i-1}$ and, after a given number of iterations, returns a solution $\breve{\boldsymbol{y}}^{(i)} := \mathcal{P}^{(i)}(\boldsymbol{y}) \in \mathbb{R}^{d_f M}$ defined at $\breve{t}_i$. We recall that d_f is the number of degrees-of-freedom for each one of the M computational cells of the discrete spatial domain ($d_f = 3$ in the case of the SWE). For denoting the solution of $\mathcal{P}^{(i)}$ along time and iterations, we keep the notation using the global indices of the temporal discretization. Then, $\boldsymbol{y}_n^{(i),k}$ is an approximation for $\boldsymbol{y}(t_n)$ at the $k-$th iteration of $\mathcal{P}^{(i)}$, with $t_n \in \breve{\mathbb{T}}^{(i)} = [t_{\breve{n}_{i-1}}, t_{\breve{n}_i}]$; and the initial solution for $\mathcal{P}^{(i)}$ is $\boldsymbol{y}_{\breve{n}_{i-1}}^{(i),0}$.

Eq. (5.1) is a simplified definition of the parareal instances, pointing out only its initial solution. Indeed, one should include the several parameters on which the parareal simulation depends and write $\mathcal{P}^{(i)} = \mathcal{P}^{(i)}(\boldsymbol{y}, N_{\text{itermax}}, \text{Type}, \varepsilon_{\text{TOL}}, \varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}, N_{\text{PID}}, N_{\text{ad}}, \dots)$, thus specifying the maximum number of iterations, the type of parareal simulation (classical or ROM-based), the convergence tolerance, the thresholds for the model reduction, the number of PID windows and the number of adaptive fine propagators, for listing a few examples of parameters. We remark that the N_{LTP} parareal instances do not necessarily use the same configurations, and different parameters could be considered depending on the behaviour of the solution within each time window. However, for the numerical simulations presented in this chapter, we consider that the same configuration are used by all parareal instances.

In this local-in-time parareal approach, each parareal instance $\mathcal{P}^{(i)}$ provides the initial solution for the next one, $\mathcal{P}^{(i+1)}$. It means that $\boldsymbol{y}_{\breve{n}_i}^{(i+1),0} = \boldsymbol{y}_{\breve{n}_i}^{(i),N_{\text{itermax}}} =: \breve{\boldsymbol{y}}^{(i)}$, where $\breve{\boldsymbol{y}}^{(i)} := \boldsymbol{y}_{\breve{n}_i}^{(i),N_{\text{itermax}}}$ is the final solution of $\mathcal{P}^{(i)}$, after N_{itermax} iterations, and $\boldsymbol{y}_{\breve{n}_i}^{(i+1),0}$ is the initial solution for $\mathcal{P}^{(i+1)}$, both defined at the intersecting time $\breve{t}_i = t_{\breve{n}_i}$. We remark that, instead of setting a maximum number of iterations N_{itermax}, the transition between two parareal instances can also be decided based on a parareal residual (as considered by Nielsen et al. (2018)), which would be more suitable for a practical implementation. We consider here, for simplification, that each parareal instance performs a fixed number N_{itermax} of iterations, and we perform a study of the convergence and speedup in function of this value.

Finally, we define the global parareal solution $\boldsymbol{y}_n^k, n = 0, \dots, N_{\Delta T}$ provided by the local-in-time approach at iteration $1 \leq k \leq N_{\text{itermax}}$ as the union of the solutions of each parareal instance at iteration k. In order to maintain the coherence between LTP and the global-in-time parareal methods, the solution at iteration $k = 0$ is still defined as the solution of the coarse propagator $\mathcal{G}_{\Delta t}$ simulated sequentially in the full temporal domain. Thus,

$$\boldsymbol{y}_n^k := \begin{cases} \mathcal{G}_{\Delta t}(\boldsymbol{y}_0, t_n, 0), & k = 0 \\ \sum_{i=0}^{N_{\text{LTP}}} \mathbb{1}_{\{t_n \in \,]\breve{t}_{i-1}, \breve{t}_i]\}} \boldsymbol{y}_n^{(i),k}, & k = 1, \dots, N_{\text{itermax}} \end{cases}, \qquad n = 1, \dots, N_{\Delta T}$$

where $\boldsymbol{y}_0 =: \boldsymbol{y}_0^k, k = 0, \dots, N_{\text{itermax}}$, is the global initial solution, defined at t_0.

Two main drawbacks can be identified in this-local-time approach. Firstly, it partially serializes the parareal algorithm. The parallel resources available for the simulation may be underused depending on the number of processors and LTP windows, which is discussed in details in Section 5.4.2. Secondly, this approach imposes a limitation for the convergence towards the reference solution. If the parareal simulation in $\mathcal{P}^{(i)}$ does not converge exactly, the next parareal instance $\mathcal{P}^{(i+1)}$, which uses the solution of the previous one as initial condition, will not converge either, since a different problem w.r.t. the reference one is solved. The numerical tests presented in Section 5.4.3 are used to evaluate the impact of these limitations.

Differences between the PID and LTP approaches

Note that the local-in-time approach is very similar to the PID-ROM parareal method, presented in Section 4.4. Their difference consists in the fact that, in the PID-based method, only the reduced-order models are formulated within time windows, being used for updating the full, global parareal solution at each iteration. In the LTP approach, this update is made only within each time window. Moreover, this last approach is applicable both for the classical and ROM-based methods, whereas the PID one is, by definition, only applicable in the framework using ROMs. Evidently, the PID model reduction can be performed within each time window $\breve{\mathbb{T}}^{(i)}$, *i.e.* $\breve{\mathbb{T}}^{(i)}$ can be decomposed into several smaller time windows, with a model reduction performed in each of them. For avoiding any confusion, when applying the LTP method with N_{PID} PID windows, it means that there are N_{PID} intervals per time window $\breve{\mathbb{T}}^{(i)}$.

5.4.2 Speedup estimation

For the sake of simplicity, we consider that all N_{LTP} time windows have the same length. Therefore, if the global parareal simulation contains $N_{\Delta T}$ time slices, $N_{\Delta t}$ coarse and $N_{\delta t}$ fine time steps, then each local parareal instance will be solved within $N_{\Delta T}/N_{\text{LTP}}$ time slices, $N_{\Delta t}/N_{\text{LTP}}$ coarse and $N_{\delta t}/N_{\text{LTP}}$ fine time steps. In the adaptive parareal approach, the number of time steps associated to each intermediate fine propagator is also scaled by $1/N_{\text{LTP}}$.

We place ourselves in the framework of the classical parareal method in order to develop the main ideas concerning the speedup provided by the LTP approach. By considering that the sequential, coarse prediction step has a negligible computational cost compared to the fine correction and that the number $N_{\Delta T}$ of time slices (in the entire temporal domain) is a multiple integer of the number N_p of processors, then the computational time for completing $\hat{k}$ iterations of the global, classical parareal method (eq. (3.10)) and the speedup (eq. (3.11)) simplify respectively to

$$T_{\text{classical-parareal}}(\hat{k}) \approx \hat{k} \frac{N_{\delta t}}{N_p} \tau_f \tag{5.2}$$

$$s_{\text{classical-parareal}}(\hat{k}) \approx \frac{1}{\frac{\hat{k}}{N_p}} = \frac{N_p}{\hat{k}} \tag{5.3}$$

For the local-in-time parareal method, each parareal instance is parallelized across the $N_{\Delta T}/N_{\text{LTP}}$ time slices contained in the time window. If $N_{\Delta T}/N_{\text{LTP}} < N_p$, the simulation is distributed among only $N_{\Delta T}/N_{\text{LTP}}$ processors, meaning that only a fraction of the parallel resources is used. Therefore, the computational time for completing $\hat{k}$ iterations of the LTP (*i.e.* for completing $\hat{k}$ iterations in each parareal instance) reads

$$T_{\text{classical-parareal-LTP}}(\hat{k}) = N_{\text{LTP}} \left[\hat{k} \frac{N_{\delta t}/N_{\text{LTP}}}{\min\left(N_p, N_{\Delta T}/N_{\text{LTP}}\right)} \tau_f \right] + c_{\text{LTP}}$$

where c_{LTP} is an additional computational time taking into account the initialization of the parareal instances and the transmission of the solution between them. The initialization cost was neglected in the global parareal case but may become important if N_{LTP} is large. Then, the speedup in the LTP approach can be estimated as

$$s_{\text{classical-parareal-LTP}}(\hat{k}) = \frac{T_{\text{ref}}}{T_{\text{classical-parareal-LTP}}(\hat{k})} = \frac{N_{\delta t}\tau_f}{N_{\text{LTP}} \left[\hat{k} \frac{N_{\delta t}/N_{\text{LTP}}}{\min\left(N_p, N_{\Delta T}/N_{\text{LTP}}\right)} \tau_f \right] + c_{\text{LTP}}}$$

where $T_{\text{ref}} = N_{\delta t}\tau_f$ is the computational time for performing the sequential, reference simulation (eq. (3.9)).

Therefore,

$$s_{\text{classical-parareal-LTP}}(\hat{k}) = \frac{\hat{k}}{N_p} \frac{N_{\delta t}\tau_f}{N_{\text{LTP}} \left[\hat{k} \frac{N_{\delta t}/N_{\text{LTP}}}{\min\left(N_p, N_{\Delta T}/N_{\text{LTP}}\right)} \tau_f \right] + c_{\text{LTP}}} s_{\text{classical-parareal}}(\hat{k})$$

which implies that

$$s_{\text{classical-parareal-LTP}}(\hat{k}) < \frac{\min\left(N_p, N_{\Delta T}/N_{\text{LTP}}\right)}{N_p} s_{\text{classical-parareal}}(\hat{k}) \tag{5.4}$$

evidencing that the speedup of the LTP, compared to the global parareal, is limited by the underuse of the parallel resources. Notably, the case $N_{\text{LTP}} = N_{\Delta T}$ is equivalent to a serial execution of the fine, reference propagator (since the parareal instances are performed along single time slices). Also, the case in which each LTP window contains N_{itermax} time slices is equivalent to the reference simulation (since the exact convergence is obtained at most after N_{itermax} iterations in each LTP window), unless a faster convergence is obtained. As in the study of the influence of the length of time slices (Section 5.3), these remarks indicate that the use of the adaptive approach may be necessary for ensuring a reasonable speedup of the LTP, since it improves the bound $s_{\text{classical-parareal}}(\hat{k}) < N_p/\hat{k}$.

In the case of the of the ROM-based parareal method, the speedup estimation becomes more complex. Similarly to what was discussed in the speedup estimation of the PID-ROM parareal method (Section 4.4.3), the cost of the ROM formulations is a balance between a less expensive POD (due to its quadratic dependence on the number of snapshots, which becomes smaller by dividing the temporal domain into windows) and a more expensive DEIM (by assuming that the dimension of the subspaces are the same in the global- and local-in-time approaches), and possibly important costs may take place due to the larger number of ROM matrices to be computed. Moreover, even if the parallel resources are possibly underused in the parallel fine correction step of the parallel algorithm, all parallel processors can be used for performing the model-order reduction procedures. We then omit a speedup estimation for the LTP in this case, but we retain the main conclusions discussed above concerning N_p and $N_{\Delta T}/N_{\text{LTP}}$ for guiding the application of the method.

5.4.3 Numerical examples

Investigation of the computational time using the classical parareal method

We consider Test case 1.1, presented in Section 5.2 as a motivating example. We first implement the LTP approach in the framework of the non-adaptive, classical parareal method, *i.e.* each parareal instance $\mathcal{P}^{(i)}, i = 1, \dots, N_{\text{LTP}}$ corresponds to a simulation of the original parareal (Algorithm 1). We notice that the LTP approach is not able to improve stability and performance of this test case. However, we present these simulations for analysing the impact of the local-in-time approach on the computational times and the speedup, assessing the magnitude of the additional cost c_{LTP} and comparing with the estimates presented in Section 5.4.2, which were formulated for the classical parareal method.

We perform LTP simulations for N_{LTP} taking value in $\{1, 2, 4, 5, 10, 20\}$, with $N_{\text{LTP}} = 1$ corresponding to the global parareal. For all simulations, $N_p = 20$ parallel processors are used and the parareal time slices have length $\Delta T = \Delta t = 0.2$. Since $N_{\Delta T} = 100$, the simulations with $N_{\text{LTP}} > N_{\Delta T}/N_p = 5$ underuse the available parallel processors, as discussed in Section 5.4.2. For avoiding the interruption of the simulations due to instabilities, we consider $N_{\text{itermax}} = 2$ for all time windows. Finally, in each LTP simulation the time windows are homogeneous, having the same length.

Table 5.5 presents the speedup at the end of the simulation for the global parareal ($N_{\text{LTP}} = 1$) and for each LTP configuration. Contrary to the theoretical expectations, we observe some important reductions of the speedup between $N_{\text{LTP}} = 1$, $N_{\text{LTP}} = 2$ and $N_{\text{LTP}} = 4$, whereas $N_{\text{LTP}} = 5$ has a closer speedup to $N_{\text{LTP}} = 1$. This can be explained by the unbalanced load of parallel tasks in the parallel fine correction step of the parareal algorithm. Indeed, for $N_{\text{LTP}} = 1$, all $N_p = 20$ processors perform the fine simulation along the same quantity of time slices ($N_{\Delta T}/N_p = 5$). For $N_{\text{LTP}} = 5$ the same situation occurs: since each parareal instance has $N_{\Delta T}/N_{\text{LTP}} = 20$ time slices, each parallel processor computes the fine correction along one time slice. This is it not the case for $N_{\text{LTP}} = 2$ and $N_{\text{LTP}} = 4$, for which there are respectively $N_{\Delta T}/N_{\text{LTP}} = 50$ and $N_{\Delta T}/N_{\text{LTP}} = 25$ time slices per parareal instance, being unequally distributed among $N_p = 20$ processors. The effects of this unbalanced load are clear when we compare the computational times spent in the fine correction step, as shown in Table 5.6.

Apart from that, the results meet our expectations. Firstly, we observe close speedups and computational times for $N_{\text{LTP}} = 1$ and $N_{\text{LTP}} = 5$, which use the same parallel capacity, since $\min(N_p, N_{\Delta T}/N_{\text{LTP}}) = 20$ in both cases. These close results also show that the additional computational costs c_{LTP} for initializing the parareal instances have small magnitude, as indicated in Table 5.7: these additional costs represent less than 2% of the total simulation time for all values of N_{LTP}, and their average magnitude per LTP time window slightly decreases for larger N_{LTP}. Secondly, for $N_{\text{LTP}} = 10$ and $N_{\text{LTP}} = 20$, we expected increases of the computational time by factors of 2 and 4, respectively. It is approximately observed for the speedup and mainly for the computational time spent in the parallel fine correction step of the parareal algorithm (respectively in Tables 5.5 and 5.6). We remark that the speedups presented in Table 5.5 have no practical meaning concerning the parareal solution (since the simulations are unstable and do not provide good convergence) and are considered here only for analyzing the computational times and speedups in function of N_{LTP}.

N_{LTP}	1	2	4	5	10	20
$s(N_{\text{itermax}} = 2)$	6.87	5.84	4.86	6.70	4.07	2.24

Table 5.5: Test case 1.1: speedup at the final iteration ($N_{\text{itermax}} = 2$) for various numbers of parareal instances in the local-in-time (LTP) approach, in the classical parareal framework. $N_{\text{LTP}} = 1$ corresponds to the global parareal.

N_{LTP}	1	2	4	5	10	20
Comp. time - fine correction	1.23	1.49	1.97	1.23	2.47	4.84

Table 5.6: Test case 1.1: average computational time (in seconds) spent per iteration in the fine correction step for various numbers of parareal instances in the local-in-time (LTP) approach. $N_{\text{LTP}} = 1$ corresponds to the global parareal. For $N_{\text{LTP}} > 1$, the computational times are obtained the summing up the computational times of all the parareal instances.

N_{LTP}	1	2	4	5	10	20
$c_{\text{LTP}}/T_{\text{par}}$	$--$	1.08%	1.13%	1.51%	1.52%	1.28%
$c_{\text{LTP}}/N_{\text{LTP}}$	$--$	0.022	0.014	0.011	0.009	0.007

Table 5.7: Test case 1.1: percentage of the total simulation time (T_{par}) represented by the cost c_{LTP} inherent to the local-in-time parareal (LTP) approach (corresponding to the initialization of the parareal instance and transmission of the solution between them), and average cost c_{LTP} (in seconds) per LTP time window, for various numbers of parareal instances, in the classical parareal framework. c_{LTP} is computed taking into account only the additional costs due to the local-in-time approach, therefore it does not apply to $N_{\text{LTP}} = 1$, which corresponds to the global parareal.

Test case 1.1 (pseudo-2D without spatial coarsening in larger spatial and temporal domains)

After this study of the computational times, we implement the LTP approach to investigate its impacts on the parareal performance. We consider the same configurations as used in Section 5.3 for studying the influence of the length of the time slices on Test case 1.1. Note, however, that the number N_{itermax} of parareal iterations becomes a more important parareal parameter in the local-in-time approach. Indeed, since N_{itermax} iterations must be performed by each parareal instance, the value of N_{itermax} influences the quality of the initial solution transmitted to the next parareal instance, and thus the convergence that can be achieved by it. Therefore, a sufficiently large number of iterations should be considered. However, if N_{itermax} is too large, the expected speedup becomes smaller, since N_{itermax} iterations must be necessarily performed by each parareal instance (unless exact convergence is obtained within less iterations).

We evaluate this compromise relying on N_{itermax} by modifying the adaptive approach configuration AD_{A}. In most of the simulations using this configuration, the intermediate fine propagators are used in only one parareal iteration, since the adaptive criterion threshold $\varepsilon_{\text{ad}} = 10^{-2}$ is relatively large, and the reference propagator is used in all remaining ones. We then simplify AD_{A} by fixing the number of iterations in which each fine propagator is used, *i.e.* the transition between consecutive propagators does not depend on ε_{ad}, which is equivalent to set $\varepsilon_{\text{ad}} = 0$. We recall that $N_{\text{ad}} = 5$ fine propagators are used here (the reference and four intermediate ones). We force each intermediate propagator to be used in a single iteration, and we set $N_{\text{itermax}} = 5, 6, 10$, such that the reference model is used respectively in one, two and six iterations. These configurations are named respectively as AD_5, AD_6 and AD_{10} (where the subscript indicates N_{itermax}) and are summarised in Table 5.8.

Configuration	N_{itermax}	ε_{ad}	N^4_{itermax}	N^3_{itermax}	N^2_{itermax}	N^1_{itermax}	N^0_{itermax}
AD_5	5	0	1	2	3	4	5
AD_6	6	0	1	2	3	4	6
AD_{10}	10	0	1	2	3	4	10

Table 5.8: Test case 1.1: configurations of the adaptive parareal method

We perform simulations for $N_{\text{LTP}} \in \{1, 2, 4, 5, 10, 20\}$, with homogeneous time windows. In simulations using adaptive configuration AD_{10}, $N_{\text{LTP}} = 20$ is not considered since it leads to exact convergence (since each time window contains $N_{\Delta T}/N_{\text{LTP}} = 100/20 = 5$ time slices and the reference propagator is used in six iterations).

The evolution of the maximum error per iteration is presented in Figure 5.9. For the classical parareal method, most of the simulations present instabilities at intermediate time windows. The configurations able to finish the simulation are those with the largest numbers of LTP time windows ($N_{\text{LTP}} = 10, 20$). These results reinforce the conclusions made in Section 5.3: the stability of the classical parareal method is closely related to the length ΔT of the time slices. Since these lengths are small in the simulations presented here ($\Delta T = \Delta t = 0.2$), instabilities rapidly arise in this test case. For larger values of N_{LTP}, the LTP time windows are smaller, and the local in time execution of parareal instances is able to control the error evolution in time.

Better results are obtained with the ROM-based parareal method, in which the simulations are able to complete all time windows for $N_{\text{LTP}} \geq 4$. Some instabilities are still observed in the last time window with $N_{\text{LTP}} = 4$, but they are no longer present for $N_{\text{LTP}} \geq 5$. Note that, contrary to what was observed in the study of the influence of the time slices (see Figure 5.4), we notice a clear improvement of the ROM-based method when the number of LTP time windows increases. It indicates that the method performs better when the reduced-order models are formulated and used for updating the parareal solution in smaller time domains. Also, we observe, as expected, an improvement of the convergence when more iterations are performed by each parareal instance, except in the case $N_{\text{LTP}} = 4$, due to the still present instabilities.

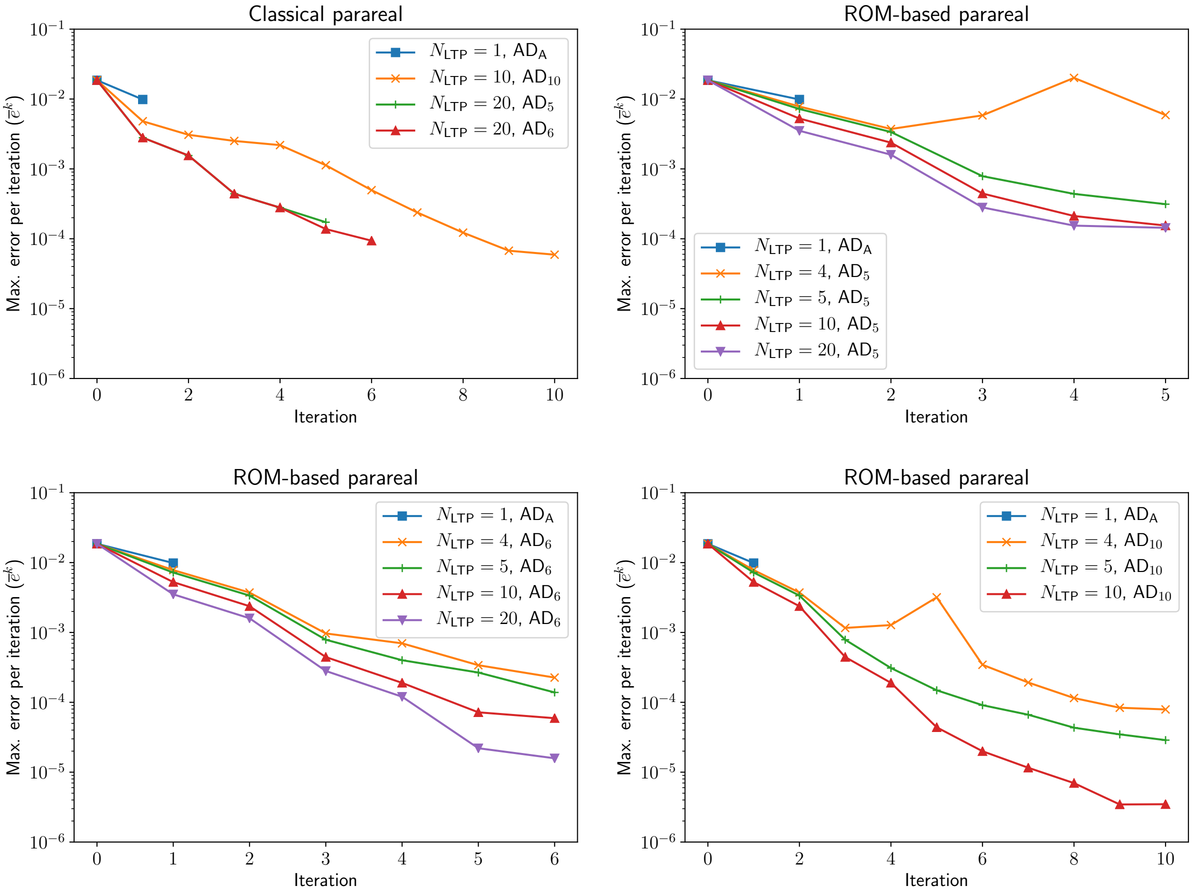


Figure 5.9: Test case 1.1: evolution of the relative error $\bar{e}^k$ for the global parareal ($N_{\text{LTP}} = 1$) and the local-in-time approaches ($N_{\text{LTP}} = 2, 4, 5, 10, 20$), for various maximum numbers of iterations (each one corresponding to an adaptive configuration). Top left: stable simulations using the classical parareal method; top right: ROM-based with configuration AD$_5$, bottom left: ROM-based with configuration AD$_6$; bottom right: ROM-based with configuration AD$_{10}$.

For illustrating the behaviour in time of the LTP approach, and also to evidence its limitations in terms of convergence, we present in Figure 5.10 the evolution of the relative errors e_n^k in the case AD$_5$, for each value of N_{LTP}. We clearly observe the local-in-time behaviour of the errors induced by the execution of sequential, smaller instances of the parareal method. The error decrease at each parareal instance $\mathcal{P}^{(i)}$ is limited by the final error of the previous instance, *i.e.* by the accuracy of the received initial solution. We notice, however, that this approach is able to improve the stability, compared to the global parareal simulation, and a better convergence behaviour is observed. As said above, instabilities are still observed for advanced time steps for $N_{\text{LTP}} = 4$, but for $N_{\text{LTP}} \geq 5$ all simulations are stable.

Table 5.9 presents the speedups at the end of the simulations. Note that, in the LTP approach, there is no meaning in analysing the speedup at an intermediate iteration $k < N_{\text{itermax}}$, since N_{itermax} iterations are necessarily performed by each parareal instance. We then report only the speedups at the final iterations. We first observe that faster simulations are obtained with $N_{\text{LTP}} = 5$ than with $N_{\text{LTP}} = 4$, due to the unbalanced distribution of parallel tasks in the latter case, as already discussed. The adaptive approach, by performing less expensive iterations, ensures speedups larger than the unity in this test case in almost all simulations. The speedups and the errors presented in Figure 5.9 reveal that a trade-off should be established when choosing the number of time windows and the number of parareal iterations.

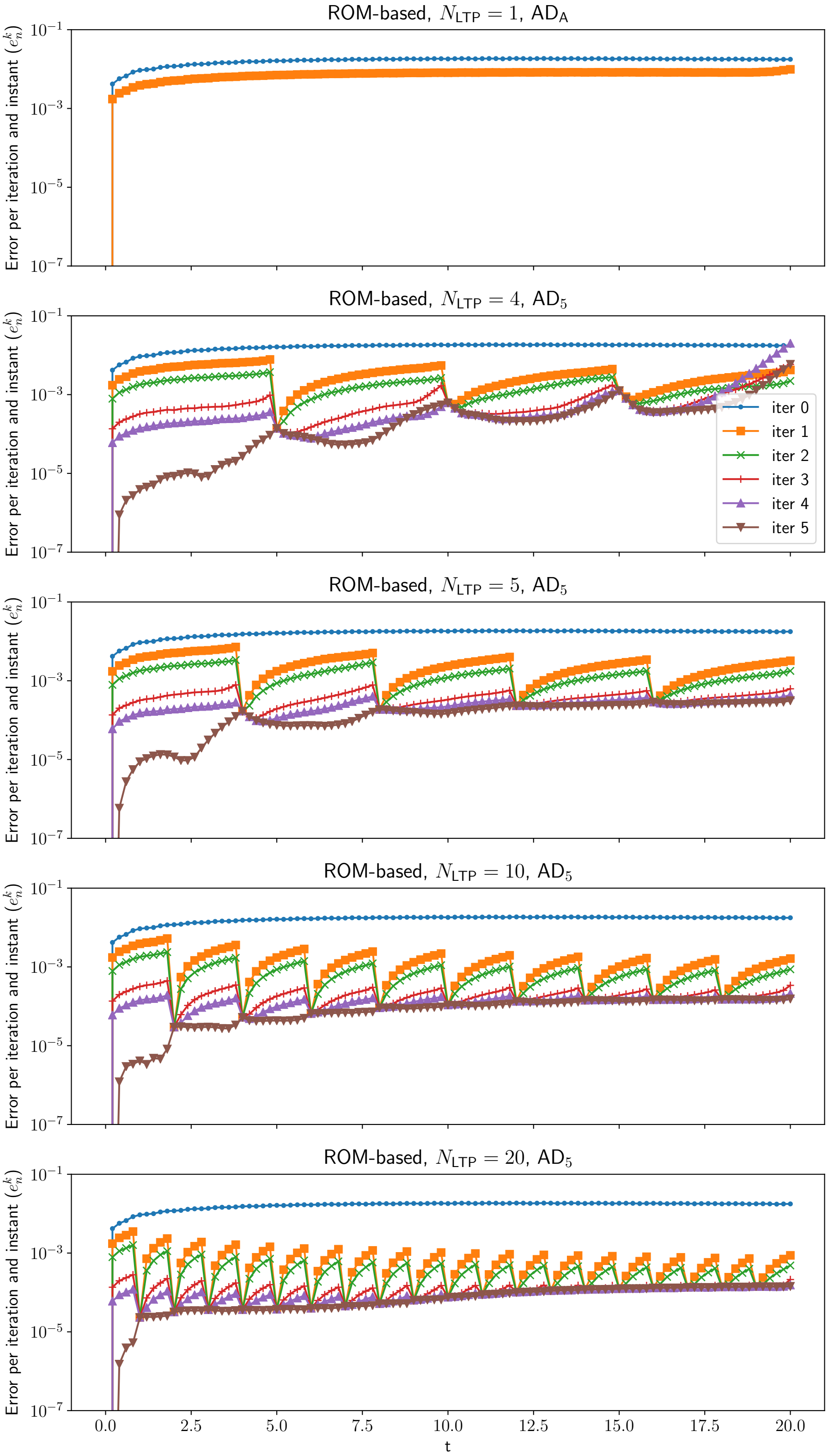


Figure 5.10: Test case 1.1: evolution of the relative error e_n^k for the global parareal ($N_{\text{LTP}} = 1$) and the local-in-time approaches ($N_{\text{LTP}} = 4, 5, 10, 20$), under the same parareal configurations ($\alpha = 1/2$, $N_{\text{PID}} = 1$, adaptive configuration AD$_5$). The global parareal simulation is unstable and does not complete two iterations.

For example, for $N_{\text{LTP}} = 5$, the error $e^{N_{\text{itermax}}}$ provided by AD_5 and AD_6 are respectively equal to 3×10^{-4} and 1×10^{-4}, approximately, but the latter provides 70% of the speedup of the former. Also, for AD_6 fixed, the error for $N_{\text{LTP}} = 10$ reduces to 6×10^{-5}, but at a higher computational cost. Finally, the small speedups for configuration AD_{10} show that the number of iterations performed by each parareal instance should be kept as small as possible.

	AD_5		AD_6		AD_{10}	
N_{LTP}	CL	ROM	CL	ROM	CL	ROM
1	*	*	*	*	*	*
2	*	*	*	*	*	*
4	*	2.06	*	1.42	*	0.60
5	*	2.34	*	1.63	*	0.74
10	*	2.13	*	1.45	1.24	0.64
20	2.57	1.58	1.65	1.06	−−	−−

Table 5.9: Test case 1.1: speedup at the final iteration for different various numbers N_{LTP} of parareal instances in the local-in-time approach. "CL" and "ROM" stand respectively for the classical and ROM-based parareal methods. $N_{\text{LTP}} = 1$ corresponds to the global parareal. Asterisks indicate that the simulation is unstable and does not complete N_{itermax} iterations. Simulations indicated with "−−" were not performed since they are equivalent to a serial execution of the reference model.

Test case 2.1 (2D in larger spatial and temporal domains)

For Test case 2.1, we consider the same configurations used in the study of the length of time slices (Section 5.3), both for the classical and ROM-based parareal methods. In this case, we considered adaptive configuration AD_{B}, in which, in general, each intermediate fine propagator is used in two iterations, due to the relative small threshold $\varepsilon_{\text{ad}} = 10^{-3}$. That is, the number of parareal iterations using the reference propagator $\widehat{\mathcal{F}}^0_{\delta t_0} = \mathcal{F}_{\delta t}$ is smaller than in configuration AD_{A}. Therefore, we do not modify configuration AD_{B}, and $N_{\text{itermax}} = 10$ iterations are performed by each parareal instance.

We perform simulations for $N_{\text{LTP}} \in \{1, 2, 4, 5, 10\}$, all the simulations being stable and completing all iterations. The case $N_{\text{LTP}} = 20$ is not considered because it is equivalent to a sequential simulation of the reference model, since, in this test case, $N_{\Delta T} = 40$ and $\mathcal{F}_{\delta t}$ is used in at least two parareal iterations. The maximum error per iteration $\overline{e}^k$ the speedups $s(N_{\text{itermax}})$ at the end of simulations are presented respectively in Figure 5.11 and Table 5.10.

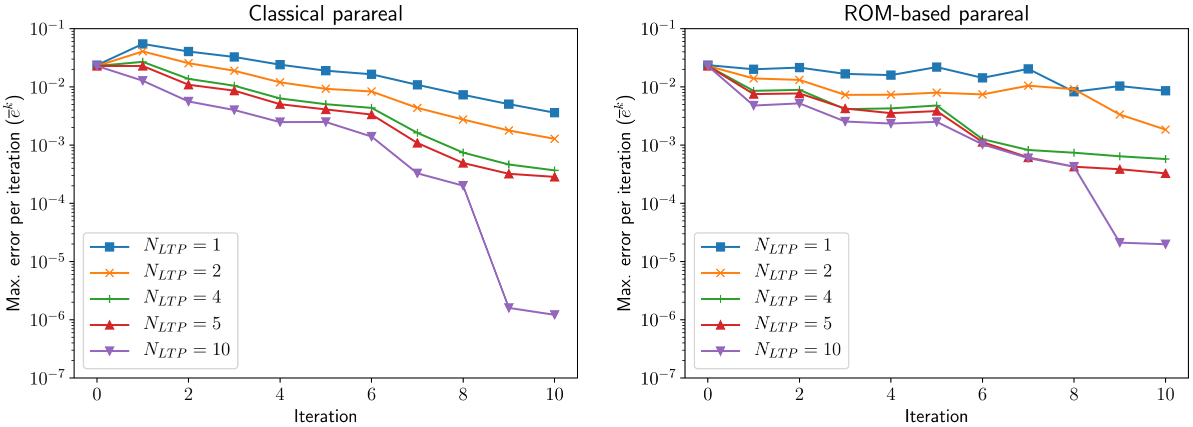


Figure 5.11: Test case 2.1: evolution of the relative error $\overline{e}^k$ for the global parareal ($N_{\text{LTP}} = 1$) and the local-in-time approaches ($N_{\text{LTP}} = 2, 4, 5, 10$), in the classical (left) and ROM-based (right) parareal frameworks.

Contrary to Test case 1.1, improvements of the classical parareal method in function of N_{LTP} are clearly observed. Indeed, in Test case 2.1 there are no instabilities due to the chosen length of time slices

N_{LTP}	1	2	4	5	10
Classical	2.65	2.77	1.95	1.69	0.97
ROM-based	1.75	1.49	1.02	0.89	0.58

Table 5.10: Test case 2.1: speedup at the final iteration for various numbers of parareal instances in the local-in-time (LTP) approach, in the adaptive ROM-based parareal framework. $N_{\text{LTP}} = 1$ corresponds to the global parareal.

($\Delta T = \Delta t$), and a relatively good convergence is obtained within each parareal instance. Improvements are also observed for the ROM-based parareal, but more remarkably for $N_{\text{LTP}} \geq 4$. As in Test case 1.1, the improvements of the ROM-based method in function of N_{LTP} are much clearer than in function of the length of the time slices (Figure 5.6). The error behaviour suggests that instabilities are still present for $N_{\text{LTP}} \leq 2$. Note that the errors are still relatively large for all simulations, but a reduction from approximately 10^{-2} for $N_{\text{LTP}} = 1$ to errors smaller than 10^{-3} for $N_{\text{LTP}} = 4$ and $N_{\text{LTP}} = 5$ are observed, with still interesting speedups compared to $N_{\text{LTP}} = 1$ in the classical parareal framework. In the ROM-based case, the simulations with $N_{\text{LTP}} \geq 4$ have similar or smaller computational times than the reference one. Also, in both parareal frameworks, the simulations with very small time windows ($N_{\text{LTP}} = 10$) provide substantial error reductions in the last two iterations, in which $\mathcal{F}^0_{\delta t_0} = \mathcal{F}_{\delta t}$ is used as fine propagator, but at the cost of speedups smaller than the unity. We remark that, in this test case, there are $N_{\Delta T} = 40$ time slices and, by using $N_p = 20$ processors, the parallel capacity is underused when N_{LTP} is larger than $N_{\Delta T}/N_p = 2$. Finally, in the case of the classical parareal method, we notice that a smaller speedup is obtained for $N_{\text{LTP}} = 1$, compared to $N_{\text{LTP}} = 2$, since the adaptive transition to the reference propagator is performed one iteration before in the former case.

5.5 Conclusion of the chapter

In this chapter, we considered two different strategies for performing longer parareal simulations in time, and we compared their impacts on the performances of the classical and ROM-based parareal methods. This study was motivated by noticing that parareal simulations become more challenging in terms of stability when performed in large temporal and spatial domains.

The first strategy is to choose larger parareal time slices, *i.e.* increase ΔT. Contrary to the simulations presented in the previous chapter, in which we considered $\Delta T = \Delta t$, we now allowed each time slice to contain several coarse time steps. Improvements of the convergence were observed both for the classical and ROM-based methods, but much clearly in the former case. Indeed, stable and fast convergence behaviours were obtained by choosing larger ΔT in the classical method, even in the one-dimensional test case, for which the method presented instabilities when $\Delta T = \Delta t$. These results are coherent with conclusions presented by Ruprecht (2018), who indicates that faster convergence is obtained by increasing ΔT. Improvements were also observed in the ROM-based parareal method, but larger ΔT did not necessarily improved the convergence in all iterations. It suggests that its convergence is limited by the quality of the reduced-order model formulated in the entire temporal domain $[0, T]$.

The second approach consists in performing local-in-time parareal simulations, in small time windows, with each parareal instance providing the initial solution for the next one. In this case, clear improvements of the ROM-based method were observed, with faster convergence by considering more time windows. It reinforces that the method performs better when the reduced-order models are formulated locally in time. In the case of the classical method, also clear improvements were observed in the two-dimensional test case, for which the global-in-time method (executed in $[0, T]$) is already stable. However, in the one-dimensional test case, in which the length ΔT of the time slice is critical in terms of stability, stable solutions were obtained only by considering very small time windows, thus limiting the error increase in time.

Both studied approaches present drawbacks in terms of parallel performance, since they reduce the possible parallelization of the parareal method. Also, the local-in-time approach limits the convergence towards the reference solution, since the quality of the solution in each parareal instance depends on the solution received from the previous one. Therefore, the choices of the length of the time slices and/or number of time windows and number of parareal iterations should be guided by a compromise between accuracy, stability and expected speedup. The use of the adaptive parareal approach, described in Section 4.5, contributes to achieve good accelerations of the simulation. Also, it was shown that additional costs due to the definition of several parareal instances remain small relative to the total parareal iteration time.

Note that these approaches can obviously be combined, by dividing the simulation into several parareal instances and using large time slices. Evidently, the negative impacts on the expected speedup would be more important. Finally, we remark that the local-in-time approach opens room for using different and more suitable configurations in each parareal instance, based *e.g.* on the behaviour of the solution in each time window. These possibilities are explored in the next chapter.

CHAPTER 6

COUPLING BETWEEN THE CLASSICAL AND THE POROSITY-BASED SHALLOW WATER EQUATIONS

Contents

6.1 Introduction **147**
6.2 A simple and challenging example **148**
6.3 Description of a base test case **151**
6.3.1 Definitions 151
6.3.2 Comparison between the models 154
6.4 Influence of the coarsening between the classical and the porosity-based SWE **157**
6.4.1 Study for a fixed coarse propagator 157
6.4.2 Study for a fixed fine, reference propagator 159
6.4.3 Some general conclusions 161
6.5 Setting parareal configurations for improving its performance **161**
6.5.1 Comparison between configurations of the ROM-based parareal method 162
6.5.2 Comparison between the classical and ROM-based parareal methods 163
6.5.3 Combining the classical and ROM-based parareal methods 165
6.5.4 Some comments on the use of the adaptive approach 169
6.6 Towards more challenging examples **172**
6.6.1 Initial water depth $h_0 = 0.5$ 173
6.6.2 Initial water depth $h_0 = 0.1$ 182
6.6.3 Initial water depth $h_0 = 0$ 183
6.7 Conclusion of the chapter **184**

6.1 Introduction

After comparing and studying the performance of the classical and ROM-based parareal methods for solving the two-dimensional nonlinear shallow water equations, with both fine and coarse propagators defined as discretizations of the classical SWE, we apply the methods for coupling the classical and porosity-based shallow-water models.

