Parallel In Time Algorithms for Data Assimilation

Rishabh Bhatt

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Mathématiques Appliquées
Unité de recherche : Laboratoire Jean Kuntzmann

Algorithmes parallèles en temps pour l'assimilation de données

Parallel In Time Algorithms for Data Assimilation

Présentée par :

Rishabh BHATT

Direction de thèse :

Laurent DEBREU — Directeur de thèse
DIRECTEUR DE RECHERCHE, INRIA CENTRE GRENOBLE-RHONE-ALPES

Arthur VIDARD — Co-directeur de thèse
CHARGE DE RECHERCHE HDR, INRIA CENTRE GRENOBLE-RHONE-ALPES

Rapporteurs :

ANTOINE ROUSSEAU
DIRECTEUR DE RECHERCHE, CENTRE INRIA D'UNIVERSITE COTE D'AZUR

JULIEN SALOMON
DIRECTEUR DE RECHERCHE, INRIA CENTRE DE PARIS

Thèse soutenue publiquement le **23 novembre 2023**, devant le jury composé de :

ANTOINE ROUSSEAU — Rapporteur
DIRECTEUR DE RECHERCHE, CENTRE INRIA D'UNIVERSITE COTE D'AZUR

JULIEN SALOMON — Rapporteur
DIRECTEUR DE RECHERCHE, INRIA CENTRE DE PARIS

CAROLINE JAPHET — Examinatrice
MAITRE DE CONFERENCES, UNIVERSITE SORBONNE PARIS NORD

MARTIN SCHREIBER — Président
PROFESSEUR DES UNIVERSITES, UNIVERSITE GRENOBLE ALPES

EHOUARN SIMON — Examinateur
MAITRE DE CONFERENCES, INP - ENSEEIHT TOULOUSE

LAURENT DEBREU — Directeur de thèse
DIRECTEUR DE RECHERCHE, INRIA CENTRE GRENOBLE-RHONE-ALPES

ARTHUR VIDARD — Co-directeur de thèse
CHARGE DE RECHERCHE HDR, INRIA CENTRE GRENOBLE-RHONE-ALPES

Acknowledgements

I would first like to thank my supervisors Dr. Laurent Debreu and Dr. Arthur Vidard for their constant support and encouragement throughout my PhD journey. I am grateful for the opportunity they gave me to carry out research on a very interesting topic which involved some nice discussions. I feel this experience has taught me a lot and I have become more mature as a young researcher. I am also very grateful to the Agence Nationale de la Recherche (ANR) for providing the funding to make this PhD possible. My appreciation extends to the jury members of my thesis, for taking out their time to review my work and providing me some invaluable comments.

The past 4 years in France has been nothing short of wonderful and I had a great time meeting some great people. My smooth integration into the lab would not have been possible had it not been the colleagues from the AIRSEA team and office 196. I had some memorable time in the office all because of you guys. Not to forget my colleagues and friends from LJK who have a lot of energy and enthusiasm to keep the spirits high and maintain a lovely environment in the lab. I would certainly miss the volleyball and football sessions, trying out new cuisines in the city, and the regular meetup at Ève.

Special thanks to my friends outside the lab who have always been there for me from day one till now, helping me in all sort of situations and making me feel at home in Grenoble.

Lastly, I would like to thank my mom, my dad and my brother for their unwavering support and belief in me. This thesis is dedicated to you.

Contents

List of Algorithms

List of Figures

List of Tables

Abstract

Four dimensional variational data assimilation (4DVAR) which is based on optimisation algorithms is used by the leading meterological institutions as a means of initialising the numerical climate models. The optimal initial condition is found by minimisng a cost function which accounts for the misfits of the model trajectory with the observations of the system over a given time window. In its incremental formulation, the integration of the forward and adjoint version of the original model is required in order to compute the gradient. A common issue in the retreival of the initial condition is the enormous size of the state variable (10^9) which makes the minimisation an expensive and time consuming task. Moreover 4DVAR is an inherently sequential algorithm and to use it in parallel architectures, the models are clasically parallelised only in the spatial dimension. This limits the scope of further speed up once spatial saturation is reached and also the maximum number of computing cores that one can use. The objective of this PhD is to introduce an additional time-paralllelisation in the data assimilation framework by using the well known parareal method. Our approach is used here for running the forward integration. We use a modified version of the inexact conjugate gradient method where the matrix-vector multiplications are supplied through the parareal and thus are not exact. The associated convergence conditions of the inexact conjugate gradient allows us to use parareal adaptively by monitoring the errors in the matrix-vector product and obtaining the same levels of accuracy as with the usual conjugate gradient method at the same time. To ensure the feasibility and a practical implementation, all the norms which are hard to compute are replaced by the easily computable approximations. The results are demonstrated by considering the one and two dimensional shallow water model. They are presented in terms of the accuracy (in comparison with the original exact conjugate gradient) and in terms of the number of required iterations of the parareal algorithm. For the more complex two dimensional model we use a Krylov enhanced subspace parareal version which accelerates the convergence of the parareal and brings down the number of iterations. In the end, the ways to time-parallelise the adjoint version is also discussed as a further avenue for research.

Résumé

L'assimilation variationnelle de données (4DVAR), basée sur des algorithmes d'optimisation, est utilisée par les principales institutions météorologiques pour initialiser les modèles climatiques numériques. La condition initiale optimale est trouvée en minimisant une fonction de coût qui prend en compte les écarts entre la trajectoire du modèle et les observations du système sur une période donnée. Dans sa formulation incrémentale, l'intégration de la version directe et adjointe du modèle original est nécessaire pour calculer le gradient. Un problème courant dans l'identification de la condition initiale est la très grande taille de la variable d'état (10^9), ce qui rend très couteuse la tâche de minimisation. De plus, la technique 4DVAR est un algorithme intrinsèquement séquentiel et, pour l'utiliser dans des architectures parallèles, les modèles sont généralement parallélisés uniquement dans la dimension spatiale. ceci limite l'accélération (scalabilité) possible une fois que la saturation spatiale est atteinte et ainsi le nombre maximal de cœurs de calcul pouvant être utilisés est également restreint. L'objectif de cette thèse est d'introduire une parallélisation supplémentaire de la dimension temporelle dans le cadre de l'assimilation de données en utilisant la méthode Parareal. Notre approche est utilisée ici pour l'intégration directe. Nous utilisons une version modifiée de la méthode du gradient conjugué inexact où les multiplications matrice-vecteur sont effectuées par l'algorithme parareal et ne sont donc pas exactes. Les conditions de convergence associées du gradient conjugué inexact nous permettent d'utiliser l'algorihtme Parareal de manière adaptative en régulant les erreurs dans le produit matrice-vecteur et en obtenant les mêmes niveaux de précision qu'avec la méthode du gradient conjugué avec gradient exact. Pour garantir la faisabilité et une mise en œuvre pratique, les normes intervenant dans la méthode du gradient conjugé inexact sont remplacées par des approximations facilement calculables. Les résultats sont démontrés en considérant un modèle en eau peu profonde en dimension 1 et 2. Ils sont présentés en termes de précision (en comparaison avec la méthode exacte du gradient conjugué) et du nombre d'itérations requises par l'algorithme Parareal. Pour le modèle en dimension 2, plus complexe, nous utilisons la technique de sous-espaces de Krylov afin d'accélèrer la convergence du parareal et réduire le nombre d'itérations. Enfin, les moyens de paralléliser temporellement la version adjointe sont aussi discutés comme une voie de recherche supplémentaire.

Introduction

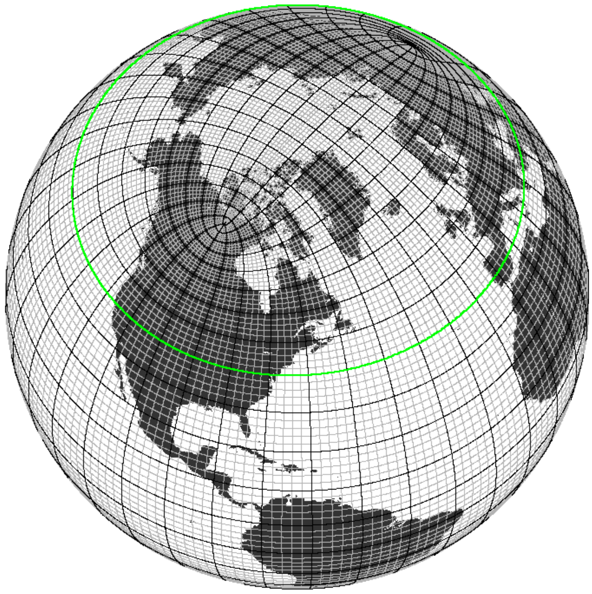
The advancements in modern computational architectures have enabled us to perform extremely large sized computational simulations such as those related to climate modelling. However, the evolution of these architectures must be accompanied by a consideration of the used mathematical algorithms in order to optimise their potential.

Ocean modelling is important from the view point of climate studies as the ocean circulation has a strong influence on the weather/climate patterns. As explained in [Griffies, 2018] the ocean provides a large reservoir for heat and other constituents of the earth's climate system and through its buffering abilities and slow time scales, the ocean represents the flywheel of the earth's climate system.

An example of where ocean models are used, coupled with the atmosphere, is in the seasonal climate forecast. The associated simulations lie at an interface between weather prediction and climate forecasting as described in [Palmer and Anderson, 1994]. On one hand it is an initial value problem where most of the predictability comes from the initial state. On the other hand apart from having a longer period of integration it is also an atmospheric-oceanic coupled model so as to make reliable seasonal predictions. On the seasonal timescale the atmospheric initial conditions start to get less important and instead the low frequency forcing from the ocean starts to provide dominant contributions to the weather predictability [Harris, 2018].

The global seasonal forecasts made by the European Centre for Medium-Range Weather Forecasts (ECMWF) uses the Integrated Forecasting System (IFS) for the atmospheric model coupled to the Nucleus for European Modelling of the Ocean (NEMO) [Madec et al., 2017] for the ocean model. A global simulation of NEMO using the ORCA-R025 which is a global ocean configuration with a tripolar grid (Fig. I) uses the pre-defined horizontal grid resolution of $1/4°$ at the equator leading to the horizontal dimensions of 1442×1021 grid points (along with

75 vertical levels). Several leading meteorological institutes like Météo France, the



Source: http:geomar.de/en/ocean-models/

Figure I: The ORCA tripolar grid

ECMWF, the Met Office, use variational data assimilation algorithms like the 3D or 4D-Var to obtain optimal initial states for making short term and medium range weather forecasts. Data assimilation is essentially an initial condition control problem which combines all the information from the model trajectory, the observations and some error statistics to give an optimal initial state of the atmosphere/ocean. The observations in data assimilation are accessible through two major sources: in-situ measurements and satellite data.

The evolving nature of the numerical models and the observations in terms of quality and size pose a challenge to the data assimilation systems with the demands to make faster and accurate predictions. The state vector in ocean data assimilation is usually of the order $\mathcal{O}(10^6 - 10^9)$ and the observation vector of the order $\mathcal{O}(10^6)$. This increases the overall computational cost and the only way to deal with this challenge is to improve the use of the massively parallel supercomputers. The incredibly rapid growth in the processing power of the high performance computers is associated to a very large increase of the number of computational units (CPU / GPU cores). This can be seen by looking at the list of the top 5 most powerful supercomputers in the world in Fig. II with the number of cores exceeding the million mark.

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	**Frontier** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE DOE/SC/Oak Ridge National Laboratory United States	8,699,904	1,194.00	1,679.82	22,703
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
3	**LUMI** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,220,288	309.10	428.70	6,016
4	**Leonardo** - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, Atos EuroHPC/CINECA Italy	1,824,768	238.70	304.47	7,404
5	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148.60	200.79	10,096

Source: `http://top500.org/`

Figure II: List of the 5 most powerful commercially available computer systems as of June 2023

In the past, traditional ways of harnessing the power of supercomputers had largely focused on taking benefit from an increase in the performance of the individual units (measured in terms of floating points operations per second) rather than from an increase in the number of these units. The recent changes in the evolution of the architectures have emphasised the need to refine mathematical algorithms so that they can achieve a much higher degree of parallelism.

As an example we again consider the ORCA-R025 ocean model grid with a horizontal domain of 1442×1021 grid points and 75 vertical levels. The usual way to parallelize ocean models is to apply a spatial domain decomposition [Toselli and Widlund, 2004, Dolean et al., 2015] method and to assign the resulting domains to individual cores. For numerical reasons, this spatial decomposition is most of the time only performed in the horizontal dimension (see Fig. III). Most ocean models utilise minimum domain sizes of 20×20 grid points, beyond which the

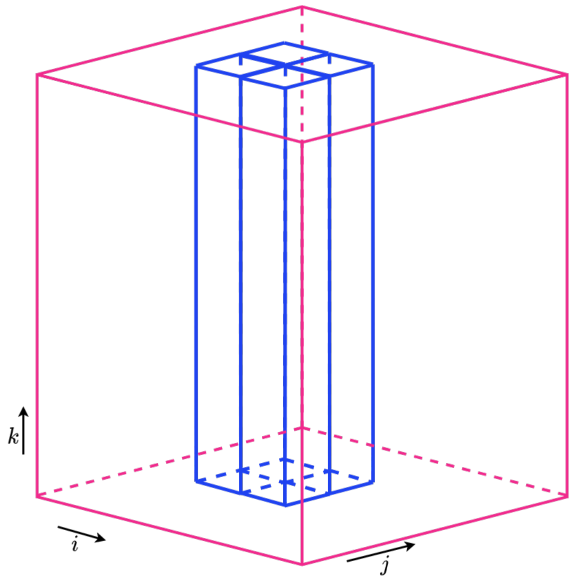


Figure III: Horizontal decomposition of an ocean domain (in red). Subdomains (in blue) are associated to one computational core.

simulation time increases, with the communication time becoming predominant over the computation time within each subdomain. For the ORCA-R025 grid, we would end up using approximately a maximum of 3672 cores. This means that even if the simulation could have access to more cores, after some point the spatial parallelisation would reach saturation and we would not be able to use them effectively.

In large time-dependent problems such as the one described in the above example, it is tentative to use the time dimension as an additional direction of parallelism. This is one of the main reasons behind turning our attention to the parallel-in-time (PinT) methods [Gander, 2015] in addition to spatial domain decomposition. This observation applies to direct modelling, but also, and this is what will be of interest in this thesis work, to data assimilation algorithms. If we take a look at the incremental 4D-Var (see Chapter 1, Sec. 1.3) which is a practical method of running a 4D-Var data assimilation scheme, the computational costs involved in its full single iteration is largely down to:

- One forward integration of the non-linear model in the outer loop
- Forward and backward integrations of the tangent linear model and its corresponding adjoint model times the number of minimisation iterations in

the inner loop

The main objective in this PhD work is to study the exploitation of time parallelism by combining a parallel-in-time method with the incremental 4D-Var. The manuscript is divided in two parts.

The first part consists of two chapters which are dedicated to the fundamentals required to understand the concepts on data assimilation and parallel-in-time methods. A concise review of the previous major studies and results in the respective domains is introduced gradually to make the reader aware about the current state of the art and also about addressing the shortcomings where there is a further scope of research.

In Chapter 1 we briefly describe what data assimilation is from the point of view of the variational approach. The concept of the variational data assimilation is explained with the help of the error statistics leading to the problem of the minimisation of a cost function. Depending on how the cost function is defined, different formulations of the variational data assimilation are explained such as the 3D-Var, the 4D-Var, the incremental 4D-Var and the weak constraint 4D-Var. Our main focus of this manuscript will be on the incremental 4D-Var algorithm. We will also talk about the adjoint method which is one of the most powerful techniques for obtaining the cost function gradient and thus allowing the application of efficient minimisation procedures. An overview of the minimisation methods used in data assimilation is given in the last section and in particular the conjugate gradient method.

Chapter 2 begins with an introductory section on the idea of time parallelisation and a historical account of the parallel-in-time or PinT algorithms. A broad classification of the PinT methods is stated from which we shift our attention to the parallel multiple shooting methods. Our focal point of discussion will be on the Parareal algorithm. The Parareal's convergence properties, speedup and efficiency are discussed. We end this chapter by citing some of the common issues that Parareal faces related to specific problems and the modifications which have been proposed in the literature to accelerate its performance.

The second part of the manuscript is devoted to our contribution towards the theoretical and numerical aspects of the coupling of the two iterative incremental 4D-Var and Parareal methods. Chapter 3 is dedicated to the methodology of formulating the inner loop problem of the incremental 4D-Var when the Parareal operator replaces the sequential forward integration. We begin with a short literature review on the previously made attempts on exploiting the time parallelisation in 4D-Var. We then define our Parareal operator, set up the data assimilation problem and show that the coupling leads to solving an approximate linear system where the error in the matrix depends on the Parareal's accuracy. To solve this linear system, we discuss the inexact conjugate gradient method which is a version of the conjugate gradient method with the presence of inaccurate matrix-vector products. In the end we make use of a modified version of the inexact conjugate gradient which allows to obtain an adaptive tolerance criterion for the Parareal method.

Chapter 4 deals with the applications part of the methodology discussed in Chapter 3. This is illustrated with the help of the numerical experiments on the one and the two dimensional linearised shallow water equations which are widely used for modelling important geophysical phenomena. Throughout the applications, we elaborate on several crucial aspects regarding the Parareal algorithm itself (choice of the coarse solver, use of the Krylov subspace enhanced acceleration, reducing the cost of the fine solver ...), as well as the adaptation of the implementation of the inexact conjugate gradient algorithm to our specific problem.

We conclude the manuscript by giving some perspectives on our work in order to envision more realistic applications.

Part I

State of the art: Data assimilation and parallel-in-time (PinT) methods

Chapter 1

Data Assimilation

Contents

The concept of data assimilation can be explained as the science of combining all the available information in terms of observations and the time-evolving numerical model and associated error statistics with the aim of improving the model output. Or as put plainly by Daley [Daley, 1997] “data assimilation is a discipline which naturally integrates theory (via models) with sampled reality (via instruments)”.

In Numerical Weather Prediction (NWP) which is an initial value problem, data assimilation is used as the technique of retrieving the *optimal initial condi-*

tion approaching the true state of the atmosphere/ocean. The numerical model incorporates the discretised form of all the physical laws governing the dynamics of the fluid such as the laws of conservation of mass and momentum. The observations provide real time information/snapshots about the atmosphere/ocean and are accessible through satellite data, in-situ measurements or other sources. There is a vast amount of literature explaining the fundamentals and the state of the art of data assimilation. For introductory texts, one can look at [Daley, 1993, Kalnay, 2003, Asch et al., 2016]. An overview of the development of data assimilation can be found in [Rabier, 2005, Navon, 2009, Bannister, 2017]. For more details see [Ghil and Malanotte-Rizzoli, 1991, Daley, 1997, Talagrand, 1997, Bouttier and Courtier, 2002, Carrassi et al., 2018].

Data assimilation is studied following two different approaches: the ensemble and the variational approach. The ensemble methods on one hand make use of an ensemble of the previous forecasts which contain valuable flow-dependent information about the background error statistics [Bannister, 2017]. The predicted ensemble of forecasts will contain a similar a priori error statistics and will be used for the future forecasts. The variational methods on the other hand use iterative minimisation algorithms on a cost function containing the discrepancies between the observations and the corresponding values from the model trajectory. Our topic of concern will be specifically on the variational approach and in this chapter we will talk about the most common variational data assimilation algorithms such as the 3D-Var, 4D-Var and the incremental 4D-Var used operationally in various meteorological institutes.

We start by first defining the notation for quantities involving the error statistics to get the cost function and later in the subsequent sections we describe the various methods. To maintain consistency in the remainder of the text we use the unified notation from [Ide et al., 1997].

1.1 Error Statistics

We begin by defining some important quantities. To describe the state of the atmosphere at some instant, we use the vector $\mathbf{x}$ as a collection of all the parameter values on which it depends. The aim is to determine numerically the best approximation for the state vector $\mathbf{x}$ which we call the *analysis* $\mathbf{x}^a$. To distinguish the analysis from the reality, we denote $\mathbf{x}^t$ as the *true state* vector which is the closest representation of the atmosphere at the time of analysis. The analysis ideally depicts the accurate image of the true state of the atmosphere. It is the starting point of a new forecast for the next assimilation cycle which gives us subsequent analysis [Schlatter, 2000].

We define the *background* state $\mathbf{x}^b$ as some a priori information for the data assimilation system and is generally obtained from a previous forecast or analysis for the same system. We suppose that the background is an approximation to the true state and has some errors, say $\boldsymbol{\epsilon}^b$. It can be written as

$$\mathbf{x}^b = \mathbf{x}^t + \boldsymbol{\epsilon}^b \tag{1.1}$$

All the observation data available at a given time can be collected together in a single vector and it is represented by $\mathbf{y}^o$. A very large volume of data ingested by the NWP data assimilation systems comes from the satellite data which is not in the same space as the model state is [Thépaut, 2003]. The satellite data we have is in the form of radiances reflected by the earth's surface which is captured by the satellite sensors. In order to quantify the differences between what we observe and what we get from the model we use an *observation operator* H which maps the state space to the observation space.

As an example, let us consider a typical radiative model which uses an atmospheric radiative transfer equation [Saunders et al., 1999]. The top of the atmosphere upwelling radiance $L(\nu, \theta)$ with frequency ν and viewing angle θ (neglecting scattering affects) is given as

$$L(\nu, \theta) = (1 - N_c)L^{\mathrm{Clr}}(\nu, \theta) + N_c\, L^{\mathrm{Cld}}(\nu, \theta) \tag{1.2}$$

where $L^{\mathrm{Clr}}(\nu, \theta)$ and $L^{\mathrm{Cld}}(\nu, \theta)$ are the clear sky and fully cloudy top of atmosphere upwelling radiances and N_c is the fractional cloud cover. For the sake of completeness see [Eyre, 1991, Matricardi et al., 2004]. In this case our observation operator $H_{\nu,\theta}(\mathbf{x})$ would take the geophysical quantities of the model and give us the radiances following the radiative transfer equation (1.2).

The observation data $\mathbf{y}^o$ itself is not entirely reliable and suffers from the so called representation and measurement errors. In brief, the representation errors encompass the error due to mismatch in the scales between the observations and the model fields, error in the observation operator itself or due to the pre-processing of the observations [Janjić et al., 2018]. The measurement errors can be attributed to the problem with the machines/sources providing the observations. One example can be some defect with the sensor of a satellite sending remotely-sensed images. Together we denote the representation and measurement error as $\boldsymbol{\epsilon}^o$ and the relation between the observation and model state can be written as

$$\mathbf{y}^o = H(\mathbf{x}^t) + \boldsymbol{\epsilon}^o \tag{1.3}$$

For a basic understanding we can look at Fig. 1.1 which describes a simple data assimilation schematic. The starting background term $\mathbf{x}^b$ is assimilated with the observations $\mathbf{y}^o$ along with the corresponding errors $\boldsymbol{\epsilon}^b$ and $\boldsymbol{\epsilon}^o$ to give an analysis $\mathbf{x}^a$ or an improved forecast of the background. We now derive the variational

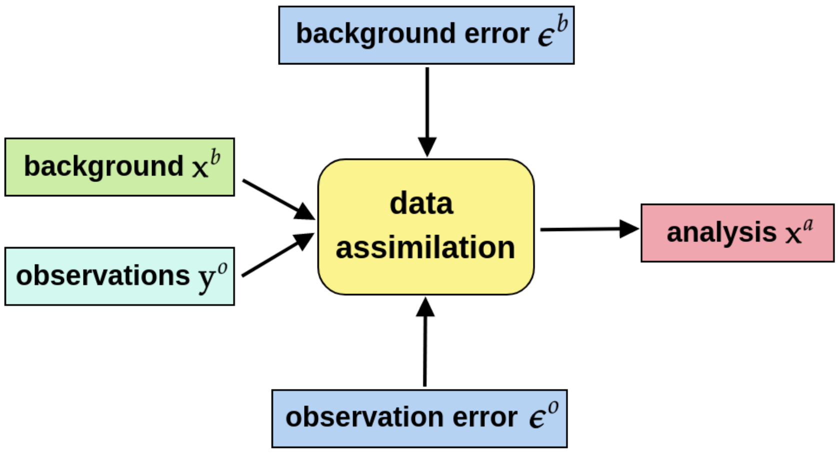


Figure 1.1: A simple schematic of data assimilation

equation for the *best analysis* using the ideas of Bayesian statistics and probability distribution functions (pdf) following [Lorenc, 1986]. We are interested in finding the conditional probability of being at the analysis state given a set of observations. In other words we want to maximise $p(\mathbf{x}|\mathbf{y}^o)$. We recall the Bayes' theorem which states that the posterior probability of any event occurring is proportional to the prior probability multiplied by the likelihood probability.

$$p(\mathbf{x}|\,\mathbf{y}^o) \propto p(\mathbf{y}^o|\,\mathbf{x})p(\mathbf{x}) \tag{1.4}$$

where $p(\mathbf{y}^o|\,\mathbf{x})$ is the likelihood pdf of the observations given model state and $p(\mathbf{x})$ is the prior pdf of the model state which is at best represented by some background information. Assuming that the errors follow a Gaussian distribution, we have

$$\boldsymbol{\epsilon}^b \sim \mathcal{N}(0, \mathbf{B}) \quad \text{and} \quad \boldsymbol{\epsilon}^o \sim \mathcal{N}(0, \mathbf{R}) \tag{1.5}$$

where $\mathcal{N}(0, \mathbf{Q})$ represents a normal distribution centered at 0 with covariance matrix $\mathbf{Q}$. In our multi-dimensional case, $\mathbf{B}$ and $\mathbf{R}$ are the background and observation covariance matrices respectively. Thus,

$$\begin{aligned} \mathbf{B} &= \left\langle (\boldsymbol{\epsilon}^b)^T (\boldsymbol{\epsilon}^b) \right\rangle \\ \mathbf{R} &= \left\langle (\boldsymbol{\epsilon}^o)^T (\boldsymbol{\epsilon}^o) \right\rangle \end{aligned} \tag{1.6}$$

and the prior and the likelihood pdfs can be expressed as

$$\begin{aligned} p(\mathbf{y}^o|\, \mathbf{x}) &\propto \exp\left\{-\frac{1}{2}(H(\mathbf{x}) - \mathbf{y}^o)^T\, \mathbf{R}^{-1}(H(\mathbf{x}) - \mathbf{y}^o)\right\} \\ p(\mathbf{x}) &\propto \exp\left\{-\frac{1}{2}(\mathbf{x} - \mathbf{x}^b)^T\, \mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}^b)\right\} \end{aligned} \tag{1.7}$$

Using the above relations in (1.4) we have

$$\begin{aligned} p(\mathbf{x}|\, \mathbf{y}^o) \propto -\exp\Bigg\{&\frac{1}{2}(\mathbf{x} - \mathbf{x}^b)^T\, \mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}^b) \\ &+ \frac{1}{2}(H(\mathbf{x}) - \mathbf{y}^o)^T\, \mathbf{R}^{-1}(H(\mathbf{x}) - \mathbf{y}^o)\Bigg\} \end{aligned} \tag{1.8}$$

In order to maximise the posterior pdf $p(\mathbf{x}|\, \mathbf{y}^o)$, we take the negative logarithm of the above relation (1.8) to obtain a cost function of the form

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^b)^T\, \mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}^b) + \frac{1}{2}(H(\mathbf{x}) - \mathbf{y}^o)^T\, \mathbf{R}^{-1}(H(\mathbf{x}) - \mathbf{y}^o) \tag{1.9}$$

This turns out to be the cost function for one of the most basic variational data assimilation algorithms called the 3D-Var which we discuss next.

1.2 3D-Var and 4D-Var

The three dimensional variational data assimilation or 3D-Var [Lorenc et al., 2000, Courtier et al., 1998] aims to determine the best state of the atmosphere/ocean at a particular time by considering just the spatial variables and not the dynamical model evolution. In other words, 3D-Var minimises the cost function without including the time dimension as a part of the control variable i.e. it is a stationary assimilation method. During the whole assimilation period window the observations which are distributed in time are assumed to be measured at a particular time (called the analysis time). All the observations are treated as if they were observed at the analysis time while calculating the cost function. Suppose

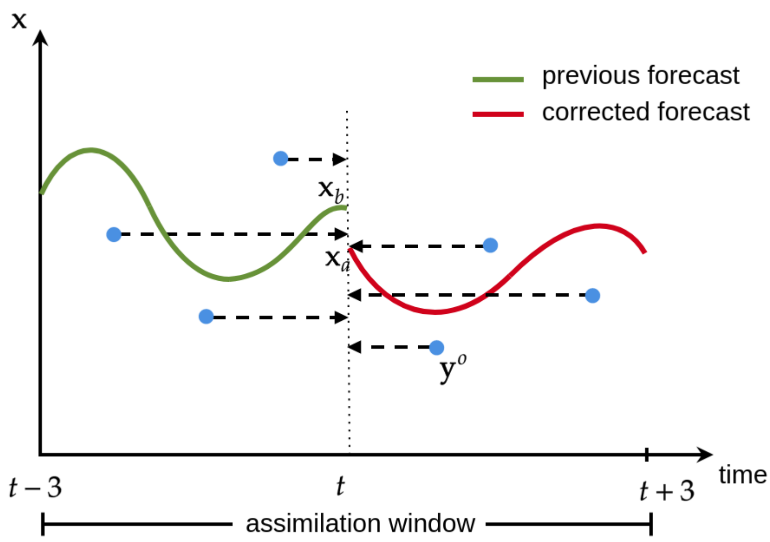


Figure 1.2: Diagram depicting the 3D-Var algorithm

we have a model state $\mathbf{x} \in \mathbb{R}^n$ (obtained by discretising the partial differential equation using a suitable numerical method like finite differences, finite element, or spectral methods). For some a priori information $\mathbf{x}^b \in \mathbb{R}^n$, a set of observations $\mathbf{y}^o \in \mathbb{R}^m$ and the observation operator $H : \mathbb{R}^n \to \mathbb{R}^m$, the 3D-Var cost function is given as

$$J(\mathbf{x}) = \frac{1}{2}\,(\mathbf{x}-\mathbf{x}^b)^T\,\mathbf{B}^{-1}\,(\mathbf{x}-\mathbf{x}^b) \;+\; \frac{1}{2}\,(H(\mathbf{x})-\mathbf{y}^o)^T\,\mathbf{R}^{-1}\,(H(\mathbf{x})-\mathbf{y}^o) \qquad (1.10)$$

The errors in the background and the observations are characterised by the corresponding covariance matrices $\mathbf{B}$ and $\mathbf{R}$. The covariances are weighted inversely which means that more weight will be given to the more reliable information and vice-versa.

The cost function J is minimised by an appropriate gradient based minimisation algorithm. In that case the 3D-Var cost function gradient is given as

$$\nabla J(\mathbf{x}) = \mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b) + \mathbf{H}^T\mathbf{R}^{-1}(H(\mathbf{x})-\mathbf{y}) \qquad (1.11)$$

where $\mathbf{H}$ is the linearised observation operator. For example, the 3D-Var implementation by the European Centre for Medium-Range Weather Forecasts (ECMWF) [Courtier et al., 1998] considered a 6-hour time window from $t-3$ to $t+3$ and the observations were assimilated at the central synoptic analysis time $t+0$ (Fig. 1.2). For a detailed information about 3D-Var, see [Rabier et al., 1998, Andersson et al., 1998].

We saw that 3D-Var lacks the knowledge about a time evolving model and a natural extension to improve it can be done by adding the fourth dimension (time) to the 3D-Var cost function resulting in another variational data assimilation method known as the 4D-Var.

The four-dimensional variational data assimilation or 4D-Var [Thepaut and Courtier, 1991, Thépaut et al., 1993, Rabier et al., 2000] minimises the cost function based on the discrepancies between the model trajectory and the observations and between the obtained analysis and the previous known background state (Fig. 1.3). For the model state vector $\mathbf{x}$ we consider the following discrete non-linear dynamical model

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x}(t_0) \\ \mathbf{x}_i &= M_i(\mathbf{x}_{i-1}) \qquad i = 1, 2, \ldots, N \end{aligned} \tag{1.12}$$

where $M_i : \mathbb{R}^n \to \mathbb{R}^n$ is the model operator describing the state evolution from time t_{i-1} to t_i and N is the total number of time windows. Given we have access to some prior estimate/background state $\mathbf{x}^b \in \mathbb{R}^n$ and a set of observations $\mathbf{y}_i^o \in \mathbb{R}^m$ at time t_i, the 4D-Var cost function can be written as

$$\begin{aligned} J(\mathbf{x}_0) = {} & \frac{1}{2} (\mathbf{x}_0 - \mathbf{x}^b)^T \, \mathbf{B}^{-1} (\mathbf{x}_0 - \mathbf{x}^b) \\ & + \frac{1}{2} \sum_{i=0}^{N} (H_i(\mathbf{x}_i) - \mathbf{y}_i^o)^T \, \mathbf{R}_i^{-1} (H_i(\mathbf{x}_i) - \mathbf{y}_i^o) \end{aligned} \tag{1.13}$$

Here $H_i : \mathbb{R}^n \to \mathbb{R}^m$ maps the evaluated model state $\mathbf{x}_i$ to the observation $\mathbf{y}_i^o$ at the correct time t_i.

Remark 1.1. *In meteorological data assimilation the role of the background term can be explained by the fact that the observations are often scarcely and irregularly distributed in time and space. This could be because the satellite covers/observes only a part of the earth (spatial domain) and so only some areas are observed during a given assimilation window. Here the covariance matrix* $\boldsymbol{B}$ *helps to extrapolate* *information from the observed regions to the unobserved areas and therefore maintaining the quality of the analysis in the areas with few observations available. In mathematical terms the problem would be under-determined in those areas in the absence of a background term [Trémolet, 2006].*

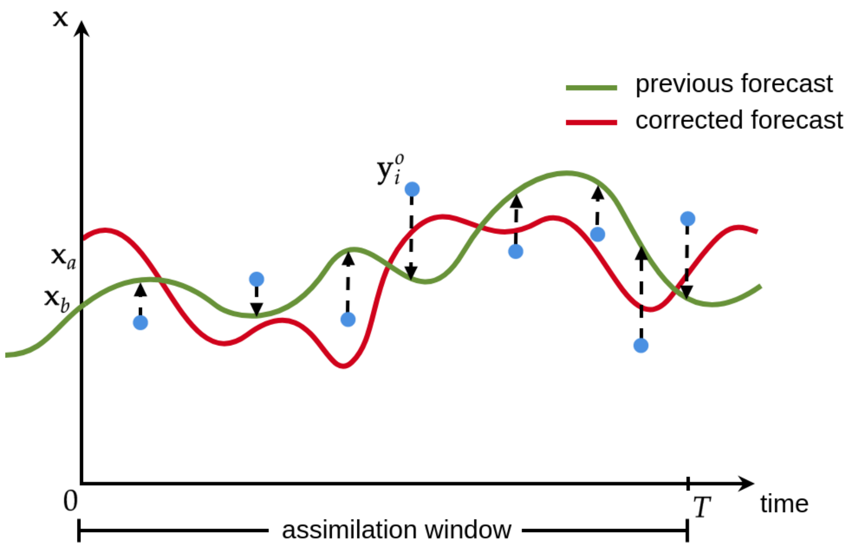


Figure 1.3: Diagram depicting the 4D-Var algorithm

Remark 1.2. *Since M and H are non-linear, from (1.13) we can say that 4D-Var is essentially a weighted non-linear least-squares fitting problem constrained by the model evolution equation (1.12). This goes by the sense that we are assuming that the model is exact or the model has no errors.*

In terms of operational implementation, it is assumed that the model error in (1.12) is small enough to be ignored and in this sense 4D-Var is sometimes also called as the *strong constraint 4D-Var formulation* [Sasaki, 1970]. Due to this assumption the whole model trajectory can be traced just from its initial condition $\mathbf{x}_0$ leading to the reduction of the control variable to a three-dimensional state.

$$\mathbf{x}_i = M_i(\mathbf{x}_{i-1}) = M_i M_{i-1} \dots M_1(\mathbf{x}_0) \tag{1.14}$$

Thus (1.13) can be re-written as

$$\begin{aligned} J(\mathbf{x}_0) &= \frac{1}{2}\left(\mathbf{x}_0 - \mathbf{x}^b\right)^T \mathbf{B}^{-1}\left(\mathbf{x}_0 - \mathbf{x}^b\right) \\ &+ \frac{1}{2}\sum_{i=0}^{N}\left\{H_i\left[M_i \dots M_1(\mathbf{x}_0)\right] - \mathbf{y}_i^o\right\}^T \mathbf{R}_i^{-1}\left\{H_i\left[M_i \dots M_1(\mathbf{x}_0)\right] - \mathbf{y}_i^o\right\} \end{aligned} \tag{1.15}$$

The minimisation of the cost function (1.15) now represents an unconstrained weighted non-linear least-squares problem with the initial states being the control variables.

Lorenc and Rawlins [Lorenc and Rawlins, 2005] explained that the difference between 3D-Var and 4D-Var is how the time dimension is treated and taken into account while assimilating the observations. In terms of producing an accurate analysis, 4D-Var has an edge over 3D-Var as it allows the comparison of each observation in full fields at the proper times and it lets the time-evolved covariances improve the quality of the analysis for longer forecasts. For more details about the operational implementation of 4D-Var in the ECMWF see [Mahfouf and Rabier, 2000, Klinker et al., 2000] and in the UK Met Office [Rawlins et al., 2007].

1.3 Incremental 4D-Var

1.3.1 Gauss-Newton Method

Solving the 4D-Var problem can prove to be very challenging because of the extremely large size of the control variable $\mathbf{x}$ ($\sim \mathcal{O}(10^7 - 10^9)$) and the immense computational resources required to run it operationally. One generally makes use of the Gauss-Newton algorithm which can be described as a modified Newton's method for solving a general non-linear least-squares problem, see [Björck, 1996, Nocedal and Wright, 1999]. We show that how this method can be adapted for solving the 4D-Var problem as done in [Gratton et al., 2007]. Notice that the 4D-Var cost function can be written as

$$J(\mathbf{x}_0) = \frac{1}{2}\|f(\mathbf{x}_0)\|_2^2 = \frac{1}{2} f(\mathbf{x}_0)^T f(\mathbf{x}_0) \tag{1.16}$$

where $f : \mathbb{R}^n \to \mathbb{R}^{N+n}$ is a non-linear twice continuously Fréchet differentiable function and

$$f(\mathbf{x}_0) = \begin{pmatrix} \mathbf{B}^{-1/2}(\mathbf{x}_0 - \mathbf{x}^b) \\ \mathbf{R}_1^{-1/2}(H_1(\mathbf{x}_1) - \mathbf{y}_1^o) \\ \vdots \\ \mathbf{R}_N^{-1/2}(H_N(\mathbf{x}_N) - \mathbf{y}_N^o) \end{pmatrix} \tag{1.17}$$

Let $\widehat{\mathbf{J}}$ be the Jacobian of f. Then the gradient and Hessian of J are written as

$$\begin{aligned}\nabla J(\mathbf{x}_0) &= \widehat{\mathbf{J}}(\mathbf{x}_0)^T f(\mathbf{x}_0)\\ \nabla^2 J(\mathbf{x}_0) &= \widehat{\mathbf{J}}(\mathbf{x}_0)^T \widehat{\mathbf{J}}(\mathbf{x}_0) + \sum_{i=1}^{N+n} f_i(\mathbf{x}_0)\nabla^2 f_i(\mathbf{x}_0)\end{aligned} \tag{1.18}$$

with f_i being the ith component of f and

$$\widehat{\mathbf{J}} = \begin{pmatrix} \mathbf{B}^{-1/2} \\ \mathbf{R}_1^{-1/2}\mathbf{H}_1\mathbf{M}_1 \\ \vdots \\ \mathbf{R}_N^{-1/2}\mathbf{H}_N\mathbf{M}_N \end{pmatrix}. \tag{1.19}$$

$\mathbf{H} = \dfrac{\partial H}{\partial \mathbf{x}}\Big|_{\mathbf{x}^{(l)}}$ and $\mathbf{M} = \dfrac{\partial M}{\partial \mathbf{x}}\Big|_{\mathbf{x}^{(l)}}$ are the linearised observation operator and the linearised model around the linearisation state $\mathbf{x}^{(l)}$ which we will discuss in detail in the next section.