This application is not straightforward. In Chapters 3 and 4, the parareal methods and the proposed modifications to the ROM-based one were tested using relatively simple test cases, defined within small spatial and temporal domains and with relatively simple and smooth solutions. In Chapter 5, it was illustrated that convergence and stability become more challenging in the case of larger problems, demanding the use of larger parareal time slices or the definition of local-in-time parareal instances, which limits the possible speedup that can be achieved by the method. Urban flood simulations usually cover larger temporal domains and use very refined spatial meshes for the classical SWE. Moreover, the solution usually presents strong discontinuities, due to the reflection of the flow on urban obstacles, imposing additional challenges for the parareal methods. As shown along this chapter, it is more challenging to ensure stability and a good convergence behaviour of the parareal methods in this scenario.

Most of the study presented in this chapter is performed considering a relatively simple, fictitious urban zone, for which we are able to define several discretizations of the classical shallow water equations, with various spatial and temporal mesh sizes. For the porosity-based equations, we consider both the single porosity (SP) and dual integral porosity (DIP) models. With this approach, we look for some indications on how do the classical and ROM-based parareal methods behave in function of the coarsening intensity between the fine and coarse propagators. More precisely, we seek to answer the following questions:

- For a given coarse propagator (a discretization of the porosity based models), how does the parareal convergence depend on the choice of reference model (a finer or coarse discretization of the classical SWE)?
- For a given fine propagator (a fine discretization of the classical SWE), how does the parareal convergence depend on the choice of the coarse propagator? What is the influence of the choice of porosity-based model (SP or DIP)? What is influence of choosing a porosity-based model or a coarse discretization of the classical SWE as coarse propagator?

This study is performed using parareal configurations that ensure relative stability (large time slices, definition of local-in-time parareal instances), even if not always optimal in terms of computational time. In a second moment, we study the influence of parareal parameters on the stability, convergence and achieved speedup. The definition of several discretizations of the classical SWE also allows to use an adaptive approach, both for the classical and ROM-based methods, and we propose a discussion on the practical utility of using it.

Finally, based on conclusions made from these preliminary studies, we use the parareal methods to solve the numerical example presented in Section 2.11 for illustrating the classical and porosity-based SWE. Even if still relatively simple, this example presents additional challenges, *e.g.* the larger temporal domain and the use of unstructured spatial meshes. We investigate the performance of the parareal methods in function of the initial water depth and we identify limitations for its application in typical situations of urban flood modelling in which the initial water depth is zero.

This chapter is organized as follows: in Section 6.2, small examples, consisting of modified versions of Test case 1, are used to illustrate the challenges that may be encountered when simulating urban floods using parareal methods. The test case used in most part of this chapter is presented in Section 6.3. The influence of the coarsening between the fine and coarse propagators is studied in Section 6.4 and the comparison of parareal configurations in terms of convergence, stability and speedup is covered in Section 6.5. The parallel-in-time simulation of the example introduced in Section 2.11 is performed in Section 6.6. Conclusions are presented in Section 6.7.

6.2 A simple and challenging example

We begin by presenting a set of simple test cases coupling the classical and porosity-based SWE, in order to illustrate that additional challenges arise in the application of the parareal method for the simulation of urban floods, due to the more complex geometry and induced discontinuities of the flow variables.

We consider Test case 1, as described in Table 3.1, but the computational domain $\Omega = [0, 20]^2$ is modified for the fine, reference propagator, by the introduction of a contraction in the $y-$direction. This contraction is symmetric w.r.t. $y = 10$, starts at $x = x_{c,0} = 8.$, ends at $x = x_{c,f} = 12$ and has a width $w_c = 20\phi_c$ in the y-direction, where ϕ_c is the fraction represented by the contracted width w.r.t. the full width in the rest of the domain. Therefore, the computational domain for $\mathcal{F}_{\delta t}$ reads $\Omega_f := [0, x_{c,0}] \times [0, 20] \cup [x_{c,0}, x_{c,f}] \times [10 - w_c/2, 10 + w_c/2] \cup [x_{c,f}, 20] \times [0, 20]$, as shown in Figure 6.1.

The computational domain for the coarse propagator has no contraction ($\Omega_c := \Omega$). Therefore, we consider a porosity approach for taking into account the reduction of area available for the flow, by defining a porosity parameter smaller than the unity for $[x_{c,0}, x_{c,f}] \times [0, 20]$. Note that, in this case, the

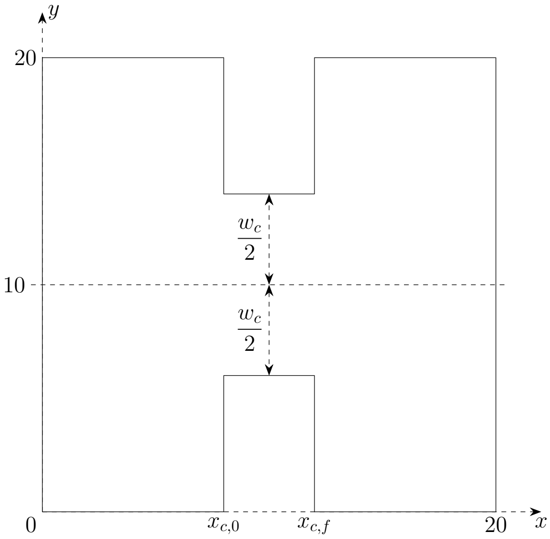


Figure 6.1: Test case 1 with contraction of the domain: spatial domain. The y-transverse section has a contraction from $x_c = x_{c,0} = 8$ to $x_c = x_{c,f} = 12$.

storage (ϕ_Ω) and conveyance (ϕ_Γ) parameters, defined in the DIP framework and accounting respectively for the available area and transverse section, are equal, since

$$\phi_\Omega = \frac{w_c(x_{c,f} - x_{c,0})}{20(x_{c,f} - x_{c,0})} = \phi_c$$
$$\phi_\Gamma = \frac{w_c}{20} = \phi_c$$

It means that the SP, IP and DIP porosity models are equivalent. Then, the porosity field for the coarse propagator $\mathcal{G}_{\Delta t}$ reads

$$\phi(\boldsymbol{x}) = \begin{cases} \phi_c, & \boldsymbol{x} \in [x_{c,0}, x_{c,f}] \times [0, 20] \\ 1, & \boldsymbol{x} \in \Omega \backslash \left([x_{c,0}, x_{c,f}] \times [0, 20]\right) \end{cases} \tag{6.1}$$

The computational meshes used by the fine and coarse models in the case $\phi_c = 0.6$ are shown in 6.2, in which the porosity field is also presented.

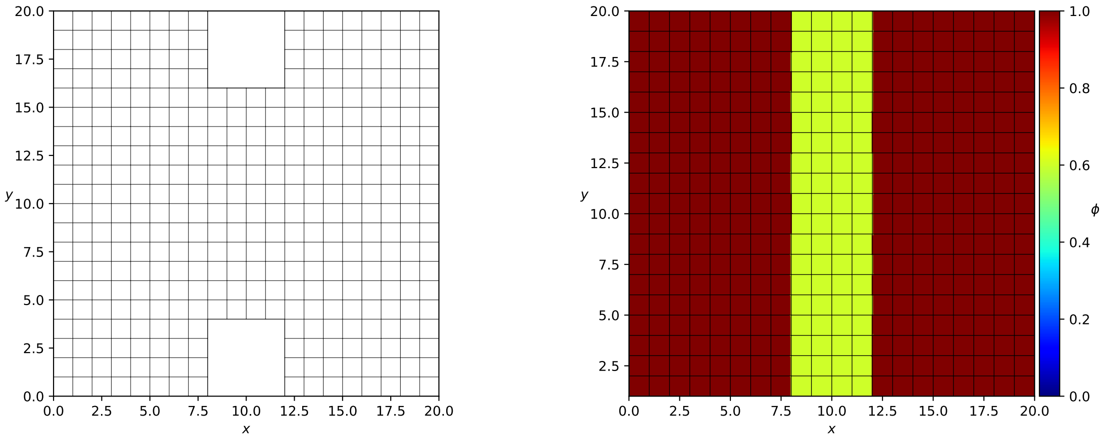


Figure 6.2: Test case 1 with contraction of the domain and $\phi_c = 0.6$: computational mesh used by the fine (left) and coarse (right) models, also representing the porosity field ϕ_c.

We perform parareal simulations considering $\phi_c \in \{0.2, 0.4, 0.6, 0.8, 1\}$. Note that $\phi_c = 1$ is equivalent to Test case 1, without contraction. Since this test case presents instabilities in the classical parareal framework under configurations presented in Table 3.1, we consider only the ROM-based approach. No modification of the method is considered here (*i.e.* $\alpha_\mathrm{s} = 1$, $N_\mathrm{PID} = 1$, non-adaptive, $N_\mathrm{LTP} = 1$). Two pairs of model reduction thresholds are considered, namely $(\varepsilon_\mathrm{sv,linear}, \varepsilon_\mathrm{sv,nonlinear}) = (10^{-5}, 10^{-5})$ and $(\varepsilon_\mathrm{sv,linear}, \varepsilon_\mathrm{sv,nonlinear}) = (10^{-3}, 10^{-5})$.

Figure 6.3 presents the maximum error per iteration for each value of ϕ_c and each pair of model reduction thresholds. For $(\varepsilon_\mathrm{sv,linear}, \varepsilon_\mathrm{sv,nonlinear}) = (10^{-5}, 10^{-5})$, all simulations with $\phi_c < 1$ present instabilities, and those with the most important domain contractions ($\phi_c = 0.2, 0.4$) are not able to complete $N_\mathrm{itermax} = 5$ iterations, whereas the simulation without contraction ($\phi_c = 1$) presents a faster convergence. By using a pair of model reduction thresholds more likely to stabilize the ROM-based parareal method, *i.e.* $(\varepsilon_\mathrm{sv,linear}, \varepsilon_\mathrm{sv,nonlinear}) = (10^{-3}, 10^{-5})$, all simulations are able to complete $N_\mathrm{itermax} = 5$ iterations and their errors decreases almost monotonically, but at a rate still much slower for $\phi_c < 1$ when compared to $\phi_c = 1$. Note that these observations hold for all simulations with $\phi_c < 1$, with only a small but not too clear outperform by $\phi_c = 0.8$ (the closest simulation to $\phi_c = 1$).

These results reveal that the introduction of a more complex geometry strongly affects the performance of the ROM-based parareal method, even in this relatively simple test case, performed within small spatial and temporal domains and with no spatial coarsening between the fine and coarse propagators. In fact, the impact on the performance of the method is analogous to what was observed, in Section 3.6.7, by using an unstructured spatial mesh. Firstly, the flow is no longer unidirectional, since non zero velocities arise along the $y-$direction due to the contraction of the domain and the reflection on the boundaries. For the same reason, strong discontinuities are formed, mainly in the velocity fields. Both these factors are challenging for the model reduction procedures and the ROM-based parareal method.

Therefore, the modifications proposed to the method in the previous chapter may be envisaged for obtaining stable and fast converging solutions. In this small example, the simplest enrichment of the snapshots sets for the model reduction, with one extra snapshot per time slice ($\alpha_\mathrm{s} = 1/2$), is able to improve the parareal performance, mainly in the case $(\varepsilon_\mathrm{sv,linear}, \varepsilon_\mathrm{sv,nonlinear}) = (10^{-5}, 10^{-5})$, as shown in Figure 6.4. However, the performances for all simulations with $\phi_c < 1$ are still far below than with $\phi_c = 1$.

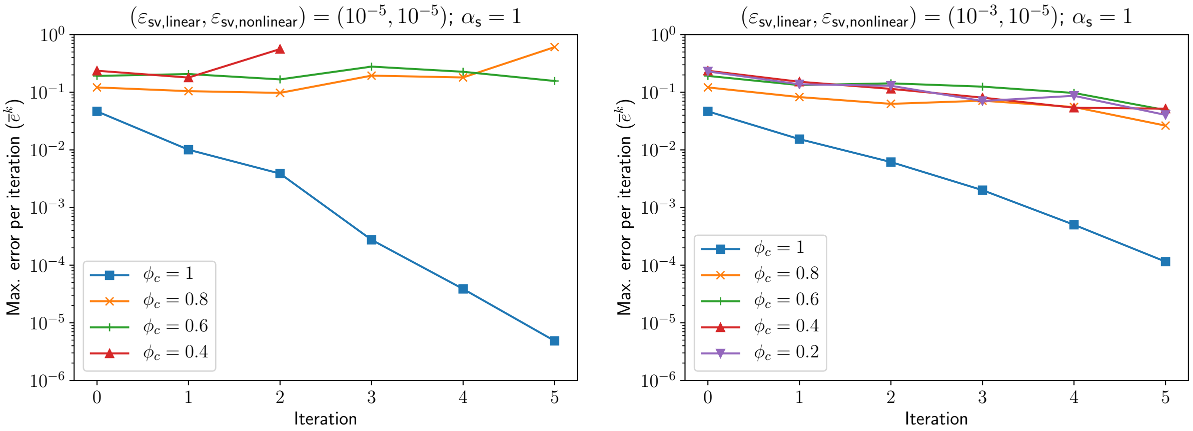


Figure 6.3: Test case 1 with contraction of the domain: evolution of the relative error $\overline{e}^k$ for various values of ϕ_c, using the ROM-based parareal method. Left: $(\varepsilon_\mathrm{sv,linear}, \varepsilon_\mathrm{sv,nonlinear}) = (10^{-5}, 10^{-5})$; right: $(\varepsilon_\mathrm{sv,linear}, \varepsilon_\mathrm{sv,nonlinear}) = (10^{-3}, 10^{-5})$. For $(\varepsilon_\mathrm{sv,linear}, \varepsilon_\mathrm{sv,nonlinear}) = (10^{-5}, 10^{-5})$, the simulation $\phi_c = 0.2$ does not complete the first parareal iteration and is not shown, and the simulation $\phi_c = 0.4$ completes only the two first iterations.

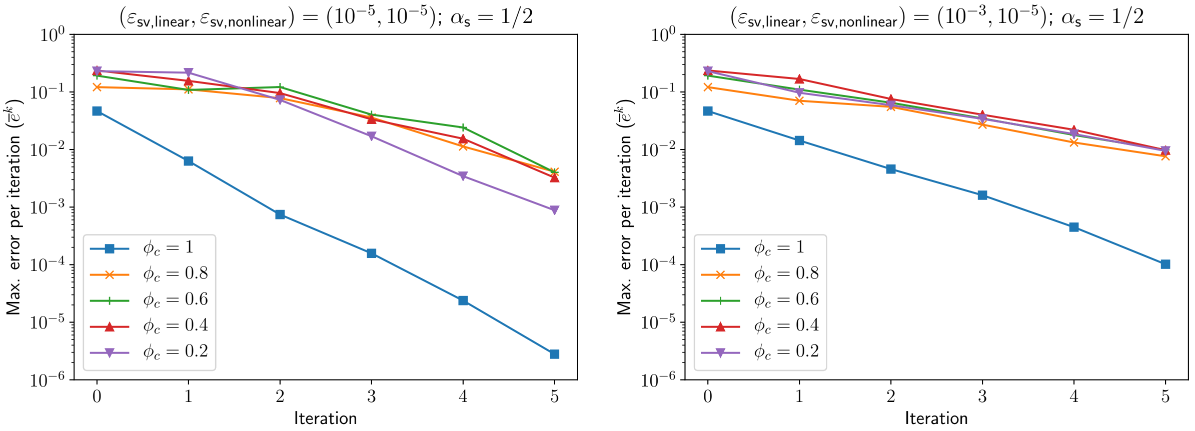


Figure 6.4: Test case 1 with contraction of the domain: evolution of the relative error $\bar{\varepsilon}^k$ for various values of ϕ_c, using the ROM-based parareal method, modified by taking $\alpha_s = 1/2$. Left: $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-5}, 10^{-5})$; right: $(\varepsilon_{\text{sv,linear}}, \varepsilon_{\text{sv,nonlinear}}) = (10^{-3}, 10^{-5})$.

6.3 Description of a base test case

We first present the test case used in most part of the work presented in this chapter. We name it hereafter as Test case 3.

6.3.1 Definitions

Domains and discretizations

We consider a square domain $\Omega_{\text{total}} = [0, L]^2$ with $L = 100$, inside which a square urban zone $\Omega_{\text{urban}} = [a_u, b_u]^2$, with $a_u = 32$ and $b_u = 68$, is defined and composed by a 5×5 Cartesian grid of square buildings with size $l_x = l_y = l = 4$ and separated one from another by $w_x = w_y = w := 4$ in each direction (see Figure 2.7 for a schematic representation). We consider two discretizations for the coarse model, named $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$, both using a Cartesian mesh, respectively with $\Delta x_0 = \Delta y_0 \approx 8.33$ (totalizing 144 cells) and $\Delta x_1 = \Delta y_1 = 5$ (totalizing 400 cells). We also define four fine propagators, named as $\mathcal{F}^0_{\delta t_0}$, $\mathcal{F}^1_{\delta t_1}$, $\mathcal{F}^2_{\delta t_2}$ and $\mathcal{F}^3_{\delta t_3}$ and defined in Cartesian meshes respectively with $\delta x_0 = \delta y_0 = 0.5$, $\delta x_1 = \delta y_1 = 1$, $\delta x_2 = \delta y_2 = 2$ and $\delta x_3 = \delta y_3 = 4$, totalizing resp. 38400, 9600, 2400 and 600 cells. Buildings are physically represented as holes in the meshes for $\mathcal{F}^0_{\delta t_0}$, $\mathcal{F}^1_{\delta t_1}$, $\mathcal{F}^2_{\delta t_2}$ and $\mathcal{F}^3_{\delta t_3}$ (since these models are discretizations of the classical SWE), but not in the meshes for $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$ (in which the presence of buildings is represented via porosity parameters). We consider here either the SP and DIP models for $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$.

Concerning the temporal domain and discretizations, the simulations are run from $t = 0$ to $t = T := 16$. All propagators are defined as to keep approximately the same CFL number and time steps as multiple integers of each other, with $\delta t_0 = 0.025$, $\delta t_1 = 0.05$, $\delta t_2 = 0.1$, $\delta t_3 = 0.2$, $\Delta t_0 = 0.4$ and $\Delta t_1 = 0.2$.

Table 6.1 summarises the basic configurations for this test case. The computational meshes used by $\mathcal{F}^1_{\delta t_1}$, $\mathcal{F}^2_{\delta t_2}$, $\mathcal{F}^3_{\delta t_3}$, $\mathcal{F}^4_{\delta t_4}$ are represented in Figure 6.5 and those used by $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$ are represented in Figure 6.6.

Porosity fields

The porosity fields used by the coarse propagators $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$ are computed analogously to the numerical example presented in Section 2.11. They take two possible values, depending on the spatial position w.r.t. Ω_{urban} (more precisely, the location of the barycenter $\boldsymbol{c}_i$ of each computational cell Ω_i). For the SP model, we have

$$\phi(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \in \Omega \backslash \Omega_{\text{urban}} \\ 1 - \dfrac{|\Omega_{\text{buildings}}|}{|\Omega_{\text{urban}}|} = 1 - \dfrac{25 l^2}{(b_u - a_u)^2} \approx 0.69, & \boldsymbol{x} \in \Omega_{\text{urban}} \end{cases}$$

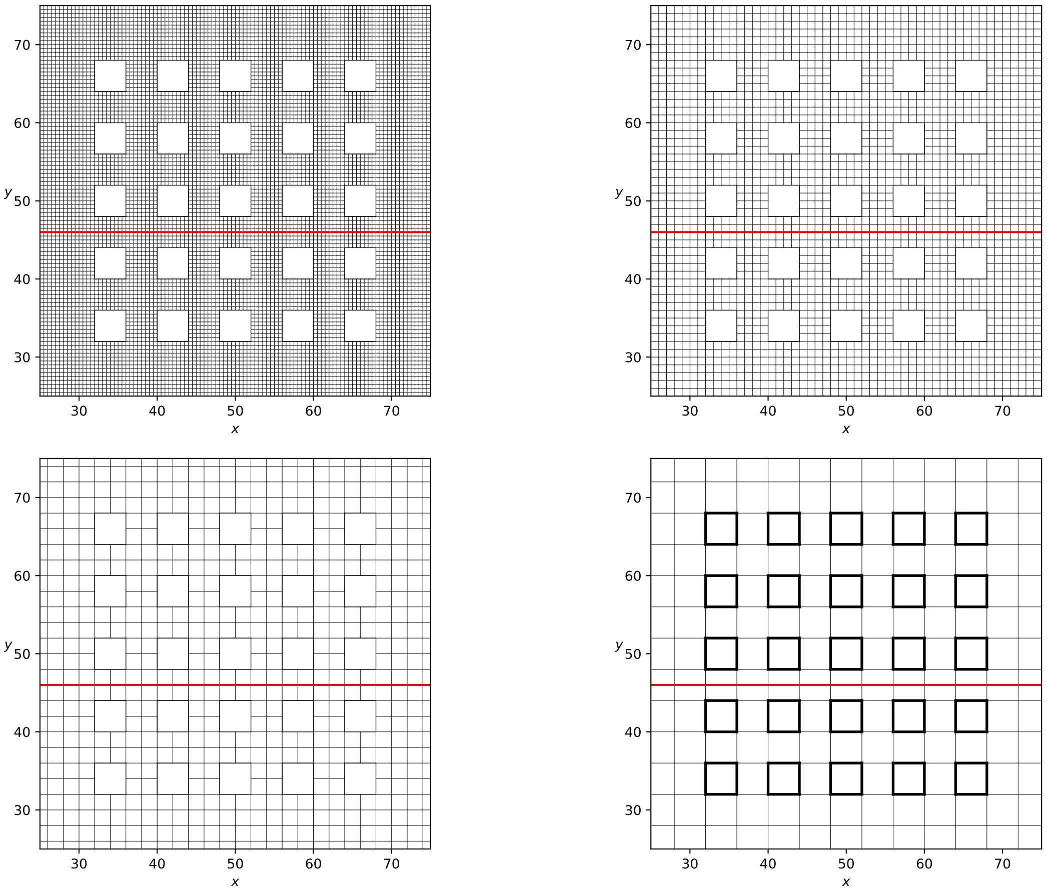


Figure 6.5: Test case 3: fine meshes (zoom on the urban zone) used respectively by the fine propagators $\mathcal{F}^0_{\delta t_0}$ (top left), $\mathcal{F}^1_{\delta t_1}$ (top right), $\mathcal{F}^2_{\delta t_2}$ (bottom left) and $\mathcal{F}^3_{\delta t_3}$ (bottom right), discretizing the classical SWE. The red lines indicate the slice $y = 46$ along which the solutions are compared.

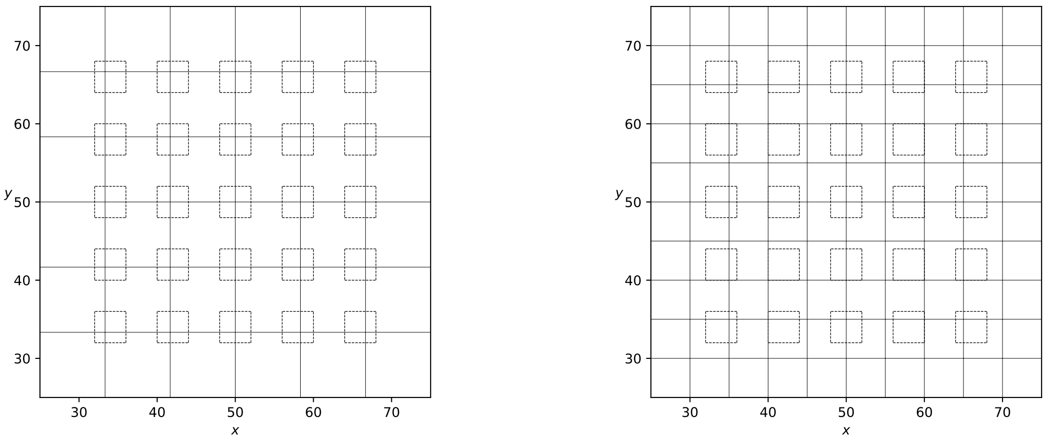


Figure 6.6: Test case 3: coarse meshes (zoom on the urban zone) used by the coarse propagators $\mathcal{G}^0_{\Delta t_0}$ (left) and $\mathcal{G}^1_{\Delta t_1}$ (right), both discretizing the porosity-based SWE. Dashed lines represent the position of the buildings (not physically represented in the mesh).

Spatial domain			$\Omega = [0, 100]^2$			
Max. simulation time			$T = 16$			
Nb. of parallel processors			$N_p = 20$			
	$\mathcal{F}^0_{\delta t_0}$	$\mathcal{F}^1_{\delta t_1}$	$\mathcal{F}^2_{\delta t_2}$	$\mathcal{F}^3_{\delta t_3}$	$\mathcal{G}^0_{\Delta t_0}$	$\mathcal{G}^1_{\Delta t_1}$
Time step	$\delta t_0 = 0.025$	$\delta t_1 = 0.05$	$\delta t_2 = 0.1$	$\delta t_3 = 0.2$	$\Delta t_0 = 0.4$	$\Delta t_1 = 0.2$
Mesh size (x-direction)	$\delta x_0 = 0.5$	$\delta x_1 = 1$	$\delta x_3 = 2$	$\delta x_4 = 4$	$\Delta x_1 \approx 8.33$	$\Delta x_0 = 5$
Mesh size (y-direction)	$\delta y_0 = 0.5$	$\delta y_1 = 1$	$\delta y_2 = 2$	$\delta y_3 = 4$	$\Delta y_0 \approx 8.33$	$\Delta y_1 = 5$

Table 6.1: Test case 3: basic configurations and definition of the fine propagators $\mathcal{F}^0_{\delta t_0}$, $\mathcal{F}^1_{\delta t_1}$, $\mathcal{F}^2_{\delta t_2}$ and $\mathcal{F}^3_{\delta t_3}$ (discretizing the classical SWE) and the coarse propagators $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$ (discretizing the porosity-based SWE).

with the porosity value in Ω_{urban} corresponding to the fraction of the area of Ω_{urban} not covered by buildings. For the DIP model, the storage and conveyance porosity fields read respectively

$$\phi_\Omega(\boldsymbol{x}) = \phi(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega$$
$$\phi_\Gamma(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \in \Omega \backslash \Omega_{\text{urban}} \\ \dfrac{4w}{(a_u - b_u)} \approx 0.44, & \boldsymbol{x} \in \Omega_{\text{urban}} \end{cases}$$

with the conveyance porosity inside Ω_{urban} corresponding to the fraction of the y-cross section of Ω_{urban} available for the flow. These porosity fields are represented in Figures 6.7 and 6.8, respectively for $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$.

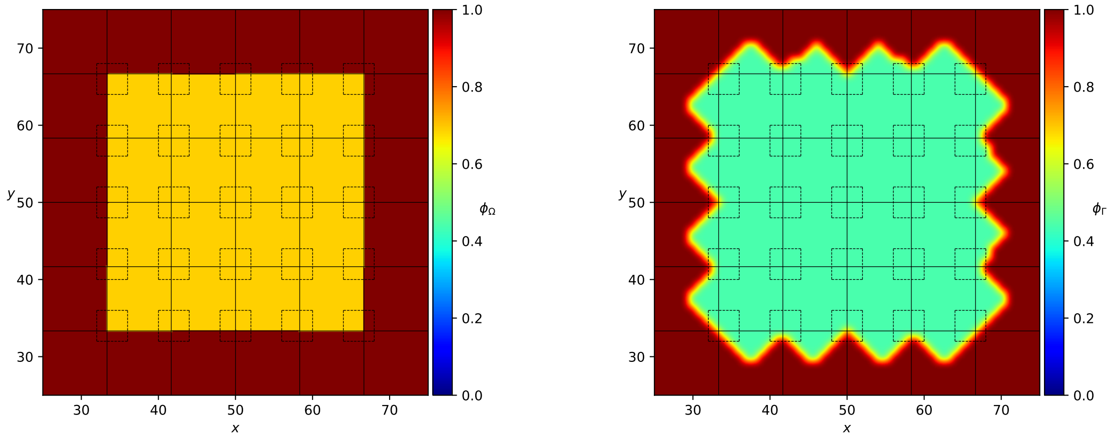


Figure 6.7: Test case 3: porosity fields used in the simulations of the porosity-based model $\mathcal{G}^0_{\Delta t_0}$. Storage porosity ϕ_Ω (left) and conveyance porosity ϕ_Γ (right) for the DIP model. The porosity field ϕ for the SP model coincides with ϕ_Ω. The conveyance porosity is presented as an interpolated field for an easier visualization. Zoom on the urban zone. All porosity fields are equal to one in the rest of the domain.

Initial and boundary conditions

We consider for these simulation a lake-at-rest as initial condition, with initial water depth $h(\boldsymbol{x}, t = 0) = 1$, and closed boundary conditions (null mass flux) for all boundaries, except for the western one, in which an inward unitary flux is defined:

$$h(\boldsymbol{x}, t = 0) = 1, \qquad u_x(\boldsymbol{x}, t = 0) = u_y(\boldsymbol{x}, t = 0) = 0, \qquad \boldsymbol{x} \in \Omega \tag{6.2}$$

$$\begin{cases} h\boldsymbol{u} \cdot \boldsymbol{n} = 1, & \boldsymbol{x} \in \partial\Omega_{\text{inward}} := \{\boldsymbol{x} \in \Omega | x = 0\}, t \in [0, T] \\ h\boldsymbol{u} \cdot \boldsymbol{n} = 0, & \boldsymbol{x} \in \partial\Omega \backslash (\partial\Omega_{\text{inward}}), t \in [0, T] \end{cases} \tag{6.3}$$

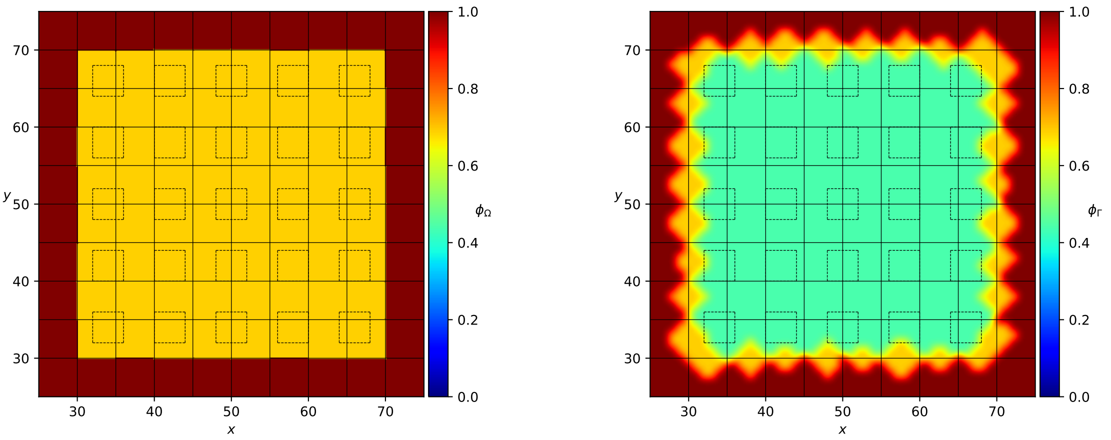


Figure 6.8: Test case 3: porosity fields used in the simulations of the porosity-based model $\mathcal{G}^1_{\Delta t_1}$. Storage porosity ϕ_Ω (left) and conveyance porosity ϕ_Γ (right) for the DIP model. The porosity field ϕ for the SP model coincides with ϕ_Ω. The conveyance porosity is presented as an interpolated field for an easier visualization. Zoom on the urban zone. All porosity fields are equal to one in the rest of the domain.

6.3.2 Comparison between the models

We first compare in Figure 6.9 the solutions provided by each model presented above ($\mathcal{F}^0_{\delta t_0}$, $\mathcal{F}^1_{\delta t_1}$, $\mathcal{F}^2_{\delta t_2}$, $\mathcal{F}^3_{\delta t_3}$ and $\mathcal{G}^0_{\Delta t_0}$, the last one using both the SP and DIP models) along $y = 46$ at two times of simulation ($t = T/2$ and $t = T$). The same results are presented in Figure 6.10, but comparing the fine models with $\mathcal{G}^1_{\Delta t_1}$. We notice that $\mathcal{G}^1_{\Delta t_1}$ provides a better approximation of the fine models, which is natural since it is has a finer discretization than $\mathcal{G}^0_{\Delta t_0}$. Both models, however, fail notably in providing a good approximation for the y-unit discharge, which presents strong discontinuities and may represent a particular challenge for the parareal methods. Also, for a fixed coarse discretization ($\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$), both porosity-based models (SP) and (DIP) provide quite similar results, with a slight outperform by the DIP, as can be assessed in Table 6.2, showing the relative errors of $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$, using the SP and DIP models, w.r.t. $\mathcal{F}^1_{\delta t_1}$.

The computational times for the simulation of each model are presented in Table 6.3. Since the models were designed in order to have nearly the same CFL number, we observe approximately a cubic dependence of the computational time on the mesh and time step sizes. The coarser reference model $\mathcal{F}^3_{\delta t_3}$ and the porosity-based ones ($\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$) present closer computational times since they are too small problems, such that overheads of the simulation (*e.g.* initialization) become relatively important.

	$t = T/2$		$t = T$	
Coarse model	SP	DIP	SP	DIP
$\mathcal{G}^0_{\Delta t_0}$	8.18E-2	7.97E-2	1.32E-1	1.22E-1
$\mathcal{G}^1_{\Delta t_1}$	4.77E-2	5.80E-2	9.96E-2	9.51E-2

Table 6.2: Test case 3: relative errors e^0_n of porosity-based models ($\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$, both with SP and DIP) w.r.t. $\mathcal{F}^0_{\delta t_0}$, for $n = N_{\Delta T}/2$ and $n = N_{\Delta T}$. These errors correspond to the iteration $k = 0$ of the parareal simulations.

$\mathcal{F}^0_{\delta t_0}$	$\mathcal{F}^1_{\delta t_1}$	$\mathcal{F}^2_{\delta t_2}$	$\mathcal{F}^3_{\delta t_3}$	$\mathcal{G}^0_{\Delta t_0}$ - SP	$\mathcal{G}^0_{\Delta t_0}$ - DIP	$\mathcal{G}^1_{\Delta t_1}$ - SP	$\mathcal{G}^1_{\Delta t_1}$ - DIP
19.84	2.71	0.30	0.09	0.05	0.06	0.06	0.06

Table 6.3: Test case 3: computational times (in seconds) for the simulation of the reference models ($\mathcal{F}^0_{\delta t_0}$, $\mathcal{F}^1_{\delta t_1}$, $\mathcal{F}^2_{\delta t_2}$, $\mathcal{F}^3_{\delta t_3}$) and the porosity-based ones ($\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$, both with SP and DIP).

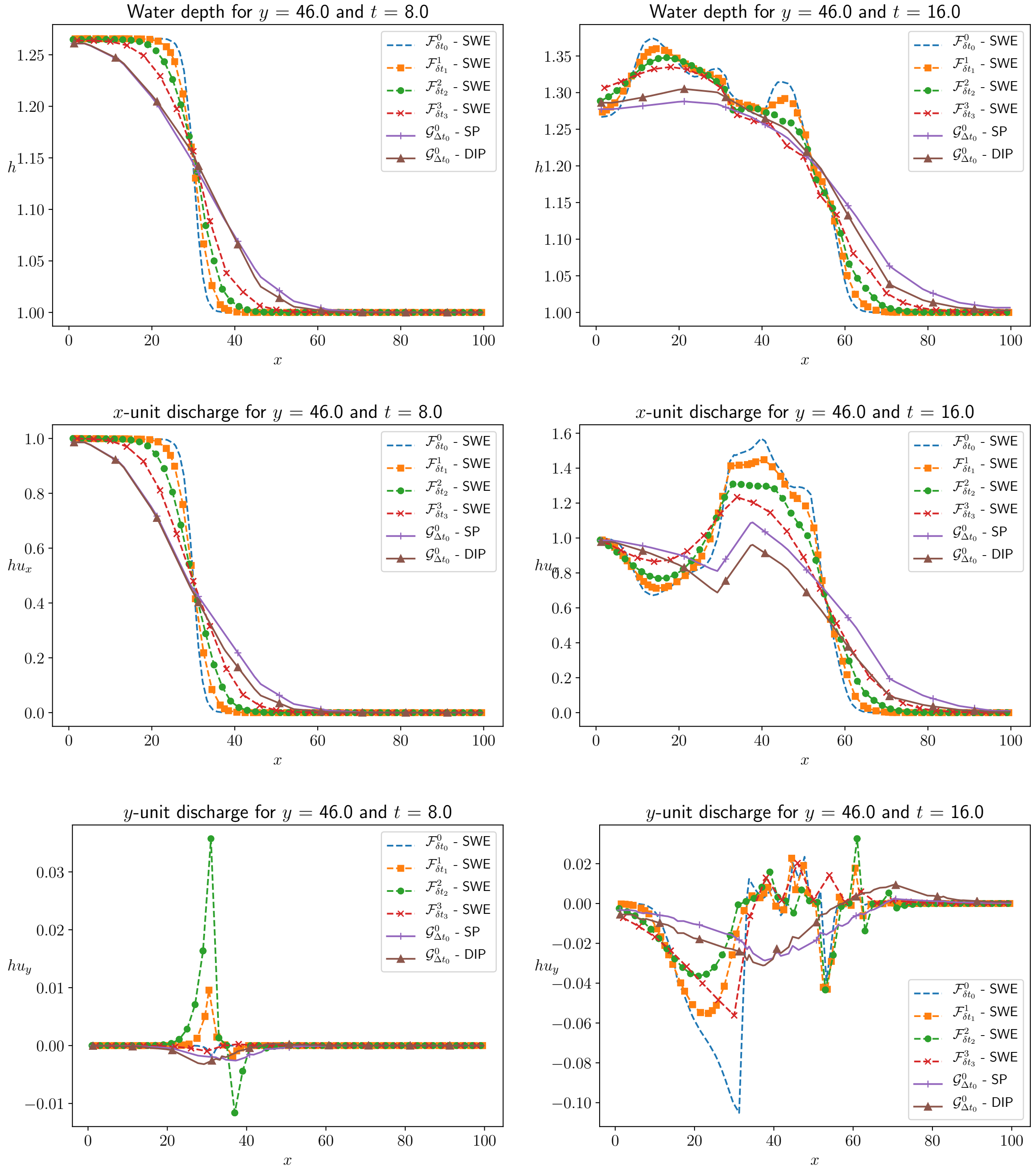
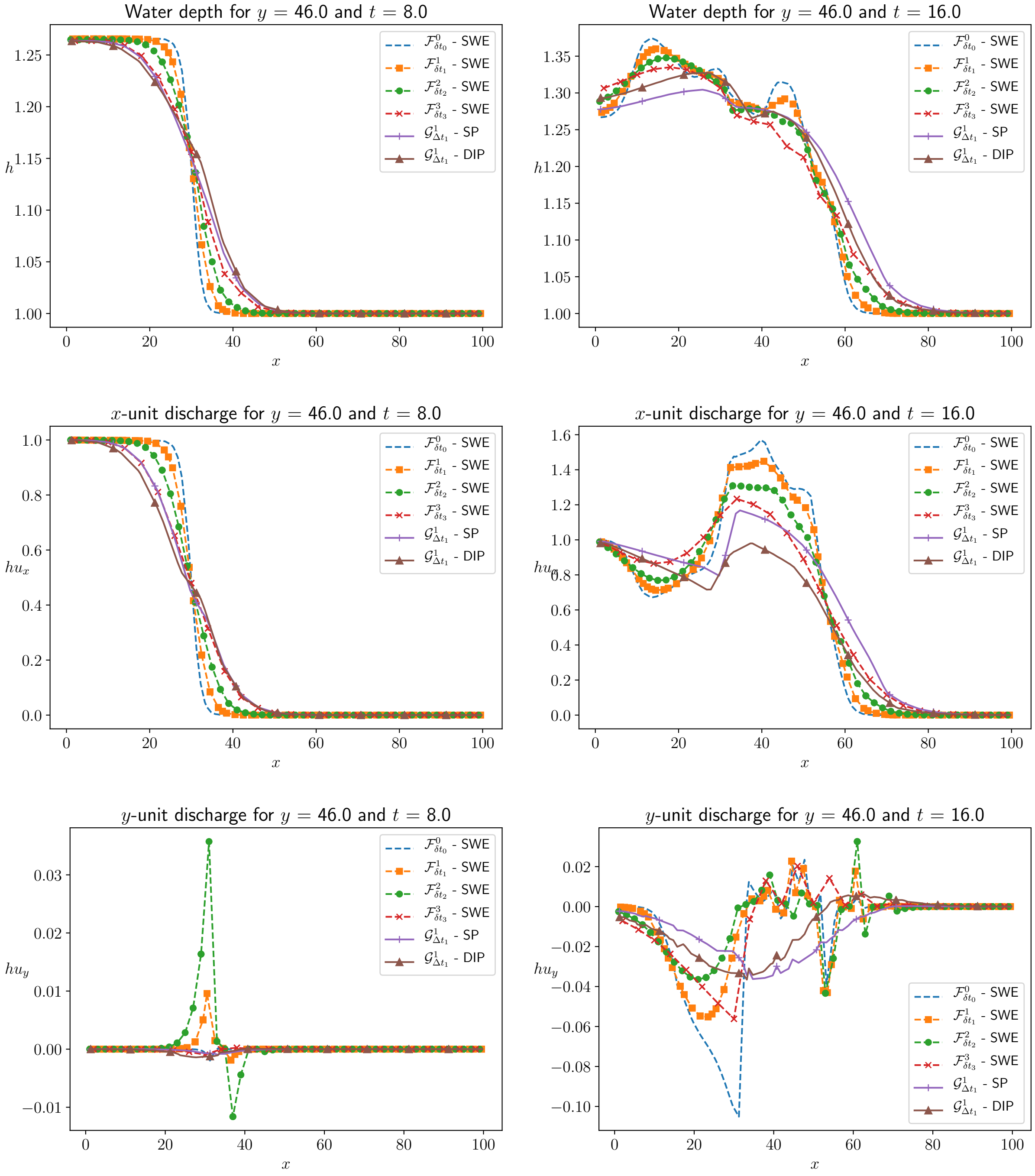


Figure 6.9: Test case 3: solution of the fine propagators (dashed lines) and the coarse ones ($\mathcal{G}^0_{\Delta t_0}$ with SP and DIP, full lines) along slice $y = 46$, for $t = T/2 = 8$ (left) and $t = T = 16$ (right). First, second and third rows: water depth, x-unit discharge and y-unit discharge, respectively. The solutions of the porosity-based models are interpolated to the fine mesh used by $\mathcal{F}^0_{\delta t_0}$.