For large scale problems such as 4D-Var, the individual second order terms $f_i \nabla^2 f_i$ of the Hessian $\nabla^2 J$ are impractical to calculate and that is where the Newton's method is approximated by neglecting those second order terms and giving the Gauss-Newton method. Therefore,

$$\nabla^2 J(\mathbf{x}_0) \approx \widehat{\mathbf{J}}(\mathbf{x}_0)^T \widehat{\mathbf{J}}(\mathbf{x}_0) \tag{1.20}$$

Algorithm 1 describes the steps for executing the Gauss-Newton algorithm. Note

Algorithm 1 GAUSS-NEWTON ALGORITHM

1. Choose an initial guess $\mathbf{x}_0 \in \mathbb{R}^n$.
2. Repeat until convergence:
   (a) Solve $\widehat{\mathbf{J}}(\mathbf{x}^{(l)})^T\widehat{\mathbf{J}}(\mathbf{x}^{(l)})\,\mathbf{s}^{(l)} = -\widehat{\mathbf{J}}(\mathbf{x}^{(l)})^T f(\mathbf{x}^{(l)})$
   (b) Set $\mathbf{x}^{(l+1)} = \mathbf{x}^{(l)} + \mathbf{s}^{(l)}$
   (c) Linearise $\mathbf{H}$ and $\mathbf{M}$ around $\mathbf{x}^{(l+1)}$ for the computation of $\widehat{\mathbf{J}}(\mathbf{x}^{(l+1)})$

that at each iteration of the step 2.(a) corresponds to solving the linearised least squares problem [Gratton et al., 2007]

$$\min_{\mathbf{s}} \frac{1}{2}\left\|\widehat{\mathbf{J}}(\mathbf{x}^{(l)})\,\mathbf{s} + f(\mathbf{x}^{(l)})\right\|_2^2 \tag{1.21}$$

In the next section we are going to talk about the incremental 4D-Var in great length and we are going to see how the Gauss-Newton algorithm is intimately related to it.

1.3.2 Implementation issues with 4D-Var

In the last section we said that implementing the full 4D-Var algorithm requires an immense amount of computational resources. One reason is that the model state usually has the number of degrees of freedom. For instance, just for a simple latitude-longitude model with a resolution of 1° and 20 vertical levels we would have $360 \times 180 \times 20 = 1.3 \times 10^6$ grid points [Kalnay, 2003]. If at each grid point we would like to store the values of the prognostic variables (two horizontal velocities, temperature, moisture) and the surface pressure for each column, we would have around 5 million variables to provide to the initial state. Another reason is the

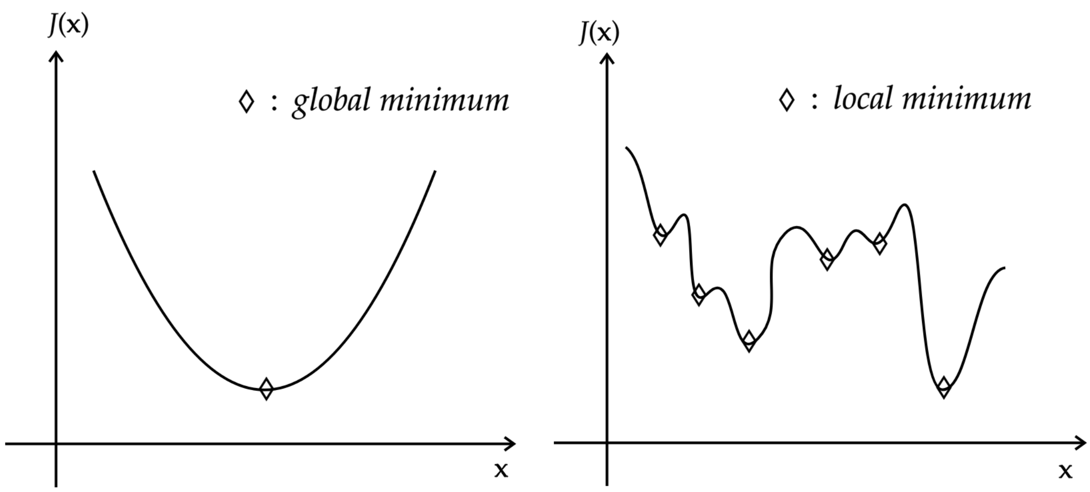


Figure 1.4: On the left the function is quadratic and has a unique global minimum; on the right the function is highly non-linear has many local minima

presence of the non-linear operators M and H which makes the cost function highly non-linear. Trying to directly minimise a fully non-linear cost-function is an arduous task where one can be in a situation of being stuck in one of the many local minima and never reaching the true optimum (Fig. 1.4). In order to use 4D-Var at an operational level, instead of minimising the full non-linear cost function at once, the problem is reduced to a series of successive minimisations of much simpler quadratic cost functions. This formulation is known as the *Incremental 4D-Var* [Courtier et al., 1994].

1.3.3 Incremental 4D-Var formulation

The incremental 4D-Var formulation is obtained by linearising the model and observation operators around the model state. For a sequence of the linearisation states $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots \mathbf{x}^{(l)}$ with corresponding increments $\delta\mathbf{x}^{(0)}, \delta\mathbf{x}^{(1)}, \ldots \delta\mathbf{x}^{(l)}$ the following tangent linear hypothesis (a first order Taylor's expansion) is made

$$\begin{aligned} M(\mathbf{x}^{(l)} + \delta\mathbf{x}^{(l)}) &\approx M(\mathbf{x}^{(l)}) + \mathbf{M}\,\delta\mathbf{x}^{(l)} \\ H(\mathbf{x}^{(l)} + \delta\mathbf{x}^{(l)}) &\approx H(\mathbf{x}^{(l)}) + \mathbf{H}\,\delta\mathbf{x}^{(l)} \end{aligned} \tag{1.22}$$

where $\mathbf{M}$ and $\mathbf{H}$ are the linearisation of the model and observation operators respectively around the linearisation state $\mathbf{x}^{(l)}$; that is,

$$\mathbf{M} = \frac{\partial M}{\partial \mathbf{x}}\Big|_{\mathbf{x}^{(l)}} \qquad \mathbf{H} = \frac{\partial H}{\partial \mathbf{x}}\Big|_{M(\mathbf{x}^{(l)})} \tag{1.23}$$

The control variable now changes from the initial model state $\mathbf{x}_0$ to the initial increment $\delta\mathbf{x}_0$ of the model state. Using the above assumption we write our cost function as

$$\begin{aligned} J(\delta\mathbf{x}_0) = {} & \frac{1}{2}\left\{\mathbf{x}_0 - (\mathbf{x}^b - \delta\mathbf{x}_0)\right\}^T \mathbf{B}^{-1} \left\{\mathbf{x}_0 - (\mathbf{x}^b - \delta\mathbf{x}_0)\right\} \\ & + \frac{1}{2}\sum_{i=0}^{N} \left\{H_i(\mathbf{x}_i + \delta\mathbf{x}_i) - \mathbf{y}_i^o\right\}^T \mathbf{R}_i^{-1} \left\{H_i(\mathbf{x}_i + \delta\mathbf{x}_i) - \mathbf{y}_i^o\right\} \end{aligned} \tag{1.24}$$

and which can be written in more simplified form after applying the tangent linear approximation

$$\begin{aligned} J(\delta\mathbf{x}_0) = {} & \frac{1}{2}\left\{\delta\mathbf{x}_0 - (\mathbf{x}^b - \mathbf{x}_0)\right\}^T \mathbf{B}^{-1}\left\{\delta\mathbf{x}_0 - (\mathbf{x}^b - \mathbf{x}_0)\right\} \\ & + \frac{1}{2}\sum_{i=0}^{N} (\mathbf{H}_i\delta\mathbf{x}_i - \mathbf{d}_i)^T \mathbf{R}_i^{-1} (\mathbf{H}_i\delta\mathbf{x}_i - \mathbf{d}_i) \end{aligned} \tag{1.25}$$

where $\mathbf{d}_i = H_i(\mathbf{x}_i) - \mathbf{y}_i$ is called the innovation vector or just departures. Clearly one has to run the non-linear model integration and store the trajectory to compute the departures since the non-linear model M is embedded within $\delta\mathbf{x}_i$. The minimisation is generally carried out using low resolution linear models to further cut the computing costs. The increment vector evolves in time through the *tangent linear model* (TLM) which is the obtained by linearising the discrete non-linear model.

$$\delta \mathbf{x}_{i+1} = \frac{\partial M[\mathbf{x}(t_i)]}{\partial \mathbf{x}} \delta \mathbf{x}_i = \mathbf{M}_i \delta \mathbf{x}_i \tag{1.26}$$

There is another way to obtain the discrete model using the *perturbation forecast*

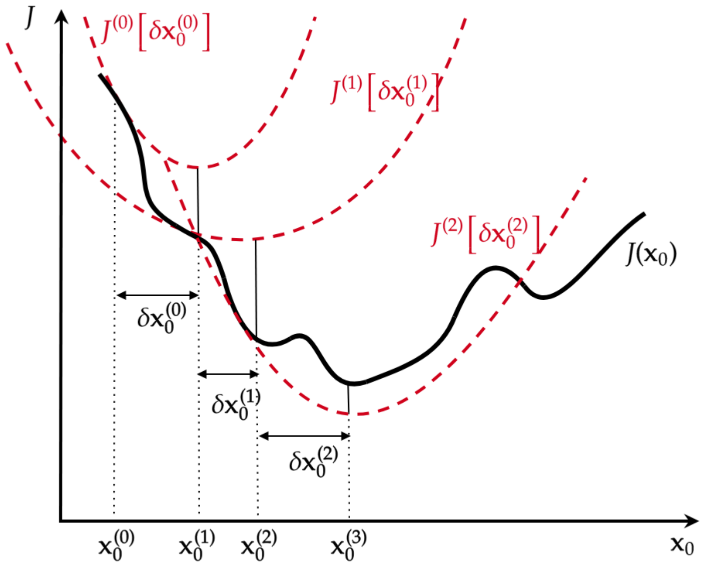


Figure 1.5: Diagram depicting the incremental 4D-Var algorithm

model (PFM) [Lawless et al., 2003] where first the continuous non-linear model is linearised to obtain a continuous linear model and then it is discretised. In a nutshell the incremental 4D-Var involves running a high resolution model, which retains all the physics, to compare the model state trajectory with the observations as part of the evaluation of the cost function and running a low resolution model with simplified physics, to minimise the cost function [Rabier et al., 2000]. An algorithmic implementation of incremental 4D-Var given below in Algorithm 2 is adopted from [Lawless et al., 2005]. For better visualisation, the first few inner and outer loop steps with the linearised cost functions, linearised states and increments are shown in Fig. 1.5.

Algorithm 2 INCREMENTAL 4D VAR

INITIALISATION

- Start with an initial guess $\mathbf{x}_0^{(l)}$. Usually for $l = 0$ we set $\mathbf{x}_0^{(0)} = \mathbf{x}^b$.

OUTER LOOP BEGINS

- Run the non-linear model (1.12) and store the model trajectory $\{\mathbf{x}_i^{(l)}\}$. Calculate the innovation vector $\mathbf{d}_i^{(l)} = H_i(\mathbf{x}_i^{(l)}) - \mathbf{y}_i^o$

- Define the increment vector $\delta\mathbf{x}_0^{(l)} = \mathbf{x}_0^{(l+1)} - \mathbf{x}_0^{(l)}$

  INNER LOOP STARTS

  - Solve the minimisation problem

$$J^{(l)}[\delta\mathbf{x}_0^{(l)}] = \frac{1}{2}(\delta\mathbf{x}_0^{(l)} - (\mathbf{x}^b - \mathbf{x}_0^{(l)}))^T\mathbf{B}^{-1}(\delta\mathbf{x}_0^{(l)} - (\mathbf{x}^b - \mathbf{x}_0^{(l)})) + \frac{1}{2}\sum_{i=0}^{N}(\mathbf{H}_i\delta\mathbf{x}_i^{(l)} - \mathbf{d}_i^{(l)})^T\mathbf{R}_i^{-1}(\mathbf{H}_i\delta\mathbf{x}_i^{(l)} - \mathbf{d}_i^{(l)}) \tag{1.27}$$

    subject to (1.26)

  INNER LOOP ENDS

- Update the guess using

$$\mathbf{x}_0^{(l+1)} = \mathbf{x}_0^{(l)} + \delta\mathbf{x}_0^{(l)} \tag{1.28}$$

- Repeat until a given convergence criterion is satisfied or a fixed number of iterations has been performed.

OUTER LOOP ENDS

Remark 1.3. *The incremental 4D-Var algorithm is equivalent to solving the iterative Gauss-Newton algorithm for non-linear least squares problem as shown in [Lawless et al., 2005].*

Remark 1.4. *An incremental 4D-Var iteration is identical to a Gauss-Newton iteration if an exact TLM is used. Whereas if an approximated TLM or PFM is used, then the incremental 4D-Var iteration can be seen as an inexact Gauss Newton iteration which makes use of an approximate Jacobian [Lawless et al., 2005, Gratton et al., 2007].*

1.3.4 Performance and convergence

As the system evolves to higher and higher resolution, the effects of small scales and the non-linear physics become even more important. In order to properly resolve them, one needs to run more outer loops because that is where the non-linearities are taken into consideration in the assimilating model [Trémolet, 2007].

The inner loop is generally run at a lower resolution with the help of a truncation operator while the inverse of the same operator is used to bring back the low resolution increment to the high resolution for updating the control variable. This results in the relative error due to the resolution difference between the inner and the outer loops. The error can be reduced by the presence of linearised physics and by adding more physical processes or by increasing the inner loop resolution which would then reduce the length of the assimilation window [Trémolet, 2004]. Experimental results have shown that the incremental 4D-Var can start to diverge

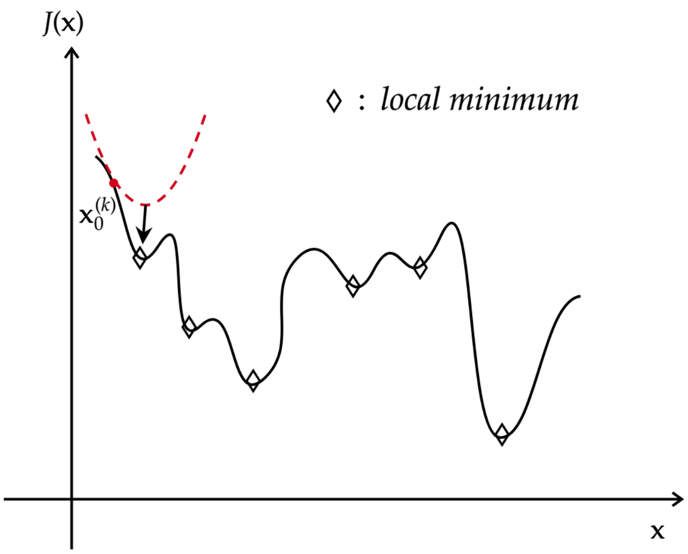


Figure 1.6: The same non-linear cost function where the red dashed curve shows the linearisation of J around the model state $\mathbf{x}_0^{(k)}$

after more than two outer loops when the TLM is too approximated. A key factor which affects the convergence behaviour is the resolution at which the inner and outer loops are run and not the linearisation itself. In fact, it is more of the temporal resolution which causes the anomaly in obtaining increments as one do more outer loops [Trémolet, 2007]. However, there is no guarantee that even after using the incremental 4D-Var one can reach the global minimum. Instead there can be a situation where after we linearise the cost function around a model state, the minimisation iteration points us towards one of the local minima and we get stuck there as shown in Fig. 1.6. A lot depends on the starting position of the initial guess, if it is closer to the global minimum then we are likely to reach it. The primary reason we use the incremental 4D-Var is to be able to use minimisation methods such as the Conjugate Gradient which is a far more efficient method than the quasi Newton methods when the cost function J is quadratic.

1.4 Weak constraint 4D-Var

So far we have talked about the data assimilation methods where the model is assumed to be perfect. A more general approach is to consider the fact that the model (1.12) is imperfect and so the presence of model errors needs to be accounted for. This leads to another formulation known as the *weak constraint 4D-Var* first introduced by Sasaki [Sasaki, 1970]. In this formulation the model is only imposed as a weak constraint in the optimisation problem because the minimising solution $\mathbf{x}$ does not satisfy the model exactly [Trémolet, 2006]. The cost function (1.13) gets an additional term due to the model error constraint. Since the values of model state are needed at all time steps, the control variable becomes the full four dimensional model state $\mathbf{x} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N)$ giving the following weak constraint 4D-Var cost function

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2}\left(\mathbf{x}_0 - \mathbf{x}^b\right)^T \mathbf{B}^{-1}\left(\mathbf{x}_0 - \mathbf{x}^b\right) + \frac{1}{2}\sum_{i=0}^{N}(H_i(\mathbf{x}_i) - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1}(H_i(\mathbf{x}_i) - \mathbf{y}_i^o)$$
$$+ \frac{1}{2}\sum_{i=1}^{N}\{\mathbf{x}_i - M_i(\mathbf{x}_{i-1})\}^T \mathbf{Q}_i^{-1}\{\mathbf{x}_i - M_i(\mathbf{x}_{i-1})\} \tag{1.29}$$

where $\mathbf{Q}_i$ is the model error covariance matrix. One thing to observe is the presence of the intermediate model states in the control variable means that the model operator is now only present in the last term of the cost function.

The error constraint term in the cost function can also be written in terms of model error forcing which is given as

$$\mathbf{x}_i = M_i(\mathbf{x}_{i-1}) + \boldsymbol{\eta}_i \tag{1.30}$$

where $\boldsymbol{\eta}_i$ represents the three-dimensional model error at time t_i. This gives rise to an another way of formulating the weak constraint problem through the change of variables in (1.30). The control variable $\mathbf{x}$ now consists of the 3D initial state $\mathbf{x}_0$ and the model error forcing term $\boldsymbol{\eta}_i$ [Trémolet, 2006] and the cost function becomes

$$\begin{aligned} J(\mathbf{x}_0, \boldsymbol{\eta}) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}^b)^T \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}^b) + \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{\eta}_i^T \mathbf{Q}_i^{-1} \boldsymbol{\eta}_i \\ + \frac{1}{2}\sum_{i=0}^{N} (H_i(\mathbf{x}_i) - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1} (H_i(\mathbf{x}_i) - \mathbf{y}_i^o) \end{aligned} \tag{1.31}$$

Alternatively, the model-error constraint can also be expressed in terms of model bias by taking the difference between the perfect model trajectory and the model state at each time step obtained from a given initial condition. It is represented as $\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_N$. Then (1.30) can also be written as

$$\mathbf{x}_i = M_{i,0}(\mathbf{x}_0) + \boldsymbol{\beta}_i \tag{1.32}$$

where $M_{i,0}$ is the model integrated from time t_0 to time t_i. With this change of variable, there is yet another control variable for the weak constraint 4D-Var. The model bias is generally taken as a constant over the assimilating window due to the ergodic assumption [Trémolet, 2006]. The corresponding cost function is written as

$$\begin{aligned} J(\mathbf{x}_0, \boldsymbol{\beta}) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}^b)^T \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}^b) + \frac{1}{2}\sum_{i=1}^{N} \boldsymbol{\beta}^T \mathbf{Q}^{-1} \boldsymbol{\beta} \\ + \frac{1}{2}\sum_{i=0}^{N} (H_i(M_{i,0}(\mathbf{x}_0) + \boldsymbol{\beta}) - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1} (H_i(M_{i,0}(\mathbf{x}_0) + \boldsymbol{\beta}) - \mathbf{y}_i^o) \end{aligned} \tag{1.33}$$

Remark 1.5. *Although the above approaches for the weak constraint formulations in (1.29), (1.31) and (1.33) are mathematically equivalent, they however lead to very different minimisation procedure and preconditioning techniques.*

1.5 Adjoint method

In variational data assimilation the gradient-based minimisation techniques [Lewis and Derber, 1985, Le Dimet and Talagrand, 1986, Talagrand and Courtier, 1987, Courtier and Talagrand, 1990] are most commonly used to minimise the cost function J. Finding the gradient of the cost function points to finding the Jacobian matrix with respect to the control variable $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$

$$\nabla J(\mathbf{x}_0) = \begin{pmatrix} \dfrac{\partial J}{\partial x_1}(\mathbf{x}_0) \\ \vdots \\ \dfrac{\partial J}{\partial x_n}(\mathbf{x}_0) \end{pmatrix} = \begin{pmatrix} \dfrac{J(\mathbf{x}_0 + \alpha \mathbf{e}_1) - J(\mathbf{x}_0)}{\alpha} \\ \vdots \\ \dfrac{J(\mathbf{x}_0 + \alpha \mathbf{e}_n) - J(\mathbf{x}_0)}{\alpha} \end{pmatrix} \tag{1.34}$$

One way of calculating the Jacobian is through finite differences which will require at least $n+1$ model runs. As discussed before in this chapter, the model state vector is usually of $\mathcal{O}(10^7 - 10^9)$ and one cannot afford to find the gradient through growth rates along all the possible directions. An alternative way which is extremely efficient in terms of making it possible to do real-time predictions is through the use of the adjoint method [Le Dimet and Talagrand, 1986].

The adjoint method makes use of the adjoint model which cannot be explained without talking first about its corresponding tangent linear model (TLM). TLM can also be represented as the Jacobian matrix around the linearisation state. It maps the variations of the control variable onto the variations in the model prediction. The adjoint model is just the adjoint of this Jacobian matrix and it maps in the reverse direction determining the effect of the control variables on a given perturbation in the model output [Giering and Kaminski, 1998].

Given the TLM in discrete form as in (1.26), the computation of the gradient requires the backward integration of the adjoint model in the sense that one starts with the final solution of the tangent linear model as the initial condition and evaluates in the reverse order with the help of the chain rule. Computation of the gradient by the backward adjoint integration is equivalent to the cost of around 2-5 model forward model runs which is extraordinarily less than the cost of n+1 model runs through finite differences. The adjoint equations for calculating the gradient of the 4D-Var cost function (1.13) are given by [Griffith and Nichols, 2000]

$$\begin{aligned}
\mathbf{u}_N &= 0 \\
\mathbf{u}_i &= \mathbf{M}_i^T(\mathbf{u}_{i+1}) - \mathbf{H}_i^T \mathbf{R}_i^{-1}(H_i(\mathbf{x}_i) - \mathbf{y}_i^o), \qquad i = N-1, \cdots, 0
\end{aligned} \tag{1.35}$$

where $\mathbf{u}_i \in \mathbb{R}^n$ is the adjoint variable and $\mathbf{M}_i \in \mathbb{R}^{n\times n}$ and $\mathbf{H}_i \in \mathbb{R}^{n\times m}$ are the Jacobians of M and H with respect to the model state $\mathbf{x}_i$. Then the gradient of the cost function (1.13) is given with respect to the initial state $\mathbf{x}_0$ is given by

$$\nabla J(\mathbf{x}_0) = \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_b) - \mathbf{u}_0 \tag{1.36}$$

The optimality condition requires that the gradient is equal to zero. If not, the gradient produces a local descent direction to provide an improved optimal estimate [Griffith and Nichols, 2000]. The adjoint model in its discrete form can be written as

$$\begin{aligned}
\mathbf{u}(t_N) &= \mathbf{u}_N \\
\mathbf{u}_i &= \mathbf{M}_i^T(\mathbf{u}_{i+1}) \quad i = N-1, \cdots, 0
\end{aligned} \tag{1.37}$$

where $\mathbf{M}_i^T : \mathbb{R}^n \to \mathbb{R}^n$ is now the adjoint model operator or the adjoint of the tangent linear model (TLM) (1.26) describing the adjoint state evolution from time t_{i+1} to t_i. A detailed explanation about the adjoint methods and how to code them can be found in [Giering and Kaminski, 1998].

The adjoint is a powerful method for obtaining the exact gradient however a lot depends on the validity of the tangent linear approximations. There are cases when the perturbation size is large, the higher order terms in the tangent linear approximations become more and more significant and can no longer be neglected. The linearisation might become unreliable in the sense that both the tangent-linear and the adjoint model may fail to satisfy and give no meaningful results. Generally there is a time where almost any perturbation enters a non-linear regime in most atmospheric models rendering the adjoint applications limited to short time spans [Errico, 1997]. In the case of incremental 4D-Var, it is also recommended that the inner loop minimisation should yield small increments otherwise there is a high possibility that the algorithm diverges.

1.6 Minimisation methods

From section 1.5 on the adjoint method we see that (1.35), (1.36) and (1.37) point out that setting the cost function gradient $\nabla J(\mathbf{x}_0)$ to zero leads to solving a system of equations linear in $\mathbf{x}_0$. In this section we are going to talk about the minimisation techniques used in variational data assimilation such as the Krylov subspace methods for solving the linear system. In particular we will explain the conjugate gradient method which is a type of Krylov subspace method and we will show how it is linked to solving the inner loop problem of the incremental 4D-Var.

1.6.1 Krylov subspace methods

Consider the linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1.38}$$

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a large and sparse coefficient matrix and $\mathbf{b} \in \mathbb{R}^n$. When solving such system there is often the case where the access to the matrix is implicit in the form of a subroutine which gives out the matrix-vector product for a given vector as an input. Using a suitable Krylov subspace method, which has the matrix-vector multiplication as its most dominant operation, is a logical option [Ipsen and Meyer, 1998].

Krylov subspace methods [Freund et al., 1992,Saad, 2003,Simoncini and Szyld, 2007,Liesen and Strakos, 2013] are projection based methods which find a sequence of approximate solutions $\mathbf{x}_j$ from an affine subspace $\mathbf{x}_0 + \mathcal{K}_j\,(j = 1, 2, \cdots)$ of dimension j by constraining the jth residual $\mathbf{r}_j = \mathbf{b} - \mathbf{A}\mathbf{x}_j$ to satisfy the Petrov-Galerkin condition

$$\mathbf{b} - \mathbf{A}\mathbf{x}_j \perp L_j \tag{1.39}$$

where $\mathbf{x}_0$ is some initial guess and $L_j \subseteq \mathbb{R}^n$ is another subspace of dimension j. The subspace $\mathcal{K}_j(\mathbf{A}, \mathbf{v})$ is called a *Krylov subspace* of order j which is generated by the repeated multiplication of the coefficient matrix $\mathbf{A}$ with a vector $\mathbf{v}$. That is,

$$\mathcal{K}_j(\mathbf{A}, \mathbf{v}) = \text{span}\{\mathbf{v}, \mathbf{A}\mathbf{v}, \cdots, \mathbf{A}^{j-1}\mathbf{v}\} \tag{1.40}$$

Note that Krylov subspaces are nested i.e. $\mathcal{K}_j \subseteq \mathcal{K}_{j+1}$. Also note that for the Krylov subspace methods, the affine subspace $\mathbf{x}_0 + \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ is used with $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ being the initial residual. Usually the initial guess is taken to be $\mathbf{x}_0 = 0$ (i.e. $\mathbf{r}_0 = \mathbf{b}$) since the space $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$ is a good space to search for the approximate solution and it is closely linked to the inverse of the matrix $\mathbf{A}$. The eigenvalues of $\mathbf{A}$ play an important role in determining the dimension of the space $\mathcal{K}$ and hence the number of iterations it would take since it is determined from the degree of the minimal polynomial of $\mathbf{A}$ [Ipsen and Meyer, 1998].

Different Krylov methods can be obtained depending upon the choice of the subspace L_j [Van der Vorst, 2003]. For $L_j = \mathbf{A}\mathcal{K}_j$ we get GMRES [Saad and Schultz, 1986] which is one of the most well known methods for solving non symmetric systems. For our discussion, we are going to talk about the *Conjugate Gradient* method which one gets when $L_j = \mathcal{K}_j$ and is a popular choice for solving symmetric systems.

1.6.2 Conjugate gradient method

The conjugate gradient method (or simply CG) is a kind of an iterative Krylov subspace method [Axelsson, 1996, Kelley, 1999, Golub and Van Loan, 2013] which was introduced by Hestenes and Stiefel [Hestenes et al., 1952] to solve symmetric positive definite linear systems

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \qquad \mathbf{A} \in \mathbb{R}^{n\times n} \text{ and } \mathbf{b} \in \mathbb{R}^n. \tag{1.41}$$

The problem (1.41) can be stated equivalently as a convex quadratic minimisation problem

$$\min_{\mathbf{x}} q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \tag{1.42}$$

as (1.41) and (1.42) have the same unique solution. Since CG is a Krylov projection method, it satisfies the Petrov-Galerkin condition for $L_j = \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ also called the Ritz-Galerkin condition [Van der Vorst, 2003]

$$\mathbf{b} - \mathbf{A}\mathbf{x}_j \perp \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0). \tag{1.43}$$

The search directions $\mathbf{p}_j$ are constructed using the residual $\mathbf{r}_j$ at each iteration j so that they are $\mathbf{A}$-conjugate to all the previous search directions. That is, a given set of search directions $\{\mathbf{p}_0, \mathbf{p}_1, \cdots \mathbf{p}_i\}$ is conjugate with respect to the matrix $\mathbf{A}$ if

$$\mathbf{p}_j^T \mathbf{A} \mathbf{p}_k = 0 \qquad \text{when } j \neq k. \tag{1.44}$$

Moreover, the residuals have a nice property that they are orthogonal to all the previous residuals and thus one always gets a linearly independent set of search directions unless the residual becomes zero [Shewchuk, 1994]. This leads to a very important result that CG is guaranteed to converge in at most n iterations when $\mathbf{A} \in \mathbb{R}^{n\times n}$.

Algorithm 3 CONJUGATE GRADIENT ALGORITHM

Given: Symmetric positive-definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, right hand side vector $\mathbf{b} \in \mathbb{R}^n$, stopping tolerance ϵ_{cg}
Set $\mathbf{x}_0 = 0$, $\mathbf{r}_0 = \mathbf{b}$, $\mathbf{p}_0 = -\mathbf{b}$
$j = 0$
while true **do**
    $\alpha_j = \mathbf{r}_j^T \mathbf{r}_j / \mathbf{p}_j \mathbf{A} \mathbf{p}_j$
    $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{p}_j$
    $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{A} \mathbf{p}_j$
    if $\|\mathbf{r}_{j+1}\|_2 \leq \epsilon_{\text{cg}}$ **then**
        break
    end if
    $\mathbf{p}_{j+1} = -\mathbf{r}_{j+1} + \left(\dfrac{\mathbf{r}_{j+1}^T \mathbf{r}_{j+1}}{\mathbf{r}_j^T \mathbf{r}_j}\right)$
    $j = j + 1$
end while

The convergence of the CG algorithm depends on the distribution of the eigenvalues of the coefficient matrix $\mathbf{A}$. If $\mathbf{A}$ has r distinct eigenvalues then the method will converge in at most r iterations [Nocedal and Wright, 1999] which becomes very handy for the case when n is very large. A convergence bound for the CG depending upon the condition number $\kappa(\mathbf{A})$ is useful in the case when we only have knowledge about the extreme eigenvalues of $\mathbf{A}$ [Greenbaum, 1997] and it is given by

$$\frac{\|\mathbf{e}_j\|_{\mathbf{A}}}{\|\mathbf{e}_0\|_{\mathbf{A}}} \leq 2\left[\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^j + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^j\right]^{-1} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^j \tag{1.45}$$

where $\|\mathbf{e}_0\|_{\mathbf{A}}$ and $\|\mathbf{e}_j\|_{\mathbf{A}}$ are the error at the initial step and at iteration j respectively.

1.6.3 Link to data assimilation

In this subsection we relate how the conjugate gradient method can be used for the incremental 4D-Var. We note that the gradient of the inner loop cost function (1.25) of incremental 4D-Var is given by

$$\nabla J(\delta \mathbf{x}_0) = \mathbf{B}^{-1}\{\delta \mathbf{x}_0 - (\mathbf{x}^b - \mathbf{x}_0)\} + \sum_{i=0}^{N} \mathbf{H}_i^T \mathbf{R}_i^{-1}(\mathbf{H}_i \delta \mathbf{x}_i - \mathbf{d}_i) \tag{1.46}$$

Rearranging the right hand side terms we get

$$\begin{aligned}\nabla J(\delta \mathbf{x}_0) &= \Big(\mathbf{B}^{-1} + \sum_{i=0}^{N} \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i \mathbf{M}_i \mathbf{M}_{i-1} \cdots \mathbf{M}_1\Big)\delta \mathbf{x}_0 - \mathbf{B}^{-1}(\mathbf{x}^b - \mathbf{x}_0) \\ &\quad - \sum_{i=0}^{N} \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{d}_i \\ &= \Big(\mathbf{B}^{-1} + \widehat{\mathbf{H}}^T \widehat{\mathbf{R}}^{-1} \widehat{\mathbf{H}}\Big)\,\delta \mathbf{x}_0 - \mathbf{B}^{-1}(\mathbf{x}^b - \mathbf{x}_0) - \widehat{\mathbf{H}}^T \widehat{\mathbf{R}}^{-1} \widehat{\mathbf{d}}\end{aligned} \tag{1.47}$$

where

$$\widehat{\mathbf{H}} = \begin{pmatrix} \mathbf{H}_0 \\ \mathbf{H}_1\mathbf{M}_1 \\ \vdots \\ \mathbf{H}_N\mathbf{M}_N \cdots \mathbf{M}_1 \end{pmatrix}, \quad \widehat{\mathbf{R}} = \begin{pmatrix} \mathbf{R}_0 & & & \\ & \mathbf{R}_1 & & \\ & & \ddots & \\ & & & \mathbf{R}_N \end{pmatrix}, \quad \widehat{\mathbf{d}} = \begin{pmatrix} \mathbf{d}_0 \\ \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_N \end{pmatrix} \tag{1.48}$$

Setting the gradient (1.47) equal to zero we have

$$\Big(\mathbf{B}^{-1} + \widehat{\mathbf{H}}^T \widehat{\mathbf{R}}^{-1} \widehat{\mathbf{H}}\Big)\,\delta \mathbf{x}_0 = \mathbf{B}^{-1}(\mathbf{x}^b - \mathbf{x}_0) + \widehat{\mathbf{H}}^T \widehat{\mathbf{R}}^{-1} \widehat{\mathbf{d}} \tag{1.49}$$

which can be put together as a linear system

$$\mathbf{A}_{\mathrm{Inc\_4dvar}} \delta \mathbf{x}_0 = \mathbf{b} \tag{1.50}$$

where

$$\mathbf{A}_{\mathrm{Inc\_4dvar}} = \mathbf{B}^{-1} + \widehat{\mathbf{H}}^T \widehat{\mathbf{R}}^{-1} \widehat{\mathbf{H}}, \qquad \mathbf{b} = \mathbf{B}^{-1}(\mathbf{x}^b - \mathbf{x}_0) + \widehat{\mathbf{H}}^T \widehat{\mathbf{R}}^{-1} \widehat{\mathbf{d}} \tag{1.51}$$

Note that the matrix $\mathbf{B}^{-1} + \widehat{\mathbf{H}}^T \widehat{\mathbf{R}}^{-1} \widehat{\mathbf{H}}$ is symmetric. Therefore we can use CG for solving the linear symmetric system (1.50) and which is equivalent to solving a convex quadratic minimisation problem where the quadratic is given as

$$q(\delta \mathbf{x}_0) = \frac{1}{2}\, \delta \mathbf{x}^T \mathbf{A}_{\text{Inc\_4dvar}} \delta \mathbf{x}_0 - \mathbf{b}^T \delta \mathbf{x}_0 \tag{1.52}$$

Chapter 2

Parallel in Time algorithms: Parareal method

Contents

The solution to large scale time-dependent problems on parallel architectures is commonly achieved by using domain-decomposition methods exploiting space parallelisation [Toselli and Widlund, 2004, Dolean et al., 2015]. A classical domain-decomposition method partitions the whole spatial domain into smaller subdomains which results in a coupled system. To each subdomain multiple cores are assigned which are run independently of one another and with the only synchronisation point during the exchange of information at the interfaces. There is however a limitation to using the spatial parallelisation when the communication overhead becomes large enough to the extent that it starts dominating the overall computation time.

In this chapter we will introduce the idea of time parallelisation and show how despite of the causality property of time domain one can exploit it as an alternative source of parallelism. The so called parallel-in-time or PinT algorithms are discussed while restricting ourselves to the category of parallel multiple shooting methods. Out of these methods our main topic of discussion will be on the well known *Parareal algorithm* in section 2.3. It can be thought of a two-level prediction-corrector algorithm which uses a sequential coarse propagator G and a parallel fine propagator G in an alternative fashion. Parareal will be talked about comprehensively for the linear problems in the rest of the chapter and we sum up some of the most important points here.

In terms of convergence properties Parareal admits a superlinear convergence (section 2.4) for the time bounded linear scalar ODEs and non-linear ODEs. A further scalar convergence analysis shows that Parareal works well for the parabolic problems but not for the hyperbolic problems in spite of a superlinear convergence. From the general Parareal error expression, we see that the term $\mathbf{F} - \mathbf{G}$ plays a significant role in governing the evolution of the Parareal error. This fact has been scrutinised by studying the eigenvalues of $\mathbf{F} - \mathbf{G}$ with the help of an implicit theta scheme in section 2.5. The error manifests as a result of the difference in the dissipation rates and dispersion between $\mathbf{F}$ and $\mathbf{G}$. A speedup analysis is also done and some bounds are provided which point out that in order to obtain a reasonable speedup one has to run as few Parareal iterations as possible. Section 2.7 talks about some of the other common challenges that Parareal faces such as the cost of the coarse solver, cost of the fine solver, computational implementation along with the modifications provided in the literature to tackle them. We end this chapter by briefly talking about the PITA algorithm, Krylov subspace enhanced Parareal and an adaptive Parareal.

2.1 Introduction to time parallelisation

The concept of time parallelisation might sound counter-intuitive but it is one of the very active areas of research for the past few decades now. Parallelisation in the time domain is a different prospect than parallelisation in the spatial domain. Time domain has the inherent property of causality which means that in order to predict the future one must know some information about the past. For the numerical solution of a simple time-dependent model, one needs to know the solution at the previous time step to proceed for the time integration and find out the solution at the next time step. This is where the so called parallel-in-time algorithms come to circumvent the causality principle and enable us to provide some additional parallelism opportunity in the time domain.