Water depth for $y = 46.0$ and $t = 8.0$

Water depth for $y = 46.0$ and $t = 16.0$

x-unit discharge for $y = 46.0$ and $t = 8.0$

x-unit discharge for $y = 46.0$ and $t = 16.0$

y-unit discharge for $y = 46.0$ and $t = 8.0$

y-unit discharge for $y = 46.0$ and $t = 16.0$

Figure 6.10: Test case 3: solution of the fine propagators (dashed lines) and the coarse ones ($\mathcal{G}^1_{\Delta t_1}$ with SP and DIP, full lines) along slice $y = 46$, for $t = T/2 = 8$ (left) and $t = T = 16$ (right). First, second and third rows: water depth, x-unit discharge and y-unit discharge, respectively. The solutions of the porosity-based models are interpolated to the fine mesh used by $\mathcal{F}^0_{\delta t_0}$.

6.4 Influence of the coarsening between the classical and the porosity-based SWE

6.4.1 Study for a fixed coarse propagator

We investigate here the influence of the coarsening between the classical (reference) and the porosity-based shallow water models on the performance of the parareal method. For that, we compare the solutions of Test case 3 using with different fine, reference propagators ($\mathcal{F}^i_{\delta t_i}, i = 0, \ldots, 3$, given by the classical SWE with different sizes of the spatial and temporal discretizations), and a fixed coarse propagator, namely the coarsest one, $\mathcal{G}^0_{\Delta t_0}$. Since the SP and DIP models provide very close results in this test case, we consider only the latter. Evidently, we do not expect to achieve speedups larger than the unity for the simulations using the coarsest reference models, since they are already low-expensive and not much refined w.r.t. the coarse propagator. Therefore, we perform, at this point, a study comparing only the convergence and stability of the simulations.

Even if these simulations are performed in a relatively small temporal domain, they are challenging in terms of convergence and stability, mainly in the ROM-based parareal framework, which can be attributed to the more complex profiles of the solutions, including discontinuities of the velocity fields, as illustrated in Figure 6.9. Therefore, the use of large time slices ($\Delta T > \Delta t_0$) or the execution of local parareal instances ($N_{\text{LTP}} > 1$) are necessary for ensuring the stability. We consider both these cases separately.

Use of large time slices

We begin by considering a small number of large time slices. In order to properly compare the convergence behaviour in function of the coarsening between the coarse and reference propagators, we choose the same configurations for all simulations, such that almost all of them are relatively stable, even if these configurations may not be optimal in terms of computational time. Notably, stable simulations are obtained for the less challenging cases (using $\mathcal{F}^2_{\delta t_2}$ or $\mathcal{F}^3_{\delta t_3}$ as fine propagator) with simpler and less restrictive parareal configurations. We set $\Delta T = 4\Delta t_0 = 1.6$, such that there are $N_{\Delta T} = 10$ time slices, and a single parareal instance ($N_{\text{LTP}} = 1$), both for the classical and ROM-based simulations. For the ROM-based parareal method, we consider $\alpha_{\text{s}} = 1/2$, and the thresholds $\varepsilon_{\text{sv,linear}}$ and $\varepsilon_{\text{sv,nonlinear}}$ for the model reduction procedures are chosen after an initial study on the non-modified ROM-based parareal method, as done in Section 3.6. As before, instabilities are observed for small $\varepsilon_{\text{sv,linear}}$ combined with large $\varepsilon_{\text{sv,nonlinear}}$, then we consider $\varepsilon_{\text{sv,linear}} = 10^{-3}$ and $\varepsilon_{\text{sv,nonlinear}} = 10^{-5}$. We perform $N_{\text{itermax}} = 3$ parareal iterations. Under these configurations, all simulations are stable, except the most challenging ones in the ROM-based case, namely the ones using $\mathcal{F}^0_{\delta t_0}$ (which does not complete two iterations) and $\mathcal{F}^1_{\delta t_1}$ as fine propagator. This last case is able to complete $N_{\text{itermax}} = 3$ iterations, but with clear instabilities that terminate the simulation in the fourth iteration.

Figure 6.11 compares the errors obtained in each simulation. Since the reference model is not the same in all cases, this information is presented as the fraction of the final parareal error (at iteration N_{itermax}) relative to the initial error (at iteration 0), *i.e.* $e_n^{N_{\text{itermax}}}/e_n^0$. We notice, both for the classical and ROM-based parareal methods, that convergence is less challenging when the reference model is less refined, thus closer to the coarse propagator. The results are coherent with conclusions made in Chapter 5: in the classical parareal framework, the use of a large ΔT ensures a stable behaviour for all simulations, even with very fine reference propagators. This is not the case in the ROM-based method, in which the formulation of reduced-order models in the entire temporal domain leads to unstable behaviours, although the large ΔT. The less challenging simulations (using $\mathcal{F}^2_{\delta t_2}$ or $\mathcal{F}^3_{\delta t_3}$ as fine propagator) present a better convergence, but a clear degradation of the solution near the end of the temporal domain is observed.

Use of local-in-time parareal instances

As in the previous case, we choose the same configurations for all test cases, in order to ensure relatively stable simulations. We consider $N_{\text{LTP}} = 5$ time windows for the local-in-time approach, with time slices covering a single coarse time step ($\Delta T = \Delta t_0 = 0.4$, *i.e.* $N_{\Delta T} = 40$), such that each LTP window contains $N_{\Delta T}/N_{\text{LTP}} = 8$ time slices (thus exact convergence would be obtained after eight iterations per window). In each parareal instance, $N_{\text{itermax}} = 5$ parareal iterations are performed. For the ROM-based method, we set $\alpha_{\text{s}} = 1/2$.

Figure 6.12 compares the error evolution for each simulation. As before, we present the fraction of the final parareal error (at iteration N_{itermax}) relative to the initial error (at iteration 0). We clearly notice,

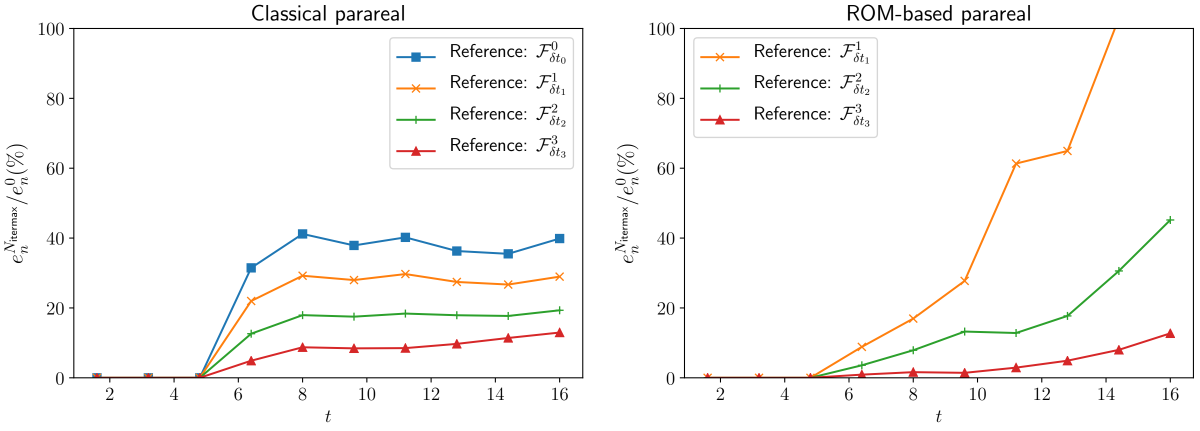


Figure 6.11: Test case 3 with $\mathcal{G}^0_{\Delta t_0}$-DIP as coarse propagator, various fine propagators, $\Delta T = 1.6$, $N_{\text{LTP}} = 1$ and $N_{\text{itermax}} = 3$: fraction of the error at the last parareal iteration relative to the error at the 0−th iteration, for each parareal time instant ($e_n^{N_{\text{itermax}}}/e_n^0$). Values expressed in percentage. Simulations using the classical (left) and ROM-based (right) parareal methods. The simulation using $\mathcal{F}^0_{\delta t_0}$ as fine propagator is unstable in the ROM-based framework and does not complete $N_{\text{itermax}} = 3$ iterations.

in the case of the classical parareal method, a dependence on the coarsening between the fine and coarse propagators, with a faster convergence in simulations using coarser reference models. The errors in the most challenging simulations (using the most refined reference propagators, $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{F}^1_{\delta t_1}$) are larger than in the case ($\Delta T = 4\Delta t_0, N_{\text{LTP}} = 1$) (Figure 6.11), indicating that the small time slice length limits the parareal convergence. In the case of the ROM-based parareal method, a less clear dependence on the coarsening intensity is observed. Indeed, except when $\mathcal{F}^0_{\delta t_0}$ is used as fine propagator, a relatively fast convergence is observed in all simulations, with errors smaller than 10% of the initial one in the entire temporal domain and outperforming the simulations using the classical parareal method, which indicates that the dynamics of the reference models are properly captured by the model reduction. By considering a very refined reference propagator, the model reduction seems to be more challenging, such that the error remains relatively small in the beginning of the simulation (in the first two time windows), but rapidly degrades at advanced times. In any case, and also in coherence with Chapter 5, the simulations are more stable and the errors smaller when compared to simulations using $N_{\text{LTP}} = 1$ (Figure 6.11), since the ROMs are formulated and used for updating the parareal solution within smaller time windows.

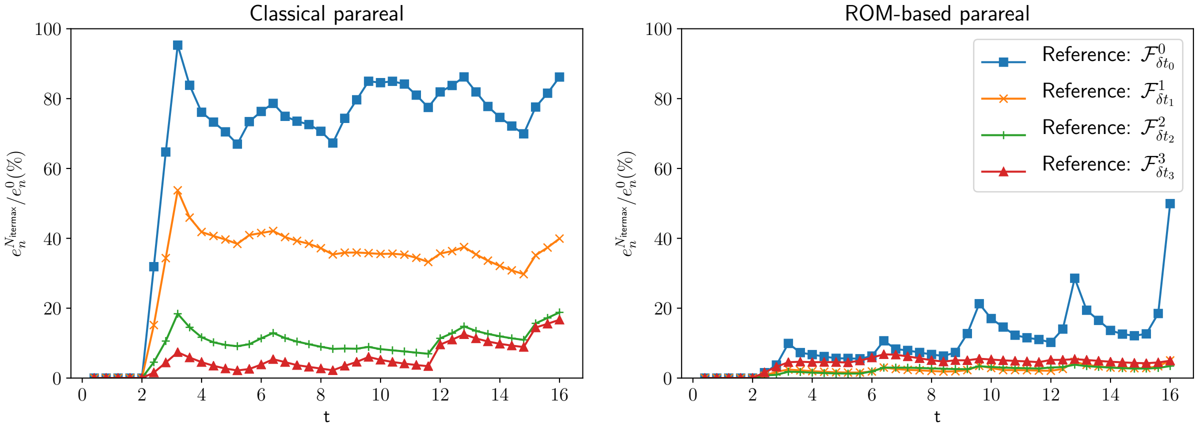


Figure 6.12: Test case 3 with $\mathcal{G}^0_{\Delta t_0}$-DIP as coarse propagator, various fine propagators, $\Delta T = 0.4$, $N_{\text{LTP}} = 5$ and $N_{\text{itermax}} = 5$: fraction of the error at the last parareal iteration relative to the error at the 0−th iteration, for each parareal time instant ($e_n^{N_{\text{itermax}}}/e_n^0$). Values expressed in percentage. Simulations using the classical (left) and the ROM-based (right) parareal methods. The legend is the same for both figures.

6.4.2 Study for a fixed fine, reference propagator

We now study the coarsening influence by an opposite approach. We fix $\mathcal{F}^0_{\delta t_0}$ as fine, reference propagator, and we perform simulations using all other propagators ($\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$ with DIP, $\mathcal{F}^3_{\delta t_3}$, $\mathcal{F}^2_{\delta t_2}$ and $\mathcal{F}^1_{\delta t_1}$) as coarse model. This comparison can also be seen as a study of the influence of the choice of coarse model, since we consider both classical and porosity-based models as coarse propagator. As before, we consider separately the cases ($\Delta T = 4\Delta t_0$, $N_{\mathrm{LTP}} = 1$) and ($\Delta T = \Delta t_0$, $N_{\mathrm{LTP}} = 5$). In each case, the same parareal configurations are set for all simulations (and identical to the ones used in Section 6.4.1).

Use of large time slices

In this case, with $\Delta T = 4\Delta t_0 = 1.6$ and $N_{\mathrm{LTP}} = 1$, all simulations using the ROM-based parareal method are unstable and do not complete $N_{\mathrm{itermax}} = 3$ iterations, indicating that the formulation of a high-quality reduced-order model, approximating the finest model $\mathcal{F}^0_{\delta t_0}$ in the entire temporal domain, remains a major issue, independently of the refinement of the coarse propagator. Therefore, only the errors for the classical parareal method are presented in Figure 6.13. We notice that smaller final errors are obtained when finer coarse propagators are used, but in this case the initial error (at iteration $k = 0$) is also smaller. Therefore, for making clearer the influence of the coarsening intensity, and also for maintaining the coherence with the results presented in Section 6.4.1, we present, also in Figure 6.13, the fraction of the final error (at iteration N_{itermax}) relative to the initial error, *i.e.* $e_n^{N_{\mathrm{itermax}}}/e_n^0$, even if the reference model is the same in all simulations. As before, we clearly observe that the convergence is faster when the coarsening intensity is smaller, *i.e.* when the coarse propagator is closer to the reference one. Notably, all simulations using a discretization of the classical SWE as coarse propagator outperform those using the porosity-based models. Indeed, similar errors are obtained in the simulations using $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$. The latter presents a slightly faster convergence, but it slows down near the end of temporal domain.

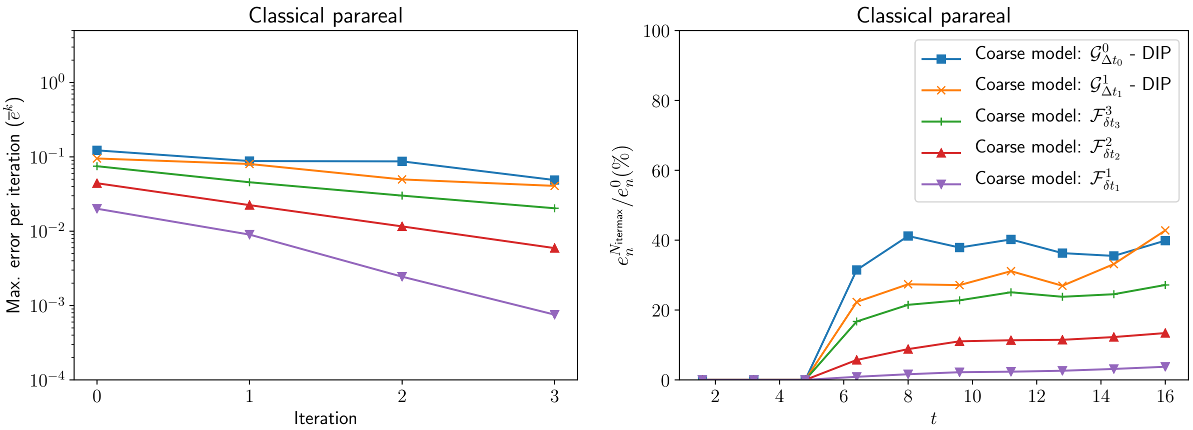


Figure 6.13: Test case 3 with various coarse propagators, $\Delta T = 1.6$, $N_{\mathrm{LTP}} = 1$ and $N_{\mathrm{itermax}} = 3$: evolution of the maximum error per iteration (left) and fraction of the error at the last parareal iteration relative to the error at the 0−th iteration, for each parareal time instant ($e_n^{N_{\mathrm{itermax}}}/e_n^0$, right), using various models as coarse propagator. Only simulations using the classical parareal method are considered, since all simulations in the ROM-based framework are unstable and do not complete $N_{\mathrm{itermax}} = 3$ iterations. The legend is the same in both figures.

Use of local-in-time parareal instances

We repeat the simulations presented above (with a fixed fine propagator and different coarse ones) by considering smaller time slices ΔT and the use of the local-in-time parareal approach. As in Section 6.4.1, we consider $N_{\mathrm{LTP}} = 5$ and $N_{\mathrm{itermax}} = 5$ iterations per parareal instance in order to ensure relative stable simulations in all cases, and a time slice length $\Delta T = \Delta t_0 = 0.4$ (equal to the time step of the coarsest propagator $\mathcal{G}^0_{\Delta t_0}$) is used in all simulations. Figure 6.14 presents, as above, the evolution of the maximum error per iteration $\overline{\varepsilon}^k$ and the fraction of the final error relative to the initial error ($e_n^{N_{\text{itermax}}}/e_n^0$), both in the classical and ROM-based parareal frameworks.

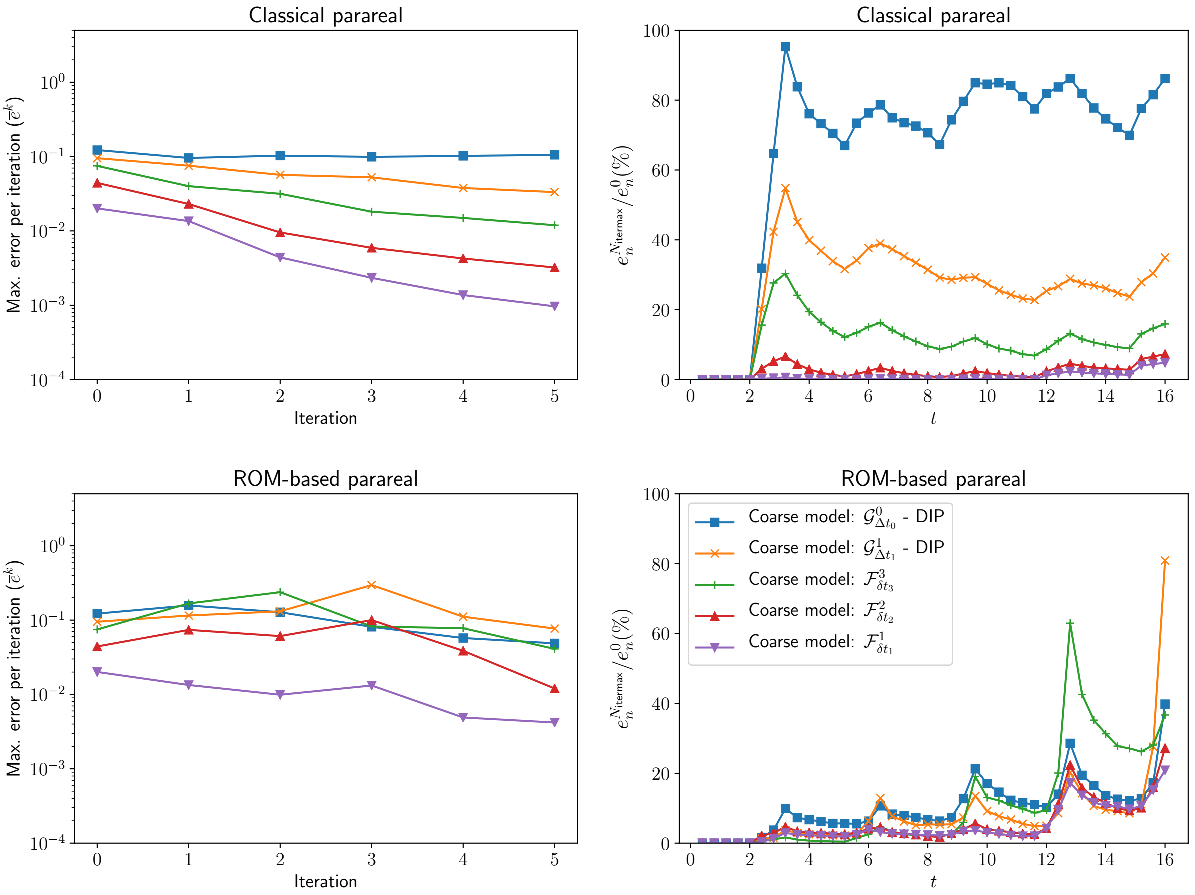


Figure 6.14: Test case 3 with $\Delta T = 0.4$, $N_{\text{LTP}} = 5$ and $N_{\text{itermax}} = 5$: evolution of the maximum error per iteration (left) and fraction of the error at the last parareal iteration relative to the error at the $0-$th iteration, for each parareal time instant ($e_n^{N_{\text{itermax}}}/e_n^0$, right), using various models as coarse propagator. Values expressed in percentage. Simulations using the classical (top) and the ROM-based (bottom) parareal methods. The legend is the same in all figures.

In the case of the classical parareal method, the conclusions are the same as with ($N_{\text{LTP}} = 1$, $\Delta T = 1.6$) (Figure 6.13), with a faster convergence when the coarse model is closer to the reference one. As observed in the study using a fixed coarse propagator (Section 6.4.1), the differences in the parareal performance in function of the coarsening intensity are more important than in the previous case, due to relatively small time slice $\Delta T = \Delta t_0$ and the execution of local-in-time parareal simulations. When a relatively fine model is used as coarse propagator, a fast convergence is obtained within each time window. However, when the coarse propagator is less refined (*e.g.* $\mathcal{G}^0_{\Delta t_0}$), this convergence slows down, also affecting negatively the convergence in the following parareal instances. An important improvement can be observed *e.g.* by using a finer discretization of the porosity-based models ($\mathcal{G}^1_{\Delta t_1}$ instead of $\mathcal{G}^0_{\Delta t_0}$), whereas both present similar error behaviours in the case ($N_{\text{LTP}} = 1$, $\Delta T = 1.6$), mainly near the end of the temporal domain.

For the ROM-based method, more stable simulations than with ($N_{\text{LTP}} = 1$, $\Delta T = 1.6$) are obtained, but a less clear behaviour of the maximum error per iteration is observed. Indeed, as shown in Figure 6.14, the errors remain relatively small in the first time windows of all simulations (with a slightly slower convergence by the simulation using $\mathcal{G}^0_{\Delta t_0}$ as coarse propagator), but strongly increase near the end of the temporal domain, indicating the formation of instabilities. Therefore, as in Section 6.4.1, we can conclude that, independently of the coarse propagator, the formulation of reduced-order models approximating the finest model $\mathcal{F}^0_{\delta t_0}$ is a major challenge for the stability and convergence of the ROM-based parareal method.

6.4.3 Some general conclusions

The numerical simulations presented in this section illustrate that the performance of the classical parareal method for solving the proposed problem is clearly influenced by the coarsening between the fine and coarse propagators. The convergence is faster when, for a fixed coarse propagator, the reference one is coarser, or when, for a fixed reference model, the coarse one is finer. Also, as already observed previously in this work, convergence is accelerated and stability improved by using larger time slices or by limiting the error increase along the temporal domain by using the local-in-time approach.

In the case of the ROM-based parareal method, the influence of the coarsening is not so clear. Indeed, convergence seems to be influenced rather by the refinement of the reference model. It was shown that the method converges faster when the reference model is coarser, but, for a fixed reference propagator (the finest one, $\mathcal{F}^0_{\delta t_0}$), the method presents instabilities for any coarse model. It can be explained by the fact that, among all fine propagators considered here, $\mathcal{F}^0_{\delta t_0}$ has the largest spatial complexity and produces solutions with stronger discontinuities and more complex profiles (see Figure 6.9), which represent a challenge for the model reduction procedures, as assessed previously in this work.

It also explains the observed convergence behaviours in time. Whereas the classical parareal method presents a quite uniform error decrease along the temporal domain, in the ROM-based one the convergence is faster in the beginning of the simulation (even with important coarsening intensities between propagators), but slower and strongly unstable near its end. Indeed, Test case 3 is characterized by a wave propagating from $x = 0$ and reaching the urban zone ($x = a_u = 32$) at a time $\bar{t}$. The analytical solution for $t \leq \bar{t}$ is the same as in Test case 1, being determined by applying the Rankine-Hugoniot condition to the propagation shock (eq. (3.54)). The analytical wave propagates with speed $\lambda \approx 3.75232$ and reaches the urban zone at $\bar{t} \approx 8.53$. Therefore, the solution in the beginning of the simulation has a simple profile (similar to Test case 1), being successfully captured by the model reduction procedures, which is favored by the use of a Cartesian mesh. For $t > \bar{t}$, and mainly near the end of the simulation, the solution profile becomes more challenging to the model reduction. In the case of less refined reference models (*e.g.* $\mathcal{F}^3_{\delta t_3}$), the solution profile is less discontinuous, and relatively good performances of the ROM-based parareal method can be obtained in the entire temporal domain, as illustrated in Figures 6.11 and 6.12.

6.5 Setting parareal configurations for improving its performance

After this initial study on the influence of the coarsening on the parareal convergence, we investigate in the following paragraphs if good results both in terms of convergence and numerical speedup can be obtained by choosing adequate parareal configurations. This choice is guided by the study performed in the previous chapters, performed in simpler sets of test cases, and also by the results observed in Section 6.4. Notably, a proper choice of time slice length, along with a combined use of the adaptive and local-in-time approaches could lead to better results, both for the classical and ROM-based methods. For the latter, the performance could also be improved by a reasonable enrichment of the input snapshots sets for the model reduction procedures and by the formulation of local ROMs in the PID framework.

This study is performed for the most challenging parareal configuration, *i.e.* by considering the finest model ($\mathcal{F}^0_{\delta t_0}$) as reference and the coarsest one ($\mathcal{G}^0_{\Delta t_0}$) as coarse propagator. It makes room for implementing the adaptive parareal approach, in which the coarser fine propagators defined in Section 6.3 act as intermediate ones. Recalling the notation introduced in Section 4.5, we consider $N_{\text{ad}} = 3$ fine propagators (the reference and the two intermediate ones, namely $\mathcal{F}^1_{\delta t_1}$ and $\mathcal{F}^2_{\delta t_2}$). We do not consider $\mathcal{F}^3_{\delta t_3}$ in this adaptive approach since it is only slightly more accurate than the porosity-based models (see Figures 6.13 and 6.14). For deciding on the transition between fine propagators, we consider a simplified approach, in which each of them is used in a fixed number of iterations, as considered in Chapter 5. It is equivalent to set $\varepsilon_{\text{ad}} = 0$, *i.e.* the transitions are not determined by a convergence criterion based on the parareal residuals (see eq. (4.18)). This choice allows to have a better control on the computational time spent at each iteration. Then, we define two adaptive configurations, named AD_3 and AD_6, in which the subscripts denote the maximum number of parareal iterations. Each fine propagator is used in one and two parareal iterations, respectively in configurations AD_3 and AD_6. We also consider a non-adaptive approach, using only the reference propagator $\mathcal{F}^0_{\delta t_0}$ and for which we define a maximum number of iterations $N_{\text{itermax}} = 3$. Table 6.4 summarises these configurations.

Configuration	N_{itermax}	ε_{ad}	N^2_{itermax}	N^1_{itermax}	N^0_{itermax}
NA	3	$--$	$--$	$--$	3
AD$_3$	3	0	1	2	3
AD$_6$	6	0	2	4	6

Table 6.4: Test case 3: adaptive parareal configurations. N^i_{itermax} is the last parareal iteration using the fine propagator $\mathcal{F}^i_{\delta t_i}, i = 0, \dots, 2$. "NA" stands for non-adaptive, for which ε_{ad} and $N^i_{\text{itermax}}, i = 1, \dots, 2$, do not apply.

6.5.1 Comparison between configurations of the ROM-based parareal method

We begin by comparing and choosing some configurations for the ROM-based parareal method (use of adaptive approach, enrichment of the snapshots sets, formulation of local-in-time reduced-order models) in terms of convergence and stability. Results presented in Section 6.4 show that, in Test case 3, using the most refined reference propagator, the ROM-based method provides a relatively good convergence in the beginning of the simulation, in which the solution profile is simpler, but the quality of the parareal solution rapidly degrades towards the end of the simulation, indicating that the model reduction cannot properly represent the highly discontinuous fine solution. Therefore, for minimizing possible instabilities, in all simulations we consider two parareal instances ($N_{\text{LTP}} = 2$), with relatively large time slices ($\Delta T = 1.6$, *i.e.* $N_{\Delta T} = 10$, with five time slices per parareal instance). We set $N_{\text{itermax}} = 3$ iterations per parareal instance.

First, we perform simulations using various adaptive configurations (the non-adaptive one and configurations AD$_3$ and AD$_6$). As shown in Figure 6.15, the use of the adaptive approach, with gradual approximations to the reference solution, greatly improves the convergence of the method. In the non-adaptive framework, strong instabilities are observed, notably in the second half of the temporal domain, leading to larger errors than in the 0-th iteration. Even if still present, these instabilities are relatively controlled and the convergence improved in the adaptive case, mainly when more iterations are performed using each fine propagator (configuration AD$_6$).

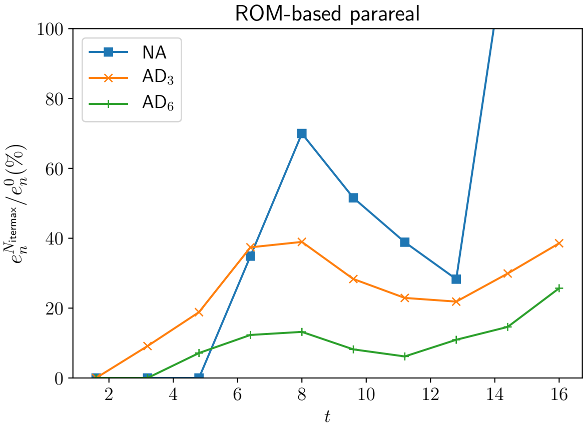


Figure 6.15: Test case 3 using the non-adaptive (NA) and adaptive configurations AD$_3$ and AD$_6$ in the ROM-based parareal method: fraction of the error at the last parareal iteration relative to the error at the 0−th iteration, for each parareal time instant ($e_n^{N_{\text{itermax}}}/e_n^0$). All simulations use $\Delta T = 1.6$, $N_{\text{LTP}} = 2$, $\alpha_{\text{s}} = 1$ and $N_{\text{PID}} = 1$. The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$.

We now fix the adaptive configuration AD$_6$ and consider various values for α_{s} and N_{PID}. For the former, we take $\alpha_{\text{s}} \in \{1, 1/2, 1/4, 1/8\}$. For the latter, we consider $N_{\text{PID}} \in \{1, 5\}$ per parareal instance. Note that these are the only possible values for N_{PID} allowing to have homogeneous PID windows, since each parareal instance contains $N_{\Delta T}/N_{\text{LTP}} = 5$ time slices. The final errors for each simulation are presented in Figure 6.16. We observe that the convergence behaviour in the first half of the simulation is similar to the results presented in the previous chapters for the one-dimensional Test case 1, since the solution has a similar and simple profile in both cases: a faster convergence is obtained by increasing the number of input snapshots used for the model reduction (smaller α_s), while the formulation of local-in-time ROMs ($N_\mathrm{PID} > 1$) increases the obtained error. However, the behaviour in the second half of the domain is quite different from previous results. Indeed, by taking too small values for α_s (1/4, 1/8), *i.e.* by using more snapshots, the solution becomes highly unstable for advanced times of simulations. In the case ($\alpha_\mathrm{s} = 1/8$, $N_\mathrm{PID} = 5$), the simulation is not able to complete all iterations. Therefore, the formulation of ROMs approximating the highly discontinuous solution in the second time window seems not to be improved by using more (also discontinuous) input information for the model reduction.

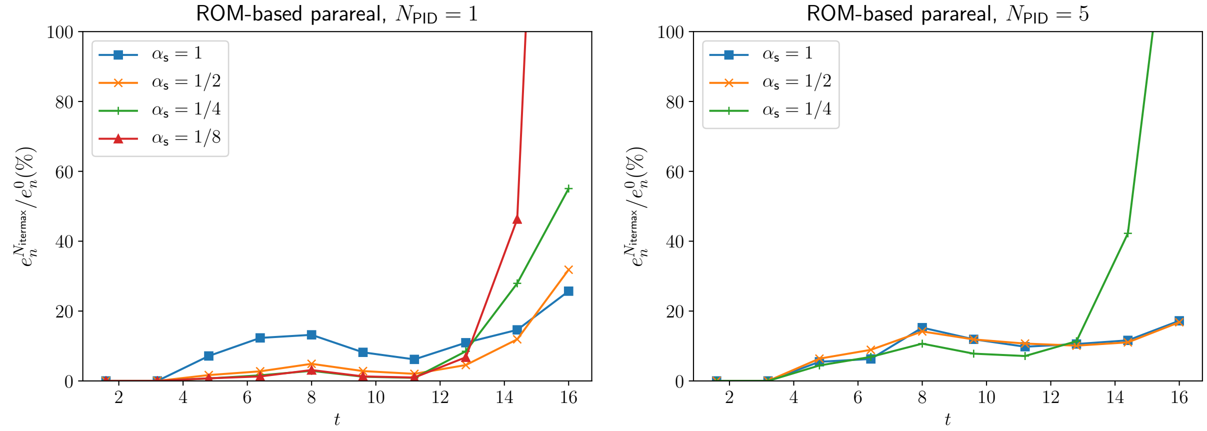


Figure 6.16: Test case 3 using various values of N_PID and α_s: fraction of the error at the last parareal iteration relative to the error at the 0−th iteration, for each parareal time instant ($e_n^{N_\mathrm{itermax}}/e_n^0$). Values expressed in percentage. Left: $N_\mathrm{PID} = 1$; right: $N_\mathrm{PID} = 5$. All simulations use $\Delta T = 1.6$, $N_\mathrm{LTP} = 2$ and adaptive configuration AD$_6$. The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$. The simulation with ($N_\mathrm{PID} = 5$, $\alpha_\mathrm{s} = 1/8$) does not complete all iterations due to instabilities.

6.5.2 Comparison between the classical and ROM-based parareal methods

We now compare, both in terms of convergence behaviour and numerical speedup, some few configurations for the classical and ROM-based simulations. For the latter, based on the preliminary study presented above, we consider $\alpha_\mathrm{s} = 1/2$ and $N_\mathrm{PID} = 1$, which provided a faster convergence in the first half of the domain and a relatively controlled stable behaviour in the second half. We also combine some time slices lengths and numbers of LTP windows, namely $\Delta T \in \{0.8, 1.6\}$ (*i.e.* $N_{\Delta T} \in \{20, 10\}$) and $N_\mathrm{LTP} \in \{1, 2\}$. The three adaptive configurations (non-adaptive, AD$_3$ and AD$_6$) are used both for the classical and ROM-based parareal methods. It results in twelve configurations in each case. The simulations are performed using $N_p = 20$ parallel processors. Therefore, only the configurations with ($N_{\Delta T} = 20$, $N_\mathrm{LTP} = 1$) use all available processors for performing the fine correction step. The case with the smallest expected speedup is ($N_{\Delta T} = 10$, $N_\mathrm{LTP} = 2$), in which the fine correction step is distributed among five processors. We recall, however, that all processors are used in the ROM-based method for performing the model reduction procedures.

The errors for the simulations in the classical parareal framework are presented in Figure 6.17. In this case, all twelve performed simulations are able to complete all iterations. It is easy to observe that the parareal solution is greatly improved by doubling the length of the time slices. The impacts of using an adaptive approach are also clear: for $\Delta T = 0.8$, the non-adaptive simulations are relatively unstable, and the final parareal error at the end of the temporal domain is larger than in the 0-th iteration. These instabilities are controlled by using adaptive configurations. For $\Delta T = 1.6$, similar results are obtained in the non-adaptive and adaptive configuration AD$_3$, with smaller errors by considering AD$_6$. Therefore, as a general conclusion, better results are obtained by performing more iterations using each intermediate fine propagator (configuration AD$_6$). Finally, important improvements are obtained in the second half of the temporal domain by dividing it into two parareal simulations ($N_\mathrm{LTP} = 2$).

In the case of the ROM-based parareal method, it is considerably more challenging to obtain stable simulations. As shown in Figure 6.18, only four among the twelve simulations complete all iterations. Notably, all non-adaptive simulations are unstable. This is also the case of configurations with a single parareal instance ($N_{\text{LTP}} = 1$), except for one of the simulations. For the simulations with $N_{\text{LTP}} = 2$, instabilities arise in the second half of the temporal domain. We can observe, for example, that the simulation with $\Delta T = 1.6$, $N_{\text{LTP}} = 2$ and adaptive configuration AD$_6$ outperforms the classical parareal simulations in $[0, T/2]$ (and even in the beginning of $[T/2, T]$), but the solution rapidly degrades in the last time slices.

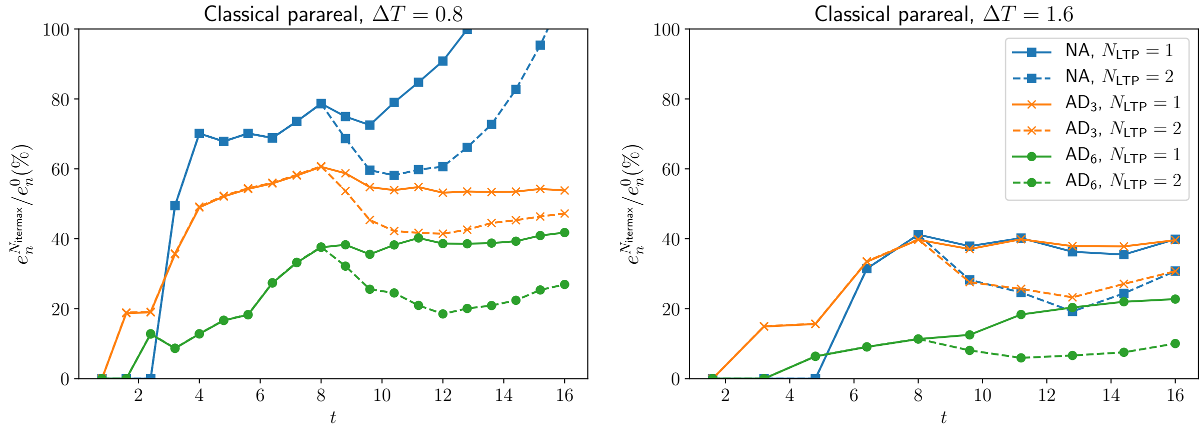


Figure 6.17: Test case 3 using various configurations of the classical parareal method: fraction of the error at the last parareal iteration relative to the error at the 0−th iteration, for each parareal time instant ($e_n^{N_{\text{itermax}}}/e_n^0$). Values expressed in percentage. Left: $\Delta T = 0.8$. Right: $\Delta T = 1.6$. Full and dashed lines indicate simulations using respectively $N_{\text{LTP}} = 1$ and $N_{\text{LTP}} = 2$. The legend is the same in both figures and the reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$.

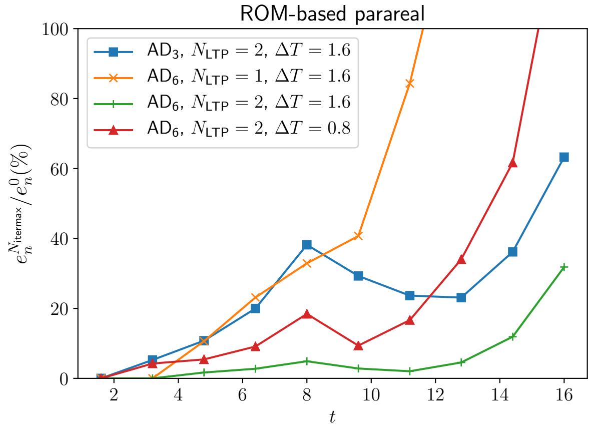


Figure 6.18: Test case 3 using various configurations of the ROM-based parareal method: fraction of the error at the last parareal iteration relative to the error at the 0−th iteration, for each parareal time instant ($e_n^{N_{\text{itermax}}}/e_n^0$). Values expressed in percentage. Only simulations able to perform all iterations are presented. The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$.

The speedups at the final iteration of each simulation are presented in Tables 6.5 and 6.6, respectively in the classical and ROM-based parareal frameworks. In the first case, larger speedups are obtained, mainly by using adaptive configurations, but they are smaller in the simulations presenting the best convergence behaviours, since more restrictive parareal configurations are necessary. The simulation with smallest final errors ($\Delta T = 1.6$, $N_{\text{LTP}} = 2$ and adaptive configuration AD$_6$) provide a speedup no larger than 2. In the ROM-based framework, the speedups are naturally smaller, but all simulations are still faster than the reference one. However, the sole simulation providing relatively small errors ($\Delta T = 1.6$, $N_{\text{LTP}} = 2$ and adaptive configuration AD$_6$) provides a speedup only slightly larger than 1.

	$\Delta T = 0.8$		$\Delta T = 1.6$	
Adaptive configuration	$N_{\mathrm{LTP}} = 1$	$N_{\mathrm{LTP}} = 2$	$N_{\mathrm{LTP}} = 1$	$N_{\mathrm{LTP}} = 2$
NA	3.01	2.04	2.46	1.47
AD_3	7.38	5.12	6.28	3.60
AD_6	3.98	2.61	3.23	1.85

Table 6.5: Test case 3 using various configurations of the classical parareal method: speedup $s(N_{\mathrm{itermax}})$ at the final iteration of each simulation. The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$.

	$\Delta T = 0.8$		$\Delta T = 1.6$	
Adaptive configuration	$N_{\mathrm{LTP}} = 1$	$N_{\mathrm{LTP}} = 2$	$N_{\mathrm{LTP}} = 1$	$N_{\mathrm{LTP}} = 2$
NA	*	*	*	*
AD_3	*	*	*	2.32
AD_6	*	1.09	1.60	1.12

Table 6.6: Test case 3 using different configurations of the ROM-based parareal method: speedup $s(N_{\mathrm{itermax}})$ at the final iteration of each simulation. The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$. Asterisks indicate that the simulation does not complete all iterations due to instabilities.

6.5.3 Combining the classical and ROM-based parareal methods

The error curves illustrated in Figures 6.17 and 6.18 and the different behaviours of the classical and ROM-based parareal simulations along the temporal domain, related to the different profiles of the solution, suggest that a combined use of both approaches could improve the quality of the solution at advanced times of simulation. Under the same configurations (AD_6, $\Delta T = 1.6$, $N_{\mathrm{LTP}} = 2$), the ROM-based method converges faster in the first time window, with a degradation in the second one, whereas the classical parareal method provides a relatively more uniform error along the entire temporal domain.

Therefore, it is natural to expect that smaller errors could be obtained by using the ROM-based parareal method in the beginning of the simulation, and the classical method for more advanced time steps, with large enough time slices for ensuring stability and a relatively fast convergence. We test two configurations, namely (AD_6, $\Delta T = 1.6$, $N_{\mathrm{LTP}} = 2$) and (AD_6, $\Delta T = 0.8$, $N_{\mathrm{LTP}} = 2$). In both cases, the ROM-based and classical parareal methods are used respectively in the first and second time windows.

Contrary to these expectations, we notice that the error behaviour at the very last time steps in the combined configuration is nearly the same as in the simulation using only the classical parareal method, even if smaller errors are obtained in the beginning of the second time window, as shown in Figure 6.19. It indicates that the convergence near the end of the simulation is conditioned by the particular choice of parareal configurations (classical method, adaptive configuration, length of time slices), and improving the quality of the initial solution received from the previous time window does not provide substantial improvements of the final solution. It is confirmed, also in Figure 6.19, by simulations in which the reference propagator $\mathcal{F}^0_{\delta t_0}$ is used in the first time window, *i.e.* the initial solution received by the second window is exact. In this case, a slight reduction of the error is observed, but at $t = T$ the error is nearly the same as in other configurations.

However, even if this combined use of the classical and ROM-based parareal methods does not greatly improve the quality of the final solution, it can be seen as a way to establish a compromise between computational time and a higher quality of the solution in the entire temporal domain, since considerably smaller errors are obtained in the first time window (in which the ROM-based method is used). As shown in Table 6.7, the combined configurations provides an intermediate speedup between the full classical and full ROM-based simulations.

The physical solutions obtained by the classical, the ROM-based and the combined configurations in the case (AD_6, $\Delta T = 1.6$, $N_{\mathrm{LTP}} = 2$) are illustrated in Figure 6.20, for $t = 4T/5 = 12.8$ and $t = T = 16$, along $y = 46$. At $t = 4T/5$ all three solutions are high-quality approximations to the reference one and greatly outperform the coarse solution. The ROM-based solution is almost visually indistinguishable from the reference solution, whereas the classical and combined parareal solutions present only small misrepresentations. We notice that even high discontinuities of the $y-$unit discharge profile are well represented. At $t = T$, the quality of the solutions is less remarkable, but the classical and combined configurations approximate well the general profile of the reference solution and also some small scale features. In the case of the ROM-based method, a quite good approximation of the general profile is also obtained, but clear instabilities can be observed. It can also be noticed in Figure 6.21, which illustrates the solution in the entire domain Ω at $t = T$.

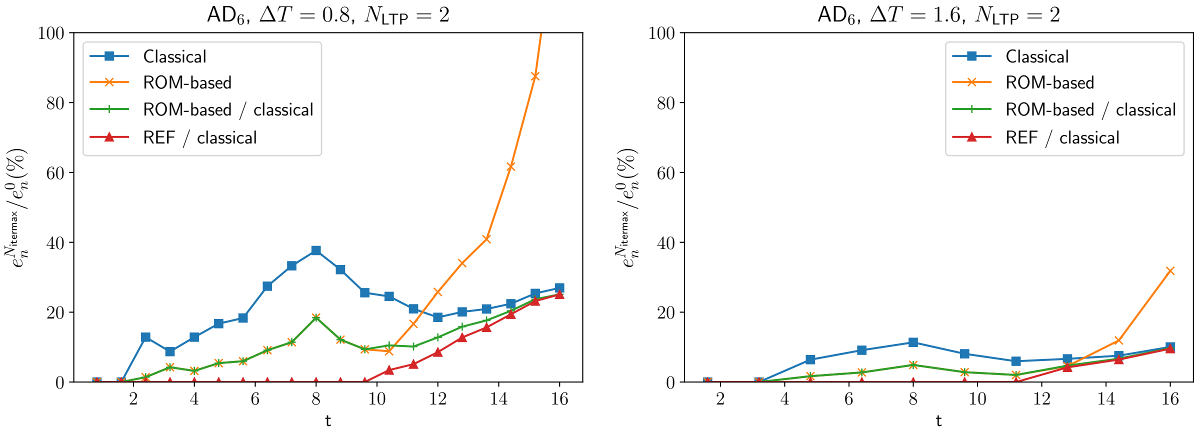


Figure 6.19: Test case 3: fraction of the error at the last parareal iteration relative to the error at the 0−th iteration, for each parareal time instant ($e_n^{N_{\mathrm{itermax}}}/e_n^0$). Values expressed in percentage. The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$. Simulations using combined ("ROM-based/classical") configurations of the ROM-based (first time window) and classical (first time windows) parareal methods. The errors obtained using only the classical and only the ROM-based methods are also presented, as well as simulations ("REF/classical") in which the solution in the first time window is exact.

$\mathrm{AD}_6,\ \Delta T = 0.8,\ N_{\mathrm{LTP}} = 2$		$\mathrm{AD}_6,\ \Delta T = 1.6,\ N_{\mathrm{LTP}} = 2$	
Config.	$s(N_{\mathrm{itermax}})$	Config.	$s(N_{\mathrm{itermax}})$
Only CL	2.61	Only CL	1.85
Only ROM	1.09	Only ROM	1.12
Combined	1.49	Combined	1.32

Table 6.7: Test case 3: speedup $s(N_{\mathrm{itermax}})$ at the final iteration for the classical ("Only CL"), ROM-based ("Only ROM") and combined parareal simulations. The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$.

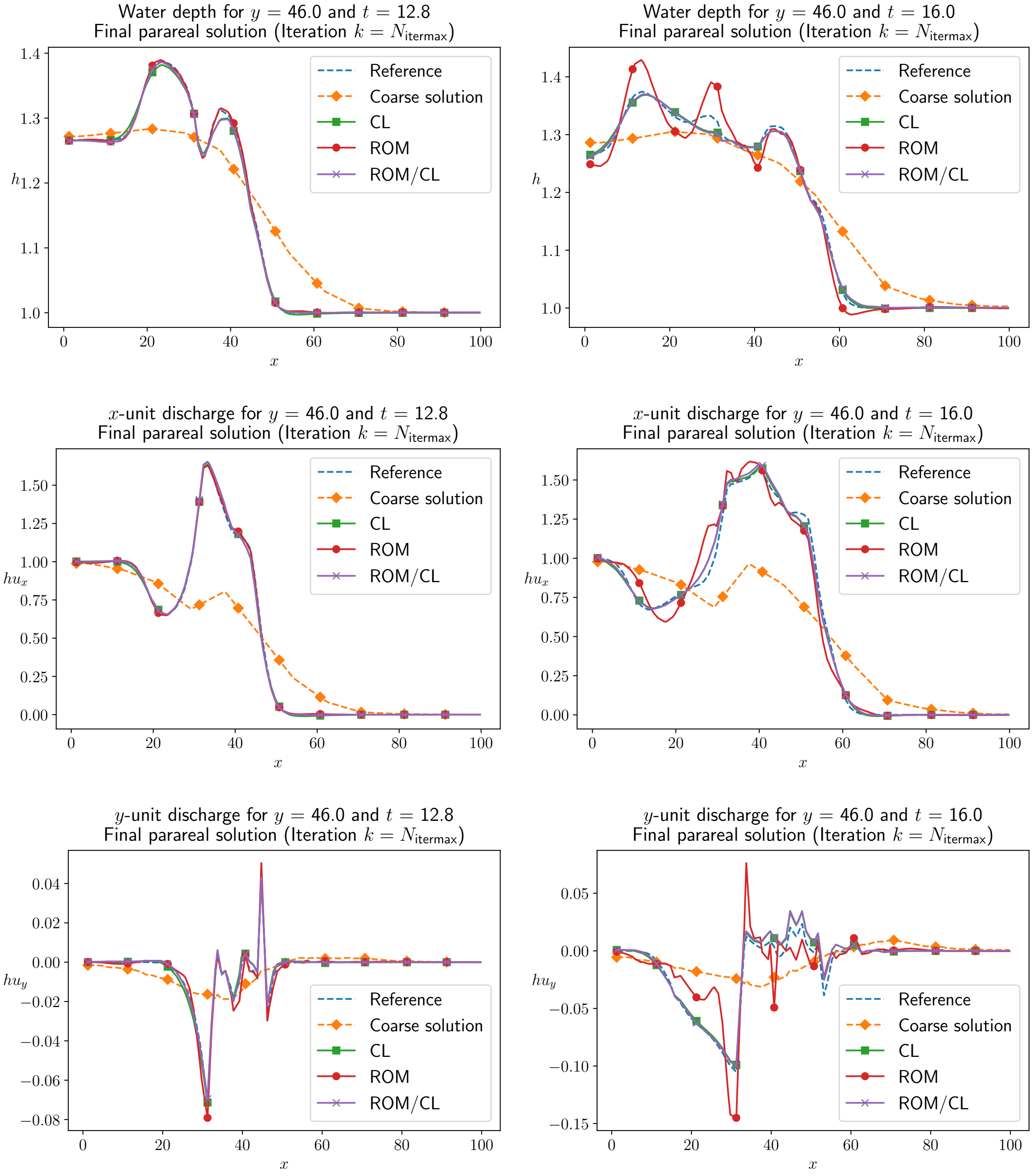


Figure 6.20: Test case 3 with adaptive configuration AD$_6$, $\Delta T = 1.6$ and $N_{\text{LTP}} = 2$: water depth along $y = 46$, at $t = 4T/5$ (left) and $t = T$ (right), for the reference, coarse (0-th parareal iteration) and parareal solutions (at iteration $N_{\text{itermax}} = 6$), for the classical ("CL"), ROM-based ("ROM") and combined ("ROM/CL") approaches. The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$.

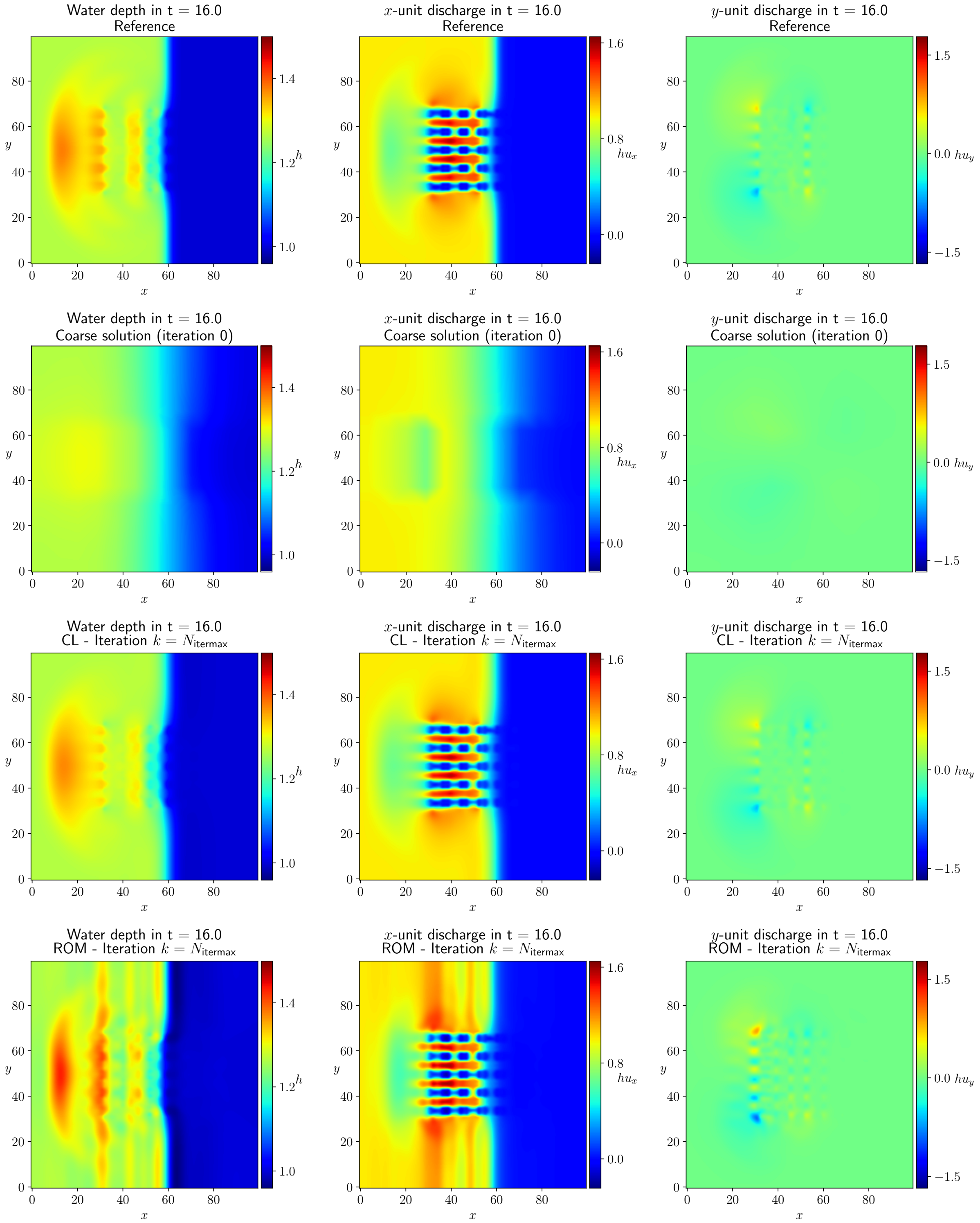


Figure 6.21: Test case 3 with adaptive configuration AD$_6$, $\Delta T = 1.6$ and $N_{\text{LTP}} = 2$: final solution ($t = T = 16$) in Ω. First, second and third columns: water depth, x-unit discharge and y-unit discharge, respectively. First, second, third and fourth rows: reference solution, coarse solution (0-th parareal iteration), final parareal solution using the classical method, and final parareal solution using the ROM-based parareal method, respectively. The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$.

6.5.4 Some comments on the use of the adaptive approach

Results presented in Figures 6.15, 6.17-6.18 and Tables 6.5-6.6 indicate that the adaptive approach is able to greatly improve the stability, convergence and speedup of the parareal method, both in the classical and ROM-based frameworks. The more stable and faster converging solutions can be attributed to the fact that the reference propagator is closer to the coarse one at intermediate iterations (thus minimizing phase speed mismatches and the associated issues for high wavenumbers of the solution), and the acceleration of the simulation is due to the use of less expensive reference models. One can observe, in Figure 6.22, that the final solutions provided by the classical and combined ROM-based/classical parareal configurations with (AD$_6$, $\Delta T = 1.6$, $N_{\text{LTP}} = 2$) outperform all intermediate fine propagators ($\mathcal{F}^i_{\delta t_i}, i = 1, 2, 3$) in the entire temporal domain, in terms of accuracy w.r.t. $\mathcal{F}^0_{\delta t_0}$, whereas the ROM-based simulation presents larger errors in the last time steps due to its unstable behaviour.

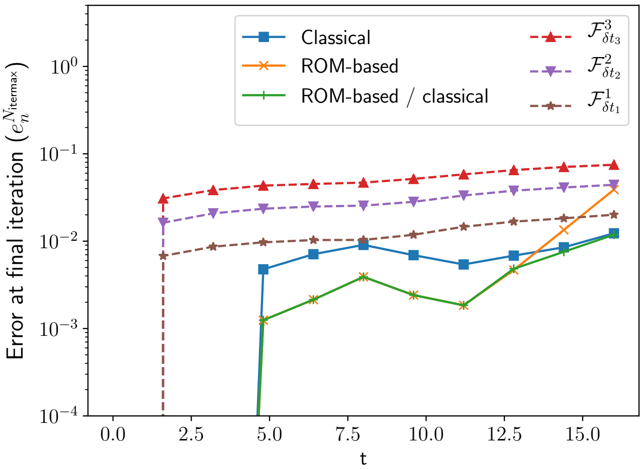


Figure 6.22: Test case 3: error $e_n^{N_{\text{itermax}}}$ at the final parareal iteration, provided by the classical, ROM-based and combined configurations (full lines) with (AD$_6$, $\Delta T = 1.6$, $N_{\text{LTP}} = 2$), compared to the accuracy w.r.t. $\mathcal{F}^0_{\delta t_0}$ of the propagators $\mathcal{F}^i_{\delta t_i}, i = 1, 2, 3$ (dashed lines). The reference and coarse propagators are respectively $\mathcal{F}^0_{\delta t_0}$ and $\mathcal{G}^0_{\Delta t_0}$.

However, one may wonder if better results could be obtained within smaller computational times by using one of the intermediate fine propagators as coarse one, instead of this adaptive approach. Surely, it would lead to more expensive iterations, since the sequential prediction step of the parareal algorithm would be performed using finer coarse models. On the other hand, Figures 6.13 and 6.14 reveal that the classical parareal simulations using one of the fine propagators $\mathcal{F}^i_{\delta t_i}, i = 1, 2, 3$, as coarse one provides faster convergence. Therefore, more accurate solutions, compared to the ones using $\mathcal{G}^0_{\Delta t_0}$, could possibly be obtained with much fewer iterations.

It is confirmed in Figure 6.23, presenting the evolution of the maximum error per iteration in the combined (ROM-based and classical) parareal simulation using $\mathcal{G}^0_{\Delta t_0}$-DIP as coarse propagator (the same simulation considered above) and in non-adaptive classical parareal simulations using $\mathcal{F}^i_{\delta t_i}, i = 1, 2, 3$, as coarse propagator (with the same configuration as in Figure 6.13). We observe that a single iteration of the parareal method using $\mathcal{F}^1_{\delta t_1}$ outperforms the simulation using $\mathcal{G}^0_{\Delta t_0}$. With $\mathcal{F}^2_{\delta t_2}$, two iterations provide nearly the same accuracy than with $\mathcal{G}^0_{\Delta t_0}$. The only simulation with a relatively slower convergence (but still faster than with $\mathcal{G}^0_{\Delta t_0}$) is the one using the coarsest fine propagator $\mathcal{F}^3_{\delta t_3}$, for which five iterations are required for outperforming $\mathcal{G}^0_{\Delta t_0}$.

Table 6.8 confirms that better accuracies are obtained within smaller computational times by using $\mathcal{F}^i_{\delta t_i}, i = 1, 2, 3$ as coarse propagators. One and two iterations of the simulation using $\mathcal{F}^1_{\delta t_1}$ provides speedups (respectively 3.01 and 1.80) larger than the one obtained with $\mathcal{G}^0_{\Delta t_0}$ after $N_{\text{itermax}} = 6$ iterations (1.32). Two iterations using $\mathcal{F}^2_{\delta t_2}$ also provides a larger speedup (2.64). For the coarsest fine propagator $\mathcal{F}^3_{\delta t_3}$, the speedup for obtaining more accurate solutions is only slightly larger ($s(5) = 1.48$).

Therefore, if intermediate propagators are available, it seems more interesting to directly use them as coarse propagator. Possibly, better results using the adaptive approach could be obtained with better guided choices of intermediate propagators and criteria for the transition between them, using *e.g.* a

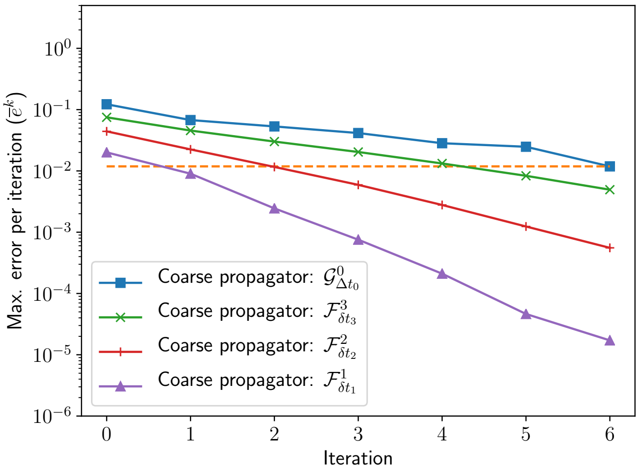


Figure 6.23: Test case 3 using $\mathcal{G}^0_{\Delta t_0}$ (combined ROM-based classical simulation, adaptive configuration AD$_6$, $\Delta T = 1.6$, $N_{\text{LTP}} = 2$) and $\mathcal{F}^i_{\delta t_i}, i = 1, 2, 3$ (classical parareal, non-adaptive, $\Delta T = 1.6$, $N_{\text{LTP}} = 1$) as coarse propagators: evolution of the maximum error per iteration ($\bar{e}^k$). The dashed, horizontal line indicates the final accuracy obtained by the simulation using $\mathcal{G}^0_{\Delta t_0}$ as coarse propagator. All simulations consider $\mathcal{F}^0_{\delta t_0}$ as reference model.

	Coarse propagator			
	$\mathcal{G}_{\Delta t}$-DIP	$\mathcal{F}^3_{\delta t_3}$	$\mathcal{F}^2_{\delta t_2}$	$\mathcal{F}^1_{\delta t_1}$
$s(1)$	18.32	6.16	4.31	3.01
$s(2)$	16.12	3.34	2.64	1.80
$s(5)$	2.28	1.48	1.27	0.88
$s(6)$	1.32	1.26	1.09	0.77

Table 6.8: Test case 3 using $\mathcal{G}^0_{\Delta t_0}$ (combined ROM-based classical simulation, AD$_6$, $\Delta T = 1.6$, $N_{\text{LTP}} = 2$) and $\mathcal{F}^i_{\delta t_i}, i = 1, 2, 3$ (classical parareal, non-adaptive, $\Delta T = 1.6$, $N_{\text{LTP}} = 1$) as coarse propagators: speedup at given iterations. All simulations consider $\mathcal{F}^0_{\delta t_0}$ as reference propagator.

posteriori error estimates, as stated by Maday and Mula (2020).

Nevertheless, in some situations it may not be possible to define intermediate propagators. For example, in simulations in urban zones with very small gaps between obstacles, making it impossible to define large computational cells (and thus limiting the time step). In this case, more restrictive parareal configurations, limiting the possible parallelization, *e.g.* with larger values of ΔT and N_{LTP}, would be necessary for ensuring stable and converging solutions, mainly in the ROM-based framework. We can also expect to obtain more accurate solutions by using finer discretizations of the porosity-based models (*e.g.* $\mathcal{G}^1_{\Delta t_1}$, defined in Table 6.1). For illustrating that, we consider the following three non-adaptive parareal simulations, which we call for simplification as configurations A, B and C:

- A: Classical parareal, with $\Delta T = 1.6$ ($N_{\Delta T} = 10$) and $N_{\text{LTP}} = 2$. This simulations has already been considered in this section (see Figure 6.17);
- B: Classical parareal, with $\Delta T = 3.2$ ($N_{\Delta T} = 5$) and $N_{\text{LTP}} = 1$;
- C: ROM-based parareal, with $\Delta T = 0.8$ ($N_{\Delta T} = 20$) and $N_{\text{LTP}} = 4$.

which are used both with $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$ as coarse propagator. Note that, in all cases, exact convergence is reached in $N_{\Delta T}/N_{\text{LTP}} = 5$ iterations. For the simulations with $N_{\text{LTP}} > 1$, we consider $N_{\text{itermax}} = 3$ performed by each parareal instance, and the maximum expected speedup is no larger than $5/3 \approx 1.67$. For the second simulation, that uses a larger time slice and $N_{\text{LTP}} = 1$, we stop the simulation at $N_{\text{itermax}} = 2$, with a maximum speedup no larger than $5/2 = 2.5$.

For all simulations, the error decrease is much slower than in the adaptive simulations, as shown in Figure 6.24. Smaller errors are obtained by using the finer coarse propagator ($\mathcal{G}^1_{\Delta t_1}$), both due to the smaller initial error (at iteration $k = 0$) and to a faster error decrease, compared to the simulations using $\mathcal{G}^0_{\Delta t_0}$, but the convergence behaviour is very similar in both cases. As already observed in previous simulations, the ROM-based configuration provides smaller errors in the beginning of the temporal domain but is unstable for advanced time steps.

In practice, this slower convergence corresponds to a rougher approximation to the reference solution, as shown in Figure 6.25. For all simulations (except for the ROM-based one at $t = T$, due to instabilities), the general profile of the solution (position and amplitude) is well represented, specially by using $\mathcal{G}^1_{\Delta t_1}$ as coarse propagator, and clearly outperforms the coarse one, but small scale oscillations are less well approximated, when compared to adaptive simulations (compare *e.g.* with Figure 6.20). Evidently, this qualitative analysis depends on user expectations in using the parareal method. As shown in Table 6.9, the classical parareal simulation using $N_{\Delta T} = 5$ time slices (configuration B) provides a speedup larger than two, since only two iterations are performed. Also, only small additional costs are observed by using the finer propagator $\mathcal{G}^1_{\Delta t_1}$. Therefore, these relatively restricted parareal configurations can be useful if one expects to obtain stable and more accurate approximations, even if still relatively rough, to the fine, reference solution. In this case, performing few iterations of the classical parareal method using a small number $N_{\Delta T}$ of time slices provides the best compromise. As seen in Table 6.9, the classical and ROM-based parareal configurations using larger time slices and $N_{\text{itermax}} = 3$ (configurations A and C, respectively) provide smaller speedups (mainly the latter, which is more expensive than the reference simulation), and even if they provide smaller errors (at least in part of the temporal domain), no remarkable qualitative improvements are observed.

	Coarse propagator	
Configuration	$\mathcal{G}^0_{\Delta t_0}$	$\mathcal{G}^1_{\Delta t_1}$
A: Classical, $\Delta T = 1.6$, $N_{\text{LTP}} = 2$	1.46	1.38
B: Classical, $\Delta T = 3.2$, $N_{\text{LTP}} = 1$	2.12	2.09
C: ROM-based, $\Delta T = 0.8$, $N_{\text{LTP}} = 4$	0.64	0.47

Table 6.9: Test case 3 with non-adaptive configuration: speedup $s(N_{\text{itermax}})$ at the final iteration of each simulation. All simulations consider $\mathcal{F}^0_{\delta t_0}$ as reference propagator.

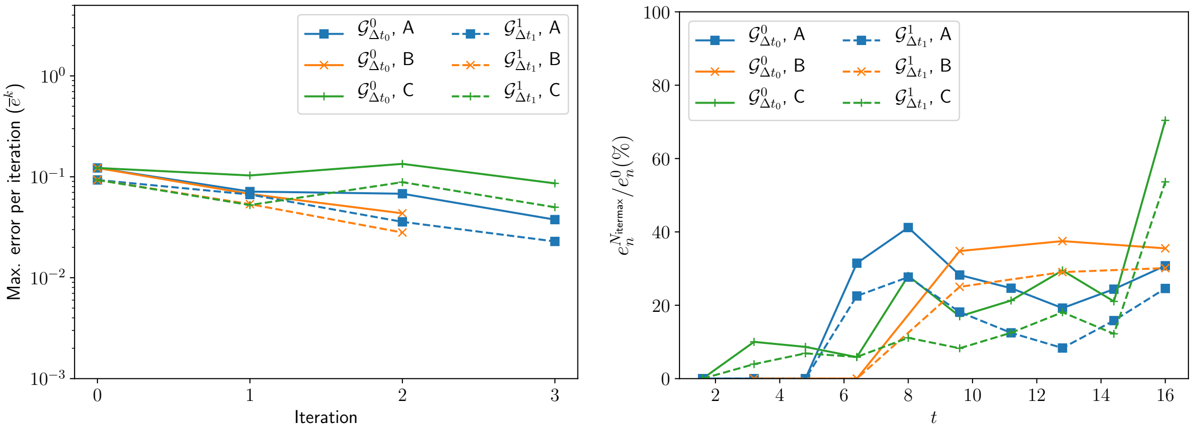


Figure 6.24: Test case 3 with non-adaptive configurations: evolution of the relative error $\overline{e}^k$ along iterations (left) and fraction of the error at the last parareal iteration relative to the error at the $0-$th iteration, for each parareal time instant ($e_n^{N_{\text{itermax}}}/e_n^0$), with values expressed in percentage (right). Full and dashed lines correspond to simulations using respectively $\mathcal{G}^0_{\Delta t_0}$ and $\mathcal{G}^1_{\Delta t_1}$ as coarse propagator. All simulations consider $\mathcal{F}^0_{\delta t_0}$ as reference propagator.

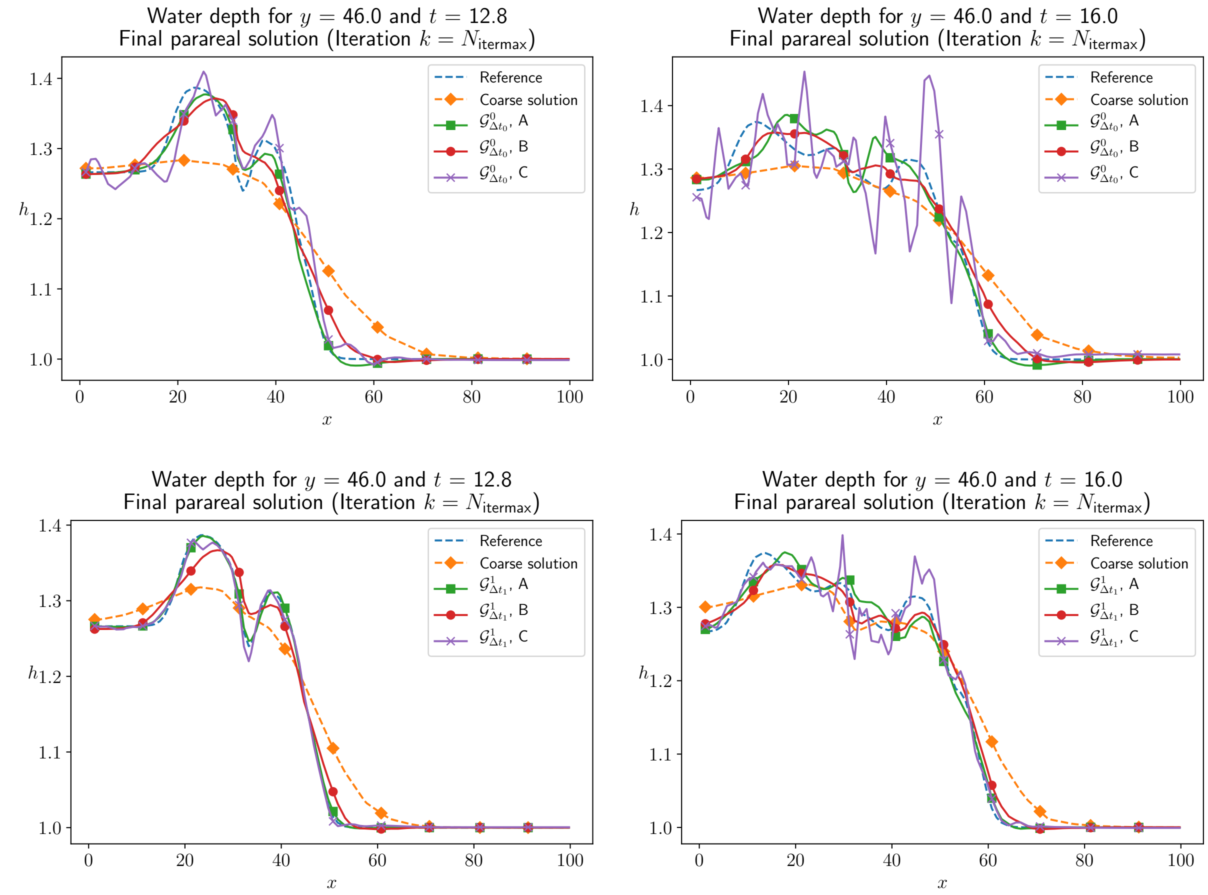


Figure 6.25: Test case 3 with non-adaptive configurations: water depth along $y = 46$, at $t = 4T/5$ (left) and $t = T$ (right), for the reference, coarse (0-th parareal iteration) and parareal solutions, using $\mathcal{G}^0_{\Delta t_0}$ (top) and $\mathcal{G}^1_{\Delta t_1}$ (bottom) as coarse propagator. All simulations consider $\mathcal{F}^0_{\delta t_0}$ as reference propagator.

6.6 Towards more challenging examples

After an initial investigation using relatively simple test cases, solved within small time domains and allowing to study the behaviour of the classical and ROM-based parareal methods for the coupling proposed in this work, we move towards more challenging simulations, by going back to the numerical example presented in Section 2.11 for illustrating the classical and porosity-based shallow water models. Note that this example is less expensive than Test case 3 with $\mathcal{F}^0_{\delta t_0}$ as reference propagator, due to the use of locally refined mesh only near the urban zone (instead of an homogeneous Cartesian mesh). However, it is solved for a larger final time $T = 120$ (against $T = 16$ in Test case 3) and uses an unstructured mesh, which may bring additional challenges, mainly in the context of the model reduction, due to discontinuities of the flow variables induced by the orientation of mesh interfaces, as discussed in Section 3.6.7.

We recall and summarise the configurations for this simulation, to which we refer hereafter as Test case 4, in Table 6.10. The reference (classical SWE) and porosity-based models (porosity-based SWE, either SP or DIP) are named respectively as $\mathcal{F}_{\delta t}$ and $\mathcal{G}_{\Delta t}$. For the purposes of studying the convergence of the parareal method using a coarser discretization of the classical SWE as coarse propagator, we also define a third propagator $\mathcal{F}^1_{\delta t_1}$, whose computational mesh is illustrated in Figure 6.26 and defined by mesh diameters equal to 10 and 2.5 respectively outside and inside the urban zone. The time steps δt (for $\mathcal{F}_{\delta t}$) , δt_1 (for $\mathcal{F}^1_{\delta t_1}$) and Δt (for $\mathcal{G}_{\Delta t}$) are chosen to be multiple integers of each other and keep close CFL numbers for all propagators.

We consider, as in the example presented in Section 2.11, a lake-in-rest as initial conditions, but with various initial water depths:

Spatial domain		$\Omega = [0, 500] \times [0, 45]$	
Maximum simulation time		$T = 120$	
Number of parallel processors		$N_p = 20$	
Maximum number of iterations		$N_{\text{itermax}} = 5$	
	$\mathcal{F}_{\delta t}$	$\mathcal{F}^1_{\delta t_1}$	$\mathcal{G}_{\Delta t}$
Time step	$\delta t_0 = 0.05$	$\delta t_1 = 0.25$	$\Delta t = 0.5$
Minimum mesh size	1	2.5	11.875

Table 6.10: Configurations of test case 4.