The first ideas could be traced back to the 1960s when Nievergelt [Nievergelt, 1964] introduced a naive approach of integrating a simple ODE problem parallel in time. His idea was to do an initial prediction with a one-step method and then running multiple integrations of the neighbouring points around the solutions simultaneously for each subinterval. The global solution was then be obtained by doing a simple interpolation between the clusters of the obtained trajectories. An another early work can be found in [Miranker and Liniger, 1967] talking about a class of numerical methods for time parallel integrations. The literature review paper by Gear [Gear, 1988] and a book by Burrage [Burrage, 1995] talk in detail about the “parallelism across the methods” for initial value problems in ODEs. Gander [Gander, 2015] has done an intensive research in a monograph combining all the major works in the field of parallel-in-time (PinT) methods from the last 5 decades. He has ordered his chapters by classifying the PinT methods into four broad categories on their mode of application : multiple shooting methods, waveform-relaxation methods, multigrid methods and direct methods. Our focus is on one of the multiple shooting methods called the Parareal algorithm which is probably the most studied PinT algorithm in this domain. For a more general introduction about the different PinT algorithms see [Ong and Schroder, 2020, Gander et al., 2022].

2.2 Multiple shooting method

The concept of multiple shooting stems from the classical shooting methods which are used to treat boundary value problems as an initial value problem (IVP) by using a *shooting parameter* [Keller, 2018]. The approach of a time-parallel solution of an IVP by Nievergelt [Nievergelt, 1964] can be considered as a multiple shooting algorithm which was further improved by the ideas in [Bellen and Zennaro, 1989, Khalaf and Hutchinson, 1992, Chartier and Philippe, 1993, Kiehl, 1994]. We consider an IVP of the form

$$\begin{aligned} \mathbf{x}'(t) &= f(t, \mathbf{x}(t)), \quad t \in [0, T], \\ \mathbf{x}(0) &= \mathbf{x}_0 \end{aligned} \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{x} \in \mathbb{R}^n$, i.e. the size of the problem is n. We introduce intermediate targets $\boldsymbol{\alpha}_i$ by partitioning the time domain $[0, T]$ into subintervals $[t_i, t_{i+1}]$, $i = 0, .., N-1$ with $0 = t_0 < t_1 < \cdots < t_N = T$ and solving the original problem (2.1) as

$$\begin{aligned} \mathbf{x}_i'(t, \boldsymbol{\alpha}_i) &= f(t, \mathbf{x}_i(t, \boldsymbol{\alpha}_i)), \quad t \in [t_i, t_{i+1}] \\ \mathbf{x}_i(t_i, \boldsymbol{\alpha}_i) &= \boldsymbol{\alpha}_i \end{aligned} \tag{2.2}$$

with $\mathbf{x}_0(t_0, \boldsymbol{\alpha}_0) = \boldsymbol{\alpha}_0 = \mathbf{x}_0$. To solve the above sub-IVPs we need to know the value of the intermediate targets $\boldsymbol{\alpha}_i$ called the shooting parameters which must satisfy the system of equations

$$\begin{aligned} \boldsymbol{\alpha}_0 &= \mathbf{x}_0(t_0, \boldsymbol{\alpha}_0) = \mathbf{x}(0) = \mathbf{x}_0 \\ \boldsymbol{\alpha}_1 &= \mathbf{x}_0(t_1, \boldsymbol{\alpha}_0) \\ &\vdots \\ \boldsymbol{\alpha}_N &= \mathbf{x}_{N-1}(t_N, \boldsymbol{\alpha}_{N-1}) \end{aligned} \tag{2.3}$$

Combining the shooting parameter values in a vector of vectors $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N)^T$ we solve for a non-linear system of equations

$$\widehat{f}(\boldsymbol{\alpha}) = \begin{pmatrix} \boldsymbol{\alpha}_0 - \mathbf{x}_0 \\ \boldsymbol{\alpha}_1 - \mathbf{x}_0(t_1, \boldsymbol{\alpha}_0) \\ \vdots \\ \boldsymbol{\alpha}_N - \mathbf{x}_{N-1}(t_N, \boldsymbol{\alpha}_{N-1}) \end{pmatrix} = \mathbf{0} \tag{2.4}$$

where $\widehat{f} : \mathbb{R}^{n.N+1} \to \mathbb{R}^{n.N+1}$ and thus (2.4) becomes a root-finding problem which can be solved through Newton's method. We start with an initial guess $\widehat{\boldsymbol{\alpha}}$ and solve for $k = 0, 1, 2, \cdots$

$$\widehat{f}'(\boldsymbol{\alpha}^k)(\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k) = -\widehat{f}(\boldsymbol{\alpha}^k) \tag{2.5}$$

with $\widehat{f}'(\boldsymbol{\alpha})$ as the Jacobian

$$\widehat{f}'(\boldsymbol{\alpha}^k) = \begin{pmatrix} \mathbf{I} & & & \\ -\dfrac{\partial \mathbf{x}_0}{\partial \boldsymbol{\alpha}_0}(t_1, \boldsymbol{\alpha}_0^k) & \mathbf{I} & & \\ & \ddots & \ddots & \\ & & -\dfrac{\partial \mathbf{x}_{N-1}}{\partial \boldsymbol{\alpha}_{N-1}}(t_N, \boldsymbol{\alpha}_{N-1}^k) & \mathbf{I} \end{pmatrix} \tag{2.6}$$

Multiplying the Newton iteration (2.5) with the Jacobian matrix on both the sides we have the following recurrence relation for $\boldsymbol{\alpha}^0 = \widehat{\boldsymbol{\alpha}}$ and for $k = 0, 1, 2, \cdots$

$$\begin{aligned} \boldsymbol{\alpha}_0^{k+1} &= \mathbf{x}_0 \\ \boldsymbol{\alpha}_{i+1}^{k+1} &= \mathbf{x}_i(t_{i+1}, \boldsymbol{\alpha}_i^k) + \frac{\partial \mathbf{x}_i}{\partial \boldsymbol{\alpha}_i}(t_{i+1}, \boldsymbol{\alpha}_i^k)(\boldsymbol{\alpha}_i^{k+1} - \boldsymbol{\alpha}_i^k), \quad i = 0, \cdots, N-1 \end{aligned} \tag{2.7}$$

where $\dfrac{\partial \mathbf{x}_i}{\partial \boldsymbol{\alpha}_i}$ is the Jacobian matrix evaluated at $\boldsymbol{\alpha}_i^k$.

The multiple shooting algorithm (2.7) has a locally quadratic convergence if the function $f(t, \mathbf{x}(t))$ is twice continuously differentiable in the second argument [Chartier and Philippe, 1993].

2.3 Parareal method

In this section and the remaining part of the chapter we are going to talk about the Parareal method which is one of the most extensively studied parallel-in-time algorithm. We are going to use Parareal as our source of time parallelism in all of our studies and analysis.

The Parareal algorithm was introduced by Lions, Maday and Turinici [Lions et al., 2001] (also in [Maday and Turinici, 2002, Maday, 2008]) with the objective of solving evolution problems parallel in time. It is widely used with applications in fractional differential equations [Xu et al., 2015, Wu and Zhou, 2017, Wu and

Zhou, 2018], Navier-Stokes equation [Trindade and Pereira, 2004, Fischer et al., 2005, Steiner et al., 2015], optimal control [Mathew et al., 2010, Maday et al., 2013], plasma turbulence [Samaddar et al., 2010, Reynolds-Barredo et al., 2012], quantum control [Maday et al., 2007, Maday and Turinici, 2003].

Parareal can be considered as a kind of a multiple shooting method where one can think of the subinterval solution $\mathbf{x}_i(t_{i+1}, \boldsymbol{\alpha}_i^k)$ in (2.7) being approximated by a fine solver $F(t_{i+1}, t_i, \boldsymbol{\alpha}_i^k)$ and the Jacobian matrix being replaced by the difference between the coarse approximations from the current and previous iteration using a coarse propagator G [Gander and Vandewalle, 2007]. That is,

$$\frac{\partial\, \mathbf{x}_i}{\partial\, \boldsymbol{\alpha}_i}(t_{i+1}, \boldsymbol{\alpha}_i^k)(\boldsymbol{\alpha}_i^{k+1} - \boldsymbol{\alpha}_i^k) \approx G(t_{i+1}, t_i, \boldsymbol{\alpha}_i^{k+1}) - G(t_{i+1}, t_i, \boldsymbol{\alpha}_i^k) \tag{2.8}$$

We formally introduce the method by considering a general class of ODE

$$\mathbf{x}'(t) = f(t, \mathbf{x}(t)), \quad t \in (0, T], \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.9}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$.

Let us consider a partition of the time domain $[0, T]$ by $0 = t_0 < t_1 < \cdots < t_N = T$ so we have N subintervals $\Omega_i = [t_i, t_{i+1}]$ with $i = 0, \cdots, N-1$. As described in [Baffico et al., 2002] in order to solve (2.9) for each subinterval Ω_i with $\mathbf{x}_i$ as the initial condition ($\mathbf{x}(t_i) = \mathbf{x}_i$), the method employs two propagators:

- a cheap coarse propagator $G(t_{i+1}, t_i, \mathbf{x}_i)$ which is fast but gives a rough solution at t_{i+1} from the initial condition $\mathbf{x}_i$.
- a fine propagator $F(t_{i+1}, t_i, \mathbf{x}_i)$ which is computationally expensive but gives very accurate solution (in theory the exact solution).

The trick is to apply the coarse solver G on the given initial condition $\mathbf{x}_0^0 = \mathbf{x}_0$ to obtain a quick initial configuration across the whole time-integration domain. So the initial Parareal iteration $k = 0$ looks like

$$\mathbf{x}_{i+1}^0 = G(t_{i+1}, t_i, \mathbf{x}_i^0) \tag{2.10}$$

Though this step is purely sequential, it circumvents the causality principle by providing initial conditions at all the intermediate subintervals so they are no longer dependent on the previous solutions for performing independent time-integrations.

This is where the fine propagator F comes into the picture to do all the heavy fine integrations in parallel. The differences between the fine and coarse integrations or the jumps are then added to the next iteration. The iterations $k = 0, 1, \cdots, N-1$ are given by

$$\mathbf{x}_{i+1}^{k+1} = \underbrace{G(t_{i+1}, t_i, \mathbf{x}_i^{k+1})}_{\text{Prediction}} + \underbrace{F(t_{i+1}, t_i, \mathbf{x}_i^{k}) - G(t_{i+1}, t_i, \mathbf{x}_i^{k})}_{\text{Correction}} \tag{2.11}$$

In other words, Parareal can be described as a predictor-corrector algorithm where at a particular iteration, the prediction made by the coarse solver is corrected by the difference between the solution of the fine and the coarse solver from the previous iteration. Fig. 2.1 illustrates a correction step of the Parareal for a

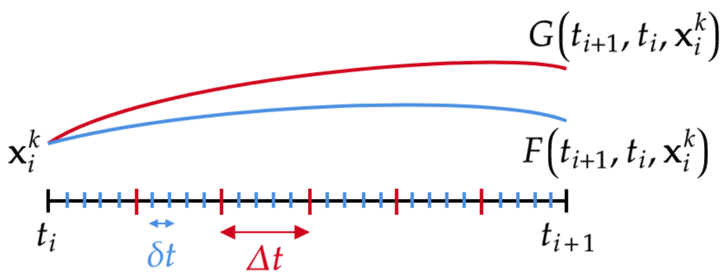


Figure 2.1: Illustration of the Parareal algorithm on a subinterval $\Omega_i = [t_i, t_{i+1}]$

given iteration k on a subinterval $\Omega_i = [t_i, t_{i+1}]$. The coarse solver G propagates with time-step Δt and the fine solver F propagates with the time-step δt. The difference of the fine and coarse solutions at t_{i+1} will be used for correcting the next prediction $\mathbf{x}_{i+1}^{k+1}$

Also an algorithmic representation of Parareal is provided in Algorithm 4. Inside the algorithm the stopping criterion of Parareal is taken to be the L^∞ norm of the error between the two successive Parareal iterates. That is for some tolerance ϵ_p,

$$\max_{1 \le i \le N} |\mathbf{x}^{k+1} - \mathbf{x}^{k}| < \epsilon_\mathrm{p} \tag{2.12}$$

The reason we use this here is that the important convergence results for Parareal which we are going to discuss in the next section use the above stopping criterion.

Algorithm 4 PARAREAL ALGORITHM

Given: initial condition $\mathbf{x}_0$, Parareal tolerance ϵ_p or maximum iterations k_max
Initialisation:
$\mathbf{x}_0^0 = \mathbf{x}_0$
for $i = 0$ to $N-1$ **do**
  $\mathbf{x}_{i+1}^0 = G(t_{i+1}, t_i, \mathbf{x}_i^0)$
end for
Iterations:
$k = 0$
repeat
  Parallel fine integrations:
  for $i = 0$ to $N-1$ **do**
    $\tilde{\mathbf{x}}_{i+1}^{k+1} = F(t_{i+1}, t_i, \mathbf{x}_i^k)$
  end for
  Sequential correction step:
  for $i = 0$ to $N-1$ **do**
    $\mathbf{x}_{i+1}^{k+1} = G(t_{i+1}, t_i, \mathbf{x}_i^{k+1}) + \tilde{\mathbf{x}}_{i+1}^{k+1} - G(t_{i+1}, t_i, \mathbf{x}_i^k)$
  end for
  $k = k+1$
until $k = k_\mathrm{max}$ or $\max\limits_{1\leq i\leq N} |\mathbf{x}^{k+1} - \mathbf{x}^k| < \epsilon_\mathrm{p}$

2.4 Convergence properties of Parareal

2.4.1 Convergence for scalar ODE

In the original Parareal publication [Lions et al., 2001] the convergence property has been studied on a scalar linear equation by taking a fixed number of iterations k. For a given scalar linear ODE

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = a\mathbf{x}, \quad \mathbf{x}(0) = \mathbf{x}_0, \quad t \in [0, T] \quad \text{with } a \in \mathbb{C} \tag{2.13}$$

we state below the convergence result in a simplified form as a proposition [Gander and Vandewalle, 2007, Nielsen, 2012].

Proposition 2.1. *Let $\Delta t = T/N$, $t_i = i\Delta t$ for $i = 0, .., N$. Consider the IVP (2.13) for $a \in \mathbb{R}$ and let us assume that the fine propagator $\mathbf{F}(t_{i+1}, t_i, \mathbf{x}_i^k)$ is exact with the initial condition $\mathbf{x}(t_i) = \mathbf{x}_i^k$. Let $\mathbf{G}(t_{i+1}, t_i, \mathbf{x}_i^k)$ be the corresponding backward Euler approximation with time step Δt. Then*

$$\max_{1 \le i \le N} |\mathbf{x}(t_i) - \mathbf{x}_i^k| \le C_k \Delta t^{k+1} \tag{2.14}$$

The proposition says that the for a fixed iteration k the error of the Parareal algorithm behaves as an $\mathcal{O}(\Delta t^{k+1})$ method in Δt. This result was later extended to a more general higher order time-integration schemes for the coarse integrator $\mathbf{G}$ in [Bal and Maday, 2002, Bal, 2005]. In those articles it was shown that if an iterative scheme of order m is taken to be the coarse propagator, then the Parareal algorithm obtained is of order $\mathcal{O}(\Delta t^{m(k+1)})$ after k iterations. This approach used a fixed number of Parareal iterations since the constant C_k grows with k.

Instead of fixing k, the convergence for the same scalar ODE (2.13) was studied in a different manner by Gander and Vandewalle [Gander and Vandewalle, 2007] by fixing the time-step Δt and letting k go to infinity. The coarse propagator was taken as a one step method $\mathbf{G}(t_{i+1}, t_i, \mathbf{x}_i^k) = \mathbf{R}(a\Delta t)\,\mathbf{x}_i^k$ where $\mathbf{R}(z)$ is the stability function for some $z \in \mathbb{C}$. With the help of the powers of a strictly lower triangular Toeplitz matrices $\mathbf{T}(\beta)$ of size N, (whose elements are fully defined by the values of the first column elements) defined as

$$\mathbf{T}_{i1} = \begin{cases} 0 & \text{if} \quad i = 1, \\ \beta^{i-2},\ \beta \in \mathbb{C} & \text{if} \quad 2 \le i \le N \end{cases} \tag{2.15}$$

a superlinear convergence was shown in the case of bounded time intervals and a linear convergence was shown in the case of infinitely long time intervals. The following theorem states the result for the bounded time interval case [Gander and Vandewalle, 2007].

Theorem 2.1. *Let $T < \infty, \Delta t = T/N$, and $t_i = i\Delta t$ for $i = 0, 1, \cdots, N$. Let $\mathbf{F}(t_{i+1}, t_i, \mathbf{x}_i^k)$ be the exact solution of (2.13) with $\mathbf{x}(t_i) = \mathbf{x}_i^k$ and let $\mathbf{G}(t_{i+1}, t_i, \mathbf{x}_i^k) = \mathbf{R}(a\Delta t)$ be a one-step method. Then, with $\mathbf{T}(\beta)$ defined in (2.15) we have the bound*

$$\max_{1 \le i \le N} |\mathbf{x}(t_i) - \mathbf{x}_i^k| \le |e^{a\Delta t} - \mathbf{R}(a\Delta t)|^k \, \|\mathbf{T}^k(\mathbf{R}(a\Delta t))\|_\infty \max_{1 \le i \le N} |\mathbf{x}(t_i) - \mathbf{x}_i^0| \tag{2.16}$$

In particular, when we consider a one step method in its region of absolute stability i.e. $|\mathbf{R}(a\Delta t)| < 1$ we obtain the following bound using the above theorem

Corollary 2.4.1. *In Theorem 2.1 if we assume that $\mathbf{G}(t_{i+1}, t_i, \mathbf{x}_i^k) = \mathbf{R}(a\Delta t)\,\mathbf{x}_i^k$ is a one-step method in its region of absolute stability then we have the bound*

$$\max_{1\le i\le N} |\mathbf{x}(t_i) - \mathbf{x}_i^k| \le \frac{|e^{a\Delta t} - \mathbf{R}(a\Delta t)|^k}{k\,!} \prod_{p=1}^{k} (N-p) \max_{1\le i\le N} |\mathbf{x}(t_i) - \mathbf{x}_i^0| \tag{2.17}$$

If the local truncation error of $\mathbf{G}$ is bounded by $C\Delta t^{m+1}$, with $m > 0$ and C a constant, then for Δt small enough we have

$$\max_{1\le i\le N} |\mathbf{x}(t_i) - \mathbf{x}_i^k| \le \frac{(CT)^k}{k\,!} \Delta t^{mk} \max_{1\le i\le N} |\mathbf{x}(t_i) - \mathbf{x}_i^0| \tag{2.18}$$

Remark 2.1. *Looking at the product term in (2.17) it can be said that the Parareal algorithm converges to the fine solver solution after $N-1$ iterations for any Δt irrespective of the choice of $\mathbf{R}(a\Delta t)$ or the coarse propagator $\mathbf{G}$. However, this is undesirable since $N-1$ Parareal iterations are equivalent to a sequential fine solver run giving us no speedup.*

Remark 2.2. *The presence of the $k\,!$ term in the denominator of (2.17) is of our interest since it indicates a superlinear convergence. The Parareal will take much fewer number of iterations than $N-1$ resulting in speedup gains.*

In the next subsection we are going to discuss in more detail about the evolution of the Parareal error in scalar equations using numerical experiments.

2.4.2 Numerical experiments for the scalar equation

In this subsection we are going to talk about the convergence results of the scalar equation for two cases: a diffusive scalar equation (parabolic equation in PDEs) and a scalar wave equation (hyperbolic equation in PDEs). Let us assume the scalar fine and coarse propagators to be f and g with time-steps δt and Δt respectively. To quantify and obtain a closed form expression of the error for the linear system when Parareal is used, we go back to the Parareal correction step

$$\mathrm{x}_i^k = g\,\mathrm{x}_{i-1}^k + (f-g)\mathrm{x}_{i-1}^{k-1}. \tag{2.19}$$

Let $\mathrm{x}_{f,i}$ be the reference solution at the start of the time window i obtained by integrating the model using the fine solver f (i.e. $\mathrm{x}_{f,i} = f^i \mathrm{x}_0$). Then the error at the kth iteration is defined as $e_i^k = \mathrm{x}_{f,i} - \mathrm{x}_i^k$ which satisfies

$$e_i^k = g e_{i-1}^k + (f-g)e_{i-1}^{k-1}. \tag{2.20}$$

Expanding the above recurrence relation we have

$$e_i^k = (f-g)\sum_{p=k}^{i-1} g^{i-p-1} e_p^{k-1} \tag{2.21}$$

where we have $e_0^0 = 0$ and also by theory, $e_i^k = 0$ for $i \leq k$. Now if the initial states are obtained by a coarse grid integration (i.e. $\mathrm{x}_i^0 = g^i \mathrm{x}_0$), we have $e_i^0 = (f-g)\mathrm{x}_i^0$ and we get a general expression for e_i^k [Staff and Rønquist, 2005]

$$e_i^k = \sum_{p=k+1}^{i} C_p^i (f-g)^p g^{i-p}, \quad i > k \tag{2.22}$$

where $C_p^i = \dfrac{i\,!}{p\,!(i-p)\,!}$ is the binomial coefficient.

Remark 2.3. *In a similar manner, the error expression* $\mathbf{e}_i^k$ *can be obtained for the matrix form given as*

$$\mathbf{e}_i^k = \sum_{p=k+1}^{i} C_p^i (\mathbf{F}-\mathbf{G})^p \mathbf{G}^{i-p}, \quad i > k \tag{2.23}$$

From the expression (2.22) we see that the e_i^k is the sum of the products involving the powers of $(f-g)$ and g. In practice (2.22) shows that when the original system is dissipative, it is much easier for Parareal to solve the system since in that case $|g| < 1$. To see that we consider the scalar equation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = -a\mathbf{x}, \qquad a \in \mathbb{R}^+. \tag{2.24}$$

We take the total number of time windows to be N, each of length Δt so that we have only one coarse time step per time window and we have the total time period of integration $T = N\Delta t$. Let γ be the ratio of the number of fine time-steps per time window to the number of coarse time-steps per time window. If δt is the fine time-step then $\delta t = \Delta t/\gamma$. Using a simple backward Euler method to discretise the above equation yields the propagators

$$g(a\Delta t) = \left(\frac{1}{1 + a\Delta t}\right), \qquad f(a\Delta t) = \left(\frac{1}{1 + a(\Delta t/\gamma)}\right)^{\gamma} \tag{2.25}$$

Fig. 2.2 below shows the Parareal errors e_N^k at the end of the integration time for the cases when $\gamma = 20$ (left) and $\gamma = 10$ (right) with $N = 10$. We observe that

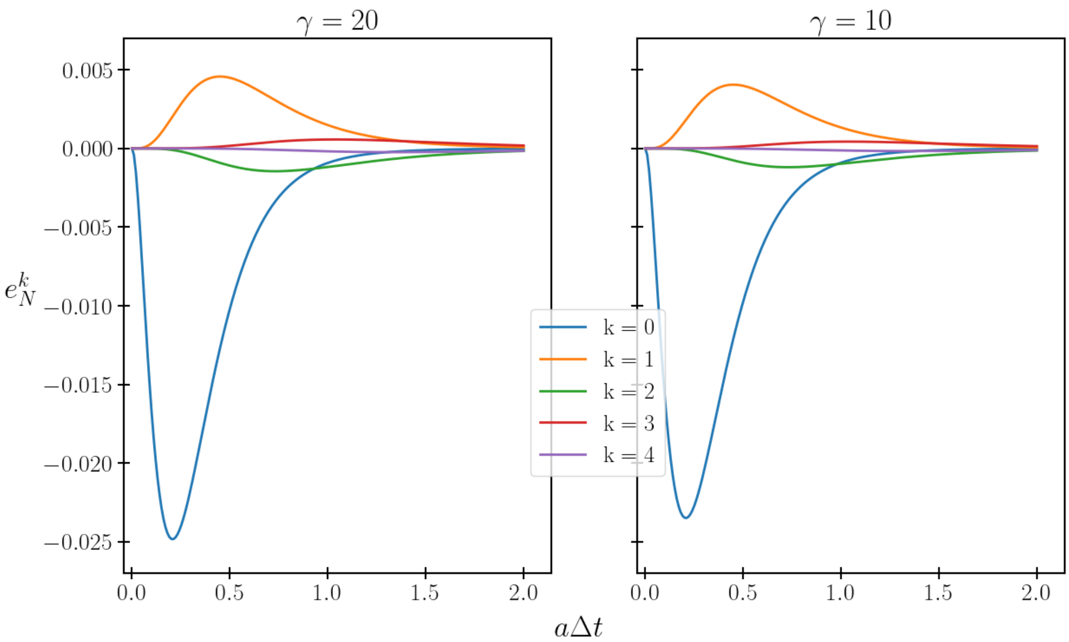


Figure 2.2: Parareal error e_N^k for successive iterates k as a function of $a\Delta t$

even after using a large value of γ the Parareal error is very small and at $k = 4$ the error is almost equal to zero. This confirms the results obtained in [Gander and Vandewalle, 2007] of a quick Parareal convergence for the diffusion equation. Next we move to the case of a simple scalar wave equation

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = i\omega\mathbf{x}, \qquad t \in [0, T] \tag{2.26}$$

where ω is some real frequency. We define the Parareal propagators for (2.26) using the implicit θ-scheme which we write as

$$\frac{\mathbf{x}^{q+1} - \mathbf{x}^{q}}{\Delta t} = i\omega\left(\theta\mathbf{x}^{q+1} + (1-\theta)\mathbf{x}^{q}\right) \tag{2.27}$$

where θ is a parameter and $0 \leq \theta \leq 1$. The propagators are given as

$$
\begin{aligned}
f(\omega\Delta t;\theta) &= \left(\frac{1 + i\omega\dfrac{\Delta t}{\gamma}(1-\theta)}{1 - i\theta\omega\dfrac{\Delta t}{\gamma}}\right)^{\gamma} \\
g(\omega\Delta t;\theta) &= \left(\frac{1 + i\omega\Delta t(1-\theta)}{1 - i\theta\omega\Delta t}\right)
\end{aligned}
\tag{2.28}
$$

For the given scalar wave equation (2.26) the exact value of the propagator over Δt is $e^{i\omega\Delta t}$, so in theory we expect our propagators to have the norm equal to 1. Numerically speaking, the propagators are expected to be dissipative (the modulus of f and g can be less than one) and dispersive (the argument of f and g is different from $\omega\Delta t$) depending upon the numerical scheme and time-stepping used.

We check the behaviour of the two propagators in Fig. 2.3 where the norms of f and g are plotted as a function of $\omega\Delta t$ with the varying values of θ. The total time windows are $N = 10$ and $\gamma = 10$. Naturally the coarse propagator has

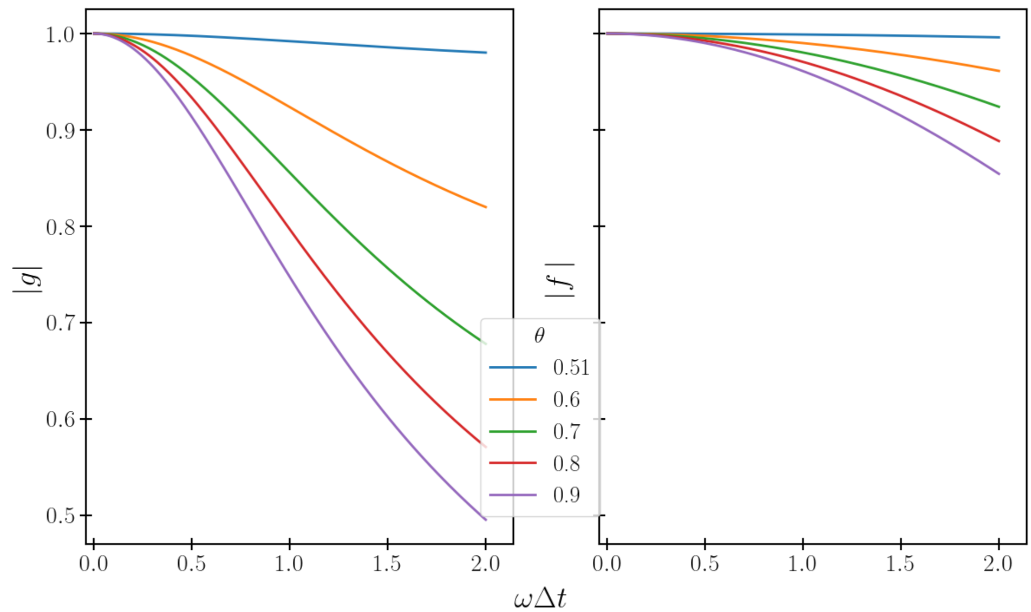


Figure 2.3: $|f|$ and $|g|$ as a function of $\omega\Delta t$ with varying θ; $N = 10, \gamma = 10$

a larger time-step and is more dissipative than the fine solver as can be observed in Fig. 2.3. We also notice that f and g become less and less dissipative when the value of θ is close to 0.5, the value for which we recover the Crank-Nicholson scheme which is known to be non-dissipative.

Next we plot the norm of the difference between the propagators $|f-g|$ for the same parameter values ($N=10, \gamma=10$) in Fig. 2.4. We see that $|f-g|$

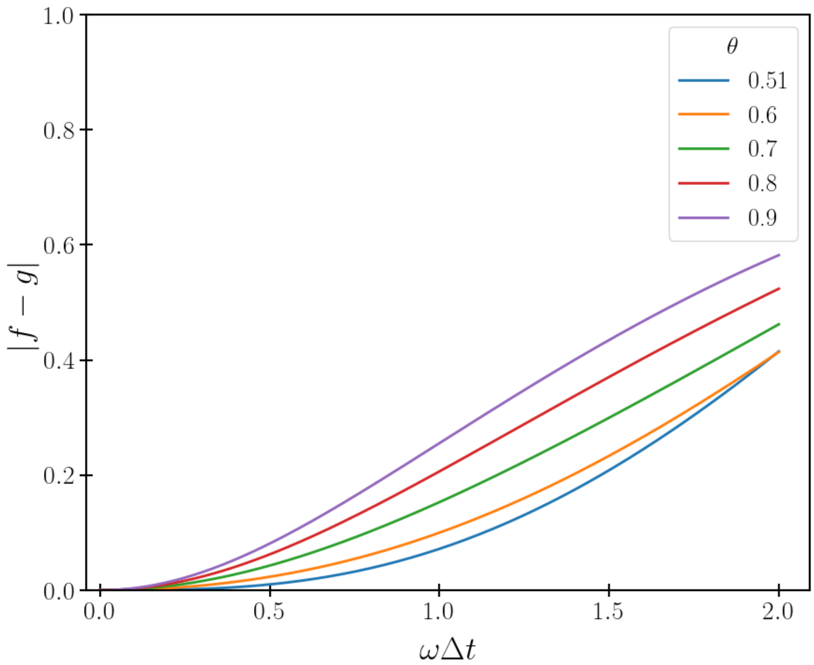


Figure 2.4: Norm of $f-g$ as a function of $\omega\Delta t$ with varying θ; $N=10, \gamma=10$

increases as θ goes to 1 and this indicates that the difference in the dissipation rates of the propagators start to vary sharply even at the relatively smaller scales. Thus $|f-g|$ will have the largest value if the backward Euler scheme is used for the propagators ($\theta=1$).

Note that the norm of $f-g$ and the norm of g play an important role in determining the error in the Parareal. From the expression (2.22), it can be seen that the error will decrease quickly with increasing k if $|f-g| \ll 1$ which simply means that the coarse solver is essentially the fine solver. In practice we need to wisely choose the ratio γ and the θ parameter. We want γ to be large enough so that we have a low-accuracy coarse solver and a small $|f-g|$ at the same time. Also we want a value of θ between 0.5 and 1 for which both $|f-g|$ and $|g|$ are small.

We consider two instances of the Parareal error for the given scalar wave equation (2.26). The errors are for the initial prediction made by the coarse solver ($k = 0$) and the first 4 Parareal iterations when the propagators are defined using $\theta = 0.55$ in Fig. 2.5 and $\theta = 0.8$ in Fig. 2.6. The left side of both the images shows the error for a coarse solver with low accuracy ($\gamma = 10$) while the right side shows the error for a coarse solver with high accuracy ($\gamma = 2$).

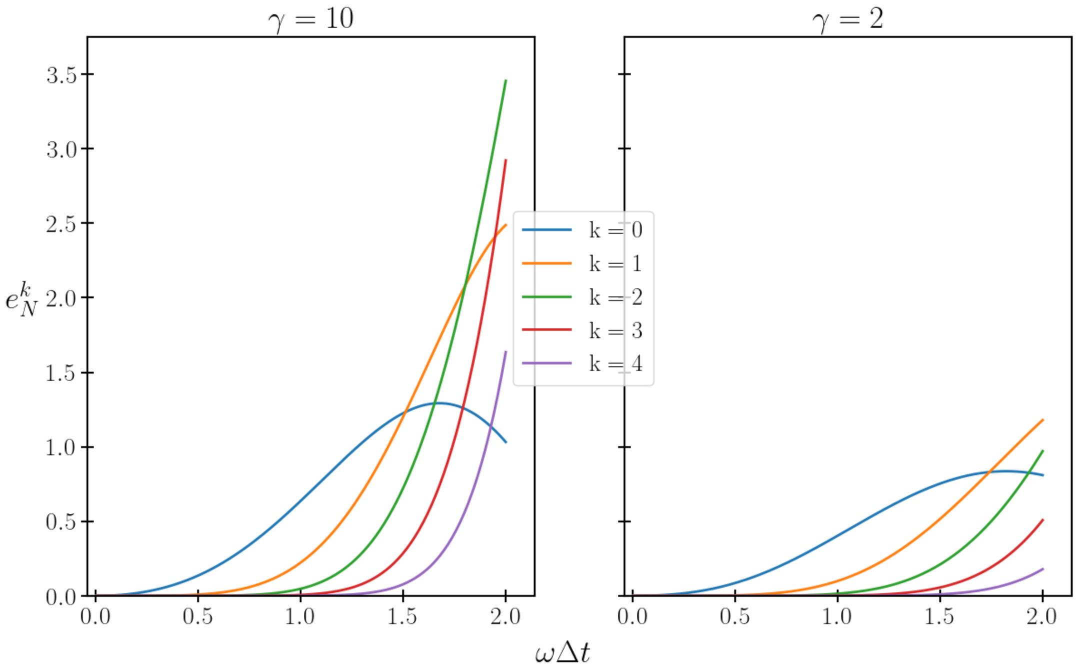


Figure 2.5: Parareal error e_N^k for successive iterates k as a function of $\omega\Delta t$ when $\theta = 0.55$

As expected, we see that the Parareal error is comparatively small for the large value of $\theta = 0.8$ (see Fig. 2.6) due to the implicit diffusion. We also see that for all Parareal iterations the error first follows a gradual increase, reaches a peak and then decreases. This is explained in [Gander and Vandewalle, 2007] that for an advection equation even though there is a superlinear convergence, the convergence factors of various one-step methods do not allow Parareal to converge any faster. To maintain the error to be less than 1 when γ is large (like Fig 2.5, $\gamma = 10$) we need to use a very small value of $\omega\Delta t$.

In the next subsection we are going to state the convergence result for the case of non-linear ODEs along with a numerical example.

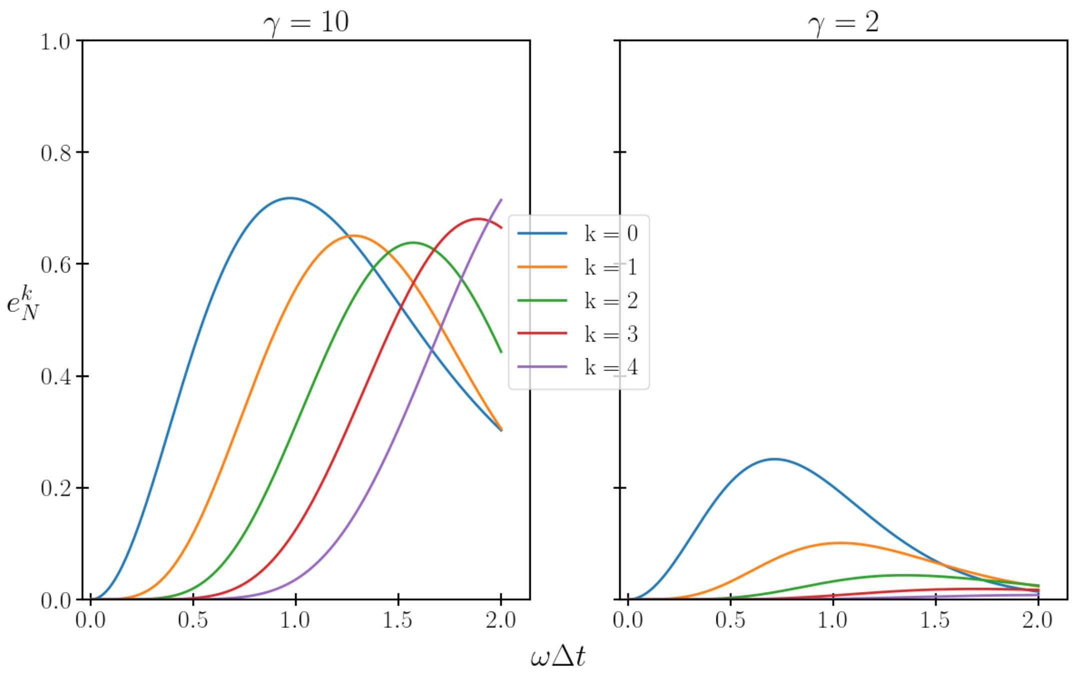


Figure 2.6: Parareal error e_N^k for successive iterates k as a function of $\omega\Delta t$ when $\theta = 0.8$

2.4.3 Non-linear convergence

The convergence study for a general non-linear system of ODE (2.9) was studied by Gander and Hairer [Gander and Hairer, 2008]. They showed that the Parareal exhibits a superlinear convergence on bounded intervals by using generating functions on an error recurrence relation. The following theorem states this result with the following assumptions

1. The difference between the approximate solution from G and the exact solution can be expanded for Δt small

$$F(t_{i+1}, t_i, \mathbf{x}) - G(t_{i+1}, t_i, \mathbf{x}) = c_{m+1}(\mathbf{x})\Delta t^{m+1} + c_{m+2}(\mathbf{x})\Delta t^{m+2} + \dots \quad (2.29)$$

2. G satisfies the Lipschitz condition

$$\|G(t_{i+1}, t_i, \mathbf{x}) - G(t_{i+1}, t_i, \mathbf{y})\| \leq (1 + C_2\Delta t)\|\mathbf{x} - \mathbf{y}\| \quad (2.30)$$

Theorem 2.2. *Let $F(t_{i+1}, t_i, \mathbf{x}_i^k)$ be the exact solution on the time subdomain Ω_i and let $G(t_{i+1}, t_i, \mathbf{x}_i^k)$ be an approximate solution with local truncation error* *bounded by* $C_3 \Delta t^{m+1}$ *and satisfying* (2.29) *where* c_j, $j = m+1, m+2, \dots$ *are continuously differentiable, and assume that* G *satisfies* (2.30)*. Then at the kth Parareal iteration we have*

$$\begin{aligned} \|\mathbf{x}(t_i) - \mathbf{x}_i^k\| &\leq \frac{C_3}{C_1} \frac{(C_1 \Delta t^{m+1})^{k+1}}{(k+1)!} (1 + C_2 \Delta t)^{i-k-1} \prod_{j=0}^{k} (i-j) \\ &\leq \frac{C_3}{C_1} \frac{(C_1 t_i)^{k+1}}{(k+1)!} \, e^{C_2 (t_i - t_{k+1})} \Delta t^{m(k+1)} \end{aligned} \tag{2.31}$$

The superlinear bound in (2.31) is evident from the presence of the $(k+1)!$ term in the denominator.

2.4.4 A numerical example

As an example for demonstrating the Parareal algorithm on a non-linear ODE we consider the Lorenz equations which were introduced by Edward Lorenz [Lorenz, 1963] in 1963 as a simplified convection model. These equations are frequently used for numerical weather prediction (NWP) experiments [Palmer, 1993] and they are a system of 3 non-linear ODEs given by

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= \sigma x + \sigma y, & x(0) &= x_0 \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= -xz + \rho x - y, & y(0) &= y_0 \\ \frac{\mathrm{d}z}{\mathrm{d}t} &= xy - \beta z, & z(0) &= z_0 \end{aligned} \tag{2.32}$$

where x, y, z are the state variables and σ, ρ, β are the physical parameters. For the specific parameter values of $\sigma = 10, \rho = 28, \beta = 8/3$ we get the famous Lorenz attractor which has the shape of a butterfly's wings. We use the explicit Euler scheme to construct our fine and the coarse propagator. The time period of integration is taken to be $[0, 5]$ with $N = 500$ time windows so that we have the coarse time step $\Delta t = 0.01$. The fine time step is taken as $\delta t = 0.001$ and the initial condition is chosen to be $(x_0, y_0, z_0) = (20, 5, -5)$.

Fig 2.7 shows the coarse initialisation and the first 8 iterations out of the total 18 Parareal iterations when a tolerance of $\epsilon_{\mathrm{p}} = 10^{-6}$ is used. The blue solid lines represent the reference solution while the yellow dots represent the Parareal

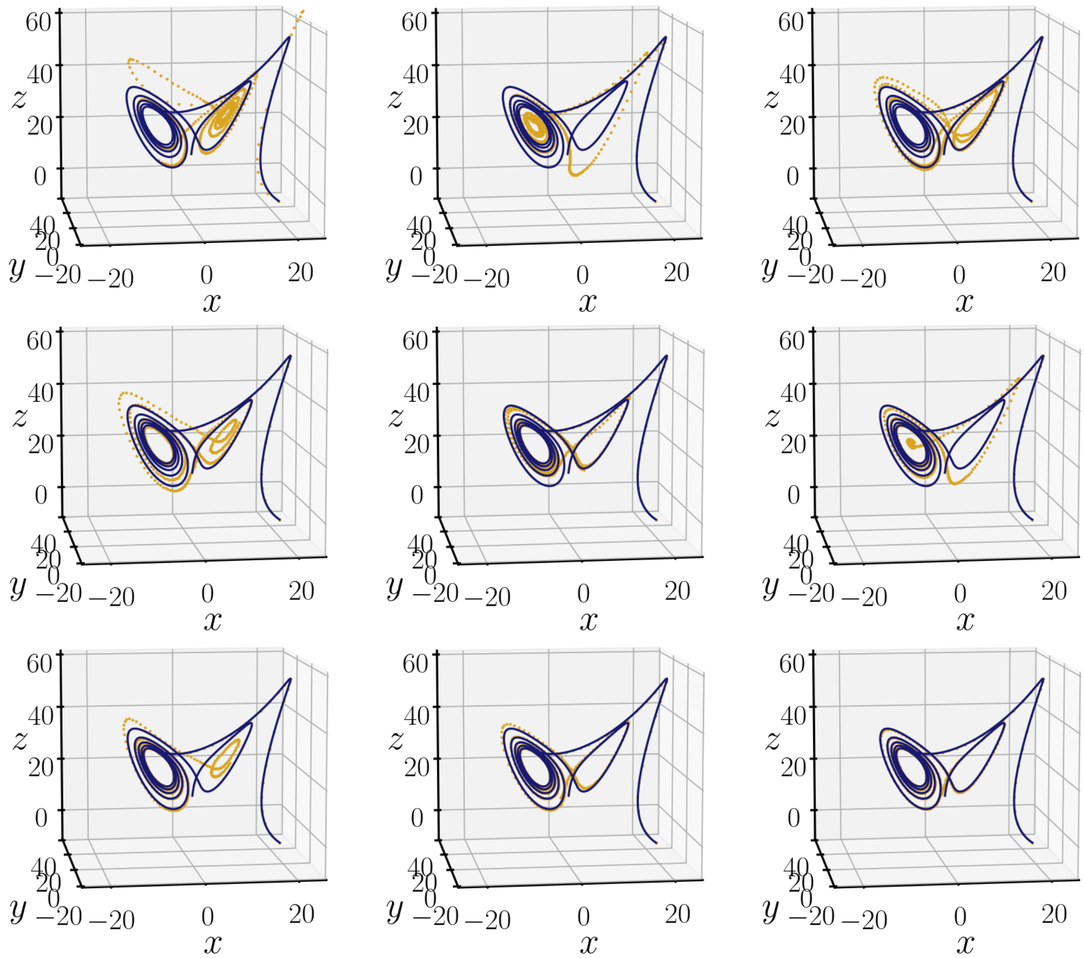


Figure 2.7: (From left to right) The coarse initialisation and first 8 Parareal iterations for the Lorenz model (2.32)

solution. We see in the first subfigure that the initial states obtained from the coarse solver are nowhere near the reference solution. It takes almost 7 Parareal iterations to correct and bring the solution closer to the reference solution.

Fig. 2.8 shows the maximum error one gets at the kth Parareal iteration ($k = 1, \cdots, 18$). The back dashed line is the Parareal tolerance ϵ_{p}. It can be seen that after the 6th iteration the error starts following a superlinear convergence as described in [Gander and Hairer, 2008] in the previous section. We have seen in this section that the Parareal error depends upon the term $f - g$ when solving scalar equations. In the next section we will do a more in depth analysis and illustrations of how the eigenvalues of $\mathbf{F} - \mathbf{G}$ affect the Parareal convergence when Parareal is used for a more general case of system of linear ODEs.

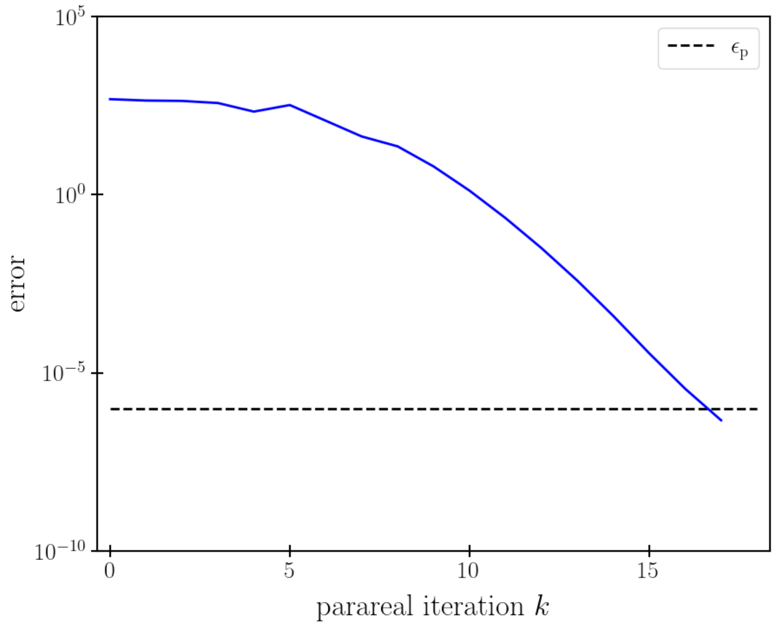


Figure 2.8: Error in the Lorenz model as a function of Parareal iterations

2.5 Analysis of the eigenvalues of $\mathbf{F} - \mathbf{G}$

Our objective in this section is to investigate how the eigenvalues of $\mathbf{F} - \mathbf{G}$ affect the Parareal convergence. Let us consider the following linear system

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{C}\mathbf{x} \tag{2.33}$$

where the matrix $\mathbf{C}$ comes from the spatial discretisation of a PDE and $\mathbf{x} \in \mathbb{R}^n$. We consider an implicit θ-scheme ($0 \leq \theta \leq 1$) for the temporal integration of (2.33)

$$\begin{aligned} \frac{\mathbf{x}^{q+1} - \mathbf{x}^q}{\Delta T} &= (1-\theta)\,\mathbf{C}\mathbf{x}^q + \theta\,\mathbf{C}\mathbf{x}^{q+1} \\ \implies [\mathbf{I} - \theta\Delta T\,\mathbf{C}]\mathbf{x}^{q+1} &= [\mathbf{I} + (1-\theta)\Delta T\,\mathbf{C}]\,\mathbf{x}^q \end{aligned} \tag{2.34}$$

and which could also be written as

$$\mathbf{x}^{q+1} = \mathbf{D}\,\mathbf{x}^q \tag{2.35}$$

with

$$\mathbf{D} = [\mathbf{I} - \theta\Delta T\,\mathbf{C}]^{-1}\,[\mathbf{I} + (1-\theta)\Delta T\,\mathbf{C}]\,. \tag{2.36}$$

We first note that $\mathbf{D}$ and $\mathbf{C}$ share the same eigenvectors. Indeed, if $(\mu_{\mathbf{C}}, \mathbf{x_C})$ is an eigenpair of $\mathbf{C}$ then

$$\begin{aligned}
\mathbf{D}\,\mathbf{x_C} &= [\mathbf{I} - \theta\Delta T\,\mathbf{C}]^{-1}\,[\mathbf{I} + (1-\theta)\Delta T\,\mathbf{C}]\,\mathbf{x_C} \\
&= [\mathbf{I} - \theta\Delta T\,\mathbf{C}]^{-1}\,[\mathbf{x_C} + (1-\theta)\Delta T\,\mathbf{C}\mathbf{x_C}] \\
&= [1 + (1-\theta)\Delta T\mu_{\mathbf{C}}]\,[\mathbf{I} - \theta\Delta T\mathbf{C}]^{-1}\,\mathbf{x_C} \\
&= [1 + (1-\theta)\Delta T\mu_{\mathbf{C}}]\,[1 - \theta\Delta T\mu_{\mathbf{C}}]^{-1}\,\mathbf{x_C} \\
&= \mu_{\mathbf{D}}\mathbf{x_C}
\end{aligned} \tag{2.37}$$

where we used the fact that if $\mu_{\mathbf{C}}$ is an eigenvalue of $\mathbf{C}$, then $(1 - \theta\Delta T\mu_{\mathbf{C}})$ is an eigenvalue of $(\mathbf{I} - \theta\Delta T\mathbf{C})$ and therefore,

$$(\mathbf{I} - \theta\Delta T\mathbf{C})^{-1}\mathbf{x_C} = \frac{1}{(1 - \theta\Delta T\mu_{\mathbf{C}})}\mathbf{x_C} \tag{2.38}$$

Thus $(\mu_{\mathbf{D}}, \mathbf{x_C})$ is an eigenpair of $\mathbf{D}$ with

$$\mu_{\mathbf{D}} = \frac{1 + (1-\theta)\Delta T\,\mu_{\mathbf{C}}}{1 - \theta\Delta T\mu_{\mathbf{C}}} \tag{2.39}$$

Let us suppose that the fine solver $\mathbf{F}$ and the coarse solver $\mathbf{G}$ will both use this θ-scheme with different time steps (and with the same spatial discretisation corresponding to $\mathbf{C}$). On a given time window of length ΔT, we assume that the fine solver performs N_{fine} time steps $\Delta t_{\mathbf{F}}$ and that the coarse solver performs N_{coarse} time steps $\Delta t_{\mathbf{G}}$. That is,

$$\Delta t_{\mathbf{F}} = \frac{\Delta T}{N_{\text{fine}}}, \quad \Delta t_{\mathbf{G}} = \frac{\Delta T}{N_{\text{coarse}}} \tag{2.40}$$

On the similar lines it can be shown that $\mathbf{C}$, $\mathbf{F}$ and $\mathbf{G}$ have the same eigenvectors. We have for the corresponding eigenpairs $(\mu_{\mathbf{F}}, \mathbf{x_C})$ and $(\mu_{\mathbf{G}}, \mathbf{x_C})$

$$\begin{aligned}
\mu_{\mathbf{F}} &= \frac{1 + (1-\theta)\Delta t_{\mathbf{F}}\,\mu_{\mathbf{C}}}{1 - \theta\Delta t_{\mathbf{F}}\,\mu_{\mathbf{C}}} \\
\mu_{\mathbf{G}} &= \frac{1 + (1-\theta)\Delta t_{\mathbf{G}}\,\mu_{\mathbf{C}}}{1 - \theta\Delta t_{\mathbf{G}}\,\mu_{\mathbf{C}}} = \frac{1 + (1-\theta)\left(\frac{N_{\text{fine}}}{N_{\text{coarse}}}\right)\Delta t_{\mathbf{F}}\,\mu_{\mathbf{C}}}{1 - \theta\left(\frac{N_{\text{fine}}}{N_{\text{coarse}}}\right)\Delta t_{\mathbf{F}}\,\mu_{\mathbf{C}}}.
\end{aligned} \tag{2.41}$$

What we would like to estimate is the norm of the matrix

$$\mathbf{Z} = \mathbf{F}^{N_{\text{fine}}} - \mathbf{G}^{N_{\text{coarse}}} \tag{2.42}$$

which can be done by considering the maximum of the modulus of the eigenvalues of $\mathbf{Z}$. Matrix $\mathbf{Z}$ is the difference between the fine and the coarse solver at the end of a time window. The eigenvalues of $\mathbf{Z}$ are given by

$$\mu_{\mathbf{Z}} = \mu_{\mathbf{F}}^{N_{\text{fine}}} - \mu_{\mathbf{G}}^{N_{\text{coarse}}}. \tag{2.43}$$

In our experiments, the time step of the fine solver $\Delta t_{\mathbf{F}}$ is defined according to the following criterion

$$\Delta t_{\mathbf{F}} \max |\mu_{\mathbf{C}}| \leq \omega_{\max} \tag{2.44}$$

for some constant $\omega_{\max}$. It implies that for a given eigenvalue $\mu_{\mathbf{C}}$ of the matrix $\mathbf{C}$ written in the exponential form we have

$$\Delta t_{\mathbf{F}} \mu_{\mathbf{C}} = r\mathrm{e}^{i\beta} \tag{2.45}$$

with modulus $0 \leq r \leq \omega_{\max}$ and polar angle β. Moreover, we assume that the eigenvalues of $\mathbf{C}$ have a negative real part i.e. $\Re(\mu_{\mathbf{C}}) \leq 0$ and so $\beta \in [\pi/2, 3\pi/2]$. And due to symmetry when computing the modulus, it is sufficient to consider $\beta \in [\pi/2, \pi]$. To explain why we take $\Re(\mu_{\mathbf{C}}) \leq 0$ let us consider the eigenvector $\mathbf{x_C}$ of $\mathbf{C}$ corresponding to the eigenvalue $\mu_{\mathbf{C}}$. We have

$$\frac{\mathrm{d}\mathbf{x_C}}{\mathrm{d}t} = \mathbf{C}\,\mathbf{x_C} = \mu_{\mathbf{C}}\,\mathbf{x_C}. \tag{2.46}$$

The solution to the above scalar equation is $\mathbf{x}(t) = \mathbf{x}(0)\,\mathrm{e}^{\mu_{\mathbf{C}} t}$ which implies that $|\mathbf{x}(t)| = |\mathbf{x}(0)|\,\mathrm{e}^{\Re(\mu_{\mathbf{C}})t}$. Now for any kind of discretisation method to be stable and for the solution to not blow up, we must have the condition that $\Re(\mu_{\mathbf{C}}) \leq 0$ (Fig. 2.9). In this case there are two situations when we have a wave propagation problem:

For the pure wave propagation problem (which is a hyperbolic case), all the eigenvalues of the matrix $\mathbf{C}$ are purely imaginary i.e. $\Re(\mu_{\mathbf{C}}) = 0$ or $\beta = \pi/2$. For the case when the problem is slightly parabolic (which has implicit dissipation), $\Re(\mu_{\mathbf{C}}) = -\sigma^2$ for some $\sigma \in \mathbb{R}$ which means $\Re(\mu_{\mathbf{C}}) < 0$.

Now for given values of $\omega_{\max}, N_{\text{fine}}, N_{\text{coarse}}, \theta$, the maximum of the modulus of $\mu_{\mathbf{Z}}$ can be obtained by

$$\max |\mu_{\mathbf{Z}}| = \max_{r \in [0, \omega_{\max}],\, \beta \in [\pi/2, \pi]} |\mu_{\mathbf{F}}^{N_{\text{fine}}} - \mu_{\mathbf{G}}^{N_{\text{coarse}}}| \tag{2.47}$$

which can be found numerically.

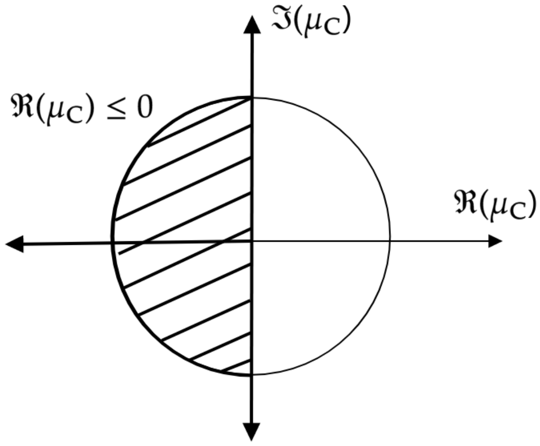


Figure 2.9: The shaded region of consideration where $\Re(\mu_{\mathbf{C}}) \leq 0$

Let γ be the ratio of the number of fine steps per time window to the number of coarse steps per time window i.e. $\gamma = N_{\text{fine}}/N_{\text{coarse}}$. We consider 3 instances of the contour plot of $|\mu_{\mathbf{Z}}|$ with different combinations of the values of $\omega_{\max}$, N_{fine}, N_{coarse}, and θ.

Plot 1

For the first example we consider the values $\omega_{\max} = 0.6, N_{\text{fine}} = 20, N_{\text{coarse}} = 4$ and $\theta = 0.6$. We get $\max|\mu_{\mathbf{Z}}| = 0.9865\ldots$ for $r = 0.48$ and $\beta = \pi/2$. This is illustrated in Fig. 2.10.

Plot 2

As a second example shown in Fig. 2.11, for another set of values $\omega_{\max} = 0.6, N_{\text{fine}} = 20, N_{\text{coarse}} = 2$ and $\theta = 0.8$ we get $\max|\mu_{\mathbf{Z}}| = 0.790\ldots$ for $r = 0.255$ and again at $\beta = \pi/2$.

Plot 3

Changing the value of θ greatly impacts the maximum eigenvalue. For another set of values $\omega_{\max} = 0.6, N_{\text{fine}} = 20, N_{\text{coarse}} = 2$ and $\theta = 0.51$, we get $\max|\mu_{\mathbf{Z}}| = 1.912\ldots$ for $r = 0.376$ and at $\beta = \pi/2$. This result is shown in Fig. 2.12.

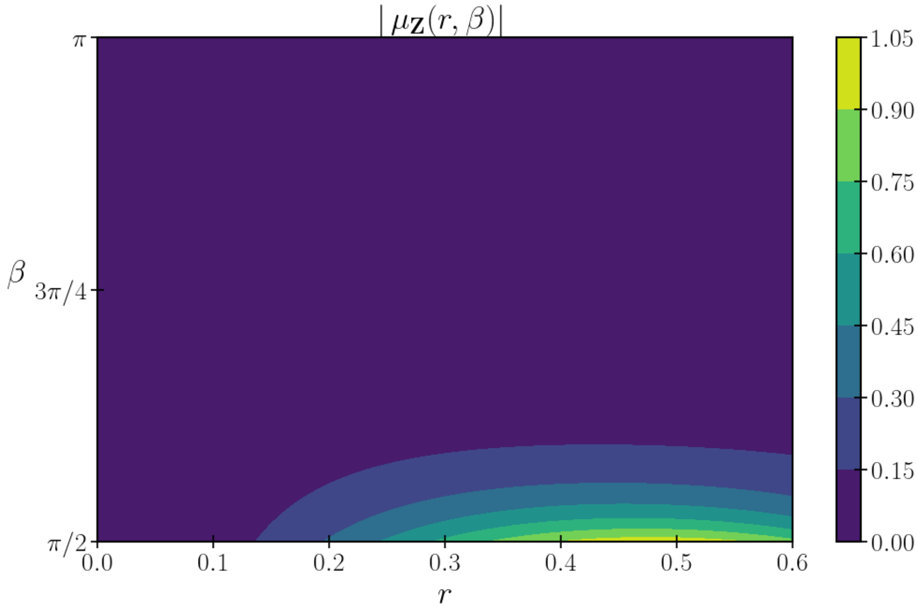


Figure 2.10: Contour plot of the modulus of the eigenvalues of $\mathbf{Z}$ for $\omega_{\max} = 0.6, \gamma = 5, \theta = 0.6$

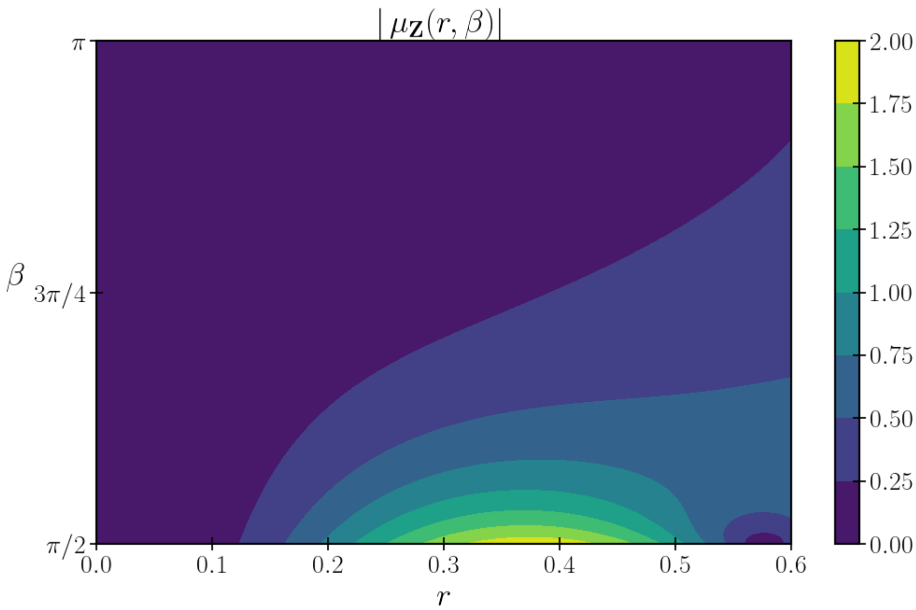


Figure 2.12: Contour plot of the modulus of the eigenvalues of $\mathbf{Z}$ for $\omega_{\max} = 0.6, \gamma = 10, \theta = 0.51$

It seems that, even if we did not try to prove it, the maximum for $|\mu_{\mathbf{Z}}|$ is always attained at $\beta = \pi/2$, which corresponds to a purely imaginary eigenvalue

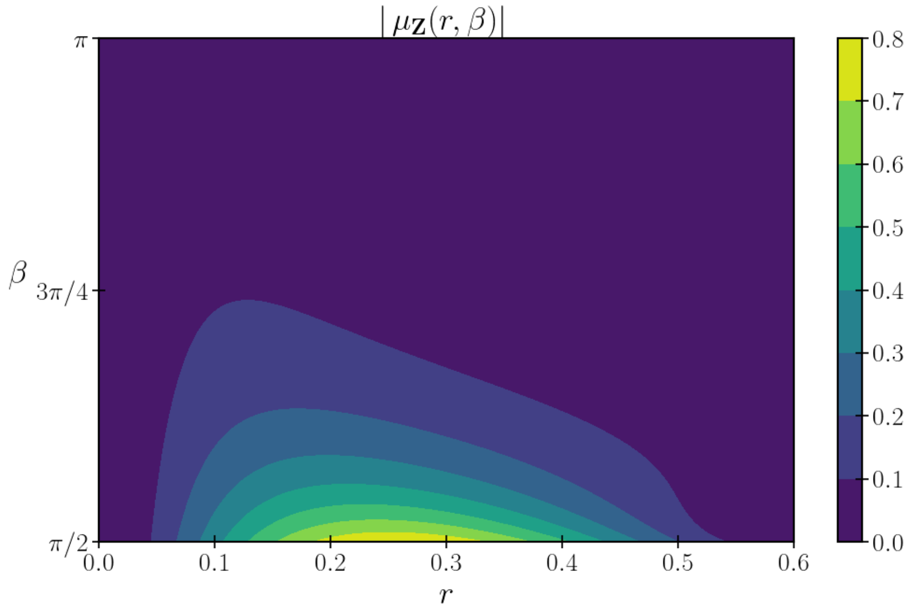


Figure 2.11: Contour plot of the modulus of the eigenvalues of $\mathbf{Z}$ for $\omega_{\max} = 0.6, \gamma = 10, \theta = 0.8$

of $\mathbf{Z}$. For example we see from Plot 3 that $\max|\mu_{\mathbf{Z}}| \approx 2$ for $\mu_{\mathbf{Z}} = 0.376\, i$ which is the "most dangerous" eigenvalue impeding the convergence of the algorithm. In practice, we would like to see how the error projects on the most dangerous eigenvalues. We have for $\mathbf{x_C}$

$$(\mathbf{F}-\mathbf{G})\mathbf{x_C} = \mu_{\mathbf{Z}}\mathbf{x_C} \tag{2.48}$$

If $\{(\mathbf{x}^p_{\mathbf{C}}, \mu^p_{\mathbf{Z}})\}_{p=1,\dots,n}$ is the set of all the eigenpairs of $\mathbf{C}$, then for any vector $\mathbf{x}$

$$(\mathbf{F}-\mathbf{G})\mathbf{x} = \sum_{p=1}^{n} \alpha_p\, \mu^p_{\mathbf{Z}}\, \mathbf{x}^p_{\mathbf{C}} \tag{2.49}$$

and if the eigenvectors are normalised then

$$\|(\mathbf{F}-\mathbf{G})\mathbf{x}\|^2 = \sum_{p=1}^{n} \alpha_p^2\, (\mu^p_{\mathbf{Z}})^2 \tag{2.50}$$

What remains is to find the weights α_p which correspond to the amplitudes of the projection of the Parareal error on the eigenvectors associated with the "dangerous" eigenvalues.

Moreover it is well known that the potentially large differences between coarse and fine solver are also due to the dispersion error. Looking at (2.41), we can write

$$\mu_{\mathbf{F}} = \mu_{\mathbf{F}}(r,\beta) = r_1 e^{i\beta_1}, \qquad \mu_{\mathbf{G}} = \mu_{\mathbf{G}}(r,\beta) = r_2 e^{i\beta_2} \tag{2.51}$$

for $0 \leq r \leq \omega_{\max}$, $\pi/2 \leq \beta \leq \pi$ and $r_1, r_2, \beta_1, \beta_2 \in \mathbb{R}$. In addition to the plots of $\mu_{\mathbf{Z}}$, we also present the individual plots for the modulus and the relative arguments of $\mu_{\mathbf{F}}^{N_{\text{fine}}}$ and $\mu_{\mathbf{G}}^{N_{\text{coarse}}}$ to clearly see the difference in terms of dissipation and dispersion.

Fig. 2.13 compares the modulus of $\mu_{\mathbf{F}}^{N_{\text{fine}}}$ (on the left) with the modulus of $\mu_{\mathbf{G}}^{N_{\text{coarse}}}$ (on the right) for given values of $\omega_{\max} = 0.6, N_{\text{fine}} = 20, N_{\text{coarse}} = 4$ and $\theta = 0.6$. The next figure (Fig. 2.14) compares the relative argument of $\mu_{\mathbf{F}}^{N_{\text{fine}}}$ (on the left) and the relative argument of $\mu_{\mathbf{G}}^{N_{\text{coarse}}}$ (on the right) with respect to the argument of the exact propagator. Note that for the ODE

$$\frac{\mathrm{d}\mathbf{x}_{\mathbf{C}}}{\mathrm{d}t} = \mu_{\mathbf{C}} \mathbf{x}_{\mathbf{C}} \tag{2.52}$$

the exact propagator is given as

$$e^{\mu_{\mathbf{C}} \Delta t} = \exp\{N_{\text{fine}} \times \Delta t_{\mathbf{F}} \mu_{\mathbf{C}}\} = \exp\{N_{\text{fine}} \times r e^{i\beta}\}. \tag{2.53}$$

The analysis of $\mu_{\mathbf{Z}}$ has been done when the original Parareal version is used for constructing the fine and coarse solver. There are several ways to which the Parareal can be modified to accelerate the convergence and this is a topic of further discussion in section 2.7.

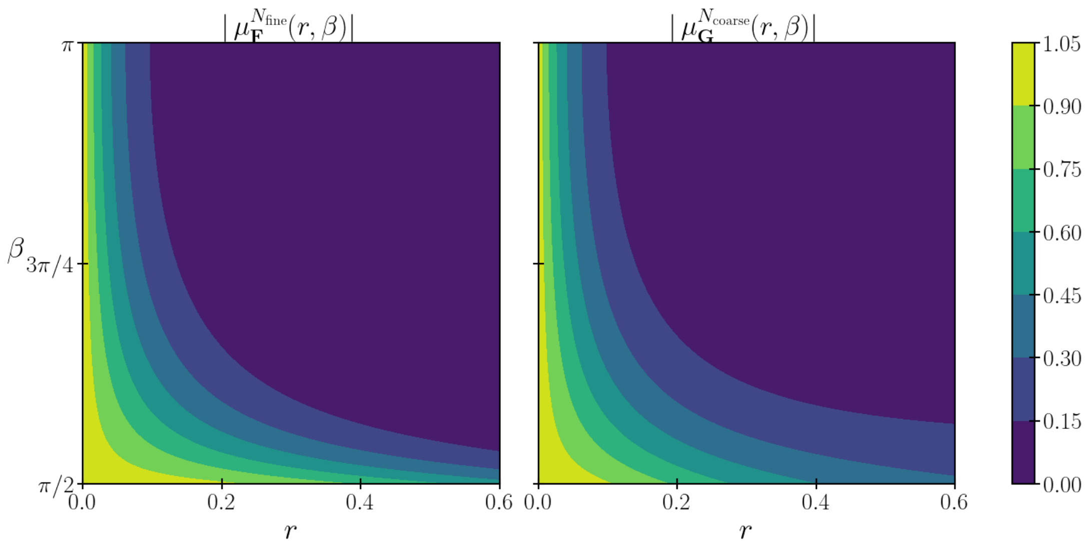


Figure 2.13: Comparison of the contour plots of the modulus of the eigenvalues of $\mathbf{F}$ and $\mathbf{G}$ for $\omega_{\max} = 0.6, \gamma = 5, \theta = 0.6$

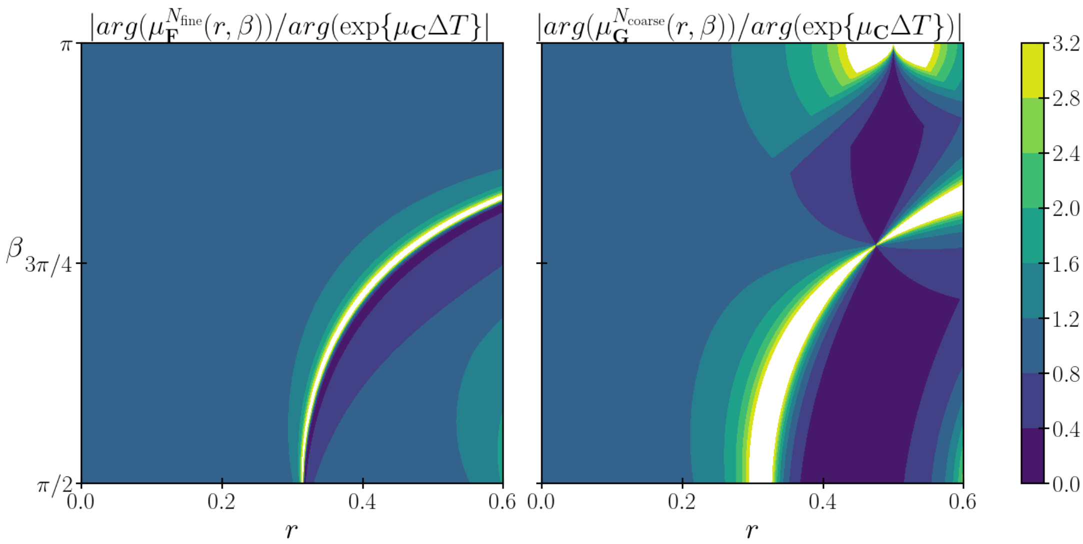


Figure 2.14: Comparison of the contour plots of the relative arguments of the eigenvalues of **F** and **G** with respect to the argument of the exact propagator for $\omega_{\max} = 0.6, \gamma = 5, \theta = 0.6$

2.6 Speedup and Efficiency

In this section we derive the theoretical speedup and efficiency which can be achieved by using the Parareal algorithm. We assume that the we have N number of cores available so that each core can be assigned to one time window. We also assume that the communication time during the prediction step between any two cores is so small that it can be ignored. The theoretical speedup S can be defined as the ratio of the time it takes to perform a task sequentially to the time it takes to do the same task in parallel. In other words,

$$S = \frac{\text{Serial computation time}}{\text{Parallel computation time}} \tag{2.54}$$

To derive the speedup for the Parareal algorithm we follow along the lines of [Minion, 2011]. We begin by defining N_{fine} and N_{coarse} to be the number of fine and coarse time steps per time window respectively. Let τ_F and τ_G be the time taken by a core to perform a single step of the numerical scheme for the fine propagator F and for the coarse propagator G respectively. Thus for a single time window, $N_{\text{fine}}\tau_F$ is the total cost of F and $N_{\text{coarse}}\tau_G$ is the total cost of G.

To simplify notations, let $\gamma_F = N_{\text{fine}}\tau_F$ and $\gamma_G = N_{\text{coarse}}\tau_G$.

Over the whole period of integration, the cost of the serial computation is the time taken by the fine solver to solve the problem sequentially which is $NN_{\text{fine}}\tau_F = N\gamma_F$. The initial configuration for the Parareal is fed through a serial coarse integration, it is given by $N\gamma_G$. Now one iteration of Parareal involves a prediction by G which is again sequential and the correction step in parallel. The total cost per Parareal iteration is given as $N\gamma_G + \gamma_F$. Therefore for k Parareal iterations,

$$\begin{aligned} S &= \frac{N\gamma_F}{N\gamma_G + k(N\gamma_G + \gamma_F)} \\ &= \frac{1}{\dfrac{\gamma_G}{\gamma_F} + k\left(\dfrac{\gamma_G}{\gamma_F} + \dfrac{1}{N}\right)}. \end{aligned} \tag{2.55}$$

Looking at (2.55) some upper bounds can be derived for the speedup [Ruprecht and Krause, 2012]

$$S \leq \frac{N}{k}, \qquad S \leq \frac{\gamma_F}{k\,\gamma_G}. \tag{2.56}$$

From the second bound, we see that in order to get some speedup it is required that

$$\frac{\gamma_F}{k\,\gamma_G} > 1 \quad \Longrightarrow \quad \gamma_G < \frac{\gamma_F}{k}. \tag{2.57}$$

The two bounds compete with each other in the sense that the coarse propagator should be really cheap and fast in order to get a good speedup according to the second bound but at the same time if the coarse approximation is too accurate there will be more Parareal iterations which will reduce the speedup according to the first bound. Ideally we would like the Parareal iterations to be as few as possible ($k \ll N$) if not equal to 1. Thus the optimal speedup must balance the two bounds in the best possible way. The Parareal efficiency is given as

$$E = \frac{\text{Parareal speedup}}{\text{Number of cores used}} = \frac{S}{N}. \tag{2.58}$$

Using (2.55) we get

$$E = \frac{1}{\left[N\dfrac{\gamma_G}{\gamma_F}(k+1)\right] + k}. \tag{2.59}$$

Clearly, the Parareal's efficiency is bounded by $1/k$. Since k is usually at least 2 this means that Parareal is already at most 50% efficient by the end of the second iteration. For a more rigorous explanation about the speedup and efficiency of the Parareal see [Bal, 2003, Maday, 2008].

2.7 Typical problems with Parareal and modifications

From our discussion in section 2.4 we saw that Parareal provides nice convergence results with parabolic problems or highly dissipative systems. But we also saw that the Parareal algorithm faces instability issues when it comes to solving hyperbolic systems [Farhat and Chandesris, 2003, Bal, 2005, Staff and Rønquist, 2005, Eghbal et al., 2017]. We list some of the studies which have been done to identify the causes of the algorithm's inability to be attractive for hyperbolic problems.

In particular Gander and Vandewalle [Gander and Vandewalle, 2007] showed that for the system of ODEs with purely imaginary eigenvalues there is a degradation in the convergence which is achieved only when the fine solver has been computed through sequentially. A similar issue of dominant imaginary eigenvalues causes numerical instability when Parareal is used to solve the non-linear Navier Stokes equation with a high Reynolds number [Steiner et al., 2015]. Dai and Maday [Dai and Maday, 2013] reported for the hyperbolic systems that it is the regularity of the initial condition and the preservation of the invariant quantities which affect the convergence. Ruprecht [Ruprecht, 2018] did an analysis on how the Parareal propagates wave like solutions and triggers instabilities by considering a semi-discrete dispersion relation for a one dimensional linear advection-diffusion equation. The instability is caused by the difference in the phase speed in high frequency modes due to different discretisations of the fine and coarse propagators. There have been several approaches to modify Parareal and make it more attractive for tackling different types of problems which we list below:

- **Coarse solver's accuracy and cost**: To improve the cost and accuracy of the coarse solver in particular for the hyperbolic problems, the PITA algorithm [Farhat and Chandesris, 2003] was proposed and later analysed as the Krylov subspace enhanced method [Gander and Petcu, 2008] which builds the coarse propagator by reusing all the information from the previously computed fine evaluations. A coarse solver based on a reduced basis approximation in space was utilised in [He, 2010, Chen et al., 2014] to accelerate the performance of Parareal.

- **Fine solver's cost**: An adaptive Parareal where the fine solver runs adaptively depending on the minimum required accuracy was presented in [Maday and Mula, 2020].

- **Computational implementation**: Aubanel [Aubanel, 2011] presented a detailed study of the scheduling of tasks within Parareal and showed better efficiency results than the original algorithm. The idea of a pipelined Parareal implementation [Minion, 2011] and an event-based implementation [Berry et al., 2012] was studied to further improve and optimise Parareal's speedup and efficiency. The cost of Parareal can also improved by using a reduced spatial resolution within the coarse propagator and using interpolation to match the solution with the fine solver in the correction step [Ruprecht, 2014]. The update iteration then becomes

$$\mathbf{x}_{i+1}^{k+1} = IG(t_{i+1}, t_i, R\mathbf{x}_i^{k+1}) + F(t_{i+1}, t_i, \mathbf{x}_i^k) - IG(t_{i+1}, t_i, R\mathbf{x}_i^k) \tag{2.60}$$

  where I and R denote the interpolation and the restriction operator respectively. The convergence of Parareal then depends on the order of the interpolation used.