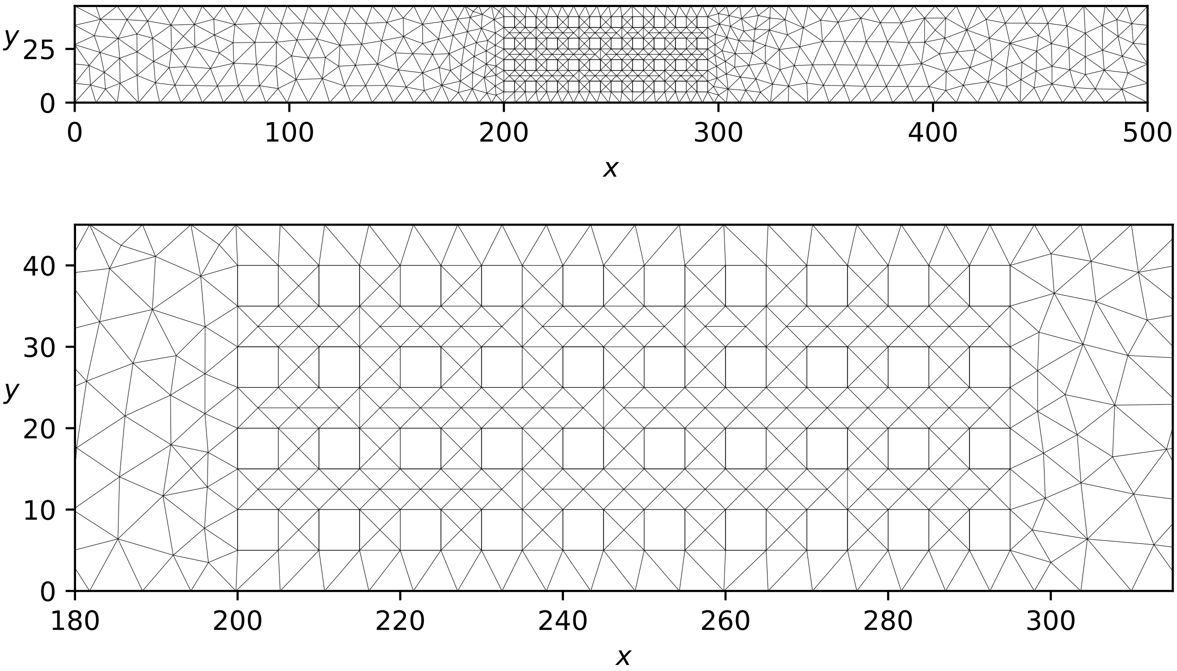


Figure 6.26: Test case 4: computational mesh used by propagator $\mathcal{F}^1_{\delta t_1}$ (classical SWE). Top: full domain; bottom: zoom on the urban zone.

$$h(\boldsymbol{x}, t = 0) = h_0, \qquad u_x(\boldsymbol{x}, t = 0) = u_y(\boldsymbol{x}, t = 0) = 0, \qquad \boldsymbol{x} \in \Omega$$

with $h_0 \in \{0, 0.1, 0.5\}$. The boundary conditions are also kept the same as before (inward flow in the western boundary and imposed water depth in the eastern one), but with the imposed water depth equal to h_0:

$$\begin{cases} h\boldsymbol{u} \cdot \boldsymbol{n} = 1, & \boldsymbol{x} \in \partial\Omega_{\text{inward}}, t \in [0, T] \\ h = h_0, & \boldsymbol{x} \in \partial\Omega_{\text{outward}}, t \in [0, T] \\ h\boldsymbol{u} \cdot \boldsymbol{n} = 0, & \boldsymbol{x} \in \partial\Omega \backslash (\partial\Omega_{\text{inward}} \cup \partial\Omega_{\text{outward}}), t \in [0, T] \end{cases}$$

In what follows, we compare the performance of some parareal configurations for each of the initial water depths.

6.6.1 Initial water depth $h_0 = 0.5$

Simulation using the classical parareal method

For the classical parareal method, we perform a global-in-time parareal simulations ($N_{\text{LTP}} = 1$) with various time slice lengths. We notice that relative large time slices are necessary for ensuring stability. When a porosity-based model is used as coarse propagator ($\mathcal{G}_{\Delta t}$-SP or $\mathcal{G}_{\Delta t}$-DIP), simulations with more than $N_{\Delta T} = 10$ time slices (*i.e.* with $\Delta T < 12$) are unstable and not able to complete all iterations. When the coarser discretization $\mathcal{F}^1_{\delta t_1}$ of the classical SWE is used as coarse propagator, simulations using smaller time slices can complete all iterations, since, even if still present, instabilities are less important and do not lead to negative water depths.

Figure 6.27 presents the evolution of the maximum error per iteration for each simulation. In the case of $\mathcal{G}_{\Delta t}$ being used as coarse propagator, a small range of numbers of time slices is considered and, for a $N_{\Delta T}$ fixed, we observe that similar convergence behaviours are obtained when using the SP and DIP models, with a slightly better performance with SP. As illustrated in Figures 2.11-2.12, in this test case the DIP model gives a better representation of the water depth field, whereas the SP model approximates better the amplitude of the x-unit discharge field inside the urban zone, but overestimates on the downstream, compared to DIP. As a consequence, the errors of the two models, w.r.t. the reference solution, are relatively close (see the 0−th iteration in Figure 6.27), and also close results are obtained in the following iterations. Concerning the simulations using $\mathcal{F}^1_{\delta t_1}$ as coarse propagator, a larger variety of time slices lengths can be used. Except for $\Delta T = 1$ ($N_{\Delta T} = 120$), the error decreases along iterations for all values of ΔT, with a faster convergence for larger ΔT. Note that, for the same value of ΔT, smaller errors are obtained with $\mathcal{F}^1_{\delta t_1}$ as coarse propagator than with $\mathcal{G}_{\Delta t}$-SP or $\mathcal{G}_{\Delta t}$-DIP.

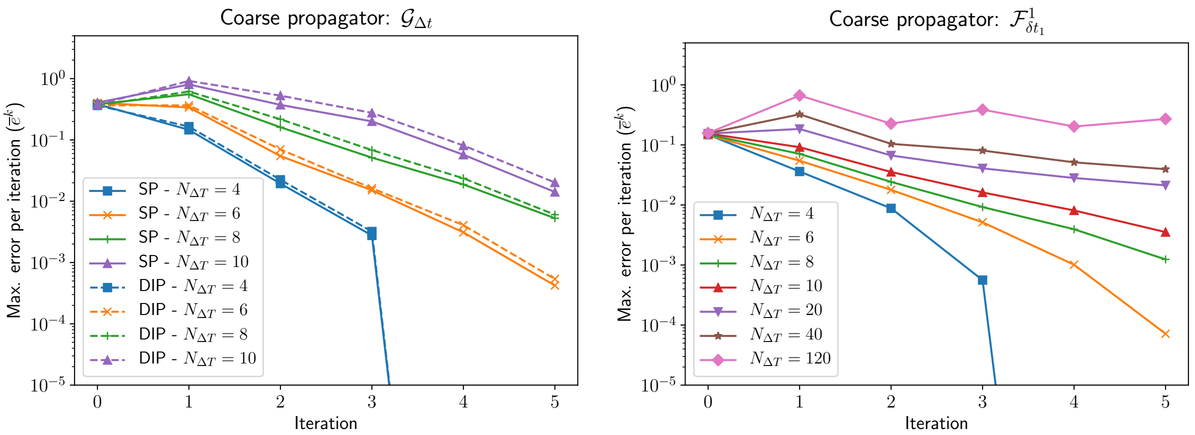


Figure 6.27: Test case 4 with $h_0 = 0.5$: evolution of the relative error $\overline{e}^k$ for various time slice lengths using the classical parareal method. Left: $\mathcal{G}_{\Delta t}$-SP (full lines) and $\mathcal{G}_{\Delta t}$-DIP (dashed lines) used as coarse propagator; right: $\mathcal{F}^1_{\delta t_1}$ used as coarse propagator.

For comparing the simulations in terms of speedup, we define, analogously to (4.26)-(4.28), a quantity k_ε corresponding to the number of iterations such that the maximum error at this iteration ($\overline{e}^{k_\varepsilon}$) is no larger than a given accuracy ε, and a quantity s_ε corresponding to the speedup for completing k_ε iterations:

$$k_\varepsilon := \min K_\varepsilon = \min\{0 \leq k \leq N_{\text{itermax}} \mid \overline{e}^k \leq \varepsilon\}$$

$$s_\varepsilon := s(k_\varepsilon), \qquad K_\varepsilon \neq \emptyset$$

The values of k_ε and s_ε for given time slice lengths are presented respectively in Tables 6.11 and 6.12. For the majority of cases, the simulations using $\mathcal{F}^1_{\delta t_1}$ as coarse propagator reach the given accuracies in less iterations than with $\mathcal{G}_{\Delta t}$, thus providing larger speedups, despite of a more expensive coarse simulation along parareal iterations.

Figure 6.28 illustrates the final water depth along $y = 22.5$ at the first three parareal iterations, compared to the coarse and reference solutions, for chosen time slice lengths ($N_{\Delta T} \in \{6, 8\}$ for $\mathcal{G}_{\Delta t}$-SP and $\mathcal{G}_{\Delta t}$-DIP as coarse propagator, and $N_{\Delta T} \in \{10, 20\}$ for $\mathcal{F}^1_{\delta t_1}$ as coarse propagator). Note that an important degradation of the solution is observed in the first iteration, mainly for the simulations using $\mathcal{G}_{\Delta t}$ as coarse propagators. The quality of the solution is improved in the following iterations, and for $k = 3$ and $N_{\Delta T} = 6$, the parareal solution is visually very close to the reference one. Also good approximations, with only small misrepresentations, are obtained for $k = 3$ and $N_{\Delta T} = 8$. As indicated in Tables 6.11 and 6.12, these results are obtained with speedups of approximately 1.66 (for $N_{\Delta T} = 6$)

		Coarse propagator: $\mathcal{G}_{\Delta t}$				Coarse propagator: $\mathcal{F}^1_{\delta t_1}$			
ε		1E-1	5E-2	1E-2	5E-3	1E-1	5E-2	1e-2	5E-3
$N_{\Delta T}$	ΔT								
4	30	2	2	3	3	1	1	2	3
6	20	2	3	4	4	1	2	3	4
8	15	3	4	5	#	1	2	3	4
10	12	4	5	#	#	1	2	4	5
20	6	*	*	*	*	2	3	#	#

Table 6.11: Test case 4: number of iterations k_ε for reaching given accuracies ε, for various time slice lengths using $\mathcal{G}_{\Delta t}$ and $\mathcal{F}^1_{\delta t_1}$ as coarse propagators, in the classical parareal framework. The speedups for $\mathcal{G}_{\Delta t}$ refer to $\mathcal{G}_{\Delta t}$-SP, and the ones for $\mathcal{G}_{\Delta t}$-DIP are similar and omitted for the sake of conciseness. "#" indicates that the given accuracy is not reached within $N_{\text{itermax}} = 5$ iterations and asterisks indicate that the solution is unstable and does not complete N_{itermax} iterations.

		Coarse propagator: $\mathcal{G}_{\Delta t}$				Coarse propagator: $\mathcal{F}^1_{\delta t_1}$			
ε		1E-1	5E-2	1E-2	5E-3	1E-1	5E-2	1E-2	5E-3
$N_{\Delta T}$	ΔT								
4	30	1.72	1.72	1.18	1.18	2.39	2.39	1.39	1.00
6	20	2.40	1.66	1.27	1.27	2.95	1.78	1.28	1.02
8	15	1.97	1.53	1.26	#	3.41	2.09	1.53	1.21
10	12	1.89	1.53	#	#	3.55	2.24	1.31	1.11
20	6	*	*	*	*	3.12	2.35	#	#

Table 6.12: Test case 4 with $h_0 = 0.5$: speedup s_ε for reaching given accuracies ε, for various time slice lengths using $\mathcal{G}_{\Delta t}$ and $\mathcal{F}^1_{\delta t_1}$ as coarse propagators, in the classical parareal framework. The speedups for $\mathcal{G}_{\Delta t}$ refer to $\mathcal{G}_{\Delta t}$-SP, and the ones for $\mathcal{G}_{\Delta t}$-DIP are similar and omitted for the sake of conciseness. "#" indicates that the given accuracy is not reached within $N_{\text{itermax}} = 5$ iterations and asterisks indicate that the solution is unstable and does not complete N_{itermax} iterations.

and 1.97 (for $N_{\Delta T} = 8$). In the cases where $\mathcal{F}^1_{\delta t_1}$ is used as coarse propagator, the degradation of the solution in the first iteration is much less important, and relatively good approximations of the reference solution are obtained after only two parareal iterations, both with $N_{\Delta T} = 10$ and $N_{\Delta T} = 20$ time slices, and corresponding respectively to speedups of 2.24 and 3.12.

The better performance of the classical parareal method using $\mathcal{F}^1_{\delta t_1}$ rather than $\mathcal{G}_{\Delta t}$ can naturally be attributed to the fact that the former is closer to the reference propagator, by using finer spatial and temporal discretizations and also by discretizing the classical SWE (instead of the porosity-based models), which lead to a better representation of the velocity fields, since the obstacles in the domain are physically taken into account in the computational mesh. Due to the unidirectionality of the flow, the use of a Cartesian mesh and the independence of the porosity fields w.r.t. y-direction (see Figure 2.10), the y-unit discharge provided by $\mathcal{G}_{\Delta t}$-SP and $\mathcal{G}_{\Delta t}$-DIP is identically zero in the entire temporal domain, whereas the reference y-unit discharge is highly discontinuous (see Figure 2.12). We choose two simulations ($\mathcal{G}_{\Delta t}$-SP as coarse propagator, with $N_{\Delta T} = 6$; and $\mathcal{F}^1_{\delta t_1}$ as coarse propagator, with $N_{\Delta T} = 10$) for illustrating, in Figure 6.29, the more challenging convergence of the unit discharge fields in the former case. With $\mathcal{G}_{\Delta t}$, we notice a very important degradation of the x-unit discharge field along $y = 22.5$ in the first iteration, including the formation of negative velocities u_x, whereas the reference one is non-negative along the entire slice. The quality of the solution increases in the following iterations, but we notice that three iterations are required for well approximating the amplitudes of the x- and y- unit discharges. When $\mathcal{F}^1_{\delta t_1}$ is used as coarse propagator, the unit discharges remain close to the reference ones in the first iteration and a good approximation is obtained in the second one.

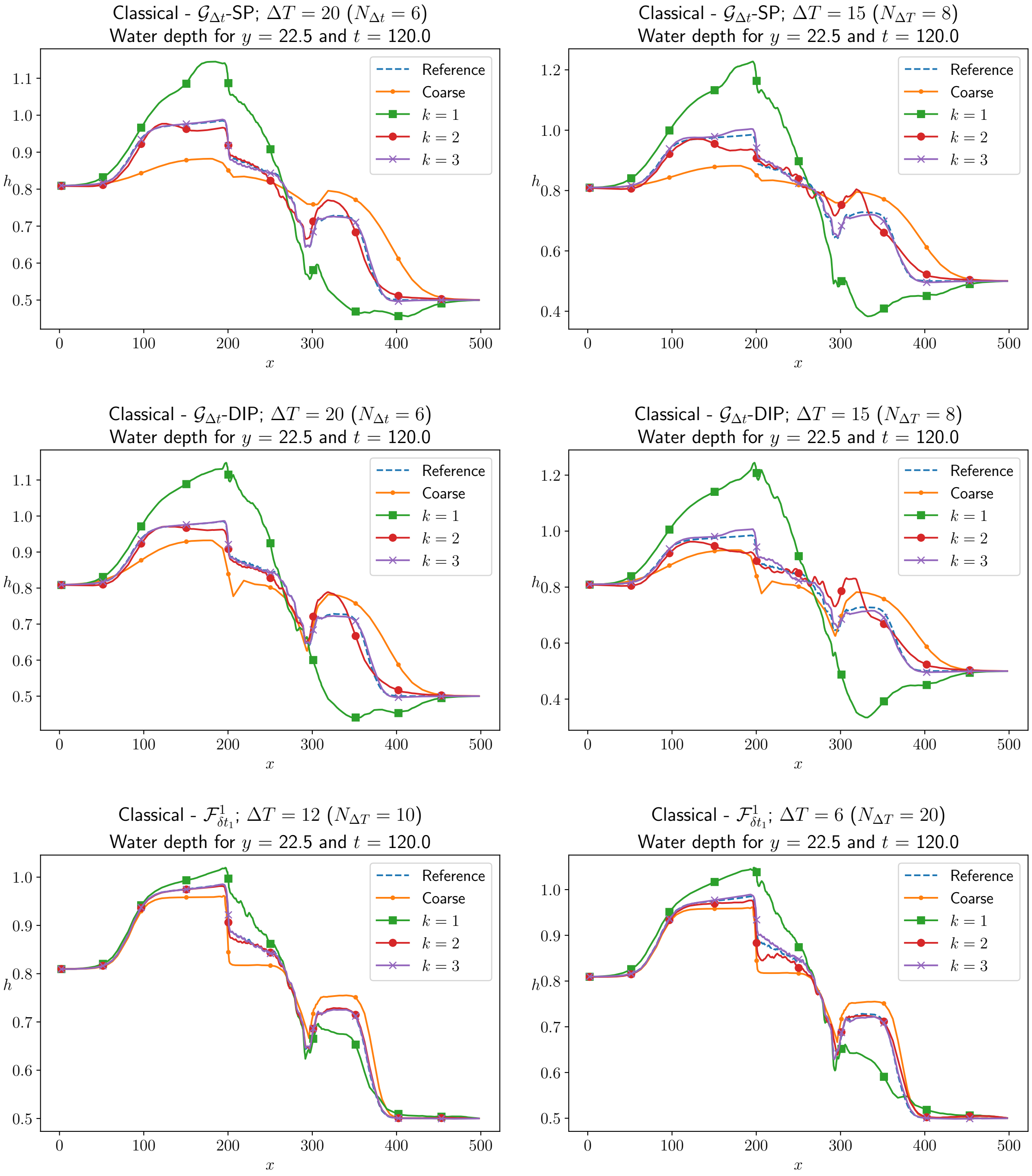


Figure 6.28: Test case 4 with $h_0 = 0.5$: water depth along $y = 46$, at $t = T$, for the reference and coarse solution (0-th parareal iteration) and first three parareal iterations, using the classical parareal method. First row: $\mathcal{G}_{\Delta t}$-SP as coarse propagator, with $N_{\Delta T} = 6$ (left) and $N_{\Delta T} = 8$ (right) time slices. Second row: $\mathcal{G}_{\Delta t}$-DIP as coarse propagator, with $N_{\Delta T} = 6$ (left) and $N_{\Delta T} = 8$ (right) time slices. Third row: $\mathcal{F}^1_{\delta t_1}$ as coarse propagator, with $N_{\Delta T} = 10$ (left) and $N_{\Delta T} = 20$ (right) time slices.

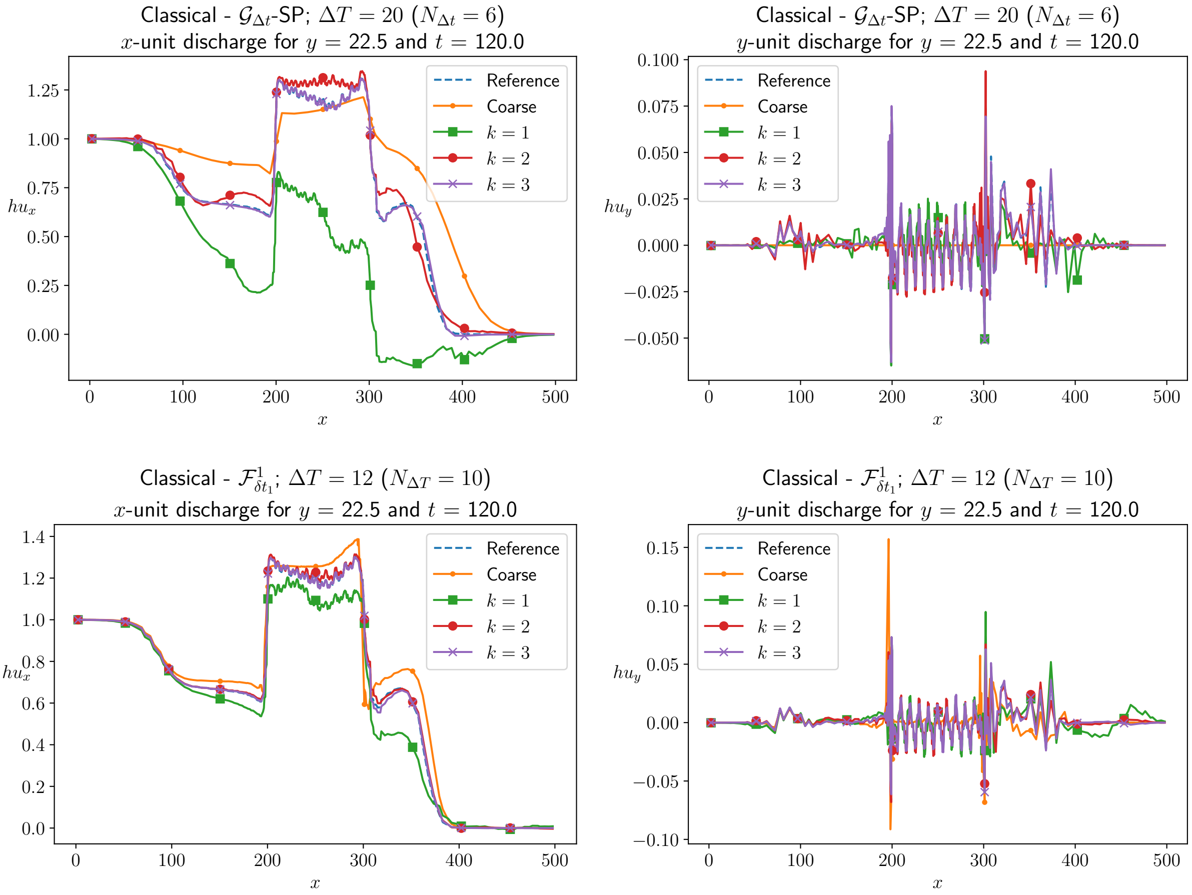


Figure 6.29: Test case 4 with $h_0 = 0.5$: x- (left) and y- (right) unit discharges along $y = 46$, at $t = T$, for the reference and coarse solution (0-th parareal iteration) and first three parareal iterations, using the classical parareal method. First row: $\mathcal{G}_{\Delta t}$-SP as coarse propagator, with $N_{\Delta T} = 6$ time slices. Second row: $\mathcal{F}^1_{\delta t_1}$ as coarse propagator, with $N_{\Delta T} = 10$ time slices.

Simulation using the ROM-based parareal method

As can naturally be expected from the results present in the previous sections (Test case 3), the application of the ROM-based parareal method for solving Test case 4 is particularly challenging in terms of stability, due to the relatively long temporal domain and complex behaviour of the solution induced by the presence of obstacles. Also, the performance of the ROM-based method is expected to be compromised even in the beginning of the simulation, before the incoming wave reaches the urban zone, since the one-dimensional reference inflow is, numerically, a discontinuous two-dimensional one due to the use of an unstructured mesh.

Indeed, it is observed that stable simulations are only obtained by considering very restrictive configurations for the ROM-based method. Namely, very low-dimensional ROMs need to be considered, by taking $\varepsilon_{\text{sv,linear}} = 10^{-1}$. It relatively ensures stability, since further POD modes that lead to instabilities are discarded, but strongly affects the parareal convergence, as discussed in Section 3.6. Therefore, in the simulations presented below, we consider $\varepsilon_{\text{sv,linear}} = 10^{-1}$ and $\varepsilon_{\text{sv,nonlinear}} = 10^{-5}$. Concerning the other configurations of the ROM-based method, we take $\alpha_{\text{s}} = 1$ and $N_{\text{PID}} = 1$. We do not consider an enrichment of the snapshots sets motivated by the observation, made in Section 6.5, that $\alpha_{\text{s}} < 1$ may lead to unstable behaviours for approximating highly discontinuous solutions.

We begin by comparing the performance of the ROM-based parareal method in function of N_{LTP} and $N_{\Delta T}$. We consider the pairs $(N_{\text{LTP}}, N_{\Delta T}) \in \{(1,4),(2,8),(5,20)\}$. In all cases, exact convergence is attained in $N_{\text{LTP}}/N_{\Delta T} = 4$ iterations, and we set $N_{\text{itermax}} = 3$. Simulations are performed using $\mathcal{G}_{\Delta t}$-SP and $\mathcal{F}^1_{\delta t_1}$ as coarse propagator, and all of them are able to complete N_{itermax} iterations, except the case ($\mathcal{G}_{\Delta t}$, $N_{\text{LTP}} = 5$, $N_{\Delta T} = 20$). Figure 6.30 presents the evolution of the relative error $\bar{e}^k$, showing that the use of local-in-time parareal instances does not contribute for improving the stability and convergence, with $N_{\text{LTP}} = 1$ providing better results. Contrary to Test case 3, presented in the previous sections, in Test case 4 there is no region of the temporal domain in which the convergence of the ROM-based method is expected to be remarkably less challenging, due to the use of unstructured meshes, as discussed above. Therefore, performing parareal simulations is small time windows seems to be less determinant for ensuring convergence and stability than the use of large time slices.

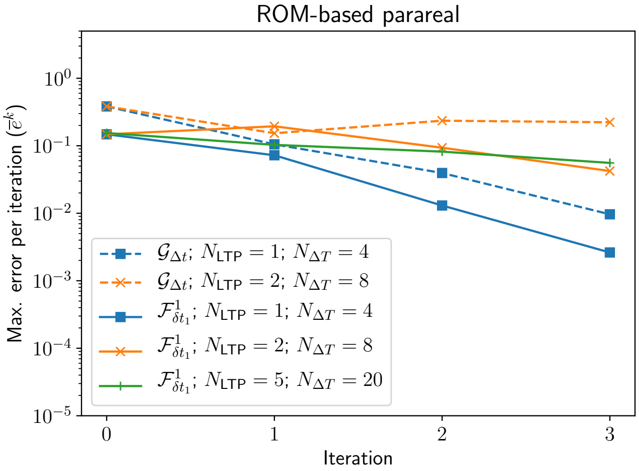


Figure 6.30: Test case 4 with $h_0 = 0.5$: evolution of the relative error $\bar{e}^k$ for various time slice lengths and numbers of LTP windows using the ROM-based parareal method, with $\mathcal{G}_{\Delta t}$-SP (dashed lines) and $\mathcal{F}^1_{\delta t_1}$ (full lines) used as coarse propagator. The simulation with ($\mathcal{G}_{\Delta t}$, $N_{\text{LTP}} = 5$, $N_{\Delta T} = 20$) is unstable and does not complete $N_{\text{itermax}} = 3$ iterations.

Therefore, we consider $N_{\text{LTP}} = 1$ hereafter and we compare the performances obtained by the classical and ROM-based parareal methods using the same configurations. We notice that, in the latter case, the range of ΔT for which the simulations complete all iterations ($N_{\text{itermax}} = 5$) is much smaller than in the former. Indeed, among the values of $N_{\Delta T}$ considered in the classical method, only $N_{\Delta T} = 4$ ($\Delta T = 30$) provides stable simulations in the ROM-based method using $\mathcal{G}_{\Delta t}$-SP and $\mathcal{G}_{\Delta t}$-DIP as coarse propagator. When $\mathcal{F}^1_{\delta t_1}$ is used as coarse propagator, a few more configurations are able to complete the simulation, namely $N_{\Delta T} \in \{4, 6, 8, 10\}$ (respectively $\Delta T \in \{30, 20, 15, 12\}$), which is, however, a range still restricted when compared to the one used in the classical method. Figure 6.31 compares the evolution of the relative errors $\bar{e}^k$ using the classical and ROM-based methods with the same numbers of time slices. In all cases, the error obtained with the ROM-based method is larger, except for the first iteration using $N_{\Delta T} = 4$ and $\mathcal{G}_{\Delta t}$ as coarse propagator. When $\mathcal{F}^1_{\delta t_1}$ is used as coarse propagator, the error does not decrease along iterations for $N_{\Delta T} \geq 8$.

We choose some of the performed simulations for illustrating the obtained physical solutions. Figure 6.32 compares the final water depth along $y = 22.5$ at the first three parareal iterations obtained in the classical and ROM-based frameworks with $\mathcal{G}_{\Delta t}$ as coarse propagator and $N_{\Delta T} = 4$ time slices; and with $\mathcal{F}^1_{\delta t_1}$ as coarse propagator and $N_{\Delta T} \in \{4, 6\}$ time slices. In all cases, the solution obtained in the first iteration of the ROM-based simulations visually outperforms the solution of the classical method, in terms of amplitude of the solution, far away from the left and right domain boundaries, but misrepresents the solution near them. After two iterations, all solutions are visually close to the reference one far away from the boundaries. This misrepresentation close to the boundaries may be linked to the fact that, in general, POD-based ROMs do not satisfy inhomogeneous boundary conditions, an issue for which alternative approaches are proposed in the literature, *e.g.* the introduction of penalty terms in the equations and the subtraction of the boundary conditions from the input snapshots sets (Stabile et al., 2017). The causes explaining why this behaviour is observed in Test case 4, but not in the previous simulations in this work, are not clear. They are possibly linked to the use of very low-dimensional POD ROMs in Test case 4. A more detailed investigation should be made, also considering the fact that boundary conditions are taken into account in the POD-EIM step of the model reduction considered in this work.

Therefore, even if the ROM-based parareal method provides visually better solutions in parts of the spatial domain, the classical method remains more attractive, since given accuracies are reached within the same or smaller numbers of iterations, as shown in Tables 6.13 and 6.14. When $\mathcal{G}_{\Delta t}$ is used as coarse propagator, two iterations of the classical and ROM-based methods provide respectively speedups of 1.72 and 1.35. Note that, when $\mathcal{F}^1_{\delta t_1}$ is used as coarse propagator, the ROM-based parareal method provides, for the same number of iterations, very close speedups compared to the classical one, since the relatively expensive coarse propagator $\mathcal{F}^1_{\delta t_1}$ is replaced by a very low-expensive reduced-order model (due to the large threshold $\varepsilon_{\text{sv,linear}} = 10^{-1}$), and the model reduction remains relatively low-expensive due to the large time slice length ΔT and the use of $\alpha_s = 1$. However, the classical parareal method is still more interesting, mainly because less iterations are required for attaining smaller ε.

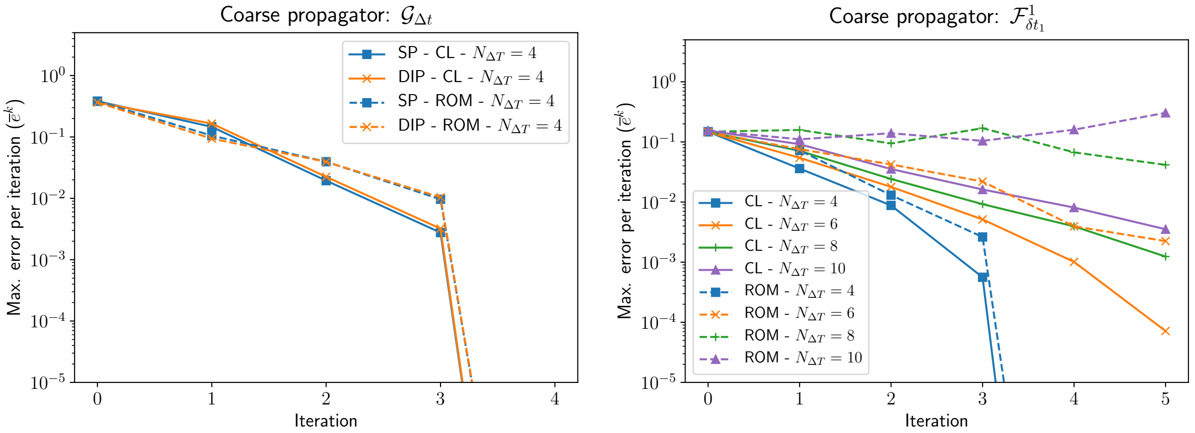


Figure 6.31: Test case 4 with $h_0 = 0.5$: evolution of the relative error $\bar{e}^k$ for various time slice lengths using the classical ("CL", full lines) and ROM-based ("ROM", dashed lines) parareal method. Left: $\mathcal{G}_{\Delta t}$-SP and $\mathcal{G}_{\Delta t}$-DIP used as coarse propagator; right: $\mathcal{F}^1_{\delta t_1}$ used as coarse propagator.

			Coarse propagator: $\mathcal{G}_{\Delta t}$				Coarse propagator: $\mathcal{F}^1_{\delta t_1}$			
	ε		1E-1	5E-2	1E-2	5E-3	1E-1	5E-2	1e-2	5E-3
	$N_{\Delta T}$	ΔT								
CL	4	30	2	2	3	3	1	1	2	3
	6	20	2	3	4	4	1	2	3	4
ROM	4	30	2	2	3	4	1	2	3	3
	6	20	*	*	*	*	1	2	4	4

Table 6.13: Test case 4 with $h_0 = 0.5$: number of iterations k_ε for reaching given accuracies ε, for $N_{\text{LTP}} = 1$, various time slice lengths and using $\mathcal{G}_{\Delta t}$ and $\mathcal{F}^1_{\delta t_1}$ as coarse propagators, in the classical ("CL") and ROM-based ("ROM") parareal frameworks. The speedups for $\mathcal{G}_{\Delta t}$ refer to $\mathcal{G}_{\Delta t}$-SP, and the ones obtained with $\mathcal{G}_{\Delta t}$-DIP are similar and omitted for simplicity. Asterisks indicate that the solution is unstable and does not complete N_{itermax} iterations.

			Coarse propagator: $\mathcal{G}_{\Delta t}$				Coarse propagator: $\mathcal{F}^1_{\delta t_1}$			
	ε		1E-1	5E-2	1E-2	5E-3	1E-1	5E-2	1e-2	5E-3
	$N_{\Delta T}$	ΔT								
CL	4	30	1.72	1.72	1.18	1.18	2.39	2.39	1.39	1.00
	6	20	2.40	1.66	1.27	1.27	2.95	1.78	1.28	1.02
ROM	4	30	1.35	1.35	0.92	0.72	2.33	1.28	0.89	0.89
	6	20	*	*	*	*	2.94	1.65	0.89	0.89

Table 6.14: Test case 4 with $h_0 = 0.5$: speedup s_ε for reaching given accuracies ε, for $N_{\text{LTP}} = 1$, various time slice lengths and using $\mathcal{G}_{\Delta t}$ and $\mathcal{F}^1_{\delta t_1}$ as coarse propagators, in the classical ("CL") and ROM-based ("ROM") parareal frameworks. The speedups for $\mathcal{G}_{\Delta t}$ refer to $\mathcal{G}_{\Delta t}$-SP, and the ones obtained with $\mathcal{G}_{\Delta t}$-DIP are similar and omitted for simplicity. Asterisks indicate that the solution is unstable and does not complete N_{itermax} iterations.

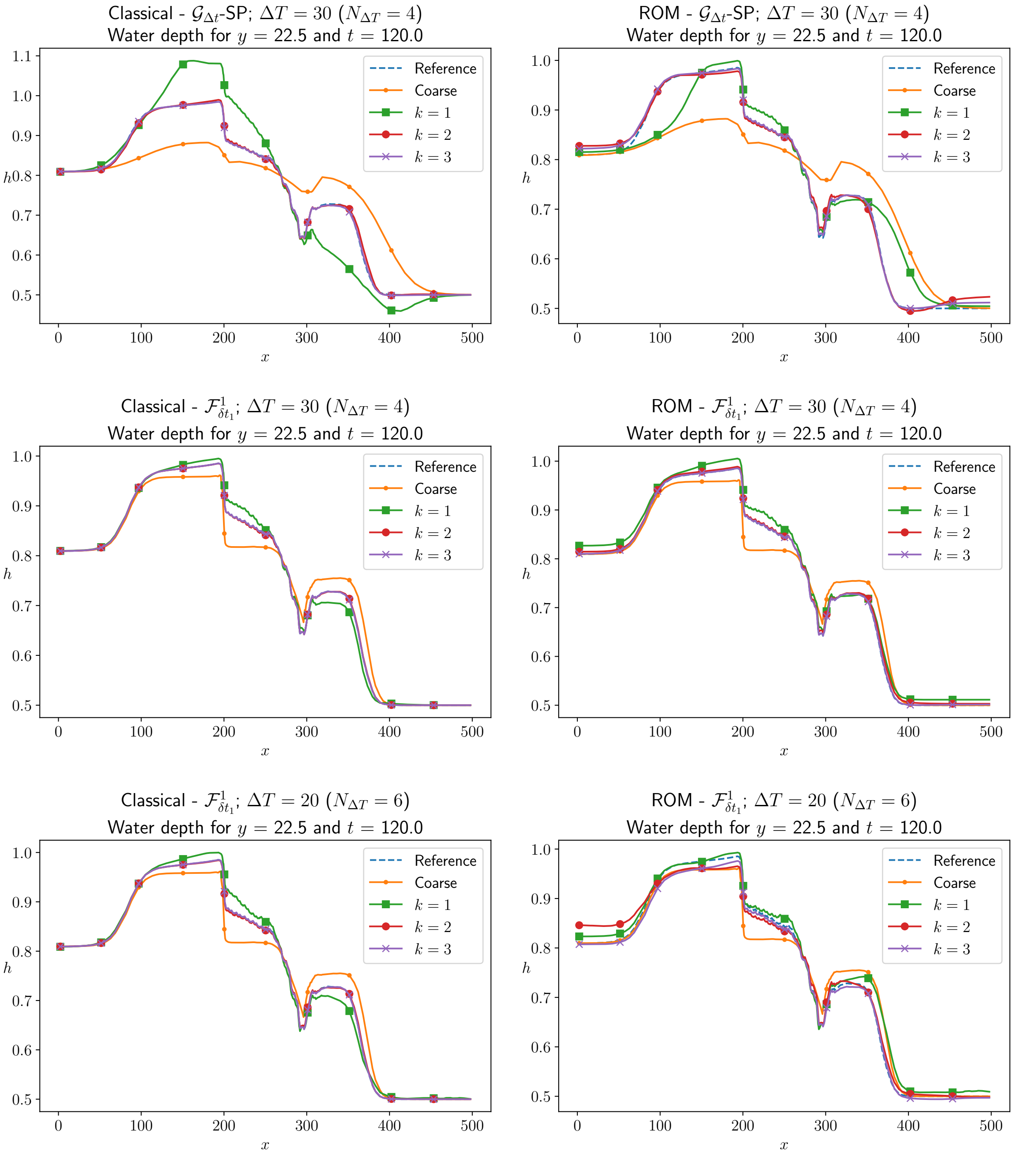


Figure 6.32: Test case 4 with $h_0 = 0.5$: water depth along $y = 46$, at $t = T$, for the reference and coarse solution (0-th parareal iteration) and first three parareal iterations, using the classical (left) and ROM-based (right) parareal methods. First row: $\mathcal{G}_{\Delta t}$-SP as coarse propagator, with $N_{\Delta T} = 4$ time slices. Second row: $\mathcal{F}^1_{\delta t_1}$ as coarse propagator, with $N_{\Delta T} = 4$ time slices. Third row: $\mathcal{F}^1_{\delta t_1}$ as coarse propagator, with $N_{\Delta T} = 6$ time slices.

Influence of the spatial interpolation order

We consider Test case 4 with $h_0 = 0.5$ for coming back to the discussion on the influence of the spatial interpolation order on the performance of the classical and ROM-based parareal methods, briefly introduced in Section 3.6.8. In that occasion, by considering small test cases using the classical SWE both as fine and coarse propagators, it was observed that the convergence of the classical method is improved by using cubic instead of linear interpolation. However, if instabilities are present, no improvements are observed. Moreover, it was illustrated that cubic interpolation has no or even negative influence on the performance of the ROM-based method.