In particular we are going to discuss the PITA algorithm, the Krylov subspace enhanced Parareal and the adaptive Parareal in more detail in the next subsections.

2.7.1 PITA algorithm

The parallel implicit time-integrator or PITA was introduced as a variant of the Parareal algorithm where the correction procedure is carried out by Newton-like iterations on a coarse time-grid [Farhat and Chandesris, 2003]. We consider a linear system of first order ODE

$$\mathbf{x}'(t) = \mathbf{D}\,\mathbf{x}(t) + \mathbf{d}(t) \tag{2.61}$$

with $\mathbf{D} \in \mathbb{R}^n \times \mathbb{R}^n$ and $\mathbf{d} \in \mathbb{R}^n$. As usual the time domain $[0, T]$ is discretised into N subintervals Ω_i of size Δt. Let δt be the fine time-step with γ partitions of the each subinterval this way obtaining a fine grid. The PITA algorithm is described in the following steps.

- Step 0: Provide initial values $\widetilde{\mathbf{x}}_i^0, i = 0, \cdots N$ by solving the model (2.61) using a sequential iterative time-integration algorithm (ITA) on the coarse grid.

For $k = 0, 1, \cdots$

- Step 1: Using the updated values as initial conditions, solve on each subinterval Ω_i the following IVP by applying ITA on a fine grid

$$\begin{aligned} (\mathbf{x}_i^k)'(t) &= \mathbf{D}\,\mathbf{x}_i^k(t) + \mathbf{d}(t) \quad \text{on } \Omega_i \\ \mathbf{x}_i^k(t_i) &= \widetilde{\mathbf{x}}_i^k \end{aligned} \tag{2.62}$$

  Note that all these computations are done in parallel.

- Step 2: Calculate the jumps

$$\mathbf{s}_i^k = \mathbf{x}_{i-1}^k(t_i) - \widetilde{\mathbf{x}}_i^k, \quad 1 \leq i \leq N \tag{2.63}$$

  on the coarse grid. If the jumps are small enough, the algorithm terminates.

- Step 3: Propagate the jumps by applying ITA on the coarse grid to the correction problem

$$(\mathbf{c}_i^k)'(t) = \mathbf{D}\,\mathbf{x}_i^k(t), \quad \mathbf{c}_0^k(t_0) = 0, \quad \mathbf{c}_i^k(t_i) = \mathbf{c}_{i-1}^k(t_i) + \mathbf{s}_i^k \tag{2.64}$$

  and compute the correction $\widetilde{\mathbf{c}}_i^k = \mathbf{c}_{i-1}^k(t_i)$.

- Step 4: Update the approximate solution

$$\begin{aligned} \widetilde{\mathbf{x}}_i^{k+1} &= \mathbf{x}_{i-1}^k(t_i) + \widetilde{\mathbf{x}}_i^k \\ &= \widetilde{\mathbf{x}}_i^k + \mathbf{s}_i^k + \widetilde{\mathbf{x}}_i^k, \quad 1 \leq i \leq N \end{aligned} \tag{2.65}$$

Remark 2.4. *The PITA algorithm and Parareal are equivalent for the linear problems [Gander and Petcu, 2008].*

Just like Parareal, PITA was shown to give good speedup for first order ODEs but for the second-order structural dynamics problem PITA fell short due to the beating phenomenon [Farhat and Chandesris, 2003]. A modification to PITA was made in [Farhat et al., 2006] as a remedy to this problem which also works well for the linear second order hyperbolic systems in structural dynamics. It was later extended and applied to the non-linear systems in [Cortial and Farhat, 2009]. The next subsection talks about the Krylov subspace enhanced Parareal algorithm which is based on the idea of PITA.

2.7.2 Krylov subspace enhanced Parareal

Notice that in the original Parareal method, all the fine evaluations are only used as correction terms. To improve convergence issues, Gander and Petcu [Gander and Petcu, 2008] used the idea from the modified PITA in [Farhat et al., 2006] to improve the coarse solver for linear first order hyperbolic equations and second order ODEs. They proposed to project the coarse solver solution onto a subspace spanned by all the previous and current snapshots of the fine evaluations at each subinterval. We explain the method following [Chen et al., 2014, Ruprecht and Krause, 2012]. We denote the space as $\mathcal{S}^k$ and it is given by

$$\mathcal{S}^k = \operatorname{span}\left\{\mathbf{x}_i^l : i = 0, \cdots, N-1; \quad l = 0, \cdots k\right\} \tag{2.66}$$

A basis for $\mathcal{S}^k$ (say $\mathbf{S}^k = \{\mathbf{s}_1, \cdots, \mathbf{s}_r\}$) can be found by using the Gram-Schmidt process. If we let $\mathbf{P}^k$ to be the projection matrix at Parareal iteration k then we can write $\mathbf{P}^k = (\mathbf{S}^k)^T\mathbf{S}^k$ and the improved coarse solver $\mathbf{K}$ can be written as

$$\mathbf{K}(t_{i+1}, t_i, \mathbf{x}_i^k) = \mathbf{F}(t_{i+1}, t_i, \mathbf{P}^k\mathbf{x}_i^k) + \mathbf{G}(t_{i+1}, t_i, (\mathbf{I} - \mathbf{P}^k)\mathbf{x}_i^k) \tag{2.67}$$

The modified coarse solver consists of two parts where the part of the solution which lies on the projection is propagated by the fine solver while the rest (projection error) is done by the coarse solver. For linear problems the fine solver part can be computed as

$$\mathbf{F}(t_{i+1}, t_i, \mathbf{P}^k\mathbf{x}_i^k) = \mathbf{F}\left(t_{i+1}, t_i, \sum_{p=1}^{r} c_p\mathbf{s}_p\right) = \sum_{p=1}^{r} c_p\mathbf{F}(t_{i+1}, t_i, \mathbf{s}_p) \tag{2.68}$$

where $\mathbf{F}(t_{i+1}, t_i, \mathbf{s}_p)$ are known from the space $\mathbf{F}(\mathcal{S}^k)$ spanned by applying the fine propagator on $\mathcal{S}^k$. The modified Parareal correction procedure looks like

$$\begin{aligned}
\mathbf{x}_{i+1}^{k+1} &= \mathbf{F}(t_{i+1}, t_i, \mathbf{x}_i^k) + \mathbf{K}(t_{i+1}, t_i, \mathbf{x}_i^{k+1} - \mathbf{x}_i^k) \\
&= \mathbf{F}(t_{i+1}, t_i, \mathbf{x}_i^k) + \mathbf{F}(t_{i+1}, t_i, \mathbf{P}^k(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k)) \\
&\qquad + \mathbf{G}(t_{i+1}, t_i, (\mathbf{I} - \mathbf{P}^k)(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k)) \\
&= \mathbf{F}(t_{i+1}, t_i, \mathbf{P}^k\mathbf{x}_i^{k+1}) + \mathbf{G}(t_{i+1}, t_i, (\mathbf{I} - \mathbf{P}^k)\,\mathbf{x}_i^{k+1})
\end{aligned} \tag{2.69}$$

where we have used the fact that $\mathbf{P}^k\mathbf{x}_i^k = \mathbf{x}_i^k$.

Algorithm 5 describes the required steps to implement the Krylov enhanced subspace method. We see that to run the Krylov subspace enhanced version no extra manipulations are required as the fine evaluations are already available. The only cost is to compute the projection matrix by singular value decomposition or a Gram-Schmidt procedure. Since all vectors are stored at each Parareal iteration, the size of the subspace increases by the factor of the number of the time windows and the size of the problem. For a large scale problem this might become computationally inefficient. It is a topic which is further discussed in chapter 4.

Algorithm 5 KRYLOV-ENHANCED PARAREAL ALGORITHM

Initialisation:
$\mathbf{x}_0^0 = \mathbf{x}_0$
for $i = 0$ to N **do**
  $\mathbf{x}_{i+1}^0 = \mathbf{G}(t_{i+1}, t_i, \mathbf{x}_i^0)$
end for
Iterations:
$k = 0$
repeat
  Parallel fine integrations:
  for $i = 0$ to N **do**
    $\tilde{\mathbf{x}}_{i+1}^k = \mathbf{F}(t_{i+1}, t_i, \mathbf{x}_i^k)$
  end for
  Sequential correction step:
  Update $\mathbf{S}^{k-1} \rightarrow \mathbf{S}^k$, $\mathbf{F}(\mathbf{S}^k)$ and $\mathbf{P}^k$
  for $i = 0$ to N **do**
    $\mathbf{x}_{i+1}^{k+1} = \mathbf{K}(t_{i+1}, t_i, \mathbf{x}_i^{k+1}) + \tilde{\mathbf{x}}_{i+1}^k - \mathbf{K}(t_{i+1}, t_i, \mathbf{x}_i^k)$
  end for
  $k = k + 1$
until $k = k_{\max}$ or $\max\limits_{1 \le i \le N} |\mathbf{x}^{k+1} - \mathbf{x}^k| < \epsilon_{\mathrm{p}}$

2.7.3 An adaptive Parareal

In the classical Parareal algorithm, the accuracy of the fine solver is fixed across all Parareal iterations and is generally taken to be that of the sequential solver. It is one of the major obstacles for achieving higher parallel efficiency. A way to improve the Parareal's performance is by adapting the fine solver accuracy depending upon the Parareal iteration [Maday and Mula, 2020].

Let $E : [0, T] \times [0, T] \times \mathbb{R}^n \to \mathbb{R}^n$ be the exact propagator. Then the idealised Parareal version using the exact propagator looks like

$$\begin{aligned} \mathbf{x}_{i+1}^0 &= G(t_i, t_{i+1}, \mathbf{x}_i^0) \\ \mathbf{x}_{i+1}^{k+1} &= G(t_i, t_{i+1}, \mathbf{x}_i^{k+1}) + E(t_i, t_{i+1}, \mathbf{x}_i^k) - G(t_i, t_{i+1}, \mathbf{x}_i^k) \end{aligned} \tag{2.70}$$

This ideal version may not be implementable because of the exact propagator E. Any approximation to $E(t, s, \mathbf{x}) \in \mathbb{R}^n$ can be done by using an approximate solver $[E(t, s, \mathbf{x}); \zeta_i^k)]$ where ζ_i^k is some accuracy which needs to be chosen carefully at each iteration k. A feasible instance of the above algorithm is to replace the exact propagator E in (2.70) by $[E(t_i, t_{i+1}, \mathbf{x}_i^k); \zeta_i^k)]$ which leads to

$$\begin{aligned} \mathbf{x}_{i+1}^0 &= G(t_i, t_{i+1}, \mathbf{x}_i^0) \\ \mathbf{x}_{i+1}^{k+1} &= G(t_i, t_{i+1}, \mathbf{x}_i^{k+1}) + [E(t_i, t_{i+1}, \mathbf{x}_i^k); \zeta_i^k] - G(t_i, t_{i+1}, \mathbf{x}_i^k) \end{aligned} \tag{2.71}$$

Maday and Mula [Maday and Mula, 2020, §2.3] have shown that the minimum accuracy ζ_i^k required at each iteration k to achieve the target accuracy ζ is bounded as

$$\zeta_i^p \leq \zeta^p := \frac{E_G^{p+2}}{(k+1)!\,\nu^p} \qquad \forall k \geq 0, \quad 0 \leq p < k \tag{2.72}$$

where E_G is the accuracy of the coarse solver G and

$$\nu^p = \frac{\max_{0 \leq i \leq N}(1 + \|\mathbf{x}_i^p\|)}{\max_{0 \leq i \leq N}(1 + \|\mathbf{x}(T)\|)} \tag{2.73}$$

The above bound shows that one does not need to run the fine solver with a very high accuracy in the initial iterations and avoid unnecessary oversolving. Since the accuracy ζ_i^k has to improve with the increasing iteration k, the subsequent approximations to $[E(t_i, t_{i+1}, \mathbf{x}_i^k); \zeta_i^k)]$ can be built using the adaptive techniques with the help of the posteriori error estimates.

The original Parareal version is obtained when $[E(t_i, t_{i+1}, \mathbf{x}_i^k)]$ is approximated by a fixed accuracy $\zeta_i^k = \zeta_F$ across all the Parareal iterations. Also by definition the fine solver accuracy is more than the coarse solver accuracy i.e. $\zeta_F < \zeta_G$. Thus we have $[E(t_i, t_{i+1}, \mathbf{x}_i^k); \zeta_F] = F(t_i, t_{i+1}, \mathbf{x}_i^k)$ and

$$\begin{aligned} \mathbf{x}_{i+1}^0 &= G(t_i, t_{i+1}, \mathbf{x}_i^0) \\ \mathbf{x}_{i+1}^{k+1} &= G(t_i, t_{i+1}, \mathbf{x}_i^{k+1}) + F(t_i, t_{i+1}, \mathbf{x}_i^k) - G(t_i, t_{i+1}, \mathbf{x}_i^k) \end{aligned} \tag{2.74}$$

In theory, one gets the exact solution at the kth time window at the kth Parareal iteration i.e. $\mathbf{x}_k^k = \mathbf{x}(t_k), 0 < k \leq N$. Using the adaptive Parareal scheme can help to achieve faster convergence and also reduce the number of iterations since now we only look for the approximated solutions with the minimal accuracies ζ_i^k to maintain the target accuracy ζ.

Remark 2.5. *When the cost of the coarse solver is negligible and there are no communication delays, the adaptive Parareal efficiency is independent of the number of Parareal iterations [Maday and Mula, 2020].*

Part II

Coupling Parareal and data assimilation

Chapter 3

Introducing Parallel In Time in data assimilation

Contents

In the first part of this manuscript, we have introduced the variational data assimilation method and the Parareal algorithm, two methods which are iterative in nature. (Variational) data assimilation on one hand is based on sequential minimisation of a cost function while Parareal on the other hand is an iterative algorithm to take profit from time parallelism.

In this chapter we describe the way in which these two iterative methods can be coupled together. We begin in Section 3.1 with a brief state of the art on some previous attempts made for a *time parallel 4D-Var*. Next we introduce a framework consisting of introducing the Parareal algorithm within the inner loop of the incremental 4D-Var in Section 3.2. In this work, we restrict ourselves to the integration of the direct model with Parareal.

One central question in our work is how the use of an approximated forward model impacts the convergence of the data assimilation minimisation step. To address this question, we rely on the inexact conjugate gradient algorithm proposed by Gratton and co-authors in [Gratton et al., 2021] and presented in Section 3.3. By its nature, this algorithm effectively allows controlling the necessary precision of matrix-vector products involved in a conjugate gradient algorithm. In addition the algorithm can also adaptively manage the unused inaccuracies obtained from the difference between the maximum allowed error and the actual error in the matrix-vector product. This *inaccuracy budget* management gives the possibility choose where/when to spend the computing power. In the following section 3.4, we explain the adaptations required within the inexact conjugate gradient to adjust it for the Parareal operator. This results in the derivation of a Parareal stopping criterion which can guarantee, in a minimum of Parareal iterations, the convergence of the (inexact) conjugate gradient. The practical estimation of this stopping criterion is finally addressed.

3.1 Previous works on a parallel 4D-Var

The inner loop of the incremental 4D-Var remains the least scalable part due to the usage of lower resolution models [Isaksen, 2012]. This means that we are with fewer grid points to provide to the processors which leads to not being able to exploit all the processors for producing the analysis increment. In fact parallelising the inner loop is by no means trivial. The most obvious choice of the minimisation method is the conjugate gradient method which is one of the Krylov-subspace based methods. It is constructed on the principle of finding the higher level Krylov subspaces at each iterations which rely on the initial gradient (residual) of the cost function. These gradients are updated at the end of the iteration to provide it for building the next Krylov subspace [Fisher et al., 2012]. This process is sequential which leaves with parallelising the computations within the iteration as the only way rather than trying to parallelise the minimisation iterations themselves.

In this section we are going to discuss some of the previous works which found a way to implement a "time parallel 4D-Var". First the notion of a parallel weak-constraint 4D-Var which has a four dimensional initial state as its control variable is presented. The outer loop of this problem is shown to be easily parallelised while the inner loop is solved in parallel by treating it as a saddle point problem. Then a different approach of finding time parallelism through a coupling of the forward and adjoint model together is discussed where the obtained non-linear system is solved by approximating the Jacobian matrix through a derivative variant of Parareal.

3.1.1 Parallel weak-constraint

The weak-constraint formulation of the 4D-Var with the full state vector (discussed in Sec. 1.4) as the control variable opens up a way of introducing potential parallelism. Looking at the corresponding cost function (1.29)

$$\begin{aligned}\mathcal{J}(\mathbf{x}) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}^b)^T \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}^b) + \frac{1}{2}\sum_{i=0}^{N}(H_i(\mathbf{x}_i) - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1}(H_i(\mathbf{x}_i) - \mathbf{y}_i^o) \\ + \frac{1}{2}\sum_{i=1}^{N}\{\mathbf{x}_i - M_i(\mathbf{x}_{i-1})\}^T \mathbf{Q}_i^{-1}\{\mathbf{x}_i - M_i(\mathbf{x}_{i-1})\}\end{aligned} \tag{3.1}$$

we can see that there is no model integration required for the observation cost function; it is only present for the model-error cost function term. Note again that the control variables are the intermediate model states $\mathbf{x}_1, \ldots, \mathbf{x}_N$ where N is the total time windows. An incremental formulation for the weak-constraint

4D-Var (3.1) can be obtained as for its natural extension for the “classical” strong constraint 4D-Var. For the control variable $\delta\mathbf{x} = (\delta\mathbf{x}_1, \dots, \delta\mathbf{x}_N)^T$ and outer loop index l we have the cost function

$$\begin{aligned} J^{(l)}(\delta\mathbf{x}) = &\frac{1}{2}(\delta\mathbf{x}_0 - \mathbf{b}^{(l)})^T\mathbf{B}^{-1}(\delta\mathbf{x}_0 - \mathbf{b}^{(l)}) \\ &+ \frac{1}{2}\sum_{i=1}^{N}(\delta\boldsymbol{\eta}_i - \mathbf{c}_i^{(l)})^T\mathbf{Q}_i^{-1}(\delta\boldsymbol{\eta}_i - \mathbf{c}_i^{(l)}) \\ &+ \frac{1}{2}\sum_{i=0}^{N}(\mathbf{H}_i^{(l)}\delta\mathbf{x}_i - \mathbf{d}_i^{(l)})^T\mathbf{R}_i^{-1}(\mathbf{H}_i^{(l)}\delta\mathbf{x}_i - \mathbf{d}_i^{(l)}) \end{aligned} \tag{3.2}$$

where $\mathbf{H}_i^{(l)}$ is the linearised observation operator and

$$\begin{aligned} \mathbf{b}^{(l)} &= \mathbf{x}^b - \mathbf{x}_0^{(l)} \\ \mathbf{c}_i^{(l)} &= \bar{\boldsymbol{\eta}} - \mathbf{x}_i^{(l)} - M_i(\mathbf{x}_{i-1}^{(l)}) \\ \mathbf{d}_i^{(l)} &= \mathbf{y}_i^o - H_i(\mathbf{x}_i^{(l)}) \end{aligned} \tag{3.3}$$

with $\bar{\boldsymbol{\eta}} = 1/N\sum_{i=1}^{N}\delta\boldsymbol{\eta}_i$ is the mean model-error. The model-error correction increment $\delta\boldsymbol{\eta}_i$ is defined through the linearisation of equation (1.30)

$$\delta\mathbf{x}_i = \mathbf{M}_i^{(l)}\delta\mathbf{x}_{i-1} + \delta\boldsymbol{\eta}_i \tag{3.4}$$

where $\mathbf{M}_i^{(l)}$ is the linearisation of the non-linear model M_i about the initial state $\mathbf{x}_{i-1}^{(l)}$. The inner loop produces a four dimensional increment $\delta\mathbf{x}^{(l)}$ and the outer loop is updated by adding this increment to the existing four dimensional model state $\mathbf{x}^{(l)}$. In other words

$$\mathbf{x}_i^{(l+1)} = \mathbf{x}_i^{(l)} + \delta\mathbf{x}_i^{(l)} \tag{3.5}$$

Fisher et al. [Fisher et al., 2012, Fisher and Gürol, 2017] gave insight into how the weak-constraint 4D-Var can be parallelised. Once $\mathbf{x}_i^{(l+1)}$ are calculated it is required that the quantities $\mathbf{b}^{(l+1)}, \mathbf{c}_i^{(l+1)}$ and $\mathbf{d}_i^{(l+1)}$ are computed for the next outer loop. The new linearised operators $\mathbf{H}_i^{(l+1)}$ and $\mathbf{M}_i^{(l+1)}$ are also needed to be updated for the new linearisation state $\mathbf{x}_i^{(l+1)}$. All these computations need the integration of the non-linear model $M_i(\mathbf{x}_i^{(l+1)})$ which can be done in parallel because the initial condition for each subinterval $[t_i, t_{i+1}]$ is $\mathbf{x}_i^{(l+1)}$ is already known. Thus the outer loop of the incremental weak-constraint 4D-Var can be inherently parallelised.

For parallelising the inner loop, consider the cost function in a more simplified vectorial form

$$\hat{J} = \frac{1}{2}(\delta\mathbf{x} - \hat{\mathbf{b}})^T \hat{\mathbf{B}}^{-1}(\delta\mathbf{x} - \hat{\mathbf{b}}) + \frac{1}{2}(\hat{\mathbf{H}}\delta\mathbf{x} - \hat{\mathbf{d}})^T \hat{\mathbf{R}}^{-1}(\hat{\mathbf{H}}\delta\mathbf{x} - \hat{\mathbf{d}}) \tag{3.6}$$

where

$$\hat{\mathbf{B}} = \begin{pmatrix} \mathbf{B} & & & \\ & \mathbf{Q}_1 & & \\ & & \ddots & \\ & & & \mathbf{Q}_n \end{pmatrix}, \quad \hat{\mathbf{R}} = \begin{pmatrix} \mathbf{R}_0 & & & \\ & \mathbf{R}_1 & & \\ & & \ddots & \\ & & & \mathbf{R}_n \end{pmatrix}$$
$$\hat{\mathbf{H}} = \begin{pmatrix} H_0 & & & \\ & H_1 & & \\ & & \ddots & \\ & & & H_n \end{pmatrix} \tag{3.7}$$

and

$$\hat{\mathbf{d}} = \begin{pmatrix} \mathbf{d}_0 \\ \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_n \end{pmatrix}, \quad \hat{\mathbf{b}} = \begin{pmatrix} \mathbf{b} \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_n \end{pmatrix} \tag{3.8}$$

In vectorial form the discrete relation (3.4) can be written as

$$\delta\mathbf{x} = \hat{\mathbf{M}}\,\delta\boldsymbol{\eta} \tag{3.9}$$

where

$$\hat{\mathbf{M}} = \begin{pmatrix} \mathbf{I} & & & & \\ \mathbf{M}_{1,1} & \mathbf{I} & & & \\ \mathbf{M}_{1,2} & \mathbf{M}_{2,2} & \mathbf{I} & & \\ \vdots & \vdots & \ddots & \ddots & \\ \mathbf{M}_{1,N} & \mathbf{M}_{2,N} & \dots & \mathbf{M}_{N,N} & \mathbf{I} \end{pmatrix} \tag{3.10}$$

and $\mathbf{M}_{i,j} = \mathbf{M}_j \cdots \mathbf{M}_i$ denotes the linear model integration from time t_{i-1} to time t_j. Rearranging the relation (3.4) we have

$$\delta\boldsymbol{\eta}_i = \delta\mathbf{x}_i^{(l)} - \mathbf{M}_i\delta\mathbf{x}_{i-1}^{(l)} \tag{3.11}$$

It is clear that $\hat{\mathbf{M}}$ is invertible and

$$\hat{\mathbf{M}}^{-1} = \begin{pmatrix} \mathbf{I} & & & & \\ -\mathbf{M}_1 & \mathbf{I} & & & \\ & -\mathbf{M}_2 & \mathbf{I} & & \\ & & \ddots & \ddots & \\ & & & -\mathbf{M}_N & \mathbf{I} \end{pmatrix} \tag{3.12}$$

Depending on the choice of the control variable $\delta\mathbf{x}$ or $\delta\boldsymbol{\eta}$, the cost function can be written in two different ways:

$$\hat{J}(\delta\mathbf{x}) = \frac{1}{2}(\hat{\mathbf{M}}^{-1}\delta\mathbf{x} - \hat{\mathbf{b}})^T \hat{\mathbf{B}}^{-1}(\hat{\mathbf{M}}^{-1}\delta\mathbf{x} - \hat{\mathbf{b}}) + \frac{1}{2}(\hat{\mathbf{H}}\delta\mathbf{x} - \mathbf{d})^T \hat{\mathbf{R}}^{-1}(\hat{\mathbf{H}}\delta\mathbf{x} - \mathbf{d}) \tag{3.13}$$

and

$$\hat{J}(\delta\boldsymbol{\eta}) = \frac{1}{2}(\delta\boldsymbol{\eta} - \hat{\mathbf{b}})^T \hat{\mathbf{B}}^{-1}(\delta\boldsymbol{\eta} - \hat{\mathbf{b}}) + \frac{1}{2}(\hat{\mathbf{H}}\hat{\mathbf{M}}\delta\boldsymbol{\eta} - \mathbf{d})^T \hat{\mathbf{R}}^{-1}(\hat{\mathbf{H}}\hat{\mathbf{M}}\delta\boldsymbol{\eta} - \mathbf{d}) \tag{3.14}$$

First for the cost function with respect to control variable $\delta\mathbf{x}$ in (3.13) the model computations $\hat{\mathbf{M}}^{-1}\delta\mathbf{x}$ can be done in parallel. Second if the cost function is represented as a function of $\delta\boldsymbol{\eta}$ as in (3.14) then the calculation of $\delta\mathbf{x}$ from $\hat{\mathbf{M}}\delta\boldsymbol{\eta}$ becomes sequential.

However the parallel implementation (3.13) suffers from a setback of preconditioning and it is in practice hard to find one. A way out is to consider the saddle point formulation of the inner loop cost function proposed by Fisher and Gürol [Fisher and Gürol, 2017]. By introducing two additional variables $\delta\mathbf{w} = \hat{\mathbf{H}}\delta\mathbf{x}$ and $\delta\boldsymbol{\eta} = \hat{\mathbf{M}}^{-1}\delta\mathbf{x}$ a Lagrangian can be defined as

$$\begin{aligned}\mathcal{L} = \frac{1}{2}(\delta\boldsymbol{\eta} - \hat{\mathbf{b}})^T \hat{\mathbf{B}}^{-1}(\delta\boldsymbol{\eta} - \hat{\mathbf{b}}) + \frac{1}{2}(\delta\mathbf{w} - \hat{\mathbf{d}})^T \hat{\mathbf{R}}^{-1}(\delta\mathbf{w} - \hat{\mathbf{d}}) \\ + \mathbf{u}^T(\delta\boldsymbol{\eta} - \hat{\mathbf{M}}^{-1}\delta\mathbf{x}) + \mathbf{v}^T(\delta\mathbf{w} - \hat{\mathbf{H}}\delta\mathbf{x})\end{aligned} \tag{3.15}$$

Finding the stationary points of the Lagrangian means setting its derivatives with respect to $\delta\boldsymbol{\eta}, \delta\mathbf{w}, \delta\mathbf{x}, \mathbf{u}$ and $\mathbf{v}$ to zero, giving

$$\begin{aligned} \hat{\mathbf{B}}^{-1}(\delta\boldsymbol{\eta} - \hat{\mathbf{b}}) + \mathbf{u} &= \mathbf{0} \\ \hat{\mathbf{R}}^{-1}(\delta\mathbf{w} - \hat{\mathbf{d}}) + \mathbf{v} &= \mathbf{0} \\ (\hat{\mathbf{M}}^{-1})^T\mathbf{u} + (\hat{\mathbf{H}}^{-1})^T\mathbf{v} &= \mathbf{0} \\ \delta\boldsymbol{\eta} - \hat{\mathbf{M}}^{-1}\delta\mathbf{x} &= \mathbf{0} \\ \delta\mathbf{w} - \hat{\mathbf{H}}\delta\mathbf{x} &= \mathbf{0}. \end{aligned} \tag{3.16}$$

Eliminating the variables $\delta\boldsymbol{\eta}$ and $\mathbf{w}$ from (3.16) leads to a system of the form

$$\begin{pmatrix} \hat{\mathbf{B}} & \mathbf{0} & \hat{\mathbf{M}}^{-1} \\ \mathbf{0} & \hat{\mathbf{R}} & \hat{\mathbf{H}} \\ (\hat{\mathbf{M}}^{-1})^T & \hat{\mathbf{H}}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \\ \delta\mathbf{x} \end{pmatrix} = \begin{pmatrix} \hat{\mathbf{b}} \\ \hat{\mathbf{d}} \\ \mathbf{0} \end{pmatrix}. \tag{3.17}$$

It can be seen form the above system that nowhere in the matrices there is a sequential term involving $\hat{\mathbf{M}}$ and instead there are terms $\hat{\mathbf{M}}^{-1}$ and $(\hat{\mathbf{M}}^{-1})^T$ which are all parallel evaluations. Moreover the quantities $\hat{\mathbf{B}}\mathbf{u}$, $\hat{\mathbf{M}}^{-1}\delta\mathbf{x}$, $\hat{\mathbf{R}}\mathbf{v}$, $\hat{\mathbf{H}}\delta\mathbf{x}$, $(\hat{\mathbf{M}}^{-1})^T\mathbf{u}$ and $\hat{\mathbf{H}}^T\mathbf{v}$ can all be computed simultaneously because the vectors $\mathbf{u}$, $\mathbf{v}$ and $\delta\mathbf{x}$ are already available before the start of the computation. Interestingly, two out of these quantities $\hat{\mathbf{M}}^{-1}\delta\mathbf{x}$ and $(\hat{\mathbf{M}}^{-1})^T\mathbf{u}$ are the tangent-linear model and the adjoint model calculations respectively, indicating that the these integrations can be done in parallel.

3.1.2 ParaOpt algorithm

The ParaOpt algorithm introduced by Gander et al. [Gander et al., 2020] for PDE constrained optimisation problems aims to solve the coupled system of the forward and backward evolution problems in time with the help of the Parareal algorithm. The algorithm is illustrated by considering the control problem of the form

$$J(\beta) = \frac{1}{2}\|\mathbf{x}_N - \mathbf{y}\|_2^2 + \frac{\alpha}{2}\sum_{i=1}^{N}\|\beta_i\|_2^2 \tag{3.18}$$

where α is a regularisation parameter and $\mathbf{y}$ is an observation at the end of the integration time. The evolution of the state vector $\mathbf{x} : \Omega = [0, T] \to \mathbb{R}^n$ is described by the discrete non-linear model

$$\mathbf{x}_i = M_i(\mathbf{x}_{i-1}) + \beta_i, \qquad \mathbf{x}(0) = \mathbf{x}_0 \tag{3.19}$$

where β_i is the control which is assumed to enter the forcing term linearly. The corresponding optimality system is then given by

$$\mathbf{x}_i = M_i(\mathbf{x}_{i-1}) - \frac{\mathbf{u}_i}{\alpha}, \quad \mathbf{u}_i = -(\mathbf{M}_i)^T\mathbf{u}_{i+1} \tag{3.20}$$

with the final condition $\mathbf{u}_N = \mathbf{x}_N - \mathbf{y}$. Now equations (3.19)-(3.20) form a coupled system and the parallelisation is introduced as follows.

Let $\{\mathbf{x}_i\}_{i=0,\ldots,N}$ and $\{\mathbf{u}_i\}_{i=1,\ldots,N}$ be the intermediate states corresponding to the model state and the adjoint state at times $t_0, \cdots, t_N$ and $t_1, \cdots, t_N$ respectively. Let P_{i+1} and Q_i be the non-linear operators solving the boundary value problem (3.20) in the subinterval $\Omega_i = [t_i, t_{i+1}]$ for initial condition $\mathbf{x}(t_i) = \mathbf{x}_i$ and final condition $\mathbf{u}(t_{i+1}) = \mathbf{u}_{i+1}$. Assume that P_{i+1} solves the forward problem to t_{i+1} and Q_i solves the adjoint problem to t_i. That is,

$$\begin{pmatrix} \mathbf{x}(t_{i+1}) \\ \mathbf{u}(t_i) \end{pmatrix} = \begin{pmatrix} P_{i+1}(\mathbf{x}_i, \mathbf{u}_{i+1}) \\ Q_i(\mathbf{x}_i, \mathbf{u}_{i+1}) \end{pmatrix} \tag{3.21}$$

These boundary value problems can be put together as a system of subproblems which satisfy

$$\begin{aligned} \mathbf{x}_0 - \mathbf{x}_0 &= 0, & & \\ \mathbf{x}_1 - P_1(\mathbf{x}_0, \mathbf{u}_1) &= 0, & \mathbf{u}_1 - Q_1(\mathbf{x}_1, \mathbf{u}_2) &= 0 \\ \mathbf{x}_2 - P_2(\mathbf{x}_1, \mathbf{u}_2) &= 0, & \mathbf{u}_2 - Q_2(\mathbf{x}_2, \mathbf{u}_3) &= 0 \\ &\vdots & &\vdots \\ \mathbf{x}_N - P_N(\mathbf{x}_{N-1}, \mathbf{u}_N) &= 0, & \mathbf{u}_N - \mathbf{x}_N + \mathbf{y} &= 0 \end{aligned} \tag{3.22}$$

This is a non-linear system of equations which is solved by using the Newton's method. If the unknowns are collected in the vectors $\mathbf{x} = (\mathbf{x}_0^T, \cdots \mathbf{x}_N^T)$ and $\mathbf{u} = (\mathbf{u}_1^T, \cdots, \mathbf{u}_N^T)$ then

$$\hat{\mathbf{F}} \begin{pmatrix} \mathbf{x} \\ \mathbf{u} \end{pmatrix} := \begin{pmatrix} \mathbf{x}_0 - \mathbf{x}_0 \\ \mathbf{x}_1 - P_1(\mathbf{x}_0, \mathbf{u}_1) \\ \mathbf{x}_2 - P_2(\mathbf{x}_1, \mathbf{u}_2) \\ \vdots \\ \mathbf{x}_N - P_N(\mathbf{x}_{N-1}, \mathbf{u}_N) \\ \mathbf{u}_1 - Q_1(\mathbf{x}_1, \mathbf{u}_2) \\ \mathbf{u}_2 - Q_2(\mathbf{x}_2, \mathbf{u}_3) \\ \vdots \\ \mathbf{u}_N - \mathbf{x}_N + \mathbf{y} \end{pmatrix} = \mathbf{0} \tag{3.23}$$

The Newton update iteration can be written as

$$\mathbf{J} \begin{pmatrix} \mathbf{x}^l \\ \mathbf{u}^l \end{pmatrix} \begin{pmatrix} \mathbf{x}^{l+1} - \mathbf{x}^l \\ \mathbf{u}^{l+1} - \mathbf{u}^l \end{pmatrix} = -\hat{\mathbf{F}} \begin{pmatrix} \mathbf{x}^l \\ \mathbf{u}^l \end{pmatrix} \tag{3.24}$$

where $\mathbf{J} =$

$$
\left(\begin{array}{cccc|cccc}
\mathbf{I} & & & & & & & \\
-(P_1)_{\mathbf{x}}(\mathbf{x}_0,\mathbf{u}_1) & \mathbf{I} & & & -(P_1)_{\mathbf{u}}(\mathbf{x}_0,\mathbf{u}_1) & & & \\
 & \ddots & \ddots & & & & \ddots & \\
 & & -(P_N)_{\mathbf{x}}(\mathbf{x}_{N-1},\mathbf{u}_N) & \mathbf{I} & & & & -(P_N)_{\mathbf{u}}(\mathbf{x}_{N-1},\mathbf{u}_N) \\
\hline
 & -(Q_1)_{\mathbf{x}}(\mathbf{x}_1,\mathbf{u}_2) & & & \mathbf{I} & -(Q_1)_{\mathbf{u}}(\mathbf{x}_1,\mathbf{u}_2) & & \\
 & & \ddots & & & \ddots & \ddots & \\
 & & -(Q_N)_{\mathbf{x}}(\mathbf{x}_{N-1},\mathbf{u}_N) & & & & \mathbf{I} & -(Q_N)_{\mathbf{u}}(\mathbf{x}_{N-1},\mathbf{u}_N) \\
 & & & -\mathbf{I} & & & & \mathbf{I}
\end{array}\right)
$$

is the Jacobian matrix. The idea of ParaOpt is to approximate the true Jacobian $\mathbf{J}$ using a derivative-variant of the Parareal algorithm. This is done by approximating the derivatives inside $\mathbf{J}$ with a coarse run of the Parareal giving
$\mathbf{J}^G =$

$$
\left(\begin{array}{cccc|cccc}
\mathbf{I} & & & & & & & \\
-(P_i)^G_{\mathbf{x}}(\mathbf{x}_0,\mathbf{u}_1) & \mathbf{I} & & & -(P_i)^G_{\mathbf{u}}(\mathbf{x}_0,\mathbf{u}_1) & & & \\
 & \ddots & \ddots & & & & \ddots & \\
 & & -(P_i)^G_{\mathbf{x}}(\mathbf{x}_{N-1},\mathbf{u}_N) & \mathbf{I} & & & & -(P_i)^G_{\mathbf{u}}(\mathbf{x}_{N-1},\mathbf{u}_N) \\
\hline
 & -(Q_1)^G_{\mathbf{x}}(\mathbf{x}_1,\mathbf{u}_2) & & & \mathbf{I} & -(Q_1)^G_{\mathbf{u}}(\mathbf{x}_1,\mathbf{u}_2) & & \\
 & & \ddots & & & \ddots & \ddots & \\
 & & -(Q_N)^G_{\mathbf{x}}(\mathbf{x}_{N-1},\mathbf{u}_N) & & & & \mathbf{I} & -(Q_N)^G_{\mathbf{u}}(\mathbf{x}_{N-1},\mathbf{u}_N) \\
 & & & -\mathbf{I} & & & & \mathbf{I}
\end{array}\right)
$$

where $\mathbf{J}^G$ is the approximation to the true Jacobian $\mathbf{J}$. The propagators $(P_i)^G_{\mathbf{x}}$, $(P_i)^G_{\mathbf{u}}$, $(Q_i)^G_{\mathbf{x}}$ and $(Q_i)^G_{\mathbf{u}}$ are the Parareal coarse propagators for the derivatives obtained from the coarse discretisation of the (now linear) subproblem by using the length of the subinterval as the time-step. The remaining non-linear solvers on the right hand side of (3.24) are the fine grid operations which can be performed by Parareal in parallel.

The computation of the product of the approximated Jacobian matrix with a vector is now embarrassingly parallel since it only involves solving the local linear derivative problems. However the resulting inexact Newton iteration now converges linearly instead of quadratically.

ParaOpt allows us to decouple the forward and adjoint computations globally by using cores in both the directions and doubling the level of parallelism. Also the only point of synchronicity involved is in the Krylov subspace method itself to solve the Jacobian system. Where ParaOpt becomes interesting is for the situations when we want to control some model parameters. But it is not easy to use it for controlling the initial condition of the model which is the core data assimilation problem we are interested in and we have to study another possibility.

Let us consider a cost function of the form

$$J(\mathbf{x}_0, \beta) = \frac{1}{2}\|\mathbf{x}_N - \mathbf{y}\|_2^2 \tag{3.25}$$

where $\mathbf{x}$ evolves following the discrete non-linear model

$$\mathbf{x}_i = M_i(\mathbf{x}_{i-1}) + \beta, \quad i = 1, \cdots, N \tag{3.26}$$

assuming that the parameter β is constant in time. Then the gradients with respect to β and $\mathbf{x}_0$ are given as

$$\begin{aligned} \nabla_\beta J &= (\mathbf{x}_N - \mathbf{y}) \\ \nabla_{\mathbf{x}_0} J &= \mathbf{M}_N^T \mathbf{M}_{N-1}^T \cdots \mathbf{M}_1^T (\mathbf{x}_N - \mathbf{y}) \end{aligned} \tag{3.27}$$

Clearly, the gradient $\nabla_\beta J$ only requires $\mathbf{x}_N$ which can be easily parallelised whereas the gradient $\nabla_{\mathbf{x}_0} J$ has implicit compositions of $\mathbf{M}_i^T$ making it harder to parallelise in a coupled system using ParaOpt.

3.2 Time parallelisation with Parareal in forward model

Each minimisation iteration of an incremental 4D-Var (or inner loop) cycle begins with the forward integration of the linear model (1.26) to calculate the quadratic cost function value. An immediate way to introduce time parallelisation is thus to apply the Parareal algorithm to the forward model integration. The adjoint model can be treated similarly once we have an initial adjoint state at the end of the integration time. In fact there is also a possibility of making use of the asynchronicity when both the forward and the adjoint model are run using Parareal. We refer to section 3.5 at the end of the manuscript for a topic of further discussion. For the moment and the rest of the manuscript we restrict ourselves only to the Parareal run for the tangent linear model.

To see how this could be done, let us suppose the length of one data assimilation cycle to be T which is made up of N time windows. Let us place ourselves in the case of a linear forward model for state vector $\delta\mathbf{x} \in \mathbb{R}^n$

$$\begin{aligned} \delta\mathbf{x}_0 &= \delta\mathbf{x}(t_0) \\ \delta\mathbf{x}_{i+1} &= \mathbf{F}_i \delta\mathbf{x}_i, \quad i = 1, \dots N \end{aligned} \tag{3.28}$$

The model operator $\mathbf{F}_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is the same as the one used in (1.26) and so

$$\mathbf{F}_i = \frac{\partial F}{\delta \mathbf{x}}\Big|_{\mathbf{x}=\mathbf{x}_i} \tag{3.29}$$

We now set up our data assimilation problem. For the sake of simplicity we remove the background term $\mathbf{x}^b$ from the cost function since it is not affected by the time integration. We also assume that there is only one observation $\mathbf{y} \in \mathbb{R}^n$ of the whole state vector at the end of the integration time $t_N = T$. Doing so the observation operator H is an identity map for each time t_i and the minimisation problem is well posed. To further simplify the covariance matrix $\mathbf{R}_i$ is set to be equal to the identity matrix. Thus our cost function can be written as

$$J(\delta \mathbf{x}_0) = \frac{1}{2} \left\| \left(\prod_{i=1}^{N} \mathbf{F}_i \right) \delta \mathbf{x}_0 - \mathbf{y} \right\|_2^2 \tag{3.30}$$

Remark 3.1. *In the rest of the thesis instead of writing the composition of the discrete model operators* $\mathbf{F}_i$ *at a model state* $\delta \mathbf{x}_i$ *as* $\prod_{i=1}^{N} \mathbf{F}_i$ *each time we are going to use the block* $\mathbf{F}^N$ *for* $\mathbf{F}_1 \mathbf{F}_2 \cdots \mathbf{F}_N$ *for convenience and it is just an abuse of the notation.*

We thus have,

$$J(\delta \mathbf{x}_0) = \frac{1}{2} \| \mathbf{F}^N \delta \mathbf{x}_0 - \mathbf{y} \|_2^2 \tag{3.31}$$

with gradient,

$$\nabla J(\delta \mathbf{x}_0) = (\mathbf{F}^N)^T (\mathbf{F}^N \delta \mathbf{x}_0 - \mathbf{y}) \tag{3.32}$$

where $(\mathbf{F}^N)^T$ is the adjoint of the operator $\mathbf{F}^N$. As stated before, if we now carry out the computation of the forward model (involved in the misfit between the model trajectory and the observation) by a Parareal based operator $\widetilde{\mathbf{P}}$, the gradient of the cost function will be approximated by

$$\begin{aligned} \nabla J(\delta \mathbf{x}_0) &\approx (\mathbf{F}^N)^T \, (\widetilde{\mathbf{P}} \, \delta \mathbf{x}_0 - \mathbf{y}) \\ &\stackrel{\text{def}}{=} \widetilde{\nabla J}(\delta \mathbf{x}_0) \end{aligned} \tag{3.33}$$

Also we have kept the same adjoint for the gradient i.e. $(\mathbf{F}^N)^T$ and not the adjoint of $\widetilde{\mathbf{P}}$. Minimising the original cost function in (3.31) requires finding $\delta \mathbf{x}_0$ such that the gradient $\nabla J(\delta \mathbf{x}_0)$ to 0 which is equivalent to solving the linear system

$$\mathbf{A} \delta \mathbf{x} = \mathbf{b} \tag{3.34}$$

where $\mathbf{A} = (\mathbf{F}^N)^T \mathbf{F}^N$, $\mathbf{b} = (\mathbf{F}^N)^T \mathbf{y}$. The corresponding approximate linear system writes

$$\mathbf{A}_{\text{para}} \delta \mathbf{x} = \mathbf{b} \tag{3.35}$$

where $\mathbf{A}_{\text{para}} = (\mathbf{F}^N)^T \ \widetilde{\mathbf{P}}$.

Since the cost function observes $\mathbf{y}$ only at the end of the integration time $t_N = T$, we accordingly define a Parareal based integrator $\mathbf{P}(k)$ over $[0, T]$ by

$$\mathbf{P}(k)\, \delta \mathbf{x}_0 = \delta \mathbf{x}_N^k \tag{3.36}$$

which acts on an initial condition $\delta \mathbf{x}_0$ and gives the Parareal solution at the last time window N after k iterations. Then the Parareal error at the end of the integration time can be written as

$$\mathbf{e}_N^k = \mathbf{F}^N \delta \mathbf{x}_0 - \mathbf{P}(k)\, \delta \mathbf{x}_0 \tag{3.37}$$

and if we recall the definition of $\mathbf{e}_N^k$ from (2.23) we have

$$\mathbf{F}^N - \mathbf{P}(k) = \sum_{p=k+1}^{N} C_p^N (\mathbf{F} - \mathbf{G})^p \, \mathbf{G}^{N-p} \tag{3.38}$$

$\widetilde{\mathbf{P}}$ will now be denoted $\mathbf{P}(k)$ with k being the number of Parareal iterations.

Now the exact linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be solved using the conjugate gradient (CG) method since the matrix $\mathbf{A}$ is symmetric. But the corresponding approximate matrix $\mathbf{A}_{\text{para}} = (\mathbf{F}^N)^T \mathbf{P}(k)$ is now non-symmetric, there is a big possibility that the CG will not give the desired solution and thus the gradient of our cost function. One way of tackling this problem is by using the inexact version of CG which is a kind of *inexact Krylov subspace method* [Golub and Ye, 1999, Bouras and Frayssé, 2005]. We first briefly talk about the inexact conjugate gradient method, what are the motivation behind using it and then introduce our own modified version of the inexact CG with the Parareal operator $\mathbf{P}(k)$ embedded in it.

3.3 Inexact conjugate gradient method

Solving the approximated linear system 3.35

$$\mathbf{A}_{\text{para}} \delta \mathbf{x} = \mathbf{b}, \quad \mathbf{A}_{\text{para}} = (\mathbf{F}^N)^T \mathbf{P}(k) \tag{3.39}$$

with the Parareal operator $\mathbf{P}(k)$ instead of the exact system 3.34

$$\mathbf{A}\delta\mathbf{x} = \mathbf{b}, \quad \mathbf{A} = (\mathbf{F}^N)^T\mathbf{F}^N \tag{3.40}$$

results in solving a linear system using approximate matrix-vector products. In the literature, this is the purpose of the inexact Krylov subspace methods. They are based on a counter-intuitive principle that the error in the matrix-vector multiplication is permitted to grow as we move ahead with the iterations [Simoncini and Szyld, 2003,Van Den Eshof and Sleijpen, 2004]. By doing so, the same level of accuracy for the minimisation process can be maintained as expected by using the usual (exact) Krylov subspace methods. In the following, we focus on the inexact conjugate gradient method described in [Gratton et al., 2021], the objective being to derive an optimal stopping criterion for the Parareal algorithm.

3.3.1 Formulation

The inexact conjugate gradient method (from now on **inexact CG**) proposed by Gratton et al. [Gratton et al., 2021] offers a way of allowing inaccuracies in the matrix-vector multiplications for solving convex optimisation problems

$$q(\delta\mathbf{x}) = \frac{1}{2}\,\delta\mathbf{x}^T\mathbf{A}\delta\mathbf{x} - \mathbf{b}^T\delta\mathbf{x} \tag{3.41}$$

They have specifically focused on monitoring the decrease in the quadratic q. The quadratic change has a direct relationship with the energy norm of the residual which provides better stopping minimisation criterion in terms of under- or over-solving. Let $\mathbf{r}(\delta\mathbf{x}) = \mathbf{b} - \mathbf{A}\delta\mathbf{x}$ be the residual and suppose that $\delta\mathbf{x}_* = \mathbf{A}^{-1}\mathbf{b}$ is the exact solution of the minimisation of the quadratic. Then this relationship can be described as

$$\begin{aligned}
\frac{1}{2}\|\mathbf{r}(\delta\mathbf{x})\|^2_{\mathbf{A}^{-1}} &= \frac{1}{2}(\mathbf{A}\delta\mathbf{x} - \mathbf{b})^T\mathbf{A}^{-1}\,(\mathbf{A}\delta\mathbf{x} - \mathbf{b}) \\
&= \frac{1}{2}(\delta\mathbf{x} - \delta\mathbf{x}_*)^T\mathbf{A}\,(\delta\mathbf{x} - \delta\mathbf{x}_*) \\
&= \frac{1}{2}(\delta\mathbf{x}^T\mathbf{A}\,\delta\mathbf{x} - 2\delta\mathbf{x}^T\mathbf{A}\,\delta\mathbf{x}_* + \delta\mathbf{x}_*^T\mathbf{A}\,\delta\mathbf{x}_*) \\
&= \left(\frac{1}{2}\delta\mathbf{x}^T\mathbf{A}\,\delta\mathbf{x} - \mathbf{b}^T\delta\mathbf{x}\right) + \frac{1}{2}\mathbf{b}^T\delta\mathbf{x}_* \\
&= q(\delta\mathbf{x}) - q(\delta\mathbf{x}_*)
\end{aligned} \tag{3.42}$$

since $q(\delta\mathbf{x}_*) = -\frac{1}{2}\mathbf{b}^T\delta\mathbf{x}_*$.

Now let $\mathbf{E}_j$ be the error in the matrix-vector product at inexact CG iteration (from now on icg-iteration) j. Thus for a given icg-iteration j and its corresponding conjugate direction vector $\mathbf{p}_j$ (see Algorithm 3 for the presentation of CG) we have access to the matrix-vector product in the form of

$$(\mathbf{A} + \mathbf{E}_j)\,\mathbf{p}_j \tag{3.43}$$

The presence of error $\mathbf{E}_j$ at each icg-iteration j also results in the computed residual $\mathbf{r}_j$ being different from the exact residual $\mathbf{r}(\delta\mathbf{x}_j) = \mathbf{b} - \mathbf{A}\delta\mathbf{x}_j$. The objective of paper [Gratton et al., 2021] is to appropriately bound the residual gap norm $\|\mathbf{r}(\delta\mathbf{x}_j) - \mathbf{r}_j\|$ in $\mathbf{A}^{-1}$ norm. This is provided in the form of a lemma as stated below

Lemma 3.3.1. *Suppose at the icg-iteration j we have,*

$$\max\left[\|\mathbf{r}(\delta\mathbf{x}_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}}, \|\mathbf{r}_j\|_{\mathbf{A}^{-1}}\right] \leq \frac{\sqrt{\epsilon}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}} \tag{3.44}$$

for some $\epsilon > 0$. Then

$$|q(\delta\mathbf{x}_j) - q(\delta\mathbf{x}_*)| \leq \epsilon\,|q(\delta\mathbf{x}_*)| \tag{3.45}$$

Since the residual gap norm is bounded in the dual space ($\mathbf{A}^{-1}$ norm), the size of the perturbation matrix $\mathbf{E}_j$ is measured in the primal-dual norm defined as

$$\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}} = \sup_{\delta\mathbf{x}\neq 0}\frac{\|\mathbf{E}_j\delta\mathbf{x}\|_{\mathbf{A}^{-1}}}{\|\delta\mathbf{x}\|_{\mathbf{A}}} = \|\mathbf{A}^{-1/2}\mathbf{E}_j\mathbf{A}^{-1/2}\|_2 \tag{3.46}$$

Now for some permissible bound on the error norm $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$ at icg-iteration j, if the inexact residual norm $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ and the residual gap norm $\|\mathbf{r}(\delta\mathbf{x}_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}}$ can be suitably bounded, then the change in the quadratic can be controlled. The following theorem captures the essence of the above idea.

Theorem 3.1 (From [Gratton et al., 2021])**.** *Let $\epsilon > 0$ and let $\boldsymbol{\phi} = (\phi_0, \phi_1, \cdots, \phi_{j-1})^T \in \mathbb{R}^j$ be a positive vector satisfying*

$$\sum_{i=1}^{j}\frac{1}{\phi_i} \leq 1 \tag{3.47}$$

Suppose that

$$\|\mathbf{E}_i\|_{\mathbf{A}^{-1},\mathbf{A}} \leq \omega_i = \frac{\sqrt{\epsilon}\,\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_i\|_{\mathbf{A}}}{2\,\phi_{i+1}\|\mathbf{r}_i\|_2^2 + \sqrt{\epsilon}\,\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_i\|_{\mathbf{A}}} \tag{3.48}$$

for all $i \in \{0, ..., j-1\}$*, with* $\mathbf{p}_i$ *being the conjugate direction at icg-iteration* i*. Then*

$$\|\mathbf{r}(\delta\mathbf{x}_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}} \leq \frac{\sqrt{\epsilon}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}} \tag{3.49}$$

Additionally if

$$\|\mathbf{r}_j\|_{\mathbf{A}^{-1}} \leq \frac{\sqrt{\epsilon}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}} \tag{3.50}$$

then $|q(\delta\mathbf{x}_j) - q(\delta\mathbf{x}_*)| \leq \epsilon|q(\delta\mathbf{x}_*)|$*.*

Remark 3.2. *Looking at* (3.48)*, the error in the matrix-vector product* $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$ *is allowed to grow as the residual norm* $\|\mathbf{r}_j\|_2$ *gradually decreases with the icg-iterations. The maximum permissible error at worst could be* $\omega_j = 1$ *when* $\|\mathbf{r}_j\|_2 = 0$*.*

We refer to Algorithm 6 below for the implementation of the inexact CG encapsulating theorem 3.1.

Algorithm 6 ICG: INEXACT CONJUGATE GRADIENT

Given: Symmetric positive definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, right hand side vector $\mathbf{b} \in \mathbb{R}^n$, tolerance ϵ_{icg}
Set $\delta\mathbf{x}_0 = 0$, $\mathbf{r}_0 = -\mathbf{b}$, $\mathbf{p}_0 = \mathbf{b}$, $\beta_0 = \|\mathbf{b}\|_2^2$, $\mathbf{u}_1 = \mathbf{b}/\beta_0$
for $j = 0, 1, \cdots,$ **do**
    Determine ω_j from the equation (3.48)
    Compute the product $\mathbf{c}_j = (\mathbf{A} + \mathbf{E}_j)\mathbf{p}_j$ with $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}} \leq \omega_j$.
    $\alpha_j = \beta_j / \mathbf{p}_j^T \mathbf{c}_j$
    $\delta\mathbf{x}_{j+1} = \delta\mathbf{x}_j + \alpha_j \mathbf{p}_j$
    $\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j \mathbf{c}_j$
    if $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}} \leq \frac{\sqrt{\epsilon_{\text{icg}}}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ **then**
        break
    end if

if (reorth) **then**
  for $i = 1, \cdots, j$ **do**
    $\mathbf{r}_{j+1} = \mathbf{r}_{j+1} - (\mathbf{u}_i^T \mathbf{r}_{j+1})\mathbf{u}_i$
  end for
  $\beta_{j+1} = \mathbf{r}_{j+1}^T \mathbf{r}_{j+1}$
  $\mathbf{u}_{j+1} = \mathbf{r}_{j+1} / \sqrt{\beta_{j+1}}$
else
  $\beta_{j+1} = \mathbf{r}_{j+1}^T \mathbf{r}_{j+1}$
end if
$\mathbf{p}_{j+1} = -\mathbf{r}_{j+1} + (\beta_{j+1}/\beta_j)\mathbf{p}_j$
end for

Remark 3.3. *As a consequence of the increasing error in the matrix-vector product, the search directions* $\mathbf{p}_j$ *are no longer conjugate to each other and the (inexact) residuals lose their orthogonality. In the theorem it is assumed that the inexact residual norm* $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ *eventually becomes smaller but it is not guaranteed. Adding a reorthogonalisation step could ensure that the residuals converge to zero after at most* n *steps.*

3.3.2 A practical implementation of inexact CG

So far the method has been discussed from a theoretical point of view with an assumption that the quantities or norms concerning the matrix $\mathbf{A}$ and its inverse are accessible. But realistically one might not even have access to the matrix $\mathbf{A}$, let alone its inverse. Thus in practice Gratton et al. [Gratton et al., 2021] have proposed the following approximations:

- $$\|\mathbf{p}_j\|_{\mathbf{A}} \approx \sqrt{\frac{1}{n}\operatorname{Tr}(\mathbf{A})}\,\|\mathbf{p}_j\|_2 \tag{3.51}$$

- $$q(\delta\mathbf{x}_j) \approx q_j \stackrel{\text{def}}{=} -\frac{1}{2}\mathbf{b}^T\delta\mathbf{x}_j \tag{3.52}$$

- $$\|\mathbf{b}\|_{\mathbf{A}^{-1}} \approx \begin{cases} \dfrac{\|\mathbf{b}\|_2}{\mu_{\max}(\mathbf{A})}, & j = 0 \\ \sqrt{2\,|q_j|}, & j = 1, \cdots, j_{\max} \end{cases} \tag{3.53}$$

- Termination criterion (3.50) by

$$q_{j-d} - q_j \leq \frac{1}{4}\,\epsilon |q_j| \tag{3.54}$$

3.3.3 Inaccuracy budget

Till now there has been no talk about the role of ϕ_j which can be used to manage the inaccuracy budget as discussed in [Gratton et al., 2021, §3.1]. By definition the values of ϕ_j are constrained by the condition (3.47) and choosing a constant value of $\phi_j = j_{\max}$ (as in Algorithm 6) limits the possibility of a larger error allowance ω_j. In fact the ϕ values can be used adaptively and the smaller the value of ϕ_{j+1}, the larger the bound for ω_j will be. This can be done by managing the inaccuracies obtained from the difference between the computed bounds for ω_j from (3.48) and the actual allowed inexactness $\hat{\omega}_j$ (the value of $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$). This difference is unused and could be utilised by distributing to the subsequent icg-iterations.

Suppose for a given ϕ_{j+1} we obtain, $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}} = \hat{\omega}_j \leq \omega_j$. The corresponding $\hat{\phi}_{j+1}$ can be obtained from $\hat{\omega}_j = \omega_j(\hat{\phi}_{j+1})$ in (3.48) as

$$\hat{\phi}_{j+1} = \frac{(1-\hat{\omega}_j)}{\hat{\omega}_j}\frac{\sqrt{\epsilon}\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_j\|_{\mathbf{A}}}{2\|\mathbf{r}_j\|_2^2} > \phi_{j+1} \tag{3.55}$$

Like in the original method, we define

$$\mathbf{\Phi}_j \stackrel{\text{def}}{=} 1 - \sum_{i=1}^{j} \hat{\phi}_i^{-1} \tag{3.56}$$

and we can distribute the unused inaccuracy evenly in the remaining $j_{\max} - j - 1$ iterations as

$$\phi_p = \left(\frac{j_{\max} - j - 1}{\mathbf{\Phi}_{j+1}}\right), \qquad p = j+2, \cdots, j_{\max} \tag{3.57}$$

Thus the resulting values of ϕ_p will be smaller which allows to have a larger perturbation and thus we end up using fewer Parareal iterations. Also the updated

ϕ_j still satisfies (3.47) as shown below

$$\begin{aligned}
\sum_{p=1}^{j_{\max}} \phi_p^{-1} &= \sum_{p=1}^{j+1} \phi_p^{-1} + \sum_{p=j+2}^{j_{\max}} \phi_p^{-1} \\
&= \sum_{p=1}^{j+1} \phi_p^{-1} + (j_{\max} - j - 1) \left(\frac{\mathbf{\Phi}_{j+1}}{j_{\max} - j - 1} \right) \\
&= \sum_{p=1}^{j+1} \phi_p^{-1} + 1 - \sum_{p=1}^{j+1} \hat{\phi}_p^{-1} \\
&< 1
\end{aligned} \tag{3.58}$$

since $\hat{\phi}_j > \phi_j$.

3.4 Inexact conjugate gradient and Parareal

In the case of the use of the Parareal operator for the forward model, by construction, the $\mathbf{E}$ matrix at any Parareal iteration k can be written as

$$\begin{aligned}
\mathbf{E}(k) &= \mathbf{A}_{\text{para}} - \mathbf{A} \\
&= (\mathbf{F}^N)^T \mathbf{P}(k) - (\mathbf{F}^N)^T \mathbf{F}^N \\
&= -(\mathbf{F}^N)^T (\mathbf{F}^N - \mathbf{P}(k))
\end{aligned} \tag{3.59}$$

Remark 3.4. *Since the inexactness is introduced by another iterative method (Parareal in this case), each CG minimisation iteration j involves some iterations of the Parareal method. The number of Parareal iterations depends on the stopping criteria (3.48) as $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$ involves $\mathbf{P}(k)$ and the number of minimisation iterations on $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ in equation (3.50).*

3.4.1 Controlling error with Parareal stopping criteria

In the previous section in Theorem 3.1, the residual gap is kept under controlled precision provided that the primal-dual norm of the perturbation matrix

$\|\mathbf{E}_j(k)\|_{\mathbf{A}^{-1},\mathbf{A}}$ is bounded by the quantity ω_j. It can also be said that ω_j acts as a threshold for how large an error can be allowed at a particular icg-iteration j by controlling the number of Parareal iterations k. As it will be seen in our numerical experiments (section 4.5.1), the direct use of $\|\mathbf{E}_j(k)\|_{\mathbf{A}^{-1},\mathbf{A}}$, does not provide a refined enough stopping criterion for our application. We show in the following that it is preferable to bound the product of the error matrix $\mathbf{E}$ by the vector $\mathbf{p}_j$ and that an estimate for this product can be easily obtained in the context of Parareal.

In the proof of Theorem 3.1, the bound (3.49) is obtained by using the inequality

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}\|\mathbf{p}_j\|_{\mathbf{A}} \leq \omega_j\|\mathbf{p}_j\|_{\mathbf{A}} \tag{3.60}$$

We noticed that utilising $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ instead of $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$ for the Parareal stopping criterion gives us much better results. To bound our new norm we introduce a new quantity ξ_j which satisfies,

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \omega_j\|\mathbf{p}_j\|_{\mathbf{A}} = \xi_j \tag{3.61}$$

Thus from (3.48) the value of ξ_j can be obtained as

$$\xi_j = \frac{\sqrt{\epsilon}\,\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_j\|_{\mathbf{A}}^2}{2\phi_{j+1}\|\mathbf{r}_j\|_2^2 + \sqrt{\epsilon}\,\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_j\|_{\mathbf{A}}} \tag{3.62}$$

The following theorem encapsulates the modifications discussed above

Theorem 3.2. *In theorem 3.1, condition (3.48) is replaced by*

$$\|\mathbf{E}_j(k)\|_{\mathbf{A}^{-1},\mathbf{A}} = \sup_{\delta\mathbf{x}\neq 0} \frac{\|\mathbf{E}_j(k)\,\delta\mathbf{x}\|_{\mathbf{A}^{-1}}}{\|\delta\mathbf{x}\|_{\mathbf{A}}} \leq \xi_j/\|\mathbf{p}_j\|_{\mathbf{A}} \tag{3.63}$$

However, using the fact that

$$\|\mathbf{E}_j(k)\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \|\mathbf{E}_j(k)\|_{\mathbf{A}^{-1},\mathbf{A}}\|\mathbf{p}_j\|_{\mathbf{A}} \tag{3.64}$$

we can get a tighter bound, while retaining the validity of the theorem.

Proof. The proof is almost identical to the proof of theorem 1 in [Gratton et al., 2021], and for this we will use Lemma 2 of the same paper:

Lemma 3.4.1. *The residual gap in the inexact CG algorithm satisfies*

$$\mathbf{r}(\delta\mathbf{x}_j) - \mathbf{r}_j = -\sum_{j=0}^{k-1}\alpha_j\mathbf{E}_j\mathbf{p}_j \tag{3.65}$$

Therefore one gets:

$$\begin{aligned}\|\mathbf{r}(\delta\mathbf{x}_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}} &\leq \sum_{j=0}^{k-1} \|\alpha_j \mathbf{E}_j \mathbf{p}_j\|_{\mathbf{A}^{-1}} \\ &\leq \sum_{j=0}^{k-1} |\alpha_j|\, \|\mathbf{E}_j \mathbf{p}_j\|_{\mathbf{A}^{-1}}\end{aligned} \tag{3.66}$$

By definition

$$\alpha_j = \frac{\|\mathbf{r}_j\|_2^2}{\mathbf{p}_j^T(\mathbf{A}+\mathbf{E}_j)\mathbf{p}_j} \tag{3.67}$$

while

$$\mathbf{p}_j^T \mathbf{E}_j \mathbf{p}_j = \mathbf{p}_j^T \mathbf{A}^{1/2}\mathbf{A}^{-1/2}\mathbf{E}_j\mathbf{p}_j = (\mathbf{A}^{1/2}\mathbf{p}_j)^T \mathbf{A}^{-1/2}\mathbf{E}_j\mathbf{p}_j \tag{3.68}$$

and using Cauchy-Schwarz inequality

$$|\mathbf{p}_j^T \mathbf{E}_j \mathbf{p}_j| \leq \|\mathbf{A}^{1/2}\mathbf{p}_j\|_2 \|\mathbf{A}^{-1/2}\mathbf{E}_j\mathbf{p}_j\|_2 = \|\mathbf{p}_j\|_{\mathbf{A}} \|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \tag{3.69}$$

Hence,

$$\begin{aligned}\alpha_j &= \frac{\|\mathbf{r}_j\|_2^2}{\mathbf{p}_j^T\mathbf{A}\mathbf{p}_j + \mathbf{p}_j^T\mathbf{E}_j\mathbf{p}_j} \\ &\leq \frac{\|\mathbf{r}_j\|_2^2}{\|\mathbf{p}_j\|_{\mathbf{A}}^2 - \|\mathbf{p}_j\|_{\mathbf{A}}\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}}\end{aligned} \tag{3.70}$$

Now going back to

$$\begin{aligned}\|\mathbf{r}(\delta\mathbf{x}_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}} &\leq \sum_{j=0}^{k-1} \left(\frac{\|\mathbf{r}_j\|_2^2\, \|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}}{\|\mathbf{p}_j\|_{\mathbf{A}}^2 - \|\mathbf{p}_j\|_{\mathbf{A}}\, \|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}} \right) \\ &\leq \sum_{j=0}^{k-1} \left(\frac{\|\mathbf{r}_j\|_2^2\, \xi_j}{\|\mathbf{p}_j\|_{\mathbf{A}}^2 - \|\mathbf{p}_j\|_{\mathbf{A}}\, \xi_j} \right) \\ &= \sum_{j=0}^{k-1} \frac{\|\mathbf{r}_j\|_2^2}{\|\mathbf{p}_j\|_{\mathbf{A}}} \left(\frac{\xi_j}{\|\mathbf{p}_j\|_{\mathbf{A}} - \xi_j} \right)\end{aligned} \tag{3.71}$$

The definition of ξ_j in (3.62) gives

$$\frac{\xi_j}{\|\mathbf{p}_j\|_{\mathbf{A}} - \xi_j} = \frac{\sqrt{\epsilon}\, \|\mathbf{b}\|_{\mathbf{A}^{-1}} \|\mathbf{p}_j\|_{\mathbf{A}}}{2\, \phi_{j+1}\, \|\mathbf{r}_j\|_2^2} \tag{3.72}$$

Putting this value back in (3.71) leads to

$$\begin{aligned}
\|\mathbf{r}(\delta\mathbf{x}_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}} &\leq \sum_{j=0}^{k-1} \frac{\|\mathbf{r}_j\|_2^2}{\|\mathbf{p}_j\|_{\mathbf{A}}} \left(\frac{\sqrt{\epsilon}\, \|\mathbf{b}\|_{\mathbf{A}^{-1}} \|\mathbf{p}_j\|_{\mathbf{A}}}{2\,\phi_{j+1}\, \|\mathbf{r}_j\|_2^2} \right) \\
&= \sum_{j=0}^{k-1} \left(\frac{\sqrt{\epsilon}\, \|\mathbf{b}\|_{\mathbf{A}^{-1}}}{2\phi_{j+1}} \right) \qquad (3.73) \\
&\leq \frac{\sqrt{\epsilon}}{2} \|\mathbf{b}\|_{\mathbf{A}^{-1}}
\end{aligned}$$

□

Remark 3.5. $\|\mathbf{E}_j(k)\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ *involves the product of the error matrix* $\mathbf{E}_j$ *and the conjugate direction vector* $\mathbf{p}_j$ *at a given Parareal iteration* k*. This means that* $\mathbf{p}_j$ *is the initial condition for the Parareal algorithm. The modified criterion does not make use of* $\|\mathbf{E}_j(k)\|_{\mathbf{A}^{-1},\mathbf{A}}$ *anywhere in the algorithm. Thus, we don't need to know the perturbation matrix explicitly. It is sufficient for us if we know it indirectly in the form of a matrix vector product. With a given conjugate direction* $\mathbf{p}_j$ *the product* $(\mathbf{F}^N)^T\mathbf{P}(k)\mathbf{p}_j - \mathbf{A}\mathbf{p}_j$ *is nothing but* $\mathbf{E}_j(k)\mathbf{p}_j$*.*

Algorithm 7 shows the inexact CG when the modified Parareal criterion is used.

Algorithm 7 ICG WITH $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ AS PARAREAL STOPPING CRITERION

Given: Symmetric positive definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, right hand side vector $\mathbf{b} \in \mathbb{R}^n$, tolerance ϵ_{cg}
initialisation:
Set $\delta\mathbf{x}_0 = 0$, $\mathbf{r}_0 = -\mathbf{b}$, $\mathbf{p}_0 = \mathbf{b}$, $\beta_0 = \|\mathbf{b}\|_2^2$, $\mathbf{u}_1 = \mathbf{b}/\beta_0$
icg iterations:
for $j = 0, 1, \cdots,$ **do**
  Determine ξ_j from the equation (3.62)
  Compute the product $\mathbf{c}_j = (\mathbf{A} + \mathbf{E}_j)\mathbf{p}_j$ with $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \xi_j$.
  $\alpha_j = \beta_j/\mathbf{p}_j^T\mathbf{c}_j$
  $\delta\mathbf{x}_{j+1} = \delta\mathbf{x}_j + \alpha_j\mathbf{p}_j$
  $\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j\mathbf{c}_j$

---

if $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}} \leq \frac{\sqrt{\epsilon}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ **then**
  break
end if
if (reorth) **then**
  for $i = 1, \cdots, j$ **do**
    $\mathbf{r}_{j+1} = \mathbf{r}_{j+1} - (\mathbf{u}_i^T \mathbf{r}_{j+1})\mathbf{u}_i$
  end for
  $\beta_{j+1} = \mathbf{r}_{j+1}^T \mathbf{r}_{j+1}$
  $\mathbf{u}_{j+1} = \mathbf{r}_{j+1}/\sqrt{\beta_{j+1}}$
else
  $\beta_{j+1} = \mathbf{r}_{j+1}^T \mathbf{r}_{j+1}$
end if
$\mathbf{p}_{j+1} = -\mathbf{r}_{j+1} + (\beta_{j+1}/\beta_j)\mathbf{p}_j$
end for

3.4.2 Illustration of the modified criterion

To illustrate how the new Parareal stopping criterion performs we consider the case when the matrix $\mathbf{A}$ comes after discretising the linearised 1D shallow water model, which we are going to discuss in detail in chapter 4, section 4.3. Fig. 3.1 below shows the comparison of the two norms used for the stopping criteria. The comparison is done by checking the number of Parareal iterations when the original criterion (involving $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$) is used on the left and when the modified criterion (involving $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$) is used on the right for a given icg-iteration j. The blue lines and the black dashed lines on both the images are the norm values and the corresponding tolerance respectively.

It can be seen that there is a huge difference between the two criteria. On the left image we see that the criterion $\|\mathbf{E}_j(k)\|_{\mathbf{A}^{-1},\mathbf{A}} \leq \omega_j$ is satisfied only at the second last Parareal iteration whereas on the right image, the one based on $\|\mathbf{E}_j(k)\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \xi_j$ is already satisfied in 6 Parareal iterations. If we look at the definition,

$$\|\mathbf{E}_j(k)\|_{\mathbf{A}^{-1},\mathbf{A}} = \sup_{\delta\mathbf{x}\neq 0} \frac{\|\mathbf{E}_j(k)\,\delta\mathbf{x}\|_{\mathbf{A}^{-1}}}{\|\delta\mathbf{x}\|_{\mathbf{A}}} \tag{3.74}$$

the relatively high values of the first criterion can be explained by the fact that it does not take into account the actual value of $\mathbf{p}_j$ and somehow is the most pessimistic one.

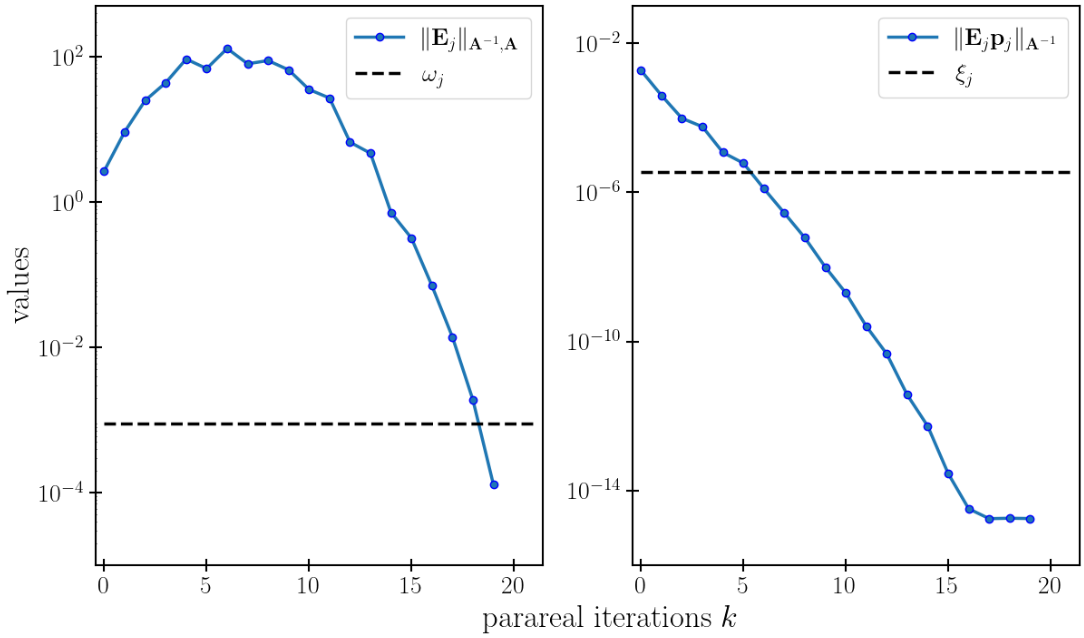


Figure 3.1: Estimates and bounds for icg-iteration j when Parareal is applied to the 1D shallow water model (4.16). The number of time windows N is equal to 20.

3.4.3 Approximation to $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$

We end this presentation by providing a practical estimate for $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$. It turns out that we can replace $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ by another equivalent and computable quantity as proved in the theorem below.