We repeat some of the simulations presented in this section, but using cubic interpolation. Figure 6.33 compares the evolution of the maximum error per iteration compared to those obtained with linear interpolation, for various time slice lengths and both with $\mathcal{G}_{\Delta t}$-SP and $\mathcal{F}^1_{\delta t_1}$ as coarse propagator.

For the simulations using $\mathcal{F}^1_{\delta t_1}$, we present the results for only some values of ΔT, for the sake of clearness. We observe that, with cubic interpolation, a faster convergence is obtained in the cases where parareal already converges in the linear case, and the initial degradation of the solution, in the first iteration, is no longer present (similarly to what was observed in Section 3.6.8 for Test case 2). However, in the case $N_{\Delta T} = 120$, in which parareal with linear interpolation is not able to reduce the error along iterations, there are no visible improvements by using cubic interpolation, indicating that the slow converging and unstable behaviour induced by the small time slice is dominant. A small error reduction is observed in the first iteration, but it increases in the following ones, such that the final error, for $k = 5$, is nearly the same for both interpolation orders.

On the other hand, these results are not verified in the simulations using $\mathcal{G}_{\Delta t}$ as coarse propagator. For a fixed ΔT, the errors with linear and cubic interpolation are nearly identical in the first iterations, and for advanced ones the linear case slightly outperforms. In these simulations, the coarsening intensity between the fine and coarse spatial meshes is relatively important (by a factor of approximately 10 in the urban zone); moreover, the coarse mesh contains a small number of cells (*e.g.* with only four cells in the y-direction, as shown in Figure 2.9). These factors can possibly explain the observed results, since there may be not enough cells for performing a proper cubic interpolation.

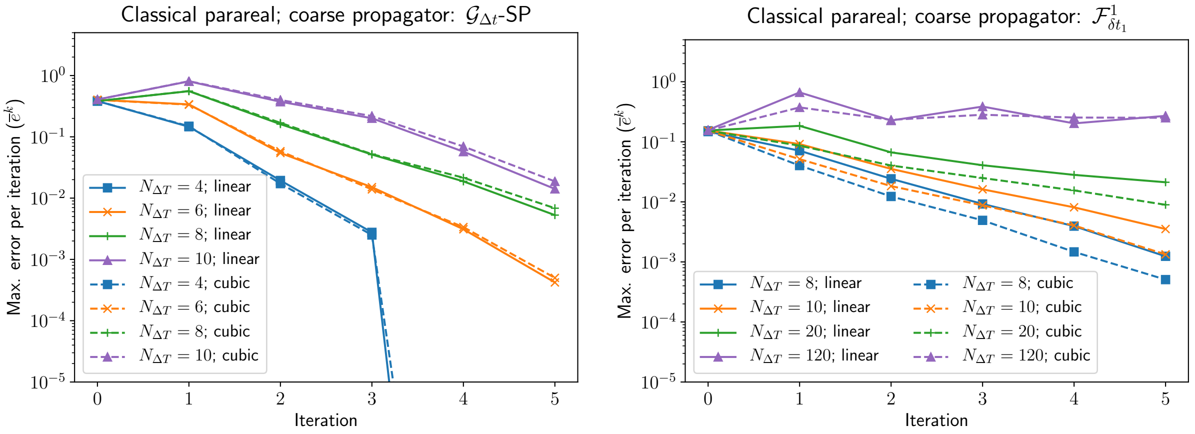


Figure 6.33: Test case 4 with $h_0 = 0.5$: evolution of error $\overline{e}^k$ for various time slice lengths using the classical parareal method, with linear (full lines) and cubic (dashed lines) spatial interpolation. Left: $\mathcal{G}_{\Delta t}$-SP used as coarse propagator; right: $\mathcal{F}^1_{\delta t_1}$ used as coarse propagator.

As discussed in Section 3.6.8, the drawback of using a higher-order spatial interpolation procedure is the increase of the computational cost. It is illustrated in Tables 6.15 and 6.16, showing the number of iterations and respective speedup for reaching given accuracies. For the sake of simplicity, results are presented only for chosen time slices lengths ($N_{\Delta T} \in \{6, 10, 20\}$), the conclusions being similar for other values. In the simulations using $\mathcal{G}_{\Delta t}$ as coarse propagator, the error behaviour is very similar with linear and cubic interpolation, and the same number of iterations are required for attaining given accuracies. Therefore, the simulations with cubic interpolation naturally provide smaller speedup. When $\mathcal{F}^1_{\delta t_1}$ is used as coarse propagator, less iterations, in general, are required with cubic interpolation, but with smaller speedups than in the linear case (with few exceptions, *e.g.* for reaching $\varepsilon = 5 \times 10^{-2}$ with $N_{\Delta T} = 6$). The negative impacts of the cubic case are more important in the simulations using $\mathcal{F}^1_{\delta t_1}$ since interpolation is performed between finer meshes.

Concerning the ROM-based framework, all simulations become highly unstable by using cubic interpolation, with a much more important initial degradation of the solution, and their execution terminates after one or two iterations, both by using $\mathcal{G}_{\Delta t}$ and $\mathcal{F}^1_{\delta t_1}$ as coarse propagator, and even with very large time slices. The influence of the interpolation order on the performance of the ROM-based method remains an unresolved question requiring more detailed investigations.

			Coarse propagator: $\mathcal{G}_{\Delta t}$				Coarse propagator: $\mathcal{F}^1_{\delta t_1}$			
	ε		1E-1	5E-2	1E-2	5E-3	1E-1	5E-2	1e-2	5E-3
	$N_{\Delta T}$	ΔT								
Linear	6	20	2	3	4	4	1	2	3	4
	10	12	4	5	#	#	1	2	4	5
	20	6	*	*	*	*	2	3	#	#
Cubic	6	20	2	3	4	4	1	1	2	3
	10	12	4	5	#	#	1	2	3	4
	20	6	*	*	*	*	1	2	5	#

Table 6.15: Test case 4 with $h_0 = 0.5$ using the classical parareal method: number of iterations k_ε for reaching given accuracies ε, for $N_{\text{LTP}} = 1$, various time slice lengths, using $\mathcal{G}_{\Delta t}$ and $\mathcal{F}^1_{\delta t_1}$ as coarse propagators and with linear and cubic spatial interpolation. The speedups for $\mathcal{G}_{\Delta t}$ refer to $\mathcal{G}_{\Delta t}$-SP, and the ones obtained with $\mathcal{G}_{\Delta t}$-DIP are similar and omitted for simplicity. "#" indicates that the given accuracy is not reached within $N_{\text{itermax}} = 5$ iterations and asterisks indicate that the solution is unstable and does not complete N_{itermax} iterations.

			Coarse propagator: $\mathcal{G}_{\Delta t}$				Coarse propagator: $\mathcal{F}^1_{\delta t_1}$			
	ε		1E-1	5E-2	1E-2	5E-3	1E-1	5E-2	1e-2	5E-3
	$N_{\Delta T}$	ΔT								
Linear	6	20	2.40	1.66	1.27	1.27	2.95	1.78	1.28	1.02
	10	12	1.89	1.53	#	#	3.55	2.24	1.31	1.11
	20	6	*	*	*	*	3.12	2.35	#	#
Cubic	6	20	2.08	1.45	1.13	1.13	2.25	2.25	1.22	0.87
	10	12	1.41	1.15	#	#	2.09	1.17	0.83	0.66
	20	6	*	*	*	*	1.41	0.77	0.35	#

Table 6.16: Test case 4 with $h_0 = 0.5$ using the classical parareal method: speedup s_ε for reaching given accuracies ε, for $N_{\text{LTP}} = 1$, various time slice lengths, using $\mathcal{G}_{\Delta t}$ and $\mathcal{F}^1_{\delta t_1}$ as coarse propagators and with linear and cubic interpolation. The speedups for $\mathcal{G}_{\Delta t}$ refer to $\mathcal{G}_{\Delta t}$-SP, and the ones obtained with $\mathcal{G}_{\Delta t}$-DIP are similar and omitted for simplicity. "#" indicates that the given accuracy is not reached within $N_{\text{itermax}} = 5$ iterations and asterisks indicate that the solution is unstable and does not complete N_{itermax} iterations.

6.6.2 Initial water depth $h_0 = 0.1$

When a smaller initial depth is considered, stability and convergence of the parareal method becomes more challenging, both when $\mathcal{G}_{\Delta t}$ and $\mathcal{F}^1_{\delta t_1}$ are used as coarse propagator, since the degradation of the solution in the first parareal iteration, as observed in several simulations in the case $h_0 = 0.5$, is more likely to produce negative water depths. Indeed, we perform simulations using the same values for $N_{\Delta T}$ as above, and relatively stable ones are obtained only with $N_{\Delta T} = 4$ time slices, whereas all others stop in the very first iteration. With $\mathcal{F}^1_{\delta t_1}$ used as coarse propagator, both simulations using the classical and ROM-based methods are able to complete N_{itermax} iterations, but with $\mathcal{G}_{\Delta t}$ (SP and DIP), only the classical method provides relatively stable results.

Figure 6.34 presents the evolution of $\bar{e}^k$ of these stable simulations, indicating that similar errors are obtained by all of them from the second iteration. It is confirmed in Figure 6.35, showing that the solution for $k = 2$ in all simulations is visually close to the reference one. Note that negative water depths are produced in the first iteration of the simulations using the porosity-based models; however, since they arise only in the final time of simulation and are not further propagated in time, the code execution is not interrupted and this unstable behaviour is controlled in the following parareal iterations. The speedups obtained in the simulations are similar to those observed previously with $h_0 = 0.5$ and $N_{\Delta T} = 4$ and are not presented.

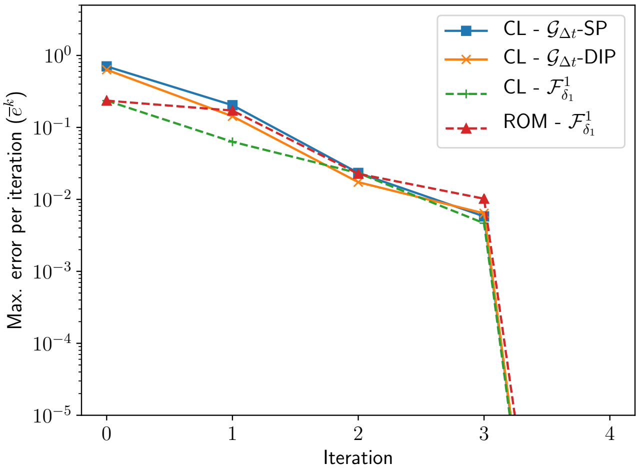


Figure 6.34: Test case 4 with $h_0 = 0.1$: evolution of error $\bar{e}^k$ using $\mathcal{G}_{\Delta t}$ (full lines) and $\mathcal{F}^1_{\delta t_1}$ as coarse propagator. In the former case, only the classical parareal method ("CL") provides stable simulations. All simulations use $N_{\Delta T} = 4$ time slices, and exact convergence is obtained after four iterations.

6.6.3 Initial water depth $h_0 = 0$

As can be expected from the previous results using decreasing initial water depths, none of the parareal configuration is able to provide stable simulations when $h_0 = 0$, since negative water depth are quickly produced. However, even if the challenging and unstable behaviour of the method when applied to the problems considered here contribute to these unsatisfactory results, it should to be noticed that they are also related to a more general drawback of the parareal method. Indeed, the parareal iteration (3.8) does not necessarily preserve the positivity of the solution (Legoll et al., 2020). In a general framework, no guarantees on the sign of the correction term $\mathcal{F}_{\delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n) - \mathcal{G}_{\Delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)$ can be obtained, and if the prediction term $\mathcal{G}_{\Delta t}(\boldsymbol{y}_n^{k+1}, t_{n+1}, t_n)$ is zero or close to zero, negative solutions can arise in the predictor-corrector procedure.

Some alternatives exist for overcoming this issue. In (Legoll et al., 2020), the parareal method is used for solving stochastic differential equations, whose solutions are probability density functions, thus supposed to be positive and with unit integral over their domain of definition (unit mass). The authors propose several approaches for satisfying these requirements: set negative values to zero and rescale for satisfying unit mass; replace the additive predictor-corrector iteration (3.8) by a multiplicative one, *i.e.* $\mathcal{G}_{\Delta t}(\boldsymbol{y}_n^{k+1}, t_{n+1}, t_n) \times \mathcal{F}_{\delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)/\mathcal{G}_{\Delta t}(\boldsymbol{y}_n^k, t_{n+1}, t_n)$; also followed by a rescaling; replace (3.8) by a norm-preserving rotation operator (an approach also proposed by Maday and Turinici (2002a)); and the use of the parareal iteration to update auxiliary statistical functions not having positivity or norm restrictions. It is illustrated that the parareal convergence strongly depends on the chosen approach, with the last one providing considerable better results.

Positivity preservation of water depths is a required property on the simulation of shallow water models and is particularly challenging in the case of urban floods since zero water depths, at least in part of the spatial domain, would be naturally considered in real applications. Therefore, some of the modifications listed above for the parareal method, such as setting negative water depths to zero or using a multiplicative parareal update, could be considered for overcoming these difficulties, accompanied by proper rescalings for ensuring mass and momentum conservation or other desired properties of the numerical solution.

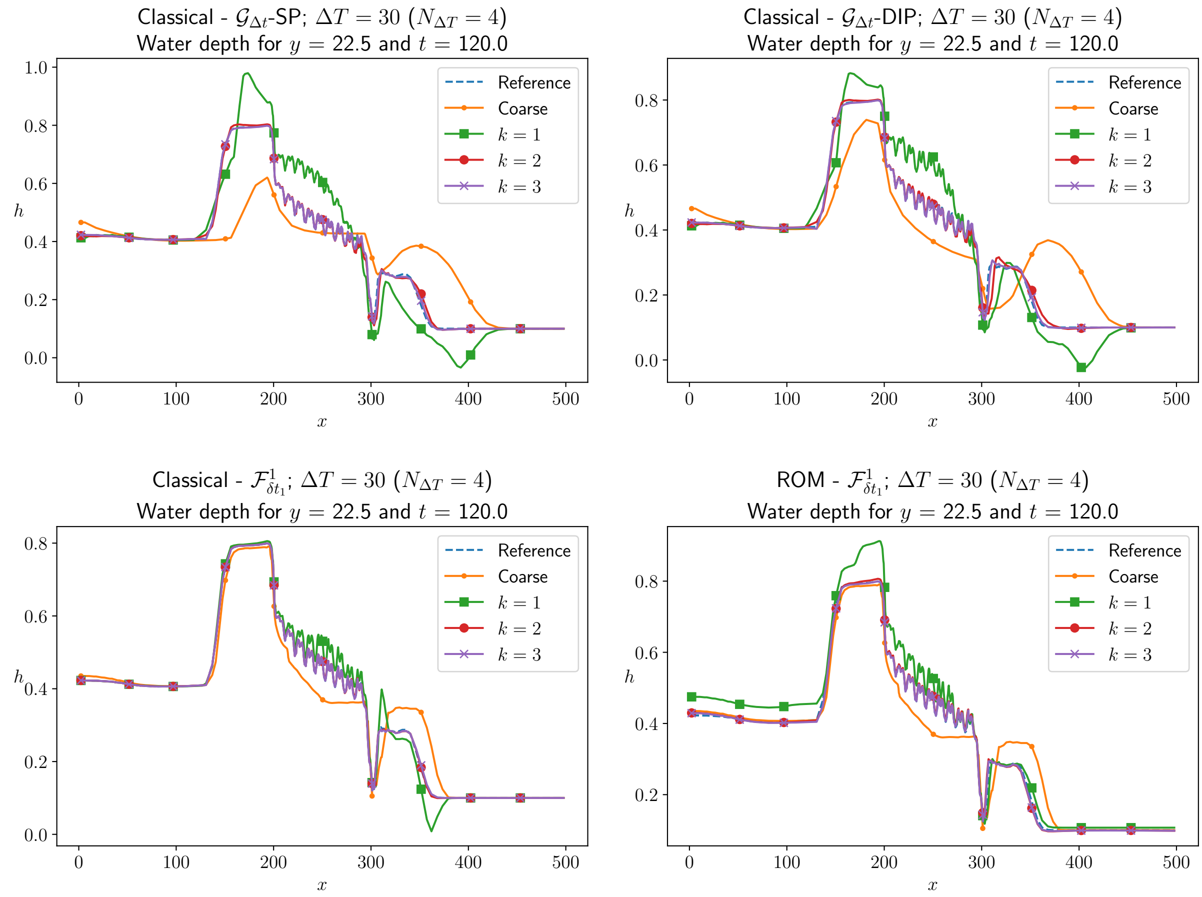


Figure 6.35: Test case 4 with $h_0 = 0.1$: water depth along $y = 46$, at $t = T$, for the reference and coarse solution (0-th parareal iteration) and first three parareal iterations, using $\mathcal{G}_{\Delta t}$-SP (top left), $\mathcal{G}_{\Delta t}$-DIP (top right) and $\mathcal{F}^1_{\delta t_1}$ (bottom) as coarse propagators. All simulations use the classical parareal method, except for the bottom right figure, in which the ROM-based method is used. In all cases, $N_{\Delta T} = 4$ time slices are considered.

6.7 Conclusion of the chapter

In this chapter, we tackled the main objective of this thesis: the coupling between the classical and porosity-based shallow water equations for the simulation of urban floods. We considered both the classical and ROM-based parareal methods, whose behaviour was studied in the previous chapters using simpler test cases, within relatively small spatial and temporal domains, simple solution profiles and using the classical shallow water equations for discretizing both the fine and coarse propagators. The nature of urban flood problems imposes several challenges for their simulation using parareal methods, mainly the ROM-based one, *e.g.* the highly discontinuous solutions induced by the presence of obstacles in the domain and by the use of unstructured computational meshes, the large temporal domains and the presence of zero or small water depths as initial solution.

In the first part of this study, we investigated the influence of the coarsening intensity between the fine and coarse propagators. It was illustrated that, in the classical parareal framework, a faster convergence is obtained when the propagators are close to each other. Therefore, if a porosity-based model is used as coarse propagator, the classical parareal method performs better when the reference model is a coarser discretization of the classical SWE. Analogously, if a very fine model is used as reference one, a faster convergence is obtained by using finer discretizations of the porosity-based SWE and, notably, it is more interesting to use a coarser discretization of the classical SWE as coarse propagator. However, even with important coarsening intensities between the fine and coarse propagators, it was possible to perform converging and stable simulations by choosing large enough parareal time slices, evidently with a negative impact on the possible speedup that can be achieved.

The execution of the same simulations, but using the ROM-based parareal method, showed to be much more challenging. Instead of the coarsening between the fine and coarse propagators, the performance of the method seems to be mostly influenced by the refinement of the reference model, which reflects the ability of the model reduction procedures to properly approximate it. Indeed, when a very fine model was defined as reference propagator, simulations presented instabilities even when using an also very fine coarse propagator. These instabilities arise mainly in regions of the temporal domain in which the solution profile is more complex and presents strong discontinuities, which are particularly challenging for the model reduction. For the same reason, the method behaves better and outperforms the classical parareal when the solution profile is simpler (*e.g.* before the inlet wave reaches the the urban zone). Therefore, stable and converging ROM-based simulations could only be obtained by using very restrictive parareal configurations, *e.g.* by dividing the temporal domain in windows separating different solution behaviours, combined with the use of large time slices. Also, the simulations performed here show that the behaviour of the ROM-based method in function of its parameters cannot be generalized both for simple and more complex problems. Namely, it was observed, in Chapter 4, that increasing the number of input snapshots for the model reduction (*i.e.* taking smaller α_s) allows to greatly improve the stability and convergence for approximating relatively simple profiles. In this chapter, however, we noticed an increasingly unstable behaviour by using more snapshots for approximating highly discontinuous solutions.

Interesting conclusions were also made by performing a more detailed discussion on the use of the adaptive parareal approach. As observed previously in this work in simpler tests cases, this approach also allowed, in the problems considered in this chapter, to improve the stability, convergence and speedup both of the classical and ROM-based parareal methods, due to the use of progressively refined fine propagators along iterations. In the examples considered here, coarser discretizations of the classical SWE act as intermediate fine propagator. It was shown, however, that more interesting results can be obtained by using non-adaptive approaches in which one of the intermediate models act as coarse propagator: even if the parareal iterations are more expensive (since a finer coarse model is used), less iterations are required for reaching given accuracies, thus resulting in larger speedups. Despite of these remarks, the observed results are meaningful since they indicate that challenging parareal simulations using very different fine and coarse propagators can be stabilized and converge faster by using intermediate models. Then, it could be explored, possibly in a more efficient way, in the framework of multilevel parallel-in-time methods, *e.g.* MGRIT (Friedhoff et al., 2013; Falgout et al., 2014), with parallelization occurring at each level.

We closed this chapter by considering a more challenging example, solved in a larger temporal domain. We studied the convergence of the parareal methods by considering either the porosity-based models or a coarser discretization of the classical SWE as coarse propagator. The challenge was considerably larger in the former case, requiring the use of few large time slices for obtaining stable and converging solutions, thus imposing severe restrictions to the expected speedup. For the latter case, smaller time slices could be used and the convergence was faster. In both cases, however, numerical solutions close to the reference one (even if still far from exact convergence) were obtained by performing few iterations (two or three) of the classical parareal method. Speedups up to 2 and 3, approximately, were observed by using a porosity-based model or and a coarser discretization of the classical SWE as coarse propagator, respectively. For the ROM-based method, convergence and stability were even more challenging, due to the large temporal domain and the use of unstructured meshes. It was necessary to use larger time slices than in the classical parareal, and to formulate very-low dimensional reduced-order models. Good approximations were also obtained after two or three iterations, but the computational time is compromised by these more restrictive configurations, with speedups no larger than 1.5.

Based on the study performed in this chapter, we list some main conclusions on the use of the parareal method for coupling the classical and porosity-based shallow water equations:

- Under adequately chosen configurations, the classical parareal method can be used for coupling the classical and porosity-based shallow water equations. Notably, large time slices are required for ensuring stable and converging solutions. The method is not able to provide a too fast convergence (*i.e.* exact convergence is only obtained at the maximum number of iterations, equal to the number of time slices), but relatively good approximations, largely outperforming the porosity-based model, can be obtained after few iterations;

- However, if available, it is more interesting to use a coarser discretization of the classical SWE as coarse propagator. In addition to spatial and temporal discretizations closer to the ones used by the reference model, a better approximation (compared to the one provided by the porosity-based model), mainly of the velocity fields, is obtained already at the 0-th parareal iterations, resulting in a faster convergence;

- The ROM-based parareal method is not suited to the problems considered here. The discontinuous nature of the solution in urban floods simulations, summed up to factors as large temporal domains, complex geometries and unstructured meshes, lead to frameworks in which good reduced-order models cannot be formulated on-the-fly, thus impacting the performance of the parareal method using them. Too severe restrictions must be imposed for ensuring the stability and convergence of the ROM-based method, thus increasing its computational cost and making it less interesting than the classical one, which has a simpler formulation and is able to provide larger speedups.
- However, even in the classical parareal framework and using a coarse discretization of the classical SWE as coarse propagator, one cannot expect to obtain large speedups, since large time slices are required in order to ensure stability and convergence, and the simulation can only be distributed to small numbers of processors. Therefore, for the problem considered here and in the current state of this work, the parareal method cannot be used as unique tool for providing important accelerations of the the simulation of urban floods, *e.g.* in massively parallel environments. It could be envisaged as a complement to spatial domain decomposition methods, thus providing a spatio-temporal parallelization.

Some other aspects were observed in this study and would require more detailed investigations to be clarified. Firstly, no remarkable differences were observed using the SP and DIP models as coarse propagator, their convergence behaviour being mostly determined by the spatial mesh and time step sizes. One could design test cases in which one of the models largely outperforms the other, in order to investigate the influence on the parareal performance. Also, other porosity-based models (*e.g.* the MP and IP ones) could be considered in this study. Secondly, the numerical simulations presented in this chapter illustrate that the parareal performance is influenced by the order of spatial interpolation, but this influence is not completely clear, mainly in the ROM-based parareal framework. In some cases, the linear interpolation provides better results, but in others the cubic one is preferable. Notably, complex urban geometries naturally impose challenges for the interpolation procedures, which should be carefully formulated for taking into account *e.g.* the presence of obstacles and the nature of the computational mesh (structured or unstructured). Finally, good parareal results could only be obtained by considering a non-zero water depth as initial solution, which is a situation not likely to be encountered in the simulation of urban floods. When a zero initial water depth is considered, parareal simulation rapidly leads to instabilities since the method does not ensure positivity of the solution. This situation could be handled *e.g.* by imposing zero water depth when it becomes negative and rescaling the solution for ensuring mass and momentum conservation.

As a last remark, we notice that only relatively small problems were considered here. The performance of the parareal method in larger problems (*e.g.* for simulations in a real urban zone) should also be investigated.

CHAPTER 7

CONCLUSION AND PERSPECTIVES

In this PhD thesis, we studied a numerical coupling between small and large scale shallow water models for the simulation of urban floods. The small scale model consists of fine spatio-temporal discretizations, with buildings and other obstacles of the urban geometry physically represented as holes in the mesh, leading to accurate numerical solutions but very large computational times, even prohibitive for some applications such as real-time forecasting. In the large scale models, an upscaled, porosity-based approach is considered, in which porosity parameters are defined for taking into account the presence or not of obstacles and the consequent availability of area and cross sections to the flow. This approach allows to use much larger computational meshes and time steps, resulting in smaller computational costs but less accurate solutions, mainly for representing small scale phenomena.

This coupling was performed using predictor-corrector iterative parallel-in-time methods. In this type of numerical scheme, predictions are provided by a coarse, cheap model computed sequentially along the entire temporal domain, and corrections are given by a fine, expensive model, solved in parallel across time steps. This coarse-fine formulation naturally fits to the purposes of this work. More precisely, we considered the parareal method, one the of the simplest and most popular parallel-in-time methods.

However, temporal parallelization is well known to be inefficient when solving hyperbolic problems, presenting slow convergence and unstable behaviour. We thus explored a variant of the parareal method with reduced-order models formulated on-the-fly along iterations, using the parareal solution as snapshots. The performance of the classical and ROM-based parareal methods for the problem proposed here was studied and compared.

We summarize in the following paragraphs the main conclusions of this work and list some guidelines for future works.

7.1 On the performance of the ROM-based parareal method

7.1.1 Synthesis and conclusions

The ROM-based parareal method proposed by Chen et al. (2014) was explored here as an alternative for overcoming stability and convergence issues of the classical method when applied for solving the two-dimensional nonlinear shallow water equations, an hyperbolic problem. The model reduction is performed using a combined POD-EIM technique, which is suitable for reducing the complexity of nonlinear problems and can be efficiently implemented by using singular value decomposition routines provided by well validated and optimized linear algebra packages. In this combined approach, reduced spaces are computed from snapshots from both the solution and nonlinear terms of the governing equations, and, in the latter case, a small set of spatial points (or interfaces, in a finite volumes discretization) is chosen for computing the nonlinear term.

Initial and promising results for small problems

In Chapter 3, it was illustrated, using a set of relatively small and simple test cases coupling the classical shallow water equations at different scales, that, when parameters for the model reduction are properly chosen, the ROM-based method provides faster convergence and more stability compared to the classical method using the same parareal configurations (coarse and fine propagators, length of time slices). Notably, the dimension of the computed reduced subspaces plays a major role, and it was illustrated that stability and faster convergence are more likely to be achieved when both subspaces formulated from the solution and nonlinear terms are enoughly high-dimensional.

Further improvements

It was illustrated, in Chapter 4, that the performance of the ROM-based parareal method can be improved by some simple modifications:

- Enrichment of the snapshots sets used for the model reduction procedures, by taking additional snapshots computed at intermediate fine time steps (instead of snapshots computed only on the extremities of the parareal time slices). In the small test cases considered here, important improvements of the quality of the ROMs formulated on-the-fly and consequently of the parareal method using them are obtained with few extra snapshots, thus providing a good compromise between convergence and computational time;

- Introduction of the adaptive approach proposed by Maday and Mula (2020), consisting in using progressively refined fine propagators along parareal iterations, instead of a fixed and expensive one. Originally formulated in the framework of the classical parareal method, this method also presents benefits in the ROM-based case, providing more stable solutions and reaching given accuracies within smaller computational times.

- Formulation of local-in-time ROMs, based on the principal interval decomposition (PID) method proposed by Ijzerman (2000), with the objective of better representing local phenomena. Less remarkably than the two previous modifications, this approach provide some improvements on the stability of the parareal solution, but may degrade its quality when ROMs formulated in the entire temporal domain already provides good approximation to the reference model.

Important challenges in more complex problems

However, although these good results observed in small problems, convergence and stability of the ROM-based parareal method become more challenging for solving larger problems. In Chapter 5, the same simple test cases considered before were solved in larger spatial and temporal domains and, in order to obtain stable and converging solutions, the simulation needed to be divided in smaller time windows, with parareal simulations performed in each of them, at the cost of smaller expected numerical speedups. It indicates that the formulation of reduced-order models approximating solutions defined in large domains is particularly challenging, thus the ROM-based parareal method is more suitable when applied to smaller domains. Another effective approach explored in Chapter 5 for overcoming these issues consists in using larger parareal time slices, which also limits the possible expected speedup. This approach is more effective for improving the classical parareal method, whose utilisation becomes more attractive due to its reduced computational cost compared to the ROM-based approach.

Additional and more important challenges for the ROM-based method arise when the problem's solution is characterized by more complex profiles, *e.g.* with discontinuities and strong variations, which is usual in the simulation of urban floods due to the presence of obstacles in the domain. As discussed in this work, it is known that POD-based model reduction is less effective for approximating discontinuous solutions. It has a strong impact on the parareal method using reduced-order models, whose stability for more challenging test cases presented in Chapter 6 was ensured by choosing very restrictive configurations and limiting the expected speedup.

Complex parametrizations and less predictable behaviour

The comparisons made in this work, mainly in Chapter 6, between the classical and the ROM-based parareal methods also allowed to assess, for the latter, additional challenges concerning more complex parametrizations and less predictable behaviour and performance in function of the chosen parameters.

Indeed, the classical parareal method already depends on several parameters (choice of coarse and fine propagators, length of time slices), whose influence may depend on the considered problem, numerical schemes and length of the temporal domain. In the ROM-based approach, other parameters, such as thresholds for the model reduction procedures and number of input snapshots, increase the difficulty in establishing and choosing optimal configurations. Moreover, by performing isolated studies of the sensitivities of the methods w.r.t. the chosen parameters, *e.g.* the length of time slices and the coarsening between the fine and coarse propagators, clear influences on the behaviour of the classical parareal method were observed. These influences are much less clear in the case of the ROM-based method. Also, the enrichment of the input snapshots sets for the model reduction, which showed itself to be effective for improving the method applied to relatively simple problems, presented contrary effects when applied to more complex ones.

Assessment of computational costs

We highlight that we provided, in this work, a first practical evaluation of the ROM-based parareal method in terms of computational cost, even if it was performed in small parallel environments. The method makes an on-the-fly use of model reduction techniques that can be excessively expensive and impede any interest in using the parareal method. We then studied its cost in function of the several parameters it depends on and analyzed the fraction of the computational time spent in each step of the algorithm. The initial study coupling the classical SWE at different scales (Test cases 1, 1c and 2) did not provide realistic computational costs and speedups (since their fine modes are largely over-resolved), but were meaningful for comparing the costs of the classical and ROM-based methods. A more adequate framework, in which the fine and coarse propagators were defined with approximately the same CFL number (as usual in explicit discretizations), was performed for coupling the classical and porosity-based SWE, and, even if speedups larger than the unity could be obtained in some simulations of the ROM-based parareal method, they were considerably smaller than in the classical one.

7.1.2 Perspectives

Application to other problems and contexts

Even if the ROM-based parareal method showed not to be suitable for the more challenging problems considered here, its use in other contexts cannot be immediately discarded. It was illustrated that the method can largely outperform the classical one when the model reduction can properly represent the dynamics of the fine model. Therefore, it can be useful *e.g.* in problems characterized by smooth solutions.

A more efficient implementation

The numerical examples presented in this work show that the ROM-based parareal method provides only a limited speedup, due to important costs introduced by the model reduction performed at each iteration. However, a more efficient implementation could be considered for minimizing these additional costs. For example, it was shown that, in the case of the 2D SWE, eight reduced subspaces need to be formulated at each iteration (three using POD and accounting for the conserved variables; and five using POD-EIM and accounting for flux and source terms). These subspaces are independent from each other, and can be computed simultaneously (which was not considered here), possibly providing further improvements of the speedup, mainly in larger problems (evidently, in situations where the ROM-based presents good stability and convergence). Similarly, the nine matrices defining the reduced 2D SWE are independent and can be simultaneously. This “parallel” model reduction implementation is discussed in Appendix B.2.

Use of other model reduction techniques

In this work, only the combined POD-EIM model reduction technique was considered for implementing the ROM-based parareal method, but other approaches could be considered. For example, a full EIM (which was also proposed by Chen et al. (2014)), or the Dynamic Mode Decomoisition (DMD) (Schmid, 2010), which has been explored in some works (Bistrian and Navon, 2015, 2017; Ahmed et al., 2020) for reducing the complexity of the SWE . Evidently, their applicability as an on-the-fly model reduction (*i.e.* their computational cost and their quality when formulated using snapshots provided by a coarse model) should be investigated in detail, as well as their performance in function of the parameters they depend on.

7.2 On the performance of adaptive parareal methods

7.2.1 Synthesis and conclusions

One of the approaches considered in this work, based on the adaptive parareal method proposed by Maday and Mula (2020), consists in using increasingly refined fine propagators along parareal iterations. This idea was extended to the ROM-based parareal method. We proposed here to define beforehand a given number of fine models, each one with a temporal and a spatial discretizations, and to decide on the transition between consecutive propagators based either on parareal residuals or fixed number of iterations. With this adaptive approach, improvements of the stability and convergence were observed both for the classical and ROM-based methods, which can possibly be attributed to the less important difference, thus less important mismatch of phase speeds, between the coarse and fine propagators along iterations. In the case of the ROM-based method, benefits could also come from the fact that relatively inaccurate snapshots are used for formulating ROMs that approximate also less accurate models, instead of a very fine one. Moreover, it was observed that this adaptive approach allows to reach given accuracies within smaller computational times than in the non-adaptive parareal method.

In practice, however, it was noticed that, if intermediate fine propagators are available, it is more interesting to use them directly as coarse propagator for the parareal method. In this case, less iterations are required for reaching given accuracies and larger speedups are obtained, despite of a more expensive sequential prediction step in the parareal algorithm.

7.2.2 Perspectives

Better oriented formulations of the adaptive approach

In the adaptive approach implemented in this work, intermediate fine propagators were arbitrarily chosen, as well as the criteria for deciding on the transition between them. Better oriented formulations, based on estimations of the parareal error, could possibly lead to a more efficient convergence and make the adaptive approach more interesting than using an intermediate fine model as coarse propagator.

Indications for a multilevel approach

Although this contestable utility of the adaptive approach as implemented here, the observed improvements on the stability and convergence are meaningful by indicating that the parareal performance in the problems considered in this work can be improved by gradually approximating the reference solution, instead of using only a coarse and fine propagators, which may be very different and lead to poor parareal behaviours. Therefore, it suggests that multilevel predictor-corrector PinT methods, *e.g.* MGRIT (Friedhoff et al., 2013; Falgout et al., 2014), could be envisaged.

7.3 On the temporal parallelization of urban floods simulations

7.3.1 Summary and conclusions

Parallelization-in-time is envisageable, but with restrictive configurations

The application of the parareal methods for the simulation of urban floods showed to be particularly challenging. Relatively good results were obtained in the numerical examples presented here, with the parareal solution well approximating, to a greater or lesser extent, the profile of the reference one, but at the cost of a limited speedup. Indeed, stability and relatively fast convergence were ensured by considering very large parareal time slices, which means that only few processors could be used for parallelizing the simulation. In practice, in the best results showed here, speedups no larger than 3, approximately, were obtained. It means, at least in the current state of this work, that parareal alone cannot be envisaged for large scale parallelism of urban flood simulations using shallow water models.

Performances of the classical and ROM-based parareal methods

Concerning the use of the classical and ROM-based parareal methods for the problem proposed here, the former presented a better performance in general, providing a good convergence behaviour with less restrictive configurations. Relatively good results were also obtained by the latter, but in very limited frameworks necessary for dealing with solutions presenting discontinuities and defined in large domains (*e.g.* by considering very large parareal time slices, defining local-in-time parareal simulations, and formulating very low-dimensional ROMs) such that the method behaves similarly or worse than the classical one, and with smaller speedups. This challenge is due to the difficulty in formulating high-quality ROMs on-the-fly along parareal iterations, using relatively inaccurate snapshots, for approximating a complex and highly discontinuous reference model. It was observed that the ROM-based parareal method is able to outperform the classical one only in very specific situations (with relatively smooth solutions, using computational meshes favoring the model reduction) that are not likely to be encountered in realistic contexts of urban flood simulations.

It is important to notice that the results presented in this work do not allow to conclude on the suitability or not of model reduction itself for the simulation of urban floods using the shallow water equations. As said above, the particularity of the ROMs considered in this work is that they are formulated on-the-fly and in a relatively cheap manner, as required by the parareal method. In a more traditional model reduction framework, a more expensive and careful offline stage would be performed, using snapshots provided by accurate simulations and possibly other model reduction methods.

Choice of coarse propagator

It was also illustrated in this work that a better performance of the parareal method (both the classical and ROM-based ones) applied to urban flood simulations can be obtained by using a coarse discretization of the classical SWE as coarse propagator, instead of a porosity-based model. It is quite natural since, in this framework, the fine and coarse models are closer to each other (they are discretizations of the same model, with smaller differences of time steps and mesh sizes), and better approximations are already obtained from the 0-th parareal iteration. As a consequence, fewer iterations are required for obtaining relatively good approximations, resulting in larger speedups (even if the iterations themselves are more expensive due to the use of a finer coarse propagator).

We also notice that no remarkable differences on the parareal performance were observed by using different porosity-based model as coarse propagator (we considered here the SP and DIP models), with the convergence behaviour depending mainly on the size of the spatial and temporal discretizations. Possibly, numerical tests specially designed for highlighting different performances of the porosity models w.r.t. the classical SWE could provide clearer differences in the parareal framework. Also, other porosity-based models could be considered in this study.

7.3.2 Perspectives

Towards more realistic problems

Even if challenging, all problems considered in Chapter 6 for coupling the classical and porosity-based SWE were relatively simple, being solved in small spatial and temporal domains and considering idealized urban zones composed by Cartesian grids of buildings blocks. An investigation of the performance of the parareal methods for solving more realistic problems, with larger domains and more complex urban geometries, needs to be investigated. Based on the conclusions of this work, we can suppose that restrictive parareal configurations would be necessary, by using larger time slices and even multiple local-in-time parareal simulations, and that the ROM-based method would be less suitable than the classical one.

Coupling with spatial parallelization

As said above, at the current state of this work, large scale parallelization and speedups cannot be obtained for the problems considered here by only using temporal paralellization with the parareal method. However, other strategies could be considered, *e.g.* a combined spatio-temporal parallelization, in which parareal is used for obtaining further speedups in spatial domain decomposition methods. A possibility would be to consider a parareal waveform relaxation method (Gander et al., 2013).

Improving interpolation

The results presented in this work show that the order of spatial interpolation has an influence on the performance of the parareal method, but more detailed investigations need to be conducted for investigating how this influence manifests itself. Indeed, improvements by using cubic rather than linear interpolation were observed in some cases, but in others the parareal performance gets worse. Notably, important degradations of the ROM-based parareal method were observed with cubic interpolation, and the causes are not clear. Moreover, the interpolation procedures used in this work, based on Delaunay triangulation of cells' barycenters, may not be the best choice for the simulation of urban zones, and better designed interpolation procedures should be developed, taking into account *e.g.* the position of the cells w.r.t. elements of the urban geometry (obstacles, transverse and longitudinal streets).