Theorem 3.3. *If* $\mathbf{F}$ *is invertible,* $\|\mathbf{E}_j(k)\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ *and* $\|\mathbf{P}(k)\mathbf{p}_j - \mathbf{F}^N\mathbf{p}_j\|_2$ *are rigorously the same whatever "the Parareal approximation" is.* $\mathbf{P}(k)\mathbf{p}_j$ *is the Parareal approximation at iteration* k *with* $\mathbf{p}_j$ *as the initial condition and* $\mathbf{F}^N\mathbf{p}_j$ *is the corresponding exact solution.*

Proof. We have,

$$\begin{aligned}
\|\mathbf{E}_j(k)\mathbf{p}_j\|_{\mathbf{A}^{-1}} &= \left\langle \mathbf{E}_j(k)\mathbf{p}_j, \mathbf{A}^{-1}\mathbf{E}_j(k)\mathbf{p}_j \right\rangle \\
&= \left\langle (\mathbf{F}^N)^T\mathbf{F}^N - (\mathbf{F}^N)^T\mathbf{P}(k)]\,\mathbf{p}_j,\ [(\mathbf{F}^N)^T\mathbf{F}^N]^{-1}\,[(\mathbf{F}^N)^T\mathbf{F}^N - (\mathbf{F}^N)^T\mathbf{P}(k)]\,\mathbf{p}_j \right\rangle \\
&= \left\langle (\mathbf{F}^N)^T[(\mathbf{F}^N - \mathbf{P}(k))\,\mathbf{p}_j\,],\ [(\mathbf{F}^N)^T\mathbf{F}^N]^{-1}\,(\mathbf{F}^N)^T[(\mathbf{F}^N - \mathbf{P}(k))\,\mathbf{p}_j\,] \right\rangle \\
&= \left\langle (\mathbf{F}^N - \mathbf{P}(k))\,\mathbf{p}_j\,,\ \mathbf{F}^N[(\mathbf{F}^N)^T\mathbf{F}^N]^{-1}\,(\mathbf{F}^N)^T[(\mathbf{F}^N - \mathbf{P}(k))\,\mathbf{p}_j\,] \right\rangle \\
&= \left\langle (\mathbf{F}^N - \mathbf{P}(k))\,\mathbf{p}_j\,,\ [\mathbf{F}^N(\mathbf{F}^N)^\dagger]\,[(\mathbf{F}^N - \mathbf{P}(k))\,\mathbf{p}_j\,] \right\rangle
\end{aligned}$$

where $(\mathbf{F}^N)^\dagger$ is the Moore-Penrose generalised inverse or pseudo-inverse of $\mathbf{F}^N$. Thus when $\mathbf{F}$ is invertible (i.e. $(\mathbf{F}^N)^\dagger = (\mathbf{F}^N)^{-1}$) we have

$$\begin{aligned}
\|\mathbf{E}_j(k)\mathbf{p}_j\|_{\mathbf{A}^{-1}} &= \left\langle (\mathbf{F}^N - \mathbf{P}(k))\,\mathbf{p}_j\,,\ (\mathbf{F}^N - \mathbf{P}(k))\,\mathbf{p}_j\,) \right\rangle \\
&= \left\langle \mathbf{F}^N\mathbf{p}_j - \mathbf{P}(k)\mathbf{p}_j\,,\ \mathbf{F}^N\mathbf{p}_j - \mathbf{P}(k)\mathbf{p}_j \right\rangle \\
&= \|\mathbf{P}(k)\mathbf{p}_j - \mathbf{F}^N\mathbf{p}_j\|_2
\end{aligned}$$

□

In the general case there is no guarantee that $\mathbf{F}$ is invertible so we can only say that $\|\mathbf{P}(k)\mathbf{p}_j - \mathbf{F}^N\mathbf{p}_j\|_2$ may be a good approximation for $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$. We managed to replace a quantity which involves the $\mathbf{A}^{-1}$ norm to a quantity which needs the 2-norm. Obviously, what remains is to find an approximation for $\mathbf{F}^N\mathbf{p}_j$. The idea is that we approximate $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ as

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} = \|\mathbf{P}(k)\mathbf{p}_j - \mathbf{F}^N\mathbf{p}_j\|_2 \approx \|\mathbf{P}(k)\mathbf{p}_j - \mathbf{P}(k+1)\mathbf{p}_j\|_2 \tag{3.75}$$

which done by taking the difference of the successive Parareal iterates in 2-norm. As the Parareal iterations k keep increasing the iterate $\mathbf{P}(k+1)\mathbf{p}_j$ will keep getting closer and closer to $\mathbf{F}^N\mathbf{p}_j$ and thus we only have to run Parareal iterations to a point where the Parareal stopping criterion is satisfied (bounded by the value of ξ_j). That is, at each icg-iteration j we keep incrementing Parareal iterations k by 1 till the time the condition $\|\mathbf{P}(k)\mathbf{p}_j - \mathbf{P}(k+1)\mathbf{p}_j\|_2 \leq \xi_j$ is satisfied.
However as a compensation of using an approximation, we end up using one more Parareal iteration than needed since the latest Parareal iterate $\mathbf{P}(k+1)\mathbf{p}_j$ replaces $\mathbf{F}^N\mathbf{p}_j$ to compute the difference with the last Parareal iterate $\mathbf{P}(k)\mathbf{p}_j$. Fig. 3.2

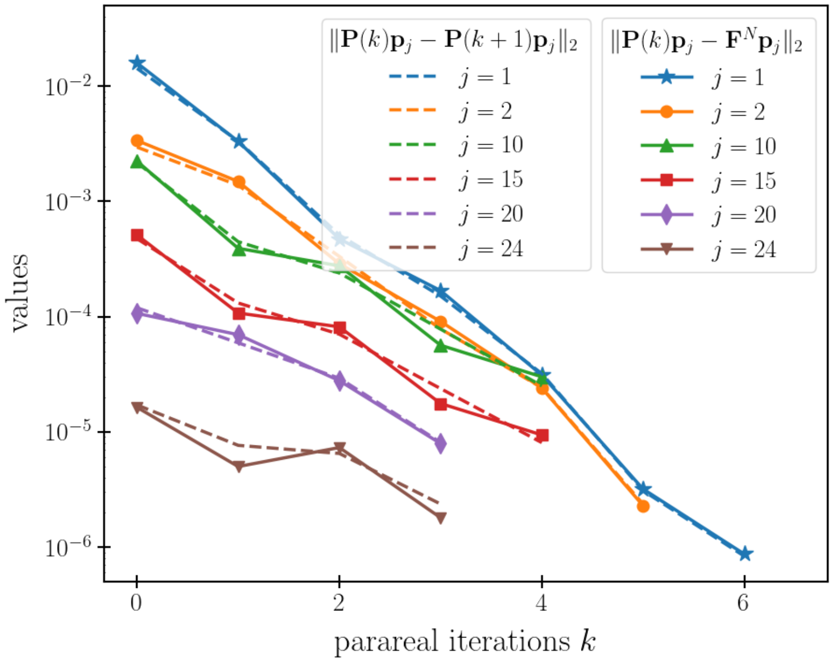


Figure 3.2: Comparison of the exact (solid lines) and approximate (dashed lines) $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ for given icg-iterations j

illustrates the idea where the exact $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ values are plotted against the approximations from the last Parareal iterate for the case of the 1D shallow water model which will be our centre of discussion in Chapter 4. The norm values are chosen for icg-iterations $1, 2, 10, 15, 20$ and 24 with the respective Parareal iterations k (on the x-axis) needed to satisfy the Parareal stopping criterion by ξ_j. It can be seen that the both the values follow closely to each other and we have a very nice approximation of $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ at the end of the Parareal iteration.

Switching to the modified Parareal criterion also changes the way of computing the inaccuracy budget which was discussed in subsection 3.3.3. Suppose now for a given ϕ_{j+1} we obtain $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} = \hat{\xi}_j \leq \xi_j$. The corresponding ξ_j is obtained from $\hat{\xi}_j(\hat{\phi}_{j+1})$ in (3.62) as

$$\hat{\phi}_{j+1} = \frac{(\|\mathbf{p}_j\|_{\mathbf{A}} - \hat{\xi}_j)}{\hat{\xi}_j} \frac{\sqrt{\epsilon}\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_j\|_{\mathbf{A}}}{2\|\mathbf{r}_j\|_2^2} > \phi_{j+1} \tag{3.76}$$

Everything else is computed in the same manner as for the original criterion except for (3.76) which replaces (3.55).

Remark 3.6. *After satisfying the Parareal stopping tolerance ξ_j, we already have a part of the matrix-vector product in the form of the last Parareal iterate $\mathbf{P}(k)\mathbf{p}_j$. We just need to store it and apply $(\mathbf{F}^N)^T$ to it to obtain the full-matrix vector product $[(\mathbf{F}^N)^T\mathbf{P}(k)]\mathbf{p}_j$. Algorithm 8 explains how everything is implemented.*

Algorithm 8 APPROXIMATION FOR $\mathbf{F}^N\mathbf{p}_j$

```
1: Given: icg-iteration j, direction vector p_j
2: Run 2 Parareal iterations for initial state p_j
3: Set k = 1
4: while true do
5:   Compute e = ||P(k)p_j − P(k+1)p_j||_2
6:   if e > ξ_j then
7:     Run one more Parareal iteration and set k = k + 1
8:   else
9:     Set ξ̂_j = ||P(k)p_j − P(k+1)p_j||_2
10:  end if
11: end while
12: Store the Parareal iterate P(k)p_j for computing the matrix-vector product
```

The final algorithm (Algorithm 9) shows all the manipulations done by including the inaccuracy budget (3.76), approximations from subsection 3.3.2 and Parareal stopping criterion involving $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$.

Algorithm 9 ICG_PARA_PRAC

```
1: Given: Symmetric positive definite matrix A ∈ ℝ^{n×n}, right hand side vector b ∈ ℝ^n, tolerance ϵ_icg
2: Set δx_0 = 0, r_0 = −b, p_0 = r, β_0 = ||b||_2^2, u_1 = b/β_0, φ_0 = j_max, Φ_0 = 1
3: for j = 0, ··· j_max do
4:   Compute the approximation to ||p_j||_A
5:   Determine ξ_j from the equation (3.62)
6:   for k = 1, ··· , N − 1 do
7:     Run Parareal with k + 1 iterations
```

Calculate $e = \|\mathbf{P}(k+1)\mathbf{p}_j - \mathbf{P}(k)\mathbf{p}_j\|_2$
if $e < \xi_j$ **then**
Set $\hat{\xi}_j = e$
break
end if
end for
Compute the product $\mathbf{c}_j = (F^N)^T \mathbf{P}(k)\mathbf{p}_j$
Find $\hat{\phi}_j$ from $\hat{\xi}_j$
$\Phi_{j+1} = \Phi_j - \hat{\phi}_j^{-1}$
if $j < j_{\max}$ **then**
$\phi_{j+1} = (j_{\max} - j)/\Phi_{j+1}$
else
$\phi_{j+1} = \phi_j$
end if
$\alpha_j = \beta_j / \mathbf{p}_j^T \mathbf{c}_j$
$\delta\mathbf{x}_{j+1} = \delta\mathbf{x}_j + \alpha_j \mathbf{p}_j$
$\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j \mathbf{c}_j$
$J_{j+1} = \frac{1}{2}\mathbf{b}^T \delta\mathbf{x}_{j+1}$
if $(J_{j+1-d} - J_{j+1}) \leq \frac{1}{4}\,\epsilon_{\text{icg}} |J_{j+1}|$ **then**
break
end if
if (reorth) **then**
for $i = 1, \cdots, j$ **do**
$\mathbf{r}_{j+1} = \mathbf{r}_{j+1} - (\mathbf{u}_i^T \mathbf{r}_{j+1})\mathbf{u}_i$
end for
$\beta_{j+1} = \mathbf{r}_{j+1}^T \mathbf{r}_{j+1}$
$\mathbf{u}_{j+1} = \mathbf{r}_{j+1} / \sqrt{\beta_{j+1}}$
else
$\beta_{j+1} = \mathbf{r}_{j+1}^T \mathbf{r}_{j+1}$
end if
$\mathbf{p}_{j+1} = -\mathbf{r}_{j+1} + (\beta_{j+1}/\beta_j)\mathbf{p}_j$
end for

In the next chapter we are going to analyse the results of the Parareal and data assimilation coupling using various settings of the minimisation algorithm. We put all the variants of the conjugate gradient method together in the Table 3.1 describing the names, their acronyms and their meanings.

Name	**Acronym**	**Meaning**
Conjugate gradient method	CG	This is the usual algorithm which uses exact matrix-vector products.
Conjugate gradient method using Parareal	CG_Para	This algorithm uses the same framework of CG except that the matrix-vector products are used by Parareal. A fixed tolerance ϵ_{p} for Parareal is used.
Inexact conjugate gradient method using Parareal	ICG_Para	In this variant of the algorithm $\lVert \mathbf{E}_j \mathbf{p}_j \rVert_{\mathbf{A}^{-1}}$ is computed exactly which serves as the stopping criterion of Parareal
Inexact conjugate gradient method using Parareal with approximated $\lVert \mathbf{E}_j \mathbf{p}_j \rVert_{\mathbf{A}^{-1}}$	ICG_Para_Approx	This variant uses the approximation of $\lVert \mathbf{E}_j \mathbf{p}_j \rVert_{\mathbf{A}^{-1}}$ using the last Parareal iterate
Inexact conjugate gradient method using Parareal with all approximations	ICG_Para_Prac	This version uses all the practical approximations along with the approximated $\lVert \mathbf{E}_j \mathbf{p}_j \rVert_{\mathbf{A}^{-1}}$

Table 3.1: Table of all the versions of the conjugate gradient used

3.5 Parareal in both directions (forward and adjoint)

In this manuscript, we focus on the integration of the forward model by a time parallel method (Parareal). In the context of variational data assimilation, a realistic application would also require the parallelisation of the adjoint model to compute the gradient of the cost function. In this section, we propose some ways to extend our work to the adjoint model.

So far when we calculate the gradient of the cost function (3.33) using the Parareal algorithm for the forward integration, we use the exact adjoint $(\mathbf{F}^N)^T$. Our modified inexact CG method is designed accordingly in a way that all the manipulations and approximations are done for the case when the forward model integration is done parallel in time.
The original cost function is defined by

$$J(\mathbf{x}_0) = \frac{1}{2}||\mathbf{F}^N\mathbf{x}_0 - \mathbf{y}||^2$$

and can be approximated by

$$\tilde{J}(\mathbf{x}_0) = \frac{1}{2}||\mathbf{P}(k)\mathbf{x}_0 - \mathbf{y}||^2,$$

whose gradient is given by

$$\nabla\tilde{J}(\mathbf{x}_0) = [\mathbf{P}(k)]^T(\mathbf{P}(k)\mathbf{x}_0 - \mathbf{y}).$$

In this formulation the adjoint of the Parareal integrator $\mathbf{P}(k)$ is used for the backward integration. This formulation provides some advantages, in particular that the corresponding matrix $[\mathbf{P}(k)]^T\mathbf{P}(k)$ is exactly symmetric and so is appropriate for a conjugate gradient method.
But more generally, the gradient of the original cost function

$$\nabla J(\mathbf{x}_0) = (\mathbf{F}^N)^T(\mathbf{F}^N\mathbf{x}_0 - \mathbf{y})$$

can be approximated by

$$\nabla J(\mathbf{x}_0) \approx \mathbf{P}^*(k_*)\mathbf{P}(k)\mathbf{x}_0 - (\mathbf{F}^N)^T\mathbf{y} \tag{3.77}$$

where $\mathbf{P}^*(k_*)$ is another backward Parareal operator which has a different stopping criterion (not the same number of iterations, for instance). Thus, we can write our gradient as

$$\nabla J(\mathbf{x}_0) \approx \mathbf{P}^*(k)[\mathbf{P}(k)\mathbf{x}_0 - \mathbf{y}] \tag{3.78}$$

The following presents three possible ways of adressing the integration of the adjoint model.

3.5.1 Forward first, then adjoint

The most intuitive implementation of the forward and the adjoint Parareal is to first perform the forward Parareal till we reach convergence and then use the solution at the end of the last time window as the initial condition for the adjoint Parareal. Here two different stopping criterion can be used, one for the forward run ϵ_{fwd} and one for the adjoint run ϵ_{adj}. We have for $k = 1, 2, \cdots$

$$\begin{aligned} \mathbf{x}_0^{k+1} &= \mathbf{x}_0 \\ \mathbf{x}_{i+1}^{k+1} &= \mathbf{G}\mathbf{x}_i^{k+1} + (\mathbf{F} - \mathbf{G})\mathbf{x}_i^k, \quad i = 0, \cdots, N-1 \end{aligned} \tag{3.79}$$

The converged Parareal solution at the end of the time window after $\bar{k}$ iterations, $\mathbf{x}_N^{\bar{k}} = \mathbf{u}_N$ (say) would serve as the initial condition for the adjoint run. Thus, the adjoint Parareal is given as

$$\begin{aligned} \mathbf{u}_N^{k+1} &= \mathbf{u}_N \\ \mathbf{u}_i^{k+1} &= \mathbf{G}^T\mathbf{u}_{i+1}^{k+1} + (\mathbf{F}^T - \mathbf{G}^T)\mathbf{u}_{i+1}^k, \quad i = N-1, \cdots, 0 \end{aligned} \tag{3.80}$$

The steps to carry out this idea is provided in Algorithm 10. It can be seen that we do not gain more degrees of parallelism compared to the case when we use Parareal only for the forward integration since the same N number of cores are used by Parareal to do the backward integration. But we still reduce the overall computation time since the adjoint is also parallelised.

3.5.2 A single forward and adjoint run

Instead of waiting for the accurate forward solution to proceed for the backward run like in the previous method, we can do a single Parareal run cycling both the

Algorithm 10 SEPARATE FORWARD AND ADJOINT PARAREAL RUN

Given: Initial condition $\mathbf{x}_0$, propagators $\mathbf{F}$ and $\mathbf{G}$

- "Parareal for the forward model"
  - Coarse initialisation
$$\mathbf{x}_i^0 = \mathbf{G}^i \mathbf{x}_0, \quad i = 0, \cdots, N \tag{3.81}$$
  - Run forward Parareal using (3.79) for given tolerance ϵ_{fwd}
- At convergence (say at $\bar{k}$ iterations), set $\mathbf{u}_N^{\bar{k}} = \mathbf{u}_N$
- "Parareal for the adjoint model"
  - Coarse initialisation
$$\mathbf{u}_i^0 = (\mathbf{G}^T)^{N-i} \mathbf{u}_N, \quad i = N-1, \cdots, 0 \tag{3.82}$$
  - Run adjoint Parareal using (3.80) for given tolerance ϵ_{adj}.

---

forward and backward integrations. This can be visualised as if we have spread the cycle in a single interval where the Parareal is run for $2N$ time windows and the propagators are changed in between (see Fig. 3.3). The difference here is that we start the adjoint computation as soon as we receive the forward solution. The initialisation is done as follows

$$\mathbf{x}_{i+1}^0 = \begin{cases} \mathbf{G}\mathbf{x}_i^0, & 0 \le i \le N \\ \mathbf{G}^T \mathbf{x}_i^0, & N \le i \le 2N-1 \end{cases} \tag{3.83}$$

and the iterates are updated as

$$\mathbf{x}_{i+1}^{k+1} = \begin{cases} \mathbf{G}\mathbf{x}_i^{k+1} + (\mathbf{F} - \mathbf{G})\mathbf{x}_i^k, & 0 \le i \le N \\ \mathbf{G}^T \mathbf{x}_i^{k+1} + (\mathbf{F}^T - \mathbf{G}^T)\mathbf{x}_i^k, & N \le i \le 2N-1 \end{cases} \tag{3.84}$$

This implementation is described in Algorithm 11. If we use Parareal for both the directions as a single loop we can use twice more the number of cores compared to the previous case of individual Parareal forward and adjoint runs. An important thing to note is that this approach will lead to a synchronous implementation since the coarse solvers $\mathbf{G}$ and $\mathbf{G}^T$ still have to integrated sequentially.

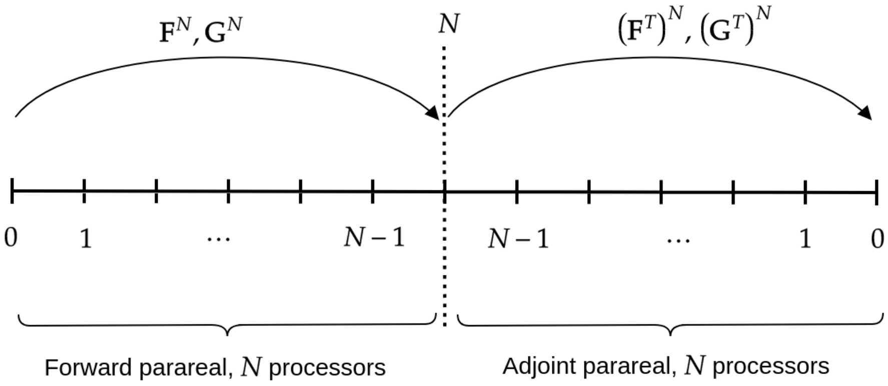


Figure 3.3: Synchronous run of the forward and adjoint Parareal

Algorithm 11 COUPLED FORWARD AND ADJOINT PARAREAL RUN

Given: Initial condition $\mathbf{x}_0$, propagators $\mathbf{F}$ and $\mathbf{G}$, stopping tolerance ϵ_{p}
$k = 0$
for $i = 0, \cdots, 2N-1$ **do**
    Obtain initial Parareal configuration using (3.83)
end for
for $k = 1, \cdots, N$ **do**
    Run the coupled Parareal using (3.84)
    if $|\mathbf{x}_{2N}^{k+1} - \mathbf{x}_{2N}^{k}| \leq \epsilon_{\mathrm{p}}$ **then**
        break
    end if
end for

3.5.3 Simultaneous forward and backward Parareal

Our final approach is to first run the sequential coarse solver for initialising the forward and the backward states and then perform one forward Parareal iteration. Once this has been done, the second forward Parareal iteration and the first backward Parareal iteration can be run independently of each other. The backward model can use the latest initial condition from the forward iteration and it can catch up on its own pace. This kind of implementation is asynchronous in nature because we are now not waiting for the current parallel forward run to update our initial condition for the backward run. We just take whatever we have from the previous iterates. The first few steps of this idea are described below:

- Run forward coarse initialisation $\mathbf{G}$
- Run adjoint coarse initialisation $\mathbf{G}^T$
- Compute fine solutions $\mathbf{F}\mathbf{x}_0^0$ and $\mathbf{F}^T\mathbf{x}_N^0$
- Run the coarse propagators $\mathbf{G}$ and $\mathbf{G}^T$
- synchronise
- Compute fine solutions $\mathbf{F}\mathbf{x}_0^1$ and $\mathbf{F}^T\mathbf{x}_N^1$
- Continue till convergence

With this implementation after one global Parareal iteration (both forward and backward), the coarse propagators can also be run in parallel. Thus the prediction step at each Parareal update iteration is no longer sequential and can be run in parallel. A synchronisation point can be added after certain Parareal iterations to use the latest iterate for the backward coarse integration.

It is also known that the cost of the adjoint run is 2-3 times more than the tangent linear run so load balancing could be done by assigning more number of processors for the adjoint run.

So far what we described in the above three approaches is simply a Parareal algorithm for a parallel forward-adjoint run. What has not been described yet is how to actually use them in the context of the data assimilation problem. If we are going to use the inexact CG, it is crucial that we find an estimate for $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ and also an another alternative for the other approximations.

Chapter 4

Applications: Numerical experiments

Contents

Chapter 3 established all the methodology of incorporating both the Parareal time parallelisation method and the incremental 4D-Var data assimilation framework. To recollect, the use of Parareal for the forward integration leads to solving a linear system through an inexact conjugate gradient (inexact CG) method where the error in the matrix-vector product is controlled by the Parareal's accuracy. For a practical implementation we also provided some feasible estimates for the various norms involved in the inexact CG method. In this chapter, we move to the application part by running different versions of the exact and the inexact CG (see the list given in table 3.1 on some numerical applications, comparing them and presenting the results. For our illustrations we are going to use the linearised 1D and 2D shallow water equations since they are one of the simplest models which can describe some of the very important geophysical phenomena.

We start off by explaining briefly how the shallow water equations are derived from the Navier-Stokes equations in section 4.1. As required by the incremental 4D-Var procedure, we linearise the obtained non-linear equations (here around a state of rest) to get the linearised forms of the 1D and 2D shallow water equations. We then set up our 1D and 2D numerical models by first semi-discretising the equations in space and then utilising the implicit theta scheme for the temporal discretisation. Section 4.2 describes the Parareal propagators which are constructed using the same implicit temporal scheme.

In the next section 4.3 we define the respective data assimilation problems for the 1D and the 2D case. In both the problems we want to retrieve the initial state when the system observes the full state vector only at the end of the integration time. In section 4.4 we discuss in much detail about the Krylov subspace enhanced Parareal which we use for the more complex and larger sized 2D problem. In particular, we address the relevancy of using a coarse solver (in addition to the Krylov basis) and we also detail the practical construction of the orthonormal basis required by the Krylov procedure. We show that a Gram-Schmidt algorithm with an additional reorthogonalisation step allows to maintain the accuracy for a reduced computational cost.

Section 4.5 is dedicated to the results of the numerical experiments carried out on the 1D and 2D models by running the different variants of the (inexact) conjugate gradient method. We first try to use Parareal without using the inexact CG bounds, i.e. with a fixed (and arbitrarily chosen) Parareal tolerance. By hit and trial we choose a Parareal stopping criterion where the method exhibits almost the same behaviour than the exact CG till convergence. Next we take benefit from the theoretical results of the inexact CG method to adopt an adaptive stopping criterion for Parareal. To check the relevancy of the approach, we start with a case where all the required norms are exactly computed. We then move to a practical implementation and show how the different norms are approximated and how it affects the behaviour of the method. The final result shows that after applying all the proposed approximations to the various norms for a practical implementation, inexact CG retains the same level of performance. The performances are measured in terms of the total number of Parareal iterations, the average number of Parareal iterations per exact/inexact CG iterations and the parallel speedup.

Before we proceed, we make the reader aware that we are going to use the notation "$\mathbf{x}$" for the increment "$\delta\mathbf{x}$" throughout the chapter for simplicity.

4.1 Shallow water equations

The shallow water equations [Holton, 1973, Gill, 1982, Vallis, 2017] are a set of equations which describe a thin layer of a homogeneous fluid in hydrostatic balance derived from the rotating Navier-Stokes equations. The name shallow water also emphasise the assumption that the horizontal length scale is much larger than the vertical scale (called the long-wave approximation in literature) in order for the hydrostatic approximation to be valid.

Remark 4.1 (Hydrostatic approximation)**.** *The hydrostatic assumption states that when the fluid is static in the vertical direction z, the pressure gradient force is balanced by the force due to gravity. In other words,*

$$\frac{\mathrm{d}p}{\mathrm{d}z} = -\rho g \tag{4.1}$$

where p is the pressure and ρ is the fluid's density.

We consider the case of a single layer fluid of constant density ρ_0. In this approximation, the pressure is simply given by $p(x, z, t) = p_0 + \rho_0 g(\eta(x, t) - z)$ where $\eta(x, t)$ is the free surface elevation. The horizontal pressure gradient is thus proportional to the horizontal free surface gradient. The total depth of the fluid $h(x, t)$ is given by $h(x, t) = H + \eta(x, t)$ where H is a constant reference depth (flat bottom). The shallow water equations for the horizontal velocity $\mathbf{u} = (u(x, t), v(x, t))^T$ are written as

$$\begin{aligned} \frac{\mathrm{D}h}{\mathrm{D}t} + h\,\nabla \cdot \mathbf{u} &= 0 && \text{(continuity equation)} \\ \frac{\mathrm{D}u}{\mathrm{D}t} + \mathbf{f} \times \mathbf{u} &= -g\,\nabla\eta && \text{(momemtum equation)} \end{aligned} \tag{4.2}$$

where $\mathbf{f} = f\,\hat{\mathbf{k}}$ is the force due to earth's rotation and

$$\frac{\mathrm{D}}{\mathrm{D}t} \equiv \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla \tag{4.3}$$

is the material derivative.

4.1.1 1D shallow water model

First we consider the one dimensional case (See Fig. 4.1) in the absence of rotation ($\mathbf{f} = 0$) and thus equations (4.2) become

$$\begin{aligned} \frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} &= -h\,\frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} &= -g\,\frac{\partial \eta}{\partial x} \end{aligned} \tag{4.4}$$

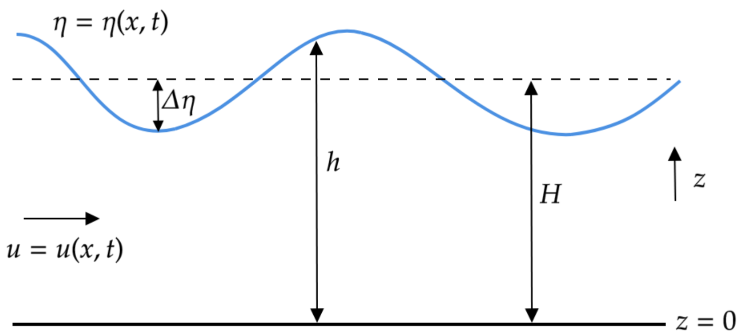


Figure 4.1: A typical one dimensional shallow water system with a flat bottom

Since we use the tangent linear model in the incremental 4D-Var approach, we linearise the above equations around a state at rest ($u_0 = 0$). That is we use the perturbation forms of the dependent variables and assume that the fluid surface is at rest (no initial velocity)

$$h = H + \eta, \qquad u = u_0 + u' = u', \qquad |\eta| \ll H,\ |u'| \ll 1 \tag{4.5}$$

Using the values of u, v and h from (4.5) we get

$$\begin{aligned} \frac{\partial \eta}{\partial t} + u'\frac{\partial \eta}{\partial x} &= -(\eta + H)\,\frac{\partial u'}{\partial x} \\ \frac{\partial u'}{\partial t} + u'\frac{\partial u'}{\partial x} &= -g\,\frac{\partial \eta}{\partial x} \end{aligned} \tag{4.6}$$

Neglecting the smaller value terms and replacing u' by u, the linearised one-dimensional shallow water equations are given by

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= -H\frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial t} &= -g\frac{\partial \eta}{\partial x} \end{aligned} \tag{4.7}$$

Lastly, we add an explicit diffusion term to the velocity equation with the help of averaging or smoothing operators [Reynolds, 1895] This is done in order to account for the affects of the unresolved small scales in terms of the resolved large scales also known as the closure of the system [Cushman-Roisin and Beckers, 2011, Wilcox et al., 1998]. In the end our model is

$$\begin{aligned} \frac{\partial \eta}{\partial t} &= -H\frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial t} &= -g\frac{\partial \eta}{\partial x} + \mu\frac{\partial^2 u}{\partial x^2} \end{aligned} \tag{4.8}$$

where μ is the diffusion constant. The physical domain $\Lambda = [0, L]$ is closed and the following physical values have been chosen and provided in table 4.1:

Length of the basin	$L = 120\text{m}$
Reference depth	$H = 0.9\text{m}$
Gravity	$g = 10\text{m.s}^{-2}$
Diffusion constant	$\mu = 0.15\text{m}^2.\text{s}^{-1}$

Table 4.1: Domain and physical parameters for the linearised 1D shallow water model

Spatial discretisation

We here introduce the spatial discretisation of the shallow water model (4.8). To achieve this we are going to consider a staggered grid discretisation [Durran, 2013] as shown in Fig. 4.2.

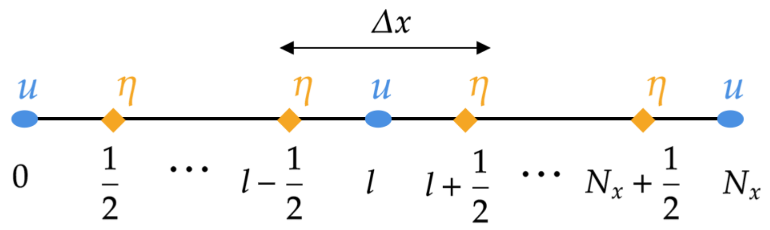


Figure 4.2: Staggered grid discretisation for 1D shallow water model

Let Δx be the spatial stepping size and N_x be the total number of spatial grid points. Keeping the index l for the spatial grid and using a second order centered finite difference scheme in space leads to

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}(\eta_{l+1/2}) &= -H\left(\frac{u_{l+1}-u_l}{\Delta x}\right), \qquad l = 0, \cdots, N_x - 1 \\
\frac{\mathrm{d}}{\mathrm{d}t}(u_l) &= -g\left(\frac{\eta_{l+1/2}-\eta_{l-1/2}}{\Delta x}\right) + \mu\left(\frac{u_{l+1}-2u_l+u_{l-1}}{\Delta x^2}\right), \\
& \qquad\qquad l = 1, \cdots, N_x - 1
\end{aligned}
\tag{4.9}
$$

The above discretisation (4.9) can be expressed as an ODE system

$$
\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{C}\,\mathbf{x} \tag{4.10}
$$

where

$$
\mathbf{C} = \left(\begin{array}{ccccc|ccccc}
 & & & & & -H/\Delta x & & & & \\
 & & & & & H/\Delta x & -H/\Delta x & & & \\
 & & \mathbf{0} & & & & \ddots & \ddots & & \\
 & & & & & & & H/\Delta x & -H/\Delta x & \\
 & & & & & & & & H/\Delta x & -H/\Delta x \\
\hline
g/\Delta x & -g/\Delta x & & & & -2\mu/\Delta x^2 & \mu/\Delta x^2 & & & \\
 & g/\Delta x & -g/\Delta x & & & \mu/\Delta x^2 & -2\mu/\Delta x^2 & \mu/\Delta x^2 & & \\
 & & \ddots & \ddots & & & \ddots & \ddots & \ddots & \\
 & & & g/\Delta x & -g/\Delta x & & & \mu/\Delta x^2 & -2\mu/\Delta x^2 & \mu/\Delta x^2 \\
 & & & & g/\Delta x & & & & \mu/\Delta x^2 & -2\mu/\Delta x^2
\end{array}\right)
$$

and $\mathbf{x} = \left(\eta_{1/2}, \dots \eta_{N_x-1/2}, u_1, \dots u_{N_x-1}\right)^T$.

Note that in our computational grid, the velocity values are placed on the boundary. We use the closed boundary conditions for our shallow water discretisation and therefore the velocity values on the boundary are set to zero i.e. $u_0 = u_{N_x} = 0$ (See Fig. 4.3). For this reason we do not include the boundary values in the state vector $\mathbf{x}$ when we do the computation for the inner grid point values. Thus $\mathbf{x}$ contains N_x grid point solutions of η and $N_x - 1$ grid point solutions of u and so the size of $\mathbf{x}$ becomes $n = N_x + N_x - 1 = 2N_x - 1$.

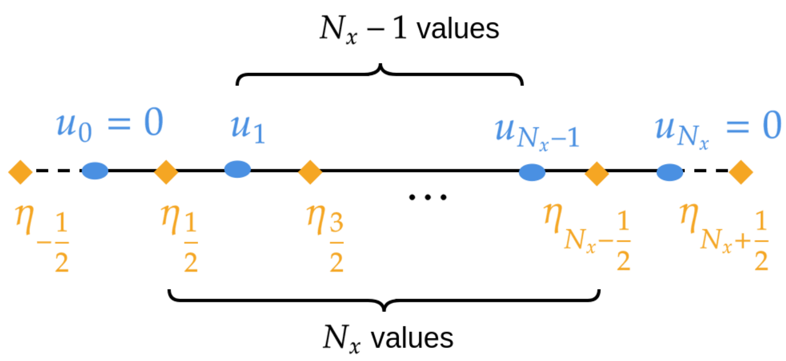


Figure 4.3: The discretised grid along with the boundary values u_0 and u_{N_x}. The grid points on the dashed line represents values outside the computational domain.

In the numerical experiments, we will use a number of grid points N_x equal to 120 ($\Delta x = L/N_x = 1$m). The dimension of the state vector is thus equal to $n = 2N_x - 1 = 239$.

4.1.2 2D shallow water model

The 1D shallow water model is useful to illustrate the methodology but is simplified both in terms of physical phenomena included (gravity waves) and also in terms of computational requirements (low dimension). A more challenging application of studying the oceanic phenomena is the extension to a linearised 2D shallow water model which brings in more parameters to play with and also more complexity in terms of solving it numerically.

Now instead of having only horizontal velocity u in the x-direction in our model we also have a horizontal velocity v in the y-direction. Since we are no longer in one dimension we have to incorporate the apparent forces due to the rotation of the earth (thus considering a rotating reference frame) most notably the Coriolis force $\mathbf{f}$ [Persson, 1998]. As a result some important geophysical phenomena such as the propagation of Kelvin waves and geostrophic balance can be modelled. Since we will use a constant Coriolis parameter, the model will however not include Rossby waves.

The Coriolis force is expressed as $\mathbf{f} = (-fv, fu)^T$ where $f \equiv 2\Omega \sin \varphi$ is the vertical component of the rotation vector $\mathbf{\Omega}$ and is called the Coriolis parameter. φ is the latitude and $\Omega = 7.292 \times 10^{-5}\, rad\, s^{-1}$ is the angular speed of rotation of the Earth. In the mid latitude regions the Coriolis force varies a lot with a slight change in the latitude. Two approximations can be made depending on the curvature of the earth by the f-plane and the β-plane approximations. The f-plane approximation assumes that the Coriolis parameter is a constant value i.e. $f = f_0$. The β-plane approximation allows the Coriolis parameter to change linearly in y (latitude). To do so we use the Taylor's expansion on f about a reference latitude φ_0. We have

$$f = 2\Omega \sin(\varphi_0 + \varphi) \approx 2\Omega \sin\varphi_0 + 2\Omega(\varphi - \varphi_0)\cos\varphi_0 \equiv f_0 + \beta y \tag{4.11}$$

where

$$\beta = \frac{\partial f}{\partial y}\Big|_{\varphi_0} = \frac{2\Omega}{a}\cos\varphi_0 \tag{4.12}$$

Here a is the Earth's radius and we have related y to the latitude φ using the relation $\mathrm{d}y = a\mathrm{d}\varphi$. As mentioned above, in our study we are going to use the f-plane approximation. Now considering the shallow water model (4.2) for the case of 2 dimensions we have

$$\begin{aligned}
\frac{\partial h}{\partial t} + u\frac{\partial h}{\partial x} + v\frac{\partial h}{\partial y} &= -h\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)\\
\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= -g\frac{\partial \eta}{\partial x} + f_0 v\\
\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= -g\frac{\partial \eta}{\partial y} - f_0 u
\end{aligned} \tag{4.13}$$

As done for the 1D model, we linearise (4.13) around the state of rest ($\mathbf{u}_0 = (u_0, v_0)^T = (0, 0)^T$)

$$h = H + \eta, \quad u = u_0 + u', \quad v = v_0 + v', \quad |\eta| \ll H, |u'| \ll 1, |v'| \ll 1 \tag{4.14}$$

to get

$$\begin{aligned}
\frac{\partial \eta}{\partial t} + u'\frac{\partial \eta}{\partial x} + v'\frac{\partial \eta}{\partial y} &= -(\eta + H)\left(\frac{\partial u'}{\partial x} + \frac{\partial v'}{\partial y}\right)\\
\frac{\partial u'}{\partial t} + u'\frac{\partial u'}{\partial x} + v'\frac{\partial u'}{\partial y} &= -g\frac{\partial \eta}{\partial x} + f_0 v'\\
\frac{\partial v'}{\partial t} + u'\frac{\partial v'}{\partial x} + v'\frac{\partial v'}{\partial y} &= -g\frac{\partial \eta}{\partial y} - f_0 u'
\end{aligned} \tag{4.15}$$

Neglecting the small value quantities, the 2D linearised shallow water equations are written as

$$\begin{aligned}\frac{\partial \eta}{\partial t} &= -H\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\\ \frac{\partial u}{\partial t} &= f_0 v - g\frac{\partial \eta}{\partial x}\\ \frac{\partial v}{\partial t} &= -f_0 u - g\frac{\partial \eta}{\partial y}\end{aligned} \tag{4.16}$$

The physical domain $\Lambda = [0, L_x] \times [0, L_y]$ is closed as well and the following physical parameters are taken and provided in table 4.2:

Length of the basin	$L_x = L_y = 2 \times 10^5$m
Reference depth	$H = 100$m
Gravity	$g = 2$m.s^{-2}
Coriolis parameter	$f_0 = 5 \times 10^{-3}$s^{-1}
Rossby radius	$\sqrt{gH}/f_0 = 2828.43$m

Table 4.2: Domain and physical parameters for the linearised 2D shallow water model

Spatial discretisation

To numerically solve (4.16), we use the staggered Arakawa C-grid discretisation (Fig. 4.4) [Arakawa and Lamb, 1977, Randall, 1994] which leads to

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}(\eta_{l+1/2,m+1/2}) &= -H\left(\frac{u_{l+1,m+1/2}-u_{l,m+1/2}}{\Delta x}+\frac{v_{l+1/2,m+1}-v_{l+1/2,m}}{\Delta y}\right),\\ &\qquad l=0,\cdots,N_x-1;\ m=0,\cdots,N_y-1\\ \frac{\mathrm{d}}{\mathrm{d}t}(u_{l,m+1/2}) &= f_0\left(\frac{v_{l-1/2,m}+v_{l+1/2,m}+v_{l-1/2,m+1}+v_{l+1/2,m+1}}{4}\right)\\ &\quad - g\left(\frac{\eta_{l+1/2,m+1/2}-\eta_{l-1/2,m+1/2}}{\Delta x}\right)\\ &\qquad l=1,\cdots,N_x-1;\ m=0,\cdots,N_y-1\\ \frac{\mathrm{d}}{\mathrm{d}t}(v_{l+1/2,m}) &= -f_0\left(\frac{u_{l,m-1/2}+u_{l,m+1/2}+u_{l+1,m-1/2}+v_{l+1,m+1/2}}{4}\right)\\ &\quad - g\left(\frac{\eta_{l+1/2,m+3/2}-\eta_{l+1/2,m+1/2}}{\Delta y}\right)\\ &\qquad l=0,\cdots,N_x-1;\ m=1,\cdots,N_y-1\end{aligned} \tag{4.17}$$

As in the 1D case, the above semi-discretised equations can be written together as a linear ODE system

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{C}\mathbf{x} \tag{4.18}$$

by stacking the discrete grid point unknowns $\boldsymbol{\eta} = (\eta_{l,m})$, $\mathbf{u} = (u_{l,m})$, and $\mathbf{v} = (v_{l,m})$ in a single vector $\mathbf{x} = (\boldsymbol{\eta}, \mathbf{u}, \mathbf{v})^T$. $\mathbf{C}$ is the coefficient matrix of the right hand side of the semi-discretised equations.