Dealing with small water depths

An important aspect of urban flood simulations, concerning the computation of very small or even zero water depths, has not been tackled in this work. It is a challenge when using parareal since the method does not ensure the positivity of the solution. Therefore, small instabilities produced along parareal iterations can lead to negative water depths. In the numerical solutions presented here, the use of large enough initial water depths showed to be necessary for ensuring stable solutions. Possibilities for dealing with this issue were cited, *e.g.* setting negative water depths to zero, and it should be studied how to guarantee mass and momentum conservation.

Flexibility on the choice of time steps

In all simulations presented in this work, the fine and coarse propagators (as well as the intermediate fine propagators in the adaptive approach) were designed such as to use homogeneous time steps, integers multiple of each other and dividing the parareal time slice length ΔT in a integer number of steps. This choice was made for simplifying the computational implementation. However, in explicit-in-time schemes, one may usually want to use the largest possible time step allowed by stability conditions, thus being computed adaptively along the simulation in function of the solution and the spatial mesh sizes. It is done *e.g.* in the SW2D-LEMON software. In can also be implemented in a parareal framework, but requiring interpolation in time or by choosing the fine and coarse time steps at the end of each parareal time slice such as to fall exactly on each parareal time instant. In larger parallel applications, other possibly issues such as a good balance of parallel tasks would also need to be dealt with.

Use of other discretization schemes

In this thesis, we restricted ourselves to the spatial and temporal discretizations adopted in SW2D-LEMON software. Namely, both the classical and porosity-based shallow water models were discretized using finite volumes and an explicit Euler time-stepping. Other numerical schemes can be considered and possibly improve the stability and convergence of the parareal method, *e.g.* if they are able to provide better representations of phase speeds. A possibility would to consider a semi-Lagrangian temporal discretization (also known as method of characteristics). Schmitt et al. (2018) applied it for discretizating the coarse propagator in the simulation of the viscous Burgers equations using the parareal method and obtained more stable and faster converging solutions than with other temporal discretizations, even with zero viscosity (*i.e.* in a purely hyperbolic problem). De Sterck et al. (2021) presented similar conclusions in the framework of parareal and MGRIT for solving the linear advection equation, stating that the coarse solver should track the solution along characteristic curves (which is the principle of semi-Lagrangian methods) for ensuring a fast convergence. These results can possibly be attributed by the fact that semi-Lagrangian temporal discretizations provide good phase speed representations (Wedi et al., 2019). Therefore, this approach could be combined with a finite volume spatial discretization and envisaged for the problem considered in this work.

Implementation in SW2D-LEMON

As discussed in Section 1.4, the work presented here was developed relying on a standalone software into which the main features of the classical and porosity-based models of SW2D-LEMON were incorporated. The following step is to move in the opposite direction, *i.e.* incorporate the parareal methods into SW2D-LEMON, allowing further studies and developments (for example, studying the influence of uneven bathymetry or other source terms on the parareal performance, or using other porosity-based models as coarse propagator).

In the case of the classical parareal method, this introduction in SW2D-LEMON could be performed in two ways. The first one is to exploit the non-intrusive character of the parareal method, with each simulation of the coarse and fine propagators corresponding to full executions of the software, and by using an external script for managing the executions and collecting and distributing input and output data. It would require minimal or any modification of the SW2D code, but possibly important costs linked to the initialization of each execution and data communication would take place. In the second approach, parallelism would be implemented directly in the SW2D code. It would be more efficient in terms of computational cost but would require important restructuring of the software, in order to allow *e.g.* the coexistence of two models in the same execution and the storage of more data. A detailed evaluation of these aspects should be considered for choosing a paradigm. Moreover, in the case of the ROM-based parareal method, an important intrusive character exists, due to the formulation and simulation of reduced-order models, and modifications of the code would be necessary.

APPENDIX A

SOME DETAILS ON THE PARALLEL IMPLEMENTATION AND EXECUTION

The numerical results presented in this thesis were obtained using a C++ program developed during the PhD work. This program implements an explicit finite volume solver for the classical and porosity-based shallow water models (reproducing the implementation in the SW2D-LEMON software) and routines for solving them using the classical and ROM-based parareal methods. We provide in this appendix some details on the parallel implementation and execution of the software.

A.1 Parallelization using OpenMP

The parallelization of the parareal methods is performed in this work using the OpenMP API (Open Multi-Processing Application Program Interface) (OpenMP Architecture Review Board, 2011). OpenMP is a shared memory parallel programming paradigm, meaning that threads (the working unit in a processor core) access the same memory. As a consequence, communication between parallel working unit is not necessary, contrary to the distributed memory paradigm MPI (Message-Passing Interface) (Message Passing Interface Forum, 2009). Indeed, parallelization of parts of a program is relatively simple with OpenMP, requiring only minor modifications of the code. As a disadvantage, the number of threads that can work simultaneously is limited by the number of processor cores available in a single computing node (since each node has its own memory), ranging in the order of tenths in computing cluster used in this work, as described below. This configuration is enough for the small-scale problems considered here, but, for larger problems, a MPI or an hybrid OpenMP-MPI implementation should be considered for executing the simulation in several nodes.

A.2 Software execution

Simulations are executed in the NEF computing platform from Inria Sophia Antipolis-Méditerranée center composed by various types of CPU and GPU clusters. In order to perform proper comparisons of computational times, all simulations, both the reference (simulations of the fine model using a single processor core) and parareal ones (simulations using several cores), are executed in the same cluster, namely in a single node dual-Xeon E5-2680 v2 with a frequency of 2.80GHz and 256GB of RAM capacity, containing 20 cores[1]. It means that at most $N_p = 20$ parallel processors can be used in parareal simulations.

[1] https://wiki.inria.fr/ClustersSophia/Hardware

A.3 Measurement of computational times

Computational times of all simulations, both the reference and parareal ones, are measured using the OpenMP function *omp_get_wtime*. Input and output operations (*e.g.* lecture of meshes and configuration files, storage of results) are not considered. All computational times and speedups reported in this work correspond to an average of five executions.

APPENDIX B

IMPLEMENTATION OF THE POD-EIM MODEL REDUCTION

B.1 Implementation using MKL linear algebra functions

As described in Sections 3.4 and 3.5, in each iteration of the ROM-based parareal method, a reduced-order model $\mathcal{F}^k_{r,\delta t}$ is constructed. This formulation consists in two steps:

1. Formulation of the reduced subspaces (*i.e.* the computation of matrices containing their basis vectors) using a chosen model reduction technique. In this work, we considered a combined application of POD (for obtaining the basis matrix V^k of the subspace $\mathcal{S}^k$) and POD-EIM (for obtaining the basis matrix $\widehat{V}^k$ of the subspace $\widehat{\mathcal{S}}^k$ and a matrix $\widehat{P}^k$ containing chosen spatial indices for computing the nonlinear term of the governing equations).

2. Computation of the N_{matrices} matrices defining the ROM $\mathcal{F}^k_{r,\delta t}$, based on V^k, $\widehat{V}^k$ and $\widehat{P}^k$.

In the case of the two-dimensional nonlinear shallow water equations, eight subspaces are computed at each iteration, the first three using POD (thus producing matrices $V^{(i)}, i = 1, 2, 3$, where the index accounting for the iteration number is omitted for simplification) and the remaining five using POD-EIM (thus producing matrices $\widehat{V}^{(i)}, \widehat{P}^{(i)}, i = 1, .., 5$). In the case where source terms are neglected (as done in this work), this computation can be simplified by directly setting $\widehat{\mathcal{S}}^{(4)}$ and $\widehat{\mathcal{S}}^{(5)}$ as null spaces. Concerning the second step of the model reduction, nine matrices need to be computed ($\widehat{B}^{(1)}, \dots, \widehat{B}^{(3,4)}$, see eq. (3.49)), with four of them not needing to be computed if source terms are neglected.

In the implementation proposed in this work, both model reduction steps rely on linear algebra functions of the *MKL-LAPACK* and *MKL-CBLAS* suites, namely:

- *LAPACKE_dgesvd*, for performing singular value decompositions required by the POD;
- *LAPACKE_dgesv*, for solving linear systems required by the DEIM step of the POD-EIM;
- *cblas_dgemm*, for performing matrix-matrix multiplications, necessary for computing the matrices of the reduced-order model;
- *LAPACKE_dgetrf* and *LAPACKE_dgetri* for inverting matrices, also necessary for computing the matrices of the reduced-order model.

By using functions provided by well validated and optimized linear algebra packages, a relatively efficient model reduction can be performed, thus reducing its impact on the computational cost of the parareal method. This efficiency is improved by the fact that *MKL* functions are parallelized using OpenMP.

B.2 Model reduction workflow

The workflow implemented in this work for the POD-EIM model reduction is schematized in Figure B.1. In the "Compute ROM subspaces" step, the subspaces are computed sequentially w.r.t. each other, and each computation is performed with inner OpenMP parallelization of the *MKL* functions using all available computing cores. The same is valid for the "Compute ROM matrices" step, with the matrices computed with inner parallelization and sequentially w.r.t. each other.

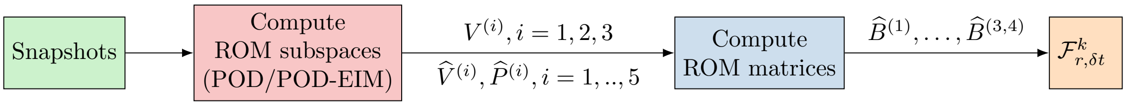

Snapshots
Compute ROM subspaces (POD/POD-EIM)
$V^{(i)}, i = 1, 2, 3$
$\widehat{V}^{(i)}, \widehat{P}^{(i)}, i = 1, .., 5$
Compute ROM matrices
$\widehat{B}^{(1)}, \ldots, \widehat{B}^{(3,4)}$
$\mathcal{F}^k_{r,\delta t}$

Figure B.1: Implemented "sequential" workflow for the POD-EIM model reduction. In each "compute" block, the computed elements (subspaces or ROM-matrices) are computed using all available computing cores and sequentially w.r.t. each other.

An alternative workflow that has not been explored in details in this work, but could be interesting in large problems, is illustrated in Figure B.2. It uses the fact that all ROM subspaces are independent, so their computation can be parallelized, and the same for all matrices defining the reduced models. Note however that these matrices depend on the formulated subspaces, and a gathering point is required for collecting the computed subspaces (*i.e.* their basis matrices) and distributing those required for computing each ROM matrix. Therefore, a possibly more efficient model reduction would consist in the following steps:

1. Parallel formulation of the subspaces (each parallel unit is responsible for computing a matrix $V^{(i)}$, or matrices $\widehat{V}^{(i)}, \widehat{P}^{(i)}$);
2. Gather all basis matrices $V^{(i)}, i = 1, 2, 3$, and $\widehat{V}^{(i)}, \widehat{P}^{(i)}, i = 1, .., 5$;
3. Distribute to each processor the required basis matrices for computing a given ROM matrix;
4. Parallel computation of the N_{matrices} ROM matrices (each parallel unit is responsible for computing a single matrix).
5. Gather the computed ROM matrices.

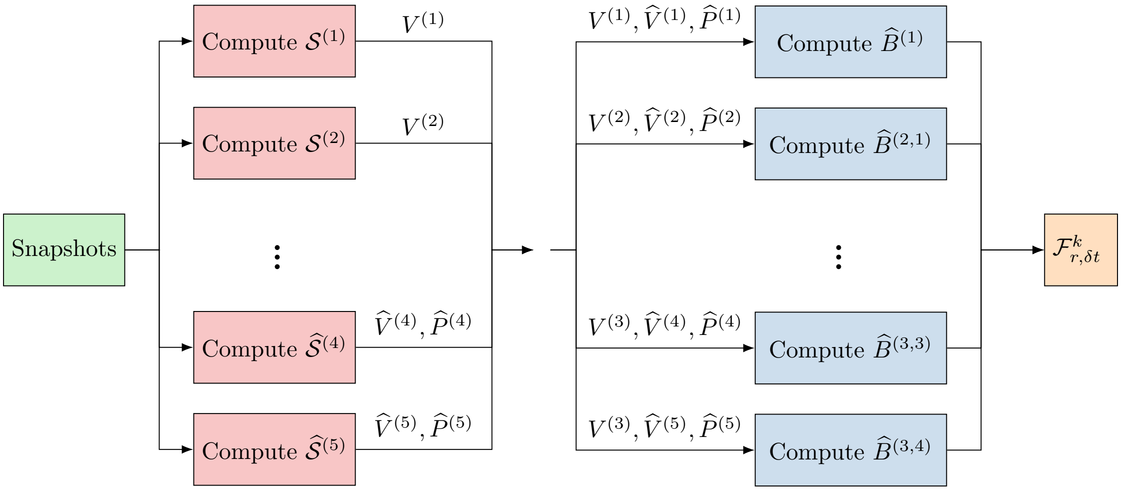

Snapshots
Compute $\mathcal{S}^{(1)}$
Compute $\mathcal{S}^{(2)}$
Compute $\widehat{\mathcal{S}}^{(4)}$
Compute $\widehat{\mathcal{S}}^{(5)}$
$V^{(1)}$
$V^{(2)}$
$\widehat{V}^{(4)}, \widehat{P}^{(4)}$
$\widehat{V}^{(5)}, \widehat{P}^{(5)}$
$V^{(1)}, \widehat{V}^{(1)}, \widehat{P}^{(1)}$
$V^{(2)}, \widehat{V}^{(2)}, \widehat{P}^{(2)}$
$V^{(3)}, \widehat{V}^{(4)}, \widehat{P}^{(4)}$
$V^{(3)}, \widehat{V}^{(5)}, \widehat{P}^{(5)}$
Compute $\widehat{B}^{(1)}$
Compute $\widehat{B}^{(2,1)}$
Compute $\widehat{B}^{(3,3)}$
Compute $\widehat{B}^{(3,4)}$
$\mathcal{F}^k_{r,\delta t}$

Figure B.2: Proposed alternative "parallel" workflow for the POD-EIM model reduction. In each "compute" block, the computed elements (subspaces or ROM-matrices) are computed in parallel w.r.t. to each other. A gathering point is implemented between the two blocks for collecting and redistributing elements necessary for computing the ROM matrices.

In this work, all simulations are performed in a single cluster node. In this case, in order to implement the "parallel" workflow described above, each parallel unit would be a single core (or few cores of the node), implying that the inner OpenMP parallelization of the *MKL* functions would be performed using less cores. Quick investigations indicated that it is less efficient than performing the "sequential" model reduction illustrated in Figure B.1, in which each *MKL* function is parallelized among all cores available in the node. Evidently, this efficiency can possibly depend on the size of the problem. Moreover, the proposed "parallel" model reduction could be more efficient in an hybrid OpenMP-MPI implementation, with each parallel unit corresponding to a different cluster node (thus allowing inner parallelization using several cores).

B.3 Complexity of the POD-EIM model reduction procedures

A detailed description of the complexity of the model reduction procedure, used for estimating the computational times $\tau_{\mathcal{S}}$ and $\tau_{\hat{\mathcal{B}}}$ in Section 3.4.4, is presented in the following paragraphs. We divide this description in two parts, accounting for the two steps for the model reduction (the POD-EIM execution and the computation of the ROM matrices).

We briefly recall some notations considered in Section 3.4.4. In parareal iteration k, each model reduction is performed on a snapshots matrix (Y^k or $\widehat{Y}^k$) of size $M_f \times k(N_{\Delta T}+1)$, since $N_{\Delta T}+1$ snapshots are collected per iteration. M_f is the spatial dimension (number of spatial cells in a FV discretization) of the fine propagator $\mathcal{F}_{\delta t}$. The number of interfaces in the fine mesh is denoted by I_f. N_{spaces} subspaces and N_{matrices} are computed per iteration (in the case of the 2D nonlinear SWE, $N_{\text{spaces}} = 8$ and $N_{\text{matrices}} = 9$). The largest dimension among all subspaces computed along the parareal simulation is denoted by $\hat{m}$.

B.3.1 Complexity of the POD-EIM procedure

We develop an estimate for the computational time $\tau_{\mathcal{S}}(k)$ necessary for computing a ROM subspace, formulated using POD or POD-EIM at parareal iteration k. We majorate $\tau_{\mathcal{S}}(k)$ by considering a subspace formulated using the POD-EIM procedure (which is obviously more expensive than the POD alone). The POD-EIM for computing a subspace in iteration k consists in two steps:

1. A POD, via a SVD (eq. (3.27)) using as input a matrix of size $M_f \times k(N_{\Delta T}+1)$. The obtained subspace has dimension $m \ll M_f$. The SVD is performed using the *dgesvd* function of the *MKL-LAPACK* suite;

2. A DEIM (Algorithm 5), consisting in the sequential solution of $m-1$ linear systems (using the *dgesv* function of the *MKL-LAPACK* suite) and m lookups for the maximum element in vectors of size M_f (using the *max_element* function of the *C++ Standard Library*). The i-th linear system has a size $i \times i$;

The *MKL-LAPACK* function *dgesvd* applied to a matrix in $\mathbb{R}^{q\times n}$ has a complexity $\mathcal{O}(qn^2+n^3)$ (Dongarra et al., 2018). Concerning *dgesv*, the complexity for a matrix in $\mathbb{R}^{n\times n}$ is $\mathcal{O}(n^3)$[1]. For the function *max_element*, the complexity for looking up a vector in $\mathbb{R}^n$ is $\mathcal{O}(n)$[2].

Thus, by majorating m by $\hat{m}$, we estimate, for the last parareal iteration $\hat{k}$,

$$\begin{aligned}
\tau_{\mathcal{S}}(\hat{k}) &= \underbrace{\mathcal{O}\left(M_f\hat{k}^2(N_{\Delta T}+1)^2 + \hat{k}^3(N_{\Delta T}+1)^3\right)}_{\text{POD}} + \underbrace{\sum_{i=2}^{\hat{m}} \mathcal{O}\left(i^3\right) + \mathcal{O}\left(\hat{m}M_f\right)}_{\text{DEIM}} \\
&= \mathcal{O}\left(\hat{k}^2 N_{\Delta T}^2(M_f + \hat{k}N_{\Delta T}) + \left(\frac{\hat{m}^2(\hat{m}+1)^2}{4} - 1\right) + \hat{m}M_f\right) \\
&= \mathcal{O}\left(M_f\hat{k}^2N_{\Delta T}^2 + \frac{\hat{m}^4}{4} + \hat{m}M_f\right)
\end{aligned}$$

where we assume $M_f \gg \hat{k}N_{\Delta T}$.

[1] http://www.netlib.org/lapack/lug/node71.html

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

B.3.2 Complexity of the computation of ROM matrices

In each iteration of the ROM-based parareal method, we compute N_{matrices} of the form

$$V^T B \widehat{V} \left(\widehat{P}^T \widehat{V} \right)^{-1} \in \mathbb{R}^{q \times m} \tag{B.1}$$

where $V \in \mathbb{R}^{M_f \times q}$, $\widehat{V}, \widehat{P} \in \mathbb{R}^{I_f \times m}$ and $B \in \mathbb{R}^{M_f \times I_f}$ is a sparse matrix. $\widehat{P}$ is also a sparse matrix, but for the practical implementation of its multiplication by $\widehat{V}$, it is considered here as a dense one.

For computing (B.1), four matrix multiplications are required. All of them are performed using the *dgemm* function of the *MKL-CBLAS* suite, whose complexity for $O(n^2)$ data is $\mathcal{O}(n^3)$, such that its complexity for multiplying two matrices in $\mathbb{R}^{q \times n}$ and $\mathbb{R}^{n \times m}$ is $\mathcal{O}(qnm)$ (Blackford and Dongarra, 1999). The only exception is the dense-sparse matrix multiplication $V^T B$, which is “handmade” implemented by looping on the few non-zero elements of B (*e.g.* in a triangular mesh, each line of B contains three non-zero elements). Therefore, the complexities for the matrix multiplications can be estimated by

- $\left[V^T\right] [B]$: $\mathcal{O}(qI_f) = \mathcal{O}(qM_f)$;
- $\left[V^T B\right] \left[\widehat{V}\right]$: $\mathcal{O}(qI_f m) = \mathcal{O}(qmM_f)$;
- $\left[\widehat{P}^T\right] \left[\widehat{V}\right]$: $\mathcal{O}(mI_f m) = \mathcal{O}(m^2 M_f)$;
- $\left[V^T B \widehat{V}\right] \left[\left(\widehat{P}^T \widehat{V}\right)^{-1}\right]$: $\mathcal{O}(qmm) = \mathcal{O}(qm^2)$

where we used $I_f = \mathcal{O}(M_f)$.

Moreover, it is necessary to invert $C := \widehat{P}^T \widehat{V} \in \mathbb{R}^{m \times m}$. In the *MKL-LAPACK* suite, it is performed in two steps: firstly, the LU decomposition $C = LU$ is computed using the *dgetrf* function, where $L, U \in \mathbb{R}^{m \times m}$ are respectively lower and upper triangular matrices. Then, C^{-1} is computed by the *dgetri* function by solving the system $C^{-1} L = U^{-1}$. Both the mentioned routines have a complexity $\mathcal{O}(m^3)$ (Blackford and Dongarra, 1999).

Therefore, by majorating both q and m by $\hat{m}$, the computational time $\tau_{\widehat{B}}(\hat{k})$ for computing (B.1) can be estimated by

$$\begin{aligned}
\tau_{\widehat{B}}(\hat{k}) &= \underbrace{\mathcal{O}\left(\hat{m} M_f + \hat{m}^2 M_f + \hat{m}^2 M_f + \hat{m}^3\right)}_{\text{multiplication}} + \underbrace{\mathcal{O}\left(\hat{m}^3 + \hat{m}^3\right)}_{\text{inversion}} \\
&= \mathcal{O}\left(\hat{m}^2 (3\hat{m} + 2M_f) + \hat{m} M_f\right) \\
&= \mathcal{O}\left(\hat{m} M_f (2\hat{m} + 1)\right) \\
&= \mathcal{O}\left(2 M_f \hat{m}^2\right)
\end{aligned}$$

by assuming $M_f \gg \hat{m}$.

APPENDIX C

SPATIAL INTERPOLATION PROCEDURES

In parareal simulations involving spatial coarsening between the fine and coarse propagators, spatial interpolation needs to be performed between their respective meshes. Depending on the dimensions of the problem, interpolation can represent an important fraction of the total computational time, mainly in the classical parareal method, in which the solution must be interpolated between the fine and coarse meshes, in both senses and in all iterations. In the ROM-based method, interpolation is performed only in the 0-th iteration, from the coarse to the fine mesh. In the following iterations, the coarse propagator is replaced by a a reduced-order model, which provides solutions computed in the fine mesh.

Let $\mathcal{T}_f$ and $\mathcal{T}_c$ be two computational meshes (associated *e.g.* to the fine and coarse propagators in the parareal framework), containing respectively M_f and M_c cells. In a finite volume scheme, the approximation of a given function ψ is considered constant at each computational cell $\Omega_{f,i}$ (of $\mathcal{T}_f$) or $\Omega_{c,i}$ (of $\mathcal{T}_c$). Let $\boldsymbol{\psi}_f \in \mathbb{R}^{M_f}$ and $\boldsymbol{\psi}_c \in \mathbb{R}^{M_c}$ be the approximations of ψ in all cells of $\mathcal{T}_f$ and $\mathcal{T}_c$, respectively. Then, the interpolation of $\boldsymbol{\psi}_f$ from $\mathcal{T}_f$ to $\mathcal{T}_c$ and the interpolation of $\boldsymbol{\psi}_c$ from $\mathcal{T}_c$ to $\mathcal{T}_f$ read respectively

$$D_{f\to c}\boldsymbol{\psi}_f \in \mathbb{R}^{M_c}, \qquad D_{c\to f}\boldsymbol{\psi}_c \in \mathbb{R}^{M_f}$$

where $D_{f\to c} \in \mathbb{R}^{M_c \times M_f}$ and $D_{c\to f} \in \mathbb{R}^{M_f \times M_c}$ are interpolation matrices. The element $[D_{f\to c}]_{i,j}$ in the i-th line and j-th column of $D_{f\to c}$ is the weight associated to cell $\Omega_{f,j} \subset \mathcal{T}_f$ for computing the interpolated solution in $\Omega_{c,i} \subset \mathcal{T}_c$, and analogously to $D_{c\to f}$. These weights satisfy

$$\sum_{j=0}^{M_f} [D_{f\to c}]_{i,j} = 1, \qquad i = 1, \dots, M_c$$

$$\sum_{j=0}^{M_c} [D_{c\to f}]_{i,j} = 1, \qquad i = 1, \dots, M_f$$

Since the computational meshes are independent of time in the problems considered here, interpolation weights can be precomputed, thus reducing the necessary time for interpolating. As described below, this precomputation is quite straightforward in the linear case, but not in the cubic one. Therefore, weights are computed, in practice, by interpolating a function that is equal to the unity in a single cell and zero elsewhere, in order to obtain each column of matrices $D_{c\to f}$ or $D_{f\to c}$. For example, if $\boldsymbol{\psi}_f = 1$ in cell $\Omega_{f,j} \in \mathcal{T}_f$ and zero elsewhere, then the interpolation of $\boldsymbol{\psi}_f$ to $\mathcal{T}_c$ is equal to the j-th column of $D_{f\to c}$.

C.1 Delaunay triangulation

In this work, we opted for implementing flexible interpolation procedures, not relying on structured grids. Indeed, unstructured meshes are usually considered for the simulation of urban floods, due to the complex domain geometries and the presence of obstacles. We then implement interpolation procedures based on the Delaunay triangulation of the set of cells' barycenters of a computational mesh. A triangulation of a set of points $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_M\} \subset \mathbb{R}^2$ is a set of triangles whose vertices are elements of $\mathcal{X}$, intersecting only on their interfaces and whose union is the convex hull of $\mathcal{X}$. The Delaunay triangulation (Delaunay, 1934) of $\mathcal{X}$ is the unique triangulation $\widetilde{\mathcal{T}}$ such that, for every edge $\boldsymbol{x}_i\boldsymbol{x}_j$ of $\widetilde{\mathcal{T}}$, with $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$, there exists a circumference passing trough $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ not containing any point of $\mathcal{X}$ in its interior. We refer the reader to (Shewchuk, 1999) for a detailed review of Delaunay triangulation, its extension to higher dimensions and its use as mesh generation and refinement method. Figure C.1 illustrates the Delaunay triangulation of the cell's barycenter of a computational mesh.

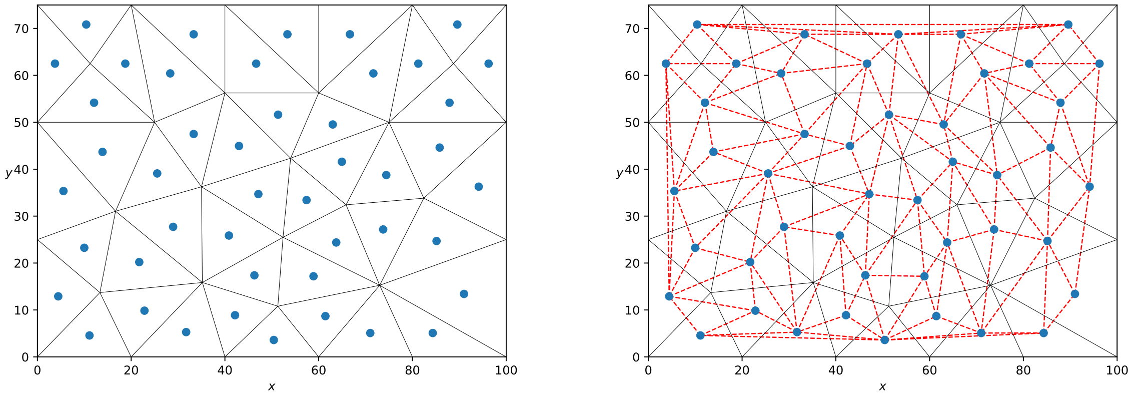


Figure C.1: Example of a Delaunay triangulation used for interpolation. Left: example of computational mesh, with the cells' barycenters marked as blue dots. Right: Delaunay triangulation (red, dashed lines) of the cells' barycenters.

In the FV scheme, we consider that the value $[\boldsymbol{\psi}_f]_i$ of $\boldsymbol{\psi}_f$ in a computational cell $\Omega_{f,i} \subset \mathcal{T}_f$ is defined on its barycenter $\boldsymbol{x}_{f,i}$ (we recall that the FV solution is constant in each cell). Therefore, if $\widetilde{\mathcal{T}}_f$ denotes the Delaunay triangulation of the barycenters $\mathcal{X}_f = \{\boldsymbol{x}_{f,1}, \dots, \boldsymbol{x}_{f,M_f}\}$ of the cells in $\mathcal{T}_f$, then each triangle of $\widetilde{\mathcal{T}}_f$ is associated to three values of $\boldsymbol{\psi}_f$ (defined on its vertices). Therefore, the interpolation from $\mathcal{T}_f$ to $\mathcal{T}_c$ consists in finding, for each cell barycenter $\boldsymbol{x}_{c,1}, \dots, \boldsymbol{x}_{c,M_c}$, of $\mathcal{T}_c$, the triangle of $\widetilde{\mathcal{T}}_f$ in which it is contained and determining the interpolation weights $D_{f\to c}$ based on a chosen interpolation order.

The interpolation from $\mathcal{T}_c$ to $\mathcal{T}_f$ is completely analogous: the Delaunay triangulation $\widetilde{\mathcal{T}}_c$ of the barycenters $\mathcal{X}_c = \{\boldsymbol{x}_{c,1}, \dots, \boldsymbol{x}_{c,M_c}\}$ is computed and each barycenter $\boldsymbol{x}_{f,1}, \dots, \boldsymbol{x}_{f,M_f}$, of $\mathcal{T}_f$ is located in a triangle of $\widetilde{\mathcal{T}}_c$ for determining the interpolation weights $D_{c\to f}$.

A special treatment must be considered in the case where the point to be located in a Delaunay triangulation lays outside its convex hull. This treatment depends on the implementation considered for each interpolation order (linear and cubic), as described in the following sections.

C.2 Linear interpolation

The linear interpolation procedure based on Delaunay triangulations is implemented and incorporated to the code developed in this work by using the open-source C++ library *delaunay_linterp*[1]. In order to interpolate from $\mathcal{T}_f$ to a cell $\Omega_{c,i} \subset \mathcal{T}_c$, its barycenter $\boldsymbol{x}_{c,i}$ is located in a triangle of the Delaunay triangulation $\widetilde{\mathcal{T}}_f$ and the interpolation weights are the barycenter coordinates of $\boldsymbol{x}_{c,i}$ relative to the found triangle. Therefore, each line of the interpolation matrix $D_{f\to c}$ contains at most three non-zero values. If $\boldsymbol{x}_{f,i_1}$, $\boldsymbol{x}_{f,i_2}$ and $\boldsymbol{x}_{f,i_3}$ denote the vertices of the found triangle, then their associated interpolation weights satisfy

$$
\begin{gathered}
\boldsymbol{x}_{c,i} = [D_{f\to c}]_{i,i_1}\boldsymbol{x}_{f,i_1} + [D_{f\to c}]_{i,i_2}\boldsymbol{x}_{f,i_2} + [D_{f\to c}]_{i,i_3}\boldsymbol{x}_{f,i_2} \\
[D_{f\to c}]_{i,i_1}, [D_{f\to c}]_{i,i_2}, [D_{f\to c}]_{i,i_3} \geq 0 \\
[D_{f\to c}]_{i,i_1} + [D_{f\to c}]_{i,i_2} + [D_{f\to c}]_{i,i_3} = 1
\end{gathered}
$$

[1] https://github.com/rncarpio/delaunay_linterp

In the case where $\boldsymbol{x}_{c,i}$ lays outside $\widetilde{\mathcal{T}}_f$, *delaunay_linterp* library projects $\boldsymbol{x}_{c,i}$ onto the closest external interface of $\widetilde{\mathcal{T}}_f$ and computes the barycenter coordinates of this projection. The computation of the interpolation matrix $D_{c\to f}$ is analogous.

C.3 Cubic interpolation

Since the library *delaunay_linterp* only implements linear interpolation, the cubic case, not incorporated to the code developed in this work, is computed previously with the function *CloughTocher2DInterpolator* of the Python Scipy library[2], which is also based on Delaunay triangulations. This function constructs piecewise cubic, continuously differentiable Bézier polynomials using a Clough-Tocher split interpolant scheme. We briefly explain these ingredients in the following paragraphs and we refer the reader to (Farin, 1986) for details.

Cubic Bézier polynomials

Let $\overline{\mathcal{T}} \in \mathbb{R}^2$ be a given triangle with vertices $\overline{\boldsymbol{x}}_1, \overline{\boldsymbol{x}}_2, \overline{\boldsymbol{x}}_3$ and barycenter $\overline{\boldsymbol{x}}_c$. Let also $\beta_1, \beta_2, \beta_3$ be the barycenter coordinates of a point $\boldsymbol{x} \in \overline{\mathcal{T}}$, *i.e.* $\boldsymbol{x} = \sum_{i=1}^{3} \beta_i \overline{\boldsymbol{x}}_i$. The Bézier polynomial of degree d over $\overline{\mathcal{T}}$ is defined as

$$b^d(\boldsymbol{x}) := b^d(\beta_1, \beta_2, \beta_3) := \sum_{\substack{i,j,k \in \mathbb{N} \\ i+j+k=d}} b_{ijk} B^d_{ijk}(\beta_1, \beta_2, \beta_3)$$

where b_{ijk} are called the Bézier ordinates of b^d and B^d_{ijk} are the Bernstein polynomials of degree d over $\overline{\mathcal{T}}$, defined by

$$B^d_{ijk}(\boldsymbol{x}) := B^d_{ijk}(\beta_1, \beta_2, \beta_3) := \frac{d!}{i!j!k!} \beta_1^i \beta_2^j \beta_3^k, \qquad i, j, k \in \mathbb{N}, \qquad i + j + k = d$$

and constituting a basis for all polynomials of degree d over $\overline{\mathcal{T}}$.

The Bézier ordinates are determined from data (function and/or derivative values, depending on the chosen interpolating shceme) provided at control points of $\overline{\mathcal{T}}$ of the form

$$\overline{\boldsymbol{x}}^d_{ijk} := \frac{i\overline{\boldsymbol{x}}_1 + j\overline{\boldsymbol{x}}_2 + k\overline{\boldsymbol{x}}_3}{d}, \qquad i, j, k \in \mathbb{N}, \qquad i + j + k = d \tag{C.1}$$

In the cubic case ($d = 3$), in which i, j, k vary from 0 to 3 and sum up 3, there are ten points under the form (C.1), given by the vertices of $\overline{\mathcal{T}}$, the partition of its edges in thirds, and its barycenter $\boldsymbol{x}_c$, as illustrated in Figure C.2.

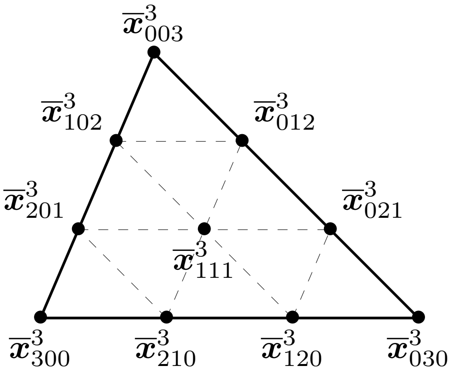

$\overline{\boldsymbol{x}}^3_{003}$
$\overline{\boldsymbol{x}}^3_{102}$
$\overline{\boldsymbol{x}}^3_{012}$
$\overline{\boldsymbol{x}}^3_{201}$
$\overline{\boldsymbol{x}}^3_{021}$
$\overline{\boldsymbol{x}}^3_{111}$
$\overline{\boldsymbol{x}}^3_{300}$
$\overline{\boldsymbol{x}}^3_{210}$
$\overline{\boldsymbol{x}}^3_{120}$
$\overline{\boldsymbol{x}}^3_{030}$

Figure C.2: Control points of a triangle $\overline{\mathcal{T}}$ used for defining a cubic Bézier polynomial.

[2] https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.CloughTocher2DInterpolator.html

Clough-Tocher split scheme

In the Clough-Tocher interpolant scheme, a given triangle $\breve{\mathcal{T}}$ with vertices $\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2, \breve{\boldsymbol{x}}_3$ and barycenter $\breve{\boldsymbol{x}}_c$ is splitted in three sub-triangles $\breve{\mathcal{T}}_1, \breve{\mathcal{T}}_2, \breve{\mathcal{T}}_3$ by joining each vertex to the barycenter, as shown in Figure C.3. Twelve data are used for constructing the cubic interpolant: the function value and gradient on the vertices $\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2, \breve{\boldsymbol{x}}_3$, in addition to normal derivatives on the midpoint of each edge of $\breve{\mathcal{T}}$.

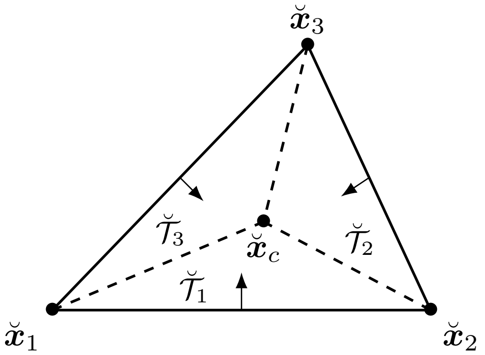

$\breve{\boldsymbol{x}}_3$
$\breve{\mathcal{T}}_3$
$\breve{\boldsymbol{x}}_c$
$\breve{\mathcal{T}}_2$
$\breve{\mathcal{T}}_1$
$\breve{\boldsymbol{x}}_1$
$\breve{\boldsymbol{x}}_2$

Figure C.3: Clough-Tocher split interpolant scheme. The triangle is divided in three sub-triangles and data for interpolation is provided on the vertices (function and gradient values) and on the midpoint of each edge (normal derivatives).

The cubic Bézier-Clough-Tocher interpolation

It can be shown that the interpolant constructed with the Clough-Tocher scheme is globally continuously differentiable if cubic polynomials are defined in each sub-triangle. It can be done, for example, by using Bézier polynomials in each $\breve{\mathcal{T}}_1, \breve{\mathcal{T}}_2, \breve{\mathcal{T}}_3$, resulting in the set of control points illustrated in Figure C.4. In the case of the interpolation scheme based on Delaunay triangulations, the represented triangle corresponds to a triangle of a Delaunay triangulation whose vertices $\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2, \breve{\boldsymbol{x}}_3$ are barycenters of the base computational mesh ($\mathcal{T}_c$ or $\mathcal{T}_f$). In Figure C.4, each control point is identified based on how its respective Bézier ordinate is computed: external points (red bullets) are defined from data defined on the vertices $\breve{\boldsymbol{x}}_1, \breve{\boldsymbol{x}}_2, \breve{\boldsymbol{x}}_3$; ordinates associate to blue square points are obtained form normal derivatives defined on edges' midpoints; ordinates at green triangle points are the average from neighbour blue squares; and for the barycenter (orange star), the ordinate is the average of green triangles.

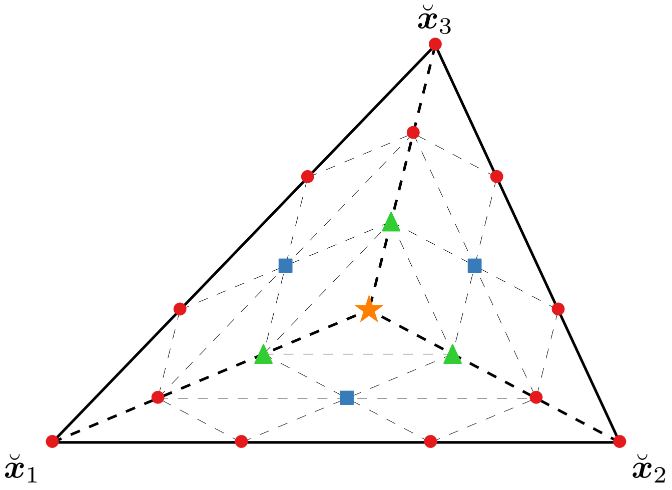

$\breve{\boldsymbol{x}}_3$
$\breve{\boldsymbol{x}}_1$
$\breve{\boldsymbol{x}}_2$

Figure C.4: Bézier Clough-Tocher split interpolation. A cubic Bézier polynomial is defined in each subtriangle obtained by the Clough-Tocher scheme. The Bézier ordinates in each control point are defined depending on the point position.

Approximation of derivatives

The interpolant construction requires the estimation of partial derivatives, which is performed in the *CloughTocher2DInterpolator* function by using global estimates (Nielson, 1983; Renka and Cline, 1984) (*i.e.* the estimated derivatives in each spatial point are influenced by the entire spatial domain). The approximate derivatives are the ones that minimize the curvature of the constructed interpolant.

As a consequence of this global estimation approach, the interpolation produces a very large number of nonzero weights, in general equal to the number of cells in the mesh (which means that the interpolation matrices $D_{c\to f}$ and $D_{f\to c}$ are dense, contrary to their sparse structures in the linear interpolation case). It implies a prohibitive cost in the framework of the parareal method, with interpolation procedures accounting for the majority of the total computational time. Therefore, in order to reduce this cost, we discard all weights smaller than 10^{-10}. It provides substantial reductions of the interpolation stencil and computational time for interpolation (even if still much larger than the linear case), with only negligible impacts on the parareal convergence behaviour.

Interpolation to points outside the Delaunay triangulation

Concerning the interpolation to a point $\boldsymbol{x}_{c,i}$ located outside the convex hull of the Delaunay triangulation $\widetilde{\mathcal{T}}_f$, the *CloughTocher2DInterpolator* function does not provide an alternative approach based on a projection of the point (as described above in the linear interpolation with *delaunay_linterp* library) and returns undefined values. Therefore, for treating these cases, we use a *nearest interpolation* procedure, *i.e.* the barycenter $\boldsymbol{x}_{f,\tilde{i}} \in \mathcal{T}_f$ with smaller distance to $\boldsymbol{x}_{c,i}$ is found and a unique weight $[D_{f\to c}]_{i,\tilde{i}} = 1$ is defined.

BIBLIOGRAPHY

Abel, N., Chaudhry, J., Falgout, R. D., and Schroder, J. (2020). Multigrid-reduction-in-time for the rotating shallow water equations.

Adikari, Y. and Yoshitani, J. (2009). Global trends in water-related disasters: an insight for policymakers. Technical report, World Water Assessment Programme Side Publication Series, Insights. The United Nations, UNESCO. International Centre for Water Hazard and Risk Management (ICHARM).

Ahmed, S. E., San, O., Bistrian, D. A., and Navon, I. M. (2020). Sampling and resolution characteristics in reduced order models of shallow water equations: Intrusive vs nonintrusive. *International Journal for Numerical Methods in Fluids*, 92(8):992–1036.

Akkari, N., Mercier, R., Lartigue, G., and Moureau, V. (2017). Stable pod-galerkin reduced order models for unsteady turbulent incompressible flows.

Allaire, G. (2010). Introduction to homogenization theory. CEA-EDF-INRIA school on homogenization. http://www.cmap.polytechnique.fr/ allaire/homog/lect1.pdf.

Anderson, T. B. and Jackson, R. (1967). Fluid mechanical description of fluidized beds. Equations of motion. *Industrial & Engineering Chemistry Fundamentals*, 6(4):527–539.

Antoulas, A. (2004). Approximation of large-scale dynamical systems: An overview. *IFAC Proceedings Volumes*, 37(11):19 – 28. 10th IFAC/IFORS/IMACS/IFIP Symposium on Large Scale Systems 2004: Theory and Applications, Osaka, Japan, 26-28 July, 2004.

Arbenz, P., Hiltebrand, A., and Obrist, D. (2012). A Parallel Space-Time Finite Difference Solver for Periodic Solutions of the Shallow-Water Equation. In Wyrzykowski, R., Dongarra, J., Karczewski, K., and Waśniewski, J., editors, *Parallel Processing and Applied Mathematics*, volume 7204 of *Lecture Notes in Computer Science*, pages 302–312. Springer Berlin Heidelberg.

Arce, P. E., Quintard, M., and Whitaker, S. (2005). *The Art and Science of Upscaling*, chapter 1, pages 1–39. John Wiley & Sons, Ltd.

Astorino, M., Chouly, F., and Quarteroni, A. (2012). Multiscale coupling of finite element and lattice Boltzmann methods for time dependent problems. Technical report.

Aubanel, E. (2011). Scheduling of tasks in the parareal algorithm. *Parallel Computing*, 37(3):172 – 182.

Audusse, E. and Bristeau, M.-O. (2003). Transport of pollutant in shallow water a two time steps kinetic method. *ESAIM: M2AN*, 37(2):389–416.

Audusse, E., Caldas Steinstraesser, J.G., Emerald, L., Heinrich, P., Paris, A., and Parisot, M. (2021). Comparison of models for the simulation of landslide generated tsunamis. *ESAIM: ProcS*, 70:14–30.

Auriault, J.-L. (2005). *Transport in Porous Media: Upscaling by Multiscale Asymptotic Expansions*, pages 3–56. Springer Vienna, Vienna.

Auriault, J.-L., Geindreau, C., and Boutin, C. (2005). Filtration law in porous media with poor separation of scales. *Transport in Porous Media*, 60:89–108.

Azaïez, M., Chacón, T., and Rubino, S. (2021). A cure for instabilities due to advection-dominance in POD solution to advection-diffusion-reaction equations. *Journal of Computational Physics*.

Baffico, L., Bernard, S., Maday, Y., Turinici, G., and Zérah, G. (2002). Parallel-in-time molecular-dynamics simulations. *Physical Review E : Statistical, Nonlinear, and Soft Matter Physics*, 66(5):057701.

Bal, G. (2005). On the convergence and the stability of the parareal algorithm to solve partial differential equations. In Barth, T. J., Griebel, M., Keyes, D. E., Nieminen, R. M., Roose, D., Schlick, T., Kornhuber, R., Hoppe, R., Périaux, J., Pironneau, O., Widlund, O., and Xu, J., editors, *Domain Decomposition Methods in Science and Engineering*, pages 425–432, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bal, G. and Maday, Y. (2002). A "parareal" time discretization for non-linear pde's with application to the pricing of an american put. In Pavarino, L. F. and Toselli, A., editors, *Recent Developments in Domain Decomposition Methods*, pages 189–202, Berlin, Heidelberg. Springer Berlin Heidelberg.

Barrault, M., Maday, Y., Nguyen, N. C., and Patera, A. T. (2004). An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339(9):667 – 672.

Barré de Saint-Venant, A. J. C. (1871). Théorie du mouvement non permanent des eaux, avec application aux crues des rivières et à l'introduction des marées dans leur lit. *Comptes Rendus des Séances de l'Académie des Sciences*, 73:147–154 and 237–240.

Bear, J. (1988). *Dynamics of Fluids in Porous Media.* Dover Publications Inc., New York.

Bellen, A. and Zennaro, M. (1989). Parallel algorithms for initial-value problems for difference and differential equations. *Journal of Computational and Applied Mathematics*, 25(3):341–350.

Bensoussan, A., Lions, J.-L., and Papanicolaou, G. (1978). *Asymptotic analysis for periodic structures.* North-Holland, Amsterdan.

Berkooz, G., Holmes, P., and Lumley, J. (2003). The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25:539–575.

Bistrian, D. and Navon, I. (2015). An improved algorithm for the shallow water equations model reduction: Dynamic mode decomposition vs pod. *International Journal for Numerical Methods in Fluids*, 78:552–580.

Bistrian, D. and Navon, I. (2017). The method of dynamic mode decomposition in shallow water and a swirling flow problem. *International Journal for Numerical Methods in Fluids*, DOI: 10.1002/fld.4257, Volume 83:73–89.

Blackford, S. and Dongarra, J. (1999). Lapack working note 41 installation guide for lapack 1.

Blumers, A., Yin, M., Nakajima, H., Hasegawa, Y., Li, Z., and Karniadakis, G. E. (2021). Multiscale parareal algorithm for long-time mesoscopic simulations of microvascular blood flow in zebrafish.

Bolten, M., Friedhoff, S., Hahne, J., and Schöps, S. (2020). Parallel-in-time simulation of an electrical machine using MGRIT. *Computing and Visualization in Science*, 23(1-4).

Borggaard, J., Hay, A., and Pelletier, D. (2007). Interval-based reduced-order models for unsteady fluid flow. *International Journal of Numerical Analysis and Modeling*, 4:353–367.

Borggaard, J., Wang, Z., and Zietsman, L. (2016). A goal-oriented reduced-order modeling approach for nonlinear systems. *Computers & Mathematics with Applications*, 71(11):2155 – 2169. Proceedings of the conference on Advances in Scientific Computing and Applied Mathematics. A special issue in honor of Max Gunzburger's 70th birthday.

Borregales, M., Kumar, K., Radu, F. A., Rodrigo, C., and Gaspar, F. J. (2019). A partially parallel-in-time fixed-stress splitting method for biot's consolidation model. *Computers & Mathematics with Applications*, 77(6):1466–1478. 7th International Conference on Advanced Computational Methods in Engineering (ACOMEN 2017).

Brocchini, M. and Dodd, N. (2008). Nonlinear shallow water equation modeling for coastal engineering. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 134(2):104–120.

Burrage, K. and Suhartanto, H. (1997). Parallel iterated method based on multistep Runge-Kutta of Radau type for stiff problems. *Advances in Computational Mathematics*, 7(1/2):59–77.

Caldas Steinstraesser, J. G., Delenne, C., Finaud-Guyot, P., Guinot, V., Kahn Casapia, J. L., and Rousseau, A. (2021a). SW2D-LEMON: a new software for upscaled shallow water modeling. In *Simhydro 2021 – 6th International Conference Models for complex and global water issues – Practices and expectations*.

Caldas Steinstraesser, J. G., Guinot, V., and Rousseau, A. (2020a). Modified parareal method for solving the two-dimensional nonlinear shallow water equations using finite volumes. *Journal of Computational Mathematics*. Submitted.

Caldas Steinstraesser, J. G., Guinot, V., and Rousseau, A. (2020b). A modified ROM-based parareal method for solving the two-dimensional nonlinear shallow water equations. CAN-J 2020 (Congrès d'Analyse Numérique pour les Jeunes). Online.

Caldas Steinstraesser, J. G., Guinot, V., and Rousseau, A. (2021b). Coupling shallow-water models at different scales using parallel-in-time methods. 8ème école EGRIN (Écoulements Gravitaires et RIsques Naturels). Online.

Caldas Steinstraesser, J. G., Vincent, and Rousseau, A. (2021c). Application of a modified parareal method for speeding up the numerical resolution of the 2D shallow water equations. In *Simhydro 2021 – 6th International Conference Models for complex and global water issues – Practices and expectations*.

Caliari, M., Einkemmer, L., Moriggl, A., and Ostermann, A. (2021). An accurate and time-parallel rational exponential integrator for hyperbolic and oscillatory pdes. *Journal of Computational Physics*, 437:110289.

Camphouse, R., Myatt, J., Schmit, R., Glauser, M., Ausseur, J., Andino, M., and Wallace, R. (2008). A snapshot decomposition method for reduced order modeling and boundary feedback control (postprint). *4th AIAA Flow Control Conference*, page 17.

Capdeville, Y., Cupillard, P., and Singh, S. (2020). Chapter six - an introduction to the two-scale homogenization method for seismology. In Moseley, B. and Krischer, L., editors, *Machine Learning in Geosciences*, volume 61 of *Advances in Geophysics*, pages 217–306. Elsevier.

Carlberg, K., Brencher, L., Haasdonk, B., and Barth, A. (2016). Data-driven time parallelism via forecasting. *CoRR*, abs/1610.09049.

Carlberg, K., Ray, J., and van Bloemen Waanders, B. (2015). Decreasing the temporal complexity for nonlinear, implicit reduced-order models by forecasting. *Computer Methods in Applied Mechanics and Engineering*, 289:79 – 103.

Chaturantabut, S. and Sorensen, D. C. (2010). Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764.

Chaturantabut, S. and Sorensen, D. C. (2011). Application of POD and DEIM on dimension reduction of non-linear miscible viscous fingering in porous media. *Mathematical and Computer Modelling of Dynamical Systems*, 17(4):337–353.

Chen, A. S., Evans, B., Djordjević, S., and Savić, D. A. (2012a). A coarse-grid approach to representing building blockage effects in 2d urban flood modelling. *Journal of Hydrology*, 426-427:1 – 16.

Chen, A. S., Evans, B., Djordjević, S., and Savić, D. A. (2012b). Multi-layered coarse grid modelling in 2d urban flood simulations. *Journal of Hydrology*, 470-471:1 – 11.

Chen, F., Hesthaven, J. S., and Zhu, X. (2014). *On the Use of Reduced Basis Methods to Accelerate and Stabilize the Parareal Method*, pages 187–214. Springer International Publishing, Cham.

Chen, H., Reuss, D., and Sick, V. (2012c). On the use and interpretation of proper orthogonal decomposition of in-cylinder engine flows. *Measurement Science & Technology - MEAS SCI TECHNOL*, 23.

Chen, W., Hesthaven, J. S., Junqiang, B., Qiu, Y., Tihao, Y., and Yang, Z. (2018). Greedy non-intrusive reduced order model for fluid dynamics. *AIAA Journal*, 56:12.

Christlieb, A. J., Macdonald, C. B., and Ong, B. W. (2010). Parallel high-order integrators. *SIAM Journal on Scientific Computing*, 32(2):818–835.

Churuksaeva, V. and Starchenko, A. (2015). Mathematical modeling of a river stream based on a shallow water approach. *Procedia Computer Science*, 66:200–209. 4th International Young Scientist Conference on Computational Science.

Cortial, J. and Farhat, C. (2009). A time-parallel implicit method for accelerating the solution of non-linear structural dynamics problems. *International Journal for Numerical Methods in Engineering*, 77(4):451–470.

CRED and UNISDR (2015). The human cost of weather related disasters - 1995 - 2015.

Dai, X. and Maday, Y. (2011). Stable parareal in time method for first and second order hyperbolic system. Technical report.

Davit, Y. and Quintard, M. (2017). Technical notes on volume averaging in porous media i: How to choose a spatial averaging operator for periodic and quasiperiodic structures. *Transport in Porous Media*, 119:1–30.

De Sterck, H., Falgout, R. D., Friedhoff, S., Krzysik, O. A., and MacLachlan, S. P. (2021). Optimizing multigrid reduction-in-time and parareal coarse-grid operators for linear advection. *Numerical Linear Algebra with Applications*.

Defina, A. (2000). Two-dimensional shallow flow equations for partially dry areas. *Water Resources Research*, 36(11):3251–3264.

Defina, A., D'Alpaos, L., and Matticchio, B. (1994). New set of equations for very shallow water and partially dry areas suitable to 2D numerical models. *Proceedings of the Specialty Conference on Modelling of Flood Propagation Over Initially Dry Areas*, pages 72–81.

Delaunay, B. (1934). Sur la sphère vide. *A la memoire de Georges Voronoi, Bulletin de l'Academie des Sciences de l'URSS. Classe des sciences mathematiques et na*, 6:793–800.

Dolean, V., Jolivet, P., and Nataf, F. (2015). *An Introduction to Domain Decomposition Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

Dongarra, J., Gates, M., Haidar, A., Kurzak, J., Luszczek, P., Tomov, S., and Yamazaki, I. (2018). The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale. *SIAM Review*, 60(4):808–865.

Duarte, M., Massot, M., and Descombes, S. (2011). Parareal operator splitting techniques for multi-scale reaction waves: Numerical analysis and strategies. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 45(5):825–852.

Eghbal, A., Gerber, A. G., and Aubanel, E. (2017). Acceleration of unsteady hydrodynamic simulations using the parareal algorithm. *Journal of Computational Science*, 19:57 – 76.

El Adawy, M., Heikal, M., Aziz, A., Adam, I., Ismael, M., Babiker, M., Baharom, M., Firmansyah, F., and Zainal A., E. Z. (2018). On the application of proper orthogonal decomposition (POD) for in-cylinder flow analysis. *Energies*, 11:2261.

Emmett, M. and Minion, M. L. (2012). Toward an Efficient Parallel in Time Method for Partial Differential Equations. *Communications in Applied Mathematics and Computational Science*, 7:105–132.

Everson, R. and Sirovich, L. (1995). Karhunen-Loeve procedure for gappy data. *Journal of the Optical Society of America A*, 12(8):1657–1664.

Falgout, R. D., Friedhoff, S., Kolev, T. V., MacLachlan, S. P., and Schroder, J. B. (2014). Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635–C661.

Farhat, C. and Chandesris, M. (2003). Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid–structure applications. *International Journal for Numerical Methods in Engineering*, 58(9):1397–1434.

Farhat, C., Cortial, J., Dastillung, C., and Bavestrello, H. (2006). Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses. *International Journal for Numerical Methods in Engineering*, 67:697–724.

Farin, G. (1986). Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3(2):83–127.

Farmer, C. L. (2002). Upscaling: a review. *International Journal for Numerical Methods in Fluids*, 40(1-2):63–78.

Filippini, A., Bellec, S., Colin, M., and Ricchiuto, M. (2015). On the nonlinear behaviour of boussinesq type models: Amplitude-velocity vs amplitude-flux forms. *Coastal Engineering*, 99:109–123.

Fischer, P. F., Hecht, F., and Maday, Y. (2005). A parareal in time semi-implicit approximation of the Navier-Stokes equations. In Barth, T. J., Griebel, M., Keyes, D. E., Nieminen, R. M., Roose, D., Schlick, T., Kornhuber, R., Hoppe, R., Périaux, J., Pironneau, O., Widlund, O., and Xu, J., editors, *Domain Decomposition Methods in Science and Engineering*, pages 433–440, Berlin, Heidelberg. Springer Berlin Heidelberg.

Fraccarollo, L. and Capart, H. (2002). Riemann wave description of erosional dam-break flows. *Journal of Fluid Mechanics*, 461:183–228.

Friedhoff, S., Falgout, R. D., Kolev, T. V., MacLachlan, S. P., and Schroder, J. B. (2013). A Multigrid-in-Time Algorithm for Solving Evolution Equations in Parallel. In *Presented at: Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, United States, Mar 17 - Mar 22, 2013*.

Gallegos, H. A., Schubert, J. E., and Sanders, B. F. (2009). Two-dimensional, high-resolution modeling of urban dam-break flooding: A case study of baldwin hills, california. *Advances in Water Resources*, 32(8):1323–1335.

Gander, M. and Vandewalle, S. (2007). Analysis of the parareal time-parallel time-integration method. *SIAM J. Scientific Computing*, 29:556–578.

Gander, M. J. (1996). *Overlapping Schwarz for Linear and Nonlinear Parabolic Problems*, pages 97–104. 9th International Conference on Domain Decomposition Methods. ID: unige:8288.

Gander, M. J. (2008). Analysis of the parareal algorithm applied to hyperbolic problems using characteristics. *Boletín de la Sociedad Española de Matemática Aplicada*, 42:21–35. ID: unige:6268.

Gander, M. J. (2015). 50 years of time parallel time integration. In Carraro, T., Geiger, M., Körkel, S., and Rannacher, R., editors, *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113, Cham. Springer International Publishing.

Gander, M. J., Jiang, Y.-L., and Li, R.-J. (2013). Parareal Schwarz waveform relaxation methods. In Bank, R., Holst, M., Widlund, O., and Xu, J., editors, *Domain Decomposition Methods in Science and Engineering XX*, pages 451–458, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gander, M. J. and Petcu, M. (2008). Analysis of a Krylov subspace enhanced parareal algorithm for linear problems. *ESAIM: Proc.*, 25:114–129.

Geiser, J. and Güttel, S. (2012). Coupling methods for heat transfer and heat flow: Operator splitting and the parareal algorithm. *Journal of Mathematical Analysis and Applications*, 388(2):873 – 887.

Gerbeau, J.-F. and Perthame, B. (2001). Derivation of viscous Saint-Venant system for laminar shallow water; numerical validation. *Discrete & Continuous Dynamical Systems - B*, 1(1531-3492_2001_1_89):89.

Grepl, M. A., Maday, Y., Nguyen, N. C., and Patera, A. T. (2007). Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. *ESAIM: M2AN*, 41(3):575–605.

Grigori, L., Hirstoaga, S. A., Nguyen, V.-T., and Salomon, J. (2021). Reduced model-based parareal simulations of oscillatory singularly perturbed ordinary differential equations. *Journal of Computational Physics*, 436:110282.

Guinot, V. (2012). Multiple porosity shallow water models for macroscopic modelling of urban floods. *Advances in Water Resources*, 37:40 – 72.

Guinot, V. (2017). Consistency and bicharacteristic analysis of integral porosity shallow water models. Explaining model oversensitivity to mesh design. *Advances in Water Resources*, 107:43–55.

Guinot, V., Delenne, C., Rousseau, A., and Boutron, O. (2018). Flux closures and source term models for shallow water models with depth-dependent integral porosity. *Advances in Water Resources*, 122:1 – 26.

Guinot, V., Sanders, B. F., and Schubert, J. E. (2017). Dual integral porosity shallow water model for urban flood modelling. *Advances in Water Resources*, 103:16 – 31.

Guinot, V. and Soares-Frazão, S. (2006). Flux and source term discretization in two-dimensional shallow water models with porosity on unstructured grids. *International Journal for Numerical Methods in Fluids*, 50(3):309–345.

Guo, K., Guan, M., and Yu, D. (2020). Urban surface water flood modelling – a comprehensive review of current models and future challenges. *Hydrology and Earth System Sciences Discussions*, 2020:1–27.

Haasdonk, Bernard and Ohlberger, Mario (2008). Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: M2AN*, 42(2):277–302.

Hamon, F. P., Schreiber, M., and Minion, M. L. (2020). Parallel-in-time multi-level integration of the shallow-water equations on the rotating sphere. *Journal of Computational Physics*, 407:109210.

Haut, T. and Wingate, B. (2014). An asymptotic parallel-in-time method for highly oscillatory PDEs. *SIAM Journal on Scientific Computing*, 36(2):A693–A713.

He, L. (2010). The reduced basis technique as a coarse solver for parareal in time simulations. *Journal of Computational Mathematics*, 28(5):676–692.

Hernández, J., Caicedo, M., and Ferrer, A. (2017). Dimensional hyper-reduction of nonlinear finite element models via empirical cubature. *Computer Methods in Applied Mechanics and Engineering*, 313:687–722.

Hervouet, J.-M., Samie, R., and Moreau, B. (2000). Modelling urban areas in dam-break flood-wave numerical simulations. In *Proceedings of the International Seminar and Workshop on Rescue Actions Based on Dambreak Flow*, Seinâjoki, Finland.

Howes, F. A. and Whitaker, S. (1985). The spatial averaging theorem revisited. *Chemical Engineering Science*, 40(8):1387–1392.

Hénonin, J., Russo, B., Mark, O., and Gourbesville, P. (2013). Real-time urban flood forecasting and modelling - a state of the art. *Journal of Hydroinformatics*, accepted.

Iizuka, M. and Ono, K. (2018). Influence of the phase accuracy of the coarse solver calculation on the convergence of the parareal method iteration for hyperbolic PDEs. *Computing and Visualization in Science*, 19(3-4):97–108.

Ijzerman, W. (2000). *Signal Representation and Modeling of Spatial Structures in Fluids.* PhD thesis, Universiteit Twente.

Jacobson, C. R. (2011). Identification and quantification of the hydrological impacts of imperviousness in urban catchments: A review. *Journal of Environmental Management*, 92(6):1438–1448.

Jha, A., Lamond, J., Bloch, R., Bhattacharya, N., Lopez, A., Papachristodoulou, N., Bird, A., Proverbs, D., Davies, J., and Barker, R. (2011). *Five Feet High and Rising: Cities and Flooding in the 21st Century.* The World Bank.

Kim, B., Sanders, B. F., Famiglietti, J. S., and Guinot, V. (2015). Urban flood modeling with porous shallow-water equations: A case study of model errors in the presence of anisotropic porosity. *Journal of Hydrology*, 523:680 – 692.

Kim, B., Sanders, B. F., Schubert, J. E., and Famiglietti, J. S. (2014). Mesh type tradeoffs in 2D hydrodynamic modeling of flooding with a godunov-based flow solver. *Advances in Water Resources*, 68:42–61.

Kosambi, D. D. (1943). Statistics in function space. *Journal of the Indian Mathematical Society*, 7:76–88.

LeFloch, P. G. and Thanh, M. D. (2007). The Riemann problem for the shallow water equations with discontinuous topography. *Communications in Mathematical Sciences*, 5(4):865 – 885.

Legoll, F., Lelièvre, T., Myerscough, K., and Samaey, G. (2020). Parareal computation of stochastic differential equations with time-scale separation: a numerical convergence study. *Computing and Visualization in Science*, 23(1-4).

Lelarasmee, E., Ruehli, A., and Sangiovanni-Vincentelli, A. (1982). The waveform relaxation method for time-domain analysis of large scale integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1(3):131–145.

Lhomme, J. (2006). *One-dimensional, two-dimensional and macroscopie approaches to urban flood modelling.* PhD thesis, University of Montpellier.

Lions, J.-L., Maday, Y., and Turinici, G. (2001). Résolution d'edp par un schéma en temps 'pararéel'. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 332(7):661 – 668.

Lions, P. L. (1988). On the Schwarz Alternating Method. I. In *1st International Symposium on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia*, pages 1–42.

Lozovskiy, A., Farthing, M., Kees, C., and Gildin, E. (2016). POD-based model reduction for stabilized finite element approximations of shallow water flows. *Journal of Computational and Applied Mathematics*, 302:50–70.

Lunet, T. (2018). *Stratégies de parallélisation espace-temps pour la simulation numérique des écoulements turbulents.* PhD thesis. Thèse de doctorat dirigée par Gratton, Serge et Bodart, Julien Mathématiques appliquées Toulouse, ISAE 2018.

Lunet, T., Bodart, J., Gratton, S., and Vasseur, X. (2018). Time-parallel simulation of the decay of homogeneous turbulence using parareal with spatial coarsening. *Computing and Visualization in Science*, 19(1-2):31–44.

Maday, Y. (2010). The 'parareal in time' algorithm. In Magoulès, F., editor, *Substructuring Techniques and Domain Decomposition Methods*, Computational science, engineering and technology series, chapter 2, pages 19–44. Saxe-Coburg Publications.

Maday, Y. and Mula, O. (2020). An Adaptive Parareal Algorithm. *Journal of Computational and Applied Mathematics.*

Maday, Y., Nguyen, N., Patera, A., and Pau, G. S. H. (2008). A general multipurpose interpolation procedure: The magic points. *Communications on Pure and Applied Analysis*, 8.

Maday, Y. and Turinici, G. (2002a). A parallel in time approach for quantum control: the parareal algorithm. In *IEEE Conference on Decision and Control*, volume 1, pages 62–66, Las Vegas, United States.

Maday, Y. and Turinici, G. (2002b). A parareal in time procedure for the control of partial differential equations. *Comptes Rendus Mathematique*, 335(4):387 – 392.

Margenberg, N. and Richter, T. (2021). Parallel time-stepping for fluid–structure interactions. *Mathematical Modelling of Natural Phenomena*, 16:20.

Markov, I. L. (2014). Limits on fundamental limits to computation. *Nature*, 512(7513):147–154.

Marquez, A., Espinosa Oviedo, J., and Odloak, D. (2013). Model reduction using proper orthogonal decomposition and predictive control of distributed reactor system. *Journal of Control Science and Engineering*, 2013.

Mercerat, E., Guillot, L., and Vilotte, J.-P. (2009). Application of the parareal algorithm for acoustic wave propagation. *AIP Conference Proceedings*, 1168:1521–1524.

Message Passing Interface Forum (2009). MPI: A message-passing interface standard, version 2.2. Specification.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, 38(8).

Moore, G. E. (1975). Progress in digital integrated electronics. In *International Electron Devices Meeting, IEEE, 11-13.*

Müller, M. (2008). *On the POD method: an abstract investigation with applications to reduced-order modeling and suboptimal control.*

Narasimha, R. (2011). Kosambi and proper orthogonal decomposition. *Resonance*, 16:574–581.

Neelz, S. and Pender, G. (2013). Benchmarking the latest generation of 2Dc1 hydraulic modelling packages. Technical report, DEFRA/Environment Agency, UK.

Nielsen, A. S., Brunner, G., and Hesthaven, J. S. (2018). Communication-aware adaptive parareal with application to a nonlinear hyperbolic system of partial differential equations. *Journal of Computational Physics*, 371:483–505.

Nielson, G. (1983). A method for interpolating scattered data based upon a minimum norm network. *Mathematics of Computation - Math. Comput.*, 40:253–253.

Nievergelt, J. (1964). Parallel methods for integrating ordinary differential equations. *Commun. ACM*, 7(12):731–733.

Ong, B. W. and Schroder, J. B. (2020). Applications of time parallelization. *Computing and Visualization in Science*, 23(1-4).

OpenMP Architecture Review Board (2011). OpenMP application program interface version 3.1.

Otero, F., Oller, S., Martinez, X., and Salomón, O. (2015). Numerical homogenization for composite materials analysis. comparison with other micro mechanical formulations. *Composite Structures*, 122:405–416.

Özgen, I., Liang, D., and Hinkelmann, R. (2016a). Shallow water equations with depth-dependent anisotropic porosity for subgrid-scale topography. *Applied Mathematical Modelling*, 40(17):7447 – 7473.

Özgen, I., Zhao, J., Liang, D., and Hinkelmann, R. (2016b). Urban flood modeling using shallow water equations with depth-dependent anisotropic porosity. *Journal of Hydrology*, 541:1165 – 1184.

Parkhurst, J., Darringer, J., and Grundmann, B. (2006). From single core to multi-core: Preparing for a new exponential. In *2006 IEEE/ACM International Conference on Computer Aided Design*, pages 67–72.

Renka, R. and Cline, A. (1984). A triangle-based c1 interpolation method. *Rocky Mountain Journal of Mathematics*, 14(1).

Rosa-Raíces, J. L., Zhang, B., and Miller, T. F. (2019). Path-accelerated stochastic molecular dynamics: Parallel-in-time integration using path integrals. *The Journal of Chemical Physics*, 151(16):164120.

Rowley, C. W. (2005). Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos*, 15(03):997–1013.

Rowley, C. W., Colonius, T., and Murray, R. M. (2004). Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1):115 – 129.

Rozza, G., Huynh, D. B. P., and Patera, A. T. (2008). Reduced Basis Approximation and a Posteriori Error Estimation for Affinely Parametrized Elliptic Coercive Partial Differential Equations. *Archives of Computational Methods in Engineering*, 15(3):229 – 275.

Ruprecht, D. (2014). Convergence of parareal with spatial coarsening. *PAMM*, 14(1):1031–1034.

Ruprecht, D. (2018). Wave propagation characteristics of Parareal. *Computing and Visualization in Science*, 19:1–17.

Ruprecht, D. and Krause, R. (2012). Explicit parallel-in-time integration of a linear acoustic-advection system. *Computers Fluids*, 59:72–83.

Ruprecht, D., Speck, R., and Krause, R. (2016). Parareal for diffusion problems with space- and time-dependent coefficients. In Dickopf, T., Gander, M. J., Halpern, L., Krause, R., and Pavarino, L. F., editors, *Domain Decomposition Methods in Science and Engineering XXII*, pages 371–378, Cham. Springer International Publishing.

Salmoiraghi, F., Ballarin, F., Corsi, G., Mola, A., Tezzele, M., and Rozza, G. (2016). Advances in geometrical parametrization and reduced order models and methods for computational fluid dynamics problems in applied sciences and engineering: Overview and perspectives. pages 1013–1031.

Samaddar, D., Coster, D., Bonnin, X., Berry, L., Elwasif, W., and Batchelor, D. (2019). Application of the parareal algorithm to simulations of elms in iter plasma. *Computer Physics Communications*, 235:246–257.

San, O. and Borggaard, J. (2015). Principal interval decomposition framework for POD reduced-order modeling of convective boussinesq flows. *International Journal for Numerical Methods in Fluids*, 78(1):37–62.

Sanders, B. F., Schubert, J. E., and Gallegos, H. A. (2008). Integral formulation of shallow-water equations with anisotropic porosity for urban flood modeling. *Journal of Hydrology*, 362(1):19 – 38.

Schilders, W. (2008). *Introduction to Model Order Reduction*, volume 13, pages 3–32.

Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28.

Schmitt, A., Schreiber, M., Peixoto, P., and Schäfer, M. (2018). A numerical study of a semi-Lagrangian parareal method applied to the viscous Burgers equation. *Computing and Visualization in Science*, 19:45–57.

Schöbel, R. and Speck, R. (2020). PFASST-ER: combining the parallel full approximation scheme in space and time with parallelization across the method. *Computing and Visualization in Science*, 23(1-4).

Schreiber, M. and Loft, R. (2019). A parallel time integrator for solving the linearized shallow water equations on the rotating sphere. *Numerical Linear Algebra with Applications*, 26(2):e2220.

Schreiber, M., Schaeffer, N., and Loft, R. (2019). Exponential integrators with parallel-in-time rational approximations for shallow-water equations on the rotating sphere. *Parallel Computing*.

Schubert, J. E. and Sanders, B. F. (2012). Building treatments for urban flood inundation models and implications for predictive skill and modeling efficiency. *Advances in Water Resources*, 41:49 – 64.

Schwarz, H. A. (1870). Über einen Grenzübergang durch alternierendes Verfahren. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich*.

Shalf, J. (2020). The future of computing beyond Moore's law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 378(2166):20190061.

Shewchuk, J. R. (1999). Lecture notes on Delaunay mesh generation. Technical report, Department of Electrical Engineering and Computer Science, University of California at Berkeley.

Slattery, J. C. (1967). Flow of viscoelastic fluids through porous media. *AIChE Journal*, 13(6):1066–1071.

Soares Frazão, S. and Guinot, V. (2007). An eigenvector-based linear reconstruction scheme for the shallow-water equations on two-dimensional unstructured meshes. *International Journal for Numerical Methods in Fluids*, 53(1):23–55.

Soares-Frazão, S., Lhomme, J., Guinot, V., and Zech, Y. (2008). Two-dimensional shallow-water model with porosity for urban flood modelling. *Journal of Hydraulic Research*, 46(1):45–64.

Stabile, G., Hijazi, S., Mola, A., Lorenzi, S., and Rozza, G. (2017). Pod-galerkin reduced order methods for cfd using finite volume discretisation: vortex shedding around a circular cylinder. *Communications in Applied and Industrial Mathematics*, 8(1):210–236.

Ştefănescu, R. and Navon, I. M. (2013). POD/DEIM nonlinear model order reduction of an adi implicit shallow water equations model. *Journal of Computational Physics*, 237:95–114.

Ştefănescu, R., Sandu, A., and Navon, I. M. (2014). Comparison of POD reduced order strategies for the nonlinear 2d shallow water equations. *International Journal for Numerical Methods in Fluids*, 76(8):497–521.

Steiner, J., Ruprecht, D., Speck, R., and Krause, R. (2015). Convergence of parareal for the Navier-Stokes equations depending on the reynolds number. In Abdulle, A., Deparis, S., Kressner, D., Nobile, F., and Picasso, M., editors, *Numerical Mathematics and Advanced Applications - ENUMATH 2013*, pages 195–202, Cham. Springer International Publishing.

Stump, B. and Plotkowski, A. (2020). Spatiotemporal parallelization of an analytical heat conduction model for additive manufacturing via a hybrid OpenMP+MPI approach. *Computational Materials Science*, 184:109861.

Sundararajan, D. (2001). *The discrete Fourier transform : theory, algorithms and applications.* World Scientific, Singapore River Edge, NJ.

Takatsu, Y. (2017). Modification of the fundamental theorem for transport phenomena in porous media. *International Journal of Heat and Mass Transfer*, 115:1109–1120.

Teng, J., Jakeman, A., Vaze, J., Croke, B., Dutta, D., and Kim, S. (2017). Flood inundation modelling: A review of methods, recent advances and uncertainty analysis. *Environmental Modelling & Software*, 90:201–216.

Trindade, J. M. F. d. and Pereira, J. F. (2004). Parallel-in-time simulation of the unsteady Navier–Stokes equations for incompressible flow. *International Journal for Numerical Methods in Fluids*, 45(10):1123–1136.

Tucci, C. M. (2007). Urban flood management. Technical report, World Meteorological Organization. Cap-Net International Network for Capacity Building in Integrated Water Resources Management. WMO/TD No. 1372.

United Nations (2019). *World Urbanization Prospects: The 2018 Revision.* United Nations, Department of Economic and Social Affairs, Population Division.

Velickovic, M., Zech, Y., and Soares-Frazão, S. (2017). Steady-flow experiments in urban areas and anisotropic porosity model. *Journal of Hydraulic Research*, 55(1):85–100.

Verkley, W. T. M. (01 Jun. 2009). A balanced approximation of the one-layer shallow-water equations on a sphere. *Journal of the Atmospheric Sciences*, 66(6):1735 – 1748.

Viero, D. P. and Valipour, M. (2017). Modeling anisotropy in free-surface overland and shallow inundation flows. *Advances in Water Resources*, 104:1 – 14.

Wang, Z., Mcbee, B., and Iliescu, T. (2015). Approximate partitioned method of snapshots for POD. *Journal of Computational and Applied Mathematics*, 307.

Washington, W. M., Buja, L., and Craig, A. (2009). The computational future for climate and earth system models: on the path to petaflop and beyond. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1890):833–846.

Wedi, N., Mengaldo, G., Wyszogrodzki, A., Diamantakis, M., Lock, S.-J., and Giraldo, F. (2019). Current and emerging time-integration strategies in global numerical weather and climate prediction. *Archives of Computational Methods in Engineering*, 26:663–684.

Wen, X.-H. and Gómez-Hernández, J. (1996). Upscaling hydraulic conductivities in heterogeneous media: An overview. *Journal of Hydrology*, 183(1):ix–xxxii.

Whitaker, S. (1967). Diffusion and dispersion in porous media. *AIChE Journal*, 13(3):420–427.

Whitaker, S. (1999). *The Method of Volume Averaging*. Kluwer Academic Publishers.

Willis, J. R. (1997). *Dynamics of Composites*, pages 265–290. Springer Vienna, Vienna.

Wu, S.-L. and Liu, J. (2020). A parallel-in-time block-circulant preconditioner for optimal control of wave equations. *SIAM Journal on Scientific Computing*, , Accepted.

Xing, Y., Liang, Q., Wang, G., Ming, X., and Xia, X. (2018). City-scale hydrodynamic modelling of urban flash floods: the issues of scale and resolution. *Natural Hazards*, 96(1):473–496.

Yang, C., Thovert, J.-F., and Debenest, G. (2015). Upscaling of mass and thermal transports in porous media with heterogeneous combustion reactions. *International Journal of Heat and Mass Transfer*, 84:862–875.

Zeng, Y., Duan, Y., and Liu, B.-S. (2019). Solving heat equations in electro-magnetics by using time parareal coupling with meshless collocation rbfs methods. *Journal of Scientific Computing*.

Zhao, P., Liu, C., and Feng, X. (2014). POD-DEIM based model order reduction for the spherical shallow water equations with turkel-zwas finite difference discretization. *J. Appl. Math.*, 2014:10 pages.

Zokagoa, J. M. and Soulaimani, A. (2018). A POD-based reduced-order model for uncertainty analyses in shallow water flows. *International Journal of Computational Fluid Dynamics*, pages 1–15.