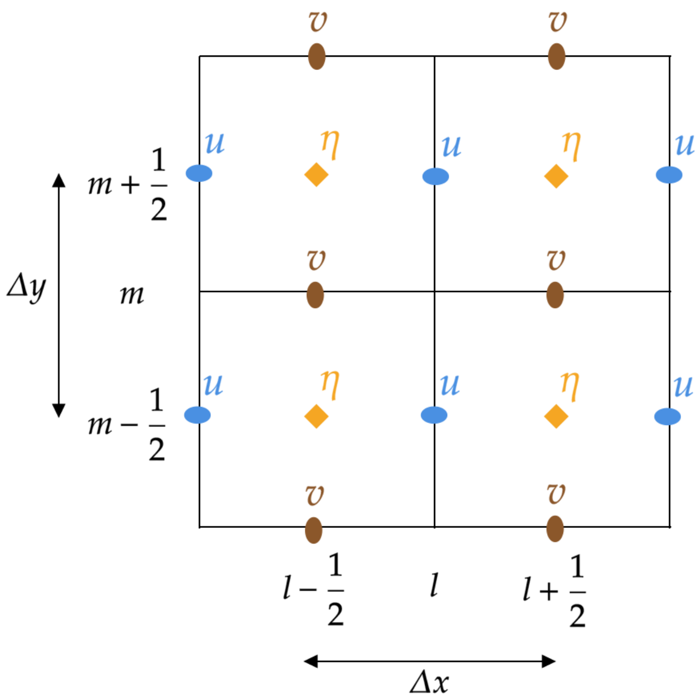


Figure 4.4: Arakawa C-grid discretisation for 2D shallow water model

In the numerical experiments, we will use the same number of grid points for both the horizontal domain setting $N_x = N_y = 80$ and thus $\Delta x = L_x/N_x = 2500\text{m}$, $\Delta y = L_y/N_y = 2500\text{m}$. As we are using closed boundary conditions we do not include the u and v values at the boundary in our state vector $\mathbf{x}$ and so the dimension of the state vector in this case is equal $n = N_xN_y + N_y(N_x - 1) + N_x(N_y - 1) = 19040$.

Remark 4.2. *An important point we make here is that while the 1D shallow water model has been implemented in Python, the 2D case uses Fortran90. In Python the matrix* $\mathbf{C}$ *can be stored explicitly and it is convenient to get its inverse and other related norms. In Fortran the matrix is only accessible through a subroutine* *which when passed a vector* $\mathbf{x}$ *returns the output as a matrix-vector product i.e.* $\mathbf{x} \mapsto \mathbf{C}\mathbf{x}$.

4.1.3 Temporal discretisation

For both the 1D and the 2D shallow water model, we see that doing the spatial discretisation leads to solving a linear system of ODE of the form $\mathbf{x}' = \mathbf{C}\mathbf{x}$. To solve this linear ODE numerically in time we introduce the temporal discretisation with the help of an implicit theta scheme. To do so we let q to be the index for the temporal grid and ΔT to be the time-stepping size. We have

$$\frac{\mathbf{x}^{q+1} - \mathbf{x}^{q}}{\Delta T} = \mathbf{C}[\,\theta \mathbf{x}^{q+1} + (1-\theta)\mathbf{x}^{q}\,], \qquad q = 0, 1, \cdots \tag{4.19}$$

where the θ is a parameter such that $0 \leq \theta \leq 1$. We can rewrite (4.19) as

$$\mathbf{x}^{q+1} = [\,\mathbf{I} - \theta \Delta T \mathbf{C}\,]^{-1}\,[\,\mathbf{I} + (1-\theta)\Delta T \mathbf{C}\,]\,\mathbf{x}^{q}, \qquad q = 0, 1, \cdots \tag{4.20}$$

According to the above time-stepping scheme in order to march ahead in time, we need to the invert the matrix $[\mathbf{I} - \theta \Delta T \mathbf{C}]$ at every time step. When working for the 1D case this can be easily done since the inverse matrix can be explicitly computed and stored in the memory. For the 2D case we have to use the GMRES algorithm (since the matrix $\mathbf{C}$ is non-symmetric) to find the inverse and solve the resulting system. We will explore more about the possibility of an efficient run of the GMRES in the next section where we talk about the parareal propagators.

4.2 Parareal parameters and propagators

Having obtained a temporal scheme (4.20) to solve the system of linear ODEs for both the 1D and the 2D case, our next step is to use it to construct the fine and the coarse propagators for implementing the Parareal algorithm. Let $\Omega = [0, T]$ be the time domain such that T is the total time period of integration. Let N be the total number of time windows and ΔT be the length of one time window. Now for each time window, we associate N_{fine} and N_{coarse} as the number of fine time steps and the number of coarse time steps respectively. Clearly N_{coarse} has to divide N_{fine}. If δt is the fine step length, then the coarse step length Δt can be calculated as

$$\Delta t = \frac{\Delta T}{N_{\text{coarse}}} = \frac{\delta t \times N_{\text{fine}}}{N_{\text{coarse}}} \tag{4.21}$$

The fine and coarse propagators are defined as

$$\begin{aligned} \mathbf{F} &= \left\{ [\mathbf{I} - \theta \delta t \mathbf{C}]^{-1} [\mathbf{I} + (1-\theta)\delta t \mathbf{C}] \right\}^{N_{\text{fine}}} \\ \mathbf{G} &= \left\{ [\mathbf{I} - \theta \Delta t \mathbf{C}]^{-1} [\mathbf{I} + (1-\theta)\Delta t \mathbf{C}] \right\}^{N_{\text{coarse}}} \end{aligned} \tag{4.22}$$

Note that the resulting total length of the numerical integrations for both the 1D and 2D case is simply

$$T = N_{\text{coarse}} \Delta t N = N_{\text{fine}} \delta t N \tag{4.23}$$

The Parareal parameter values used for both the models is put in table 4.3.

Parareal parameters	**1D model**	**2D model**
Number of time windows, N	20	40
Theta parameter, θ	0.51	0.51
Number of fine time steps per time window, N_{fine}	100	20
Number of coarse time steps per time window, N_{coarse}	20	5
Fine time step, δt	0.05s	68.763s
Coarse time step, Δt	0.25s	275.053s
Courant number, $\omega_{\max}$	0.29	1.1
Total time period of integration, T	100s	55010.4s

Table 4.3: Parareal parameters for the linearised 1D and 2D shallow water model

In the 1D shallow water model, we have used an explicit diffusion term to account for the effects of the small scales. The choice of the value of the diffusion constant also impacts the number of Parareal iterations and we show this in Fig. 4.5 when Parareal is used to solve for the system (4.10).

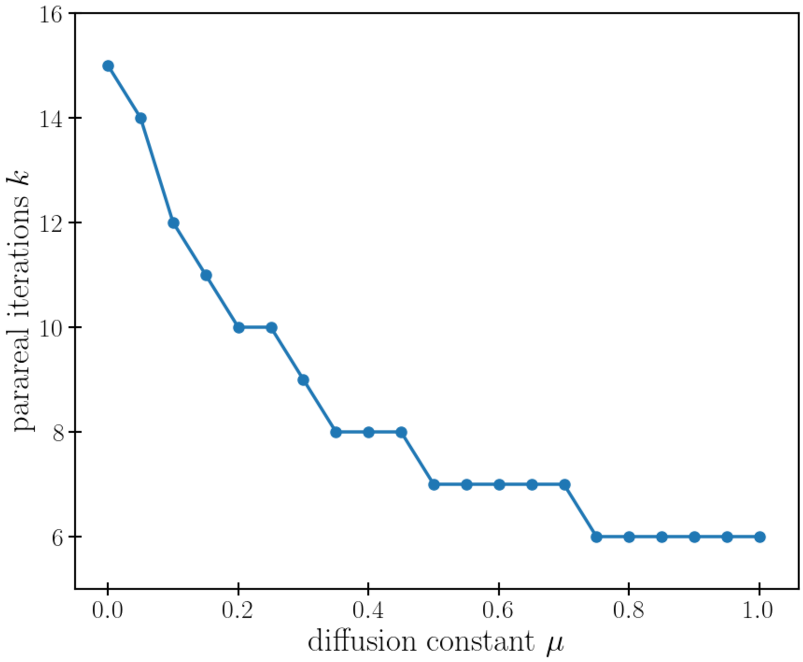


Figure 4.5: Parareal iterations needed as a function of the explicit diffusion constant μ

Note that the viscosity has to be chosen such that

$$\frac{\mu \delta t}{\Delta x^2} \leq \frac{1}{2} \tag{4.24}$$

where the quantity on the left is the parabolic Courant number.

It takes 15 Parareal iterations when no diffusion is used as compared to 6 Parareal iterations when $\mu = 1$ corresponding to the parabolic Courant number equal to 0.05.

For the 2D shallow water model, each time when we use the Parareal propagators we are calling the GMRES for the matrix inversion. We make a small manipulation within the Parareal algorithm by making use of the previous Parareal iterate which can improve its overall performance by reducing the number of GMRES iterations.

After the first integrations, for the fine parallel simulations, we use the linearity of $\mathbf{F}$ and compute $\mathbf{F}(\mathbf{x}^k - \mathbf{x}^{k-1})$ and deduce

$$\mathbf{F}(\mathbf{x}^k) = \mathbf{F}(\mathbf{x}^k - \mathbf{x}^{k-1}) + \mathbf{F}(\mathbf{x}^{k-1}) \tag{4.25}$$

instead of directly computing $\mathbf{F}(\mathbf{x}^k)$. Since $\|\mathbf{x}^k - \mathbf{x}^{k-1}\| \leq \|\mathbf{x}^{k-1}\|$ this lowers the number of GMRES iterations required as k increases which in turn allows to reduce the cost of the fine solver. When near to Parareal convergence, the successive Parareal iterates are so close that we can assume $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| \approx 0$. Then in that case it means that GMRES should converge in only 1 iteration. We

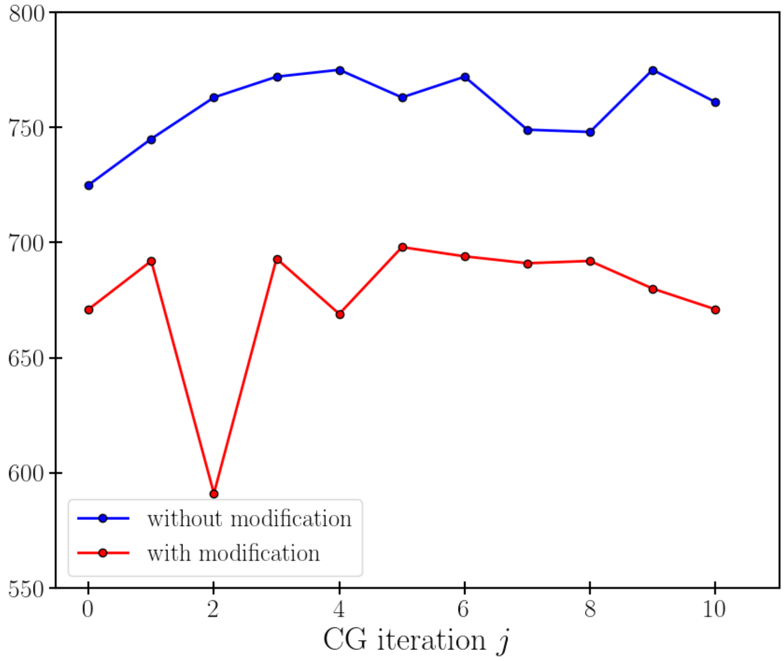


Figure 4.6: Average number of GMRES iterations per Parareal iteration (on y-axis) for each CG iteration j.

show the benefit of using this modification by plotting the evolution of the average number of GMRES iterations per Parareal iteration when used in the context of a CG run for the 2D shallow water model (See Fig. 4.6). The blue line shows the average GMRES iterations when no modifications are done and the red line shows the average GMRES iterations when modifications are applied. Clearly, we observe that the modifications helps to bring down the average number of GMRES iterations per Parareal iterations (below 700) and thus reduce the cost of the fine solver.

4.3 Data assimilation

In this section, we describe the parameters of the data assimilation procedure for both 1D and 2D case. They essentially differ in the form of the regularisation term used. Subsections 4.3.1 and 4.3.2 contain a standard description of the data assimilation problem for 1D and 2D model respectively without including Parareal and inexact CG. It is a sanity check and it represents the best we can hope for. Subsection 4.3.3 talks about approximating the gradients of the cost functions obtained from the two models by including Parareal and subsequently the inexact CG.

4.3.1 1D case

When the conditioning of $\mathbf{A}$ i.e. $\kappa(\mathbf{A})$ is large a regularisation term α can be used in the cost function which depends on the spatial derivative of the initial state $\mathbf{x}_0$. For our case without a regularisation, with the computational parameters given in section 4.1 and 4.2 $\kappa(\mathbf{A})$ has an approximate value of $\kappa(\mathbf{A}) \approx 2.98 \times 10^{18}$ which is extremely large. To improve the conditioning, we minimise the following regularised data assimilation cost function instead:

$$\min\ J(\mathbf{x}_0) = \frac{1}{2}\left\|\mathbf{F}^N\mathbf{x}_0 - \mathbf{y}\right\|_2^2 + \frac{\alpha}{2}\left\|\frac{\mathrm{d}\mathbf{x}_0}{\mathrm{d}x}\right\|_2^2 \tag{4.26}$$

We use the finite differences to discretise the spatial derivative of the regularisation term. With

$$\begin{aligned}\mathbf{x}_0 &= ((\mathbf{x}_0)_1, \cdots, (\mathbf{x}_0)_l, \cdots, (\mathbf{x}_0)_{2N_x-1})^T \\ &= ((\eta_0)_{1/2}, \cdots, (\eta_0)_{N_x-1/2}, (u_0)_1, \cdots, (u_0)_{N_x-1})^T\end{aligned} \tag{4.27}$$

then for $l = 1, \cdots, N_x$,

$$\frac{\mathrm{d}\mathbf{x}_0}{\mathrm{d}x} = \frac{(\eta_0)_{m+1/2} - (\eta_0)_{m-1/2}}{\Delta x}, \qquad m = 1, \cdots, N_x - 1 \tag{4.28}$$

and for $l = N_x + 1, \cdots, 2N_x - 1$

$$\frac{\mathrm{d}\mathbf{x}_0}{\mathrm{d}x} = \frac{(u_0)_{m+1} - (u_0)_m}{\Delta x}, \qquad m = 1, \cdots, N_x - 2 \tag{4.29}$$

Putting together we have

$$\frac{\mathrm{d}\mathbf{x}_0}{\mathrm{d}x} = \frac{1}{\Delta x}\mathbf{Q}_1\mathbf{x}_0 \tag{4.30}$$

where

$$\mathbf{Q}_1 = \left(\begin{array}{cccc|cccc} -1 & 1 & & & & & & \\ & -1 & 1 & & & \mathbf{0} & & \\ & & \ddots & \ddots & & & & \\ & & & -1 \quad 1 & & & & \\ \hline & & & & -1 & 1 & & \\ & \mathbf{0} & & & & -1 & 1 & \\ & & & & & & \ddots & \ddots \\ & & & & & & & -1 \quad 1 \end{array}\right) \tag{4.31}$$

Therefore,

$$\left\| \frac{\mathrm{d}\mathbf{x}_0}{\mathrm{d}x} \right\|_2^2 = \frac{1}{\Delta x^2} \left\langle \mathbf{Q}_1\mathbf{x}_0, \mathbf{Q}_1\mathbf{x}_0 \right\rangle = \mathbf{Q}_2\mathbf{x}_0 \tag{4.32}$$

where

$$\mathbf{Q}_2 = \frac{1}{\Delta x^2}\mathbf{Q}_1^T\mathbf{Q}_1 \tag{4.33}$$

With the addition of the regularisation term to the cost function, the updated gradient becomes

$$\nabla J(\mathbf{x}_0) = (\mathbf{F}^N)^T(\mathbf{F}^N\mathbf{x}_0 - \mathbf{y}) + \alpha\mathbf{Q}_2\mathbf{x}_0 \tag{4.34}$$

For data assimilation, the following parameters values are used as shown in table 4.4:

Regularisation constant	$\alpha = 10^{-5}$
Initial free surface value	$\eta_0(x) = \exp\left\{\left[(x - L/2)/(L/15)\right]^2\right\}$
Initial velocity value	$u_0(x) = 0 \ \forall x$

Table 4.4: Data assimilation parameters for the 1D shallow water model

Remark 4.3. *The cost function (4.26) can be written in the classical form as in chapter 1 by explicitly writing the inverse background covariance matrix* $\mathbf{B}^{-1}$ *and the inverse observation covariance matrix* $\mathbf{R}^{-1}$*. They can also be combined together and used as a weighed matrix* $\mathbf{W}$ *for the initial state* $\mathbf{x}_0$*. Here* $\mathbf{Q}_2$ *can be considered as a tridiagonal weighted matrix which acts as a smoother or a diffusion matrix.*

The conditioning of the regularised cost function after using the regularisation constant given in table 4.4 is $\kappa(\mathbf{A}) \simeq 10^6$. We now show three figures which provide a comparison between the free surface and velocity values at different times using the true initial state and the optimal initial state retrieved after the optimisation in data assimilation.

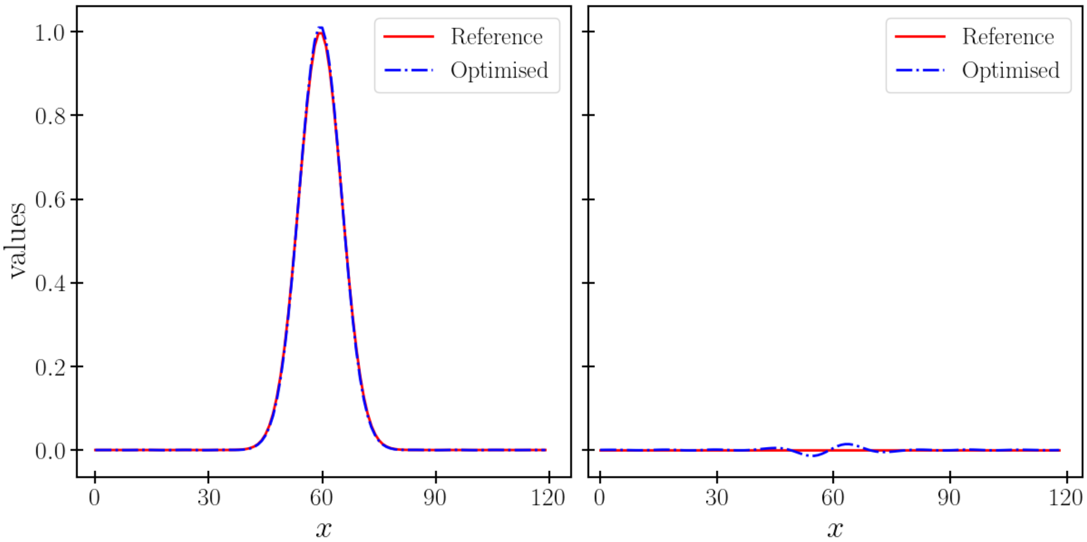


Figure 4.7: Free surface (left) and velocity (right) values at the initial time for the reference (solid red) and for the optimised (blue dashdot) states

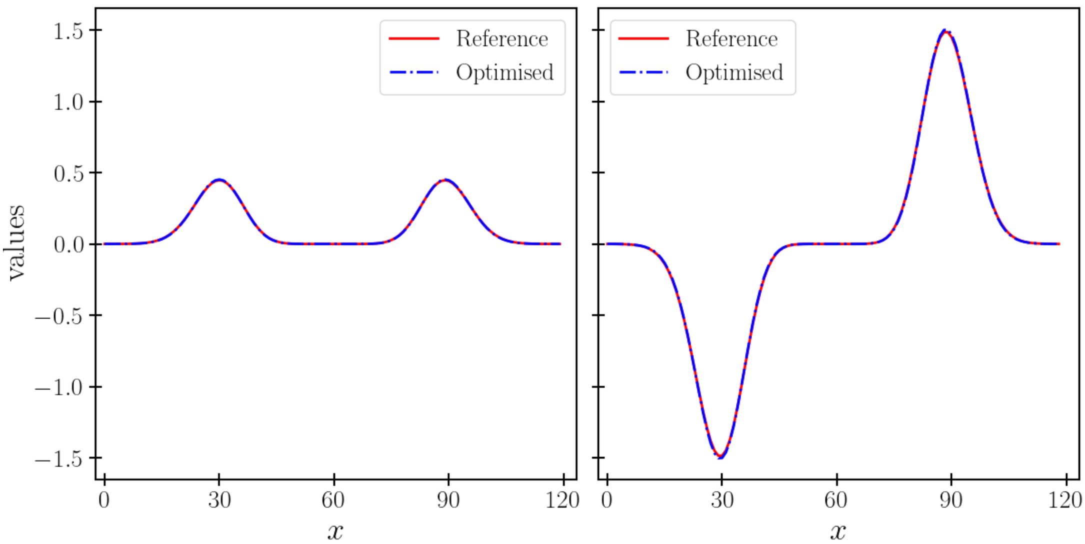


Figure 4.8: Free surface (left) and velocity (right) values at time = 50s for the reference (solid red) and for the optimised (blue dashdot) states

The background term is taken to be the zero vector i.e. $\mathbf{x}^b = (0,0)^T$. The first plot in Fig. 4.7 show the true and optimised initial states themselves and it can be observed that they are more or less identical. The other two figures show the true and the optimised free surface and velocity values obtained at time = 50 seconds (Fig. 4.8) and at time = 100 seconds which is at the end of the time window 20 (Fig. 4.9). Both the true values in solid red line and the optimised values in blue dashdot line are calculated by using the fine solver starting from the true initial state and the optimised initial state respectively. We observe here that the true and the optimised free surface and velocity fit each other well.

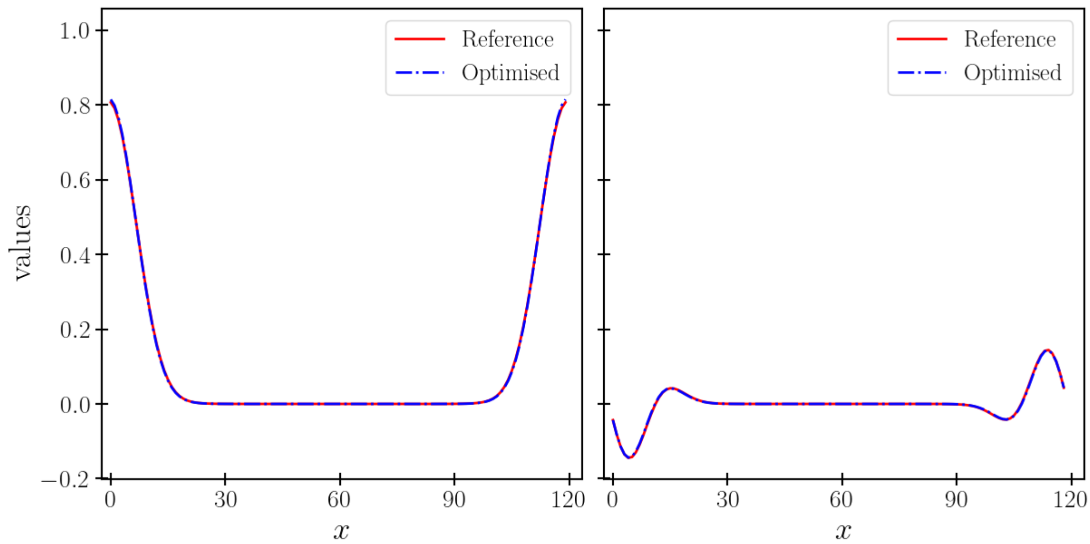


Figure 4.9: Free surface (left) and velocity (right) values at time = 100s for the reference (solid red) and for the optimised (blue dashdot) states

4.3.2 2D case

Before defining the configuration for our data assimilation problem we begin by first explaining briefly two very important terms and for which we refer to [Marshall and Plumb, 1989, Vallis, 2017].

Rossby number

The ratio between the magnitude of the acceleration due to inertial forces and the Coriolis acceleration is a dimensionless quantity which is known as the Rossby number and is denoted by Ro. If U is the characteristic velocity and L is the characteristic length of a given flow, then the Rossby number is given by

$$Ro = \frac{U}{f_0 L} \tag{4.35}$$

Geostrophic balance

Typically in mid-latitudes for large scale atmospheric and oceanic flows the Rossby number is quite small ($Ro \ll 1$) and so the Coriolis force is balanced by the horizontal pressure gradient forces. This is known as geostrophic balance. For the linearised shallow water equations,

$$f\hat{\mathbf{k}} \times \mathbf{u} = -g\nabla\eta \tag{4.36}$$

which gives

$$u_{\mathrm{g}} = -\left(\frac{g}{f_0}\right)\frac{\partial \eta_{\mathrm{g}}}{\partial y}, \qquad v_{\mathrm{g}} = \left(\frac{g}{f_0}\right)\frac{\partial \eta_{\mathrm{g}}}{\partial x} \tag{4.37}$$

where u_{g} and v_{g} are called geostrophic currents.

For the configuration we consider the case of a shallow layer of fluid in an initially unbalanced state (by taking an initial Gaussian free surface). We suppose the state is initially at rest (with zero horizontal velocities). Due to this initial perturbation to the free surface, the so called gravity-inertia or the Poincaré waves will be generated which will try to restore the shallow water system back towards the geostrophic balance also known as geostrophic adjustment. Now if $(\eta, u, v)^T$ is a solution of the 2D shallow water equations (4.16), then $(\eta+\eta_{\mathrm{g}}, u+u_{\mathrm{g}}, v+v_{\mathrm{g}})^T$ is also a solution of (4.16) if $(\eta_{\mathrm{g}}, u_{\mathrm{g}}, v_{\mathrm{g}})^T$ are in geostrophic balance. Clearly $(\eta_{\mathrm{g}}, u_{\mathrm{g}}, v_{\mathrm{g}})^T$ is divergence free. From (4.37)

$$\frac{\partial u_{\mathrm{g}}}{\partial x} + \frac{\partial v_{\mathrm{g}}}{\partial y} = -\left(\frac{g}{f_0}\right)\frac{\partial^2 \eta_{\mathrm{g}}}{\partial x \partial y} + \left(\frac{g}{f_0}\right)\frac{\partial^2 \eta_{\mathrm{g}}}{\partial x \partial y} = 0 \tag{4.38}$$

which gives

$$\frac{\partial(\eta+\eta_{\mathrm{g}})}{\partial t} + H\left(\frac{\partial(u+u_{\mathrm{g}})}{\partial x} + \frac{\partial(v+v_{\mathrm{g}})}{\partial y}\right) = 0 \tag{4.39}$$

Also,

$$f v_{\mathrm{g}} = g\frac{\partial \eta_{\mathrm{g}}}{\partial x} \quad \text{and} \quad f u_{\mathrm{g}} = -g\frac{\partial \eta_{\mathrm{g}}}{\partial y} \tag{4.40}$$

and therefore,

$$
\begin{aligned}
\frac{\partial(u+u_{\mathrm{g}})}{\partial t} &= f_0(v+v_{\mathrm{g}}) - g\frac{\partial(\eta+\eta_{\mathrm{g}})}{\partial x} \\
\frac{\partial(v+v_{\mathrm{g}})}{\partial t} &= -f_0(u+u_{\mathrm{g}}) - g\frac{\partial(\eta+\eta_{\mathrm{g}})}{\partial y}
\end{aligned}
\tag{4.41}
$$

We want to ensure that we retrieve the optimal values of the free surface height and the horizontal velocities corresponding to the true initial values after the end of the minimisation. In our case, since we know that the true values of initial velocity field are 0, one easy way to remove the indeterminate part is to just penalize the initial velocity field (in order to constraint $u_{\mathrm{g}}, v_{\mathrm{g}}$ to be 0). For this we add a regularisation term which depends on the initial state $\mathbf{x}_0$ to our cost function. For a regularisation constant α our cost function looks like

$$J(\mathbf{x}_0) = \frac{1}{2}\|\mathbf{F}^N\mathbf{x}_0 - \mathbf{y}\|_2^2 + \frac{\alpha}{2}\|\mathbf{W}\mathbf{x}_0\|_2^2 \tag{4.42}$$

whose gradient is

$$\nabla J(\mathbf{x}_0) = (\mathbf{F}^N)^T(\mathbf{F}^N\mathbf{x}_0 - \mathbf{y}) + \alpha\mathbf{W}\mathbf{x}_0 \tag{4.43}$$

and where $\mathbf{W}$ is a weighted matrix defined in a way which is driven by the knowledge of the difference in the order of magnitude of the actual variable values. We use the following parameter values provided in table 4.5 for our setup.

Regularisation constant	$\alpha = 5$
Initial free surface value	$\eta_0(x,y) = \exp\left\{\left(\frac{x-L_x/2}{L_x/10}\right)^2 - \left(\frac{y-L_y/2}{L_y/10}\right)^2\right\}$
Initial velocity value	$u_0(x,y) = v_0(x,y) = 0 \quad \forall x,y$

Table 4.5: Data assimilation parameters for the 2D shallow water model

Remark 4.4. *The cost function (4.42) can be written in the classical form by taking diagonal covariance matrices* $\mathbf{B}$ *and* $\mathbf{R}$*. Assigning values to the individual variances:* $\sigma_o^2 = 10^{-3}, \sigma_{b,\eta}^2 = 10^2, \sigma_{b,u}^2 = 10^{-1}$ *and* $\sigma_{b,v}^2 = 10^{-1}$ *we have*

$$\mathbf{R} = \begin{pmatrix}\sigma_o^2\end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix}\sigma_{b,\eta}^2 & & \\ & \sigma_{b,u}^2 & \\ & & \sigma_{b,v}^2\end{pmatrix} \tag{4.44}$$

The resulting inverse matrices $\mathbf{R}^{-1}$ *and* $\mathbf{B}^{-1}$ *are also diagonal and with the above values we will have a similar regularisation value as given in table 4.5.*

The following figures show the plots of the free surface and the horizontal velocity values from the initial state obtained by performing data assimilation. The background term is again taken to be the zero vector i.e. $\mathbf{x}^b = (0, 0, 0)^T$. The results from the optimised initial state are compared by running the fine solver for the true initial state (which is the reference plot). On the first row, the left side has the reference plot and the right side has the optimised plot for the instance when the integration has been done for 1/2 of the total period. On the second row, the left side has the reference plot and the right side has the optimised plot at the end of the total period of integration. Fig. 4.10 shows the η values, Fig. 4.11 depicts the u values and Fig. 4.12 represents the v values.

As can be observed, the solution values from the optimised initial state after data assimilation gives too familiar results with only marginal differences with the corresponding reference solution values.

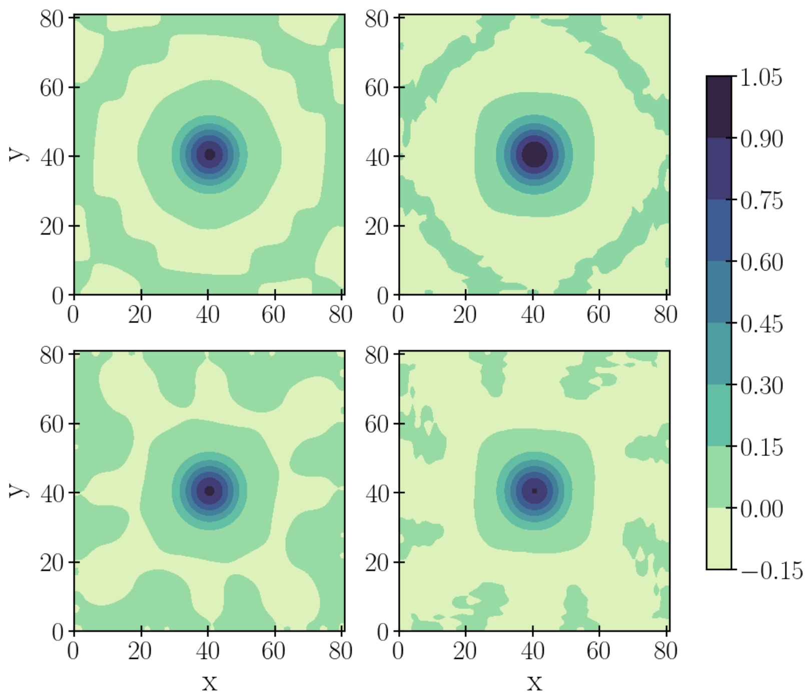


Figure 4.10: η values at different time windows. Reference (left) and optimised (right) η for 1/2 (first row) and for full (second row) period of integration

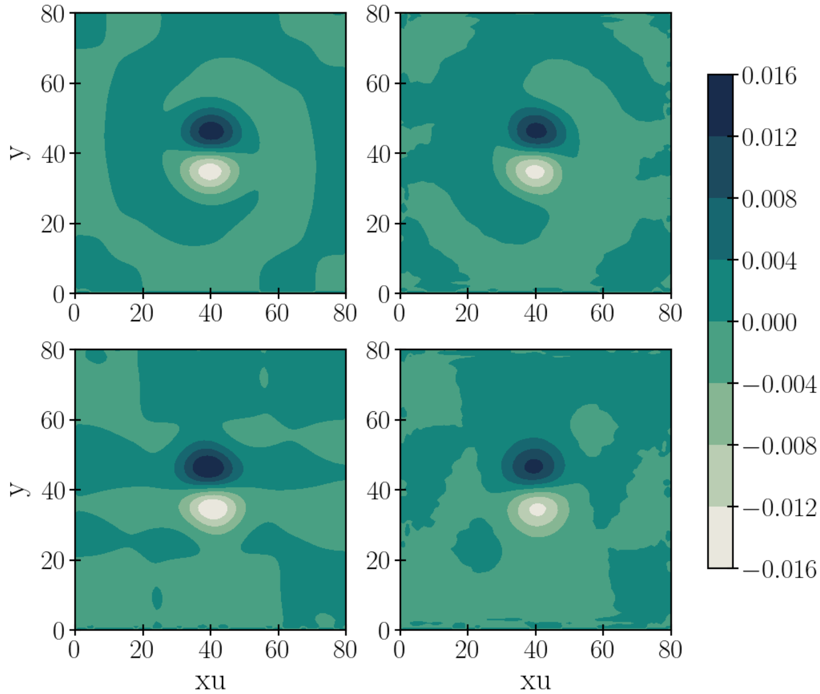


Figure 4.11: u values at different time windows. Reference (left) and optimised (right) u for 1/2 (first row) and for full (second row) period of integration

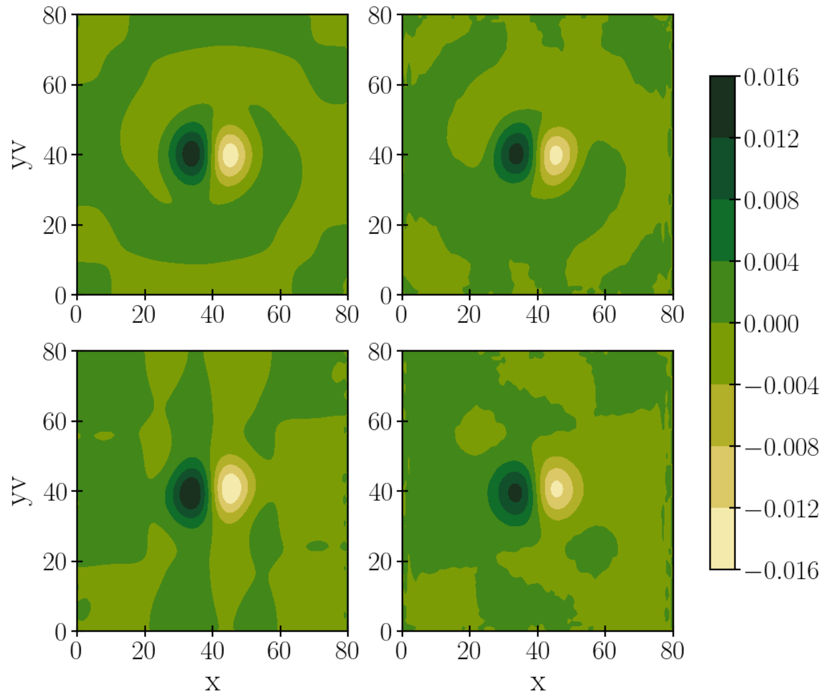


Figure 4.12: v values at different time windows. Reference (left) and optimised (right) v for 1/2 (first row) and for full (second row) period of integration

4.3.3 Approximating the gradient

We recall from the last two subsections 4.3.1 and 4.3.2 that the 1D and the 2D problem give rise to slightly different cost function gradients due to the type of regularisation terms used. We have in (4.34) the gradient for the 1D model which is written as

$$\nabla J(\mathbf{x}_0) = (\mathbf{F}^N)^T(\mathbf{F}^N\mathbf{x}_0 - \mathbf{y}) + \alpha\,\mathbf{Q}_2\mathbf{x}_0 \tag{4.45}$$

Cancelling the gradient (4.34) leads to solving the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with $\mathbf{A} = (\mathbf{F}^N)^T\mathbf{F}^N + \alpha\mathbf{Q}_2$ and $\mathbf{b} = (\mathbf{F}^N)^T\mathbf{y}$. When Parareal is introduced for the forward model, the true gradient is approximated by

$$\nabla J(\mathbf{x}_0) \approx (\mathbf{F}^N)^T(\mathbf{P}(k)\mathbf{x}_0 - \mathbf{y}) + \alpha\,\mathbf{Q}_2\mathbf{x}_0 \tag{4.46}$$

This is equivalent of solving the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A}$ is approximated as

$$\mathbf{A} \approx (\mathbf{F}^N)^T\mathbf{P}(k) + \alpha\,\mathbf{Q}_2 \quad , \qquad \mathbf{b} = (\mathbf{F}^N)^T\mathbf{y} \tag{4.47}$$

Following along the same lines one can similarly obtain the approximation of the gradient (4.43) of the 2D shallow water model. This is again equivalent of solving the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ where $\mathbf{A}$ is approximated as

$$\mathbf{A} \approx (\mathbf{F}^N)^T\mathbf{P}(k) + \alpha\,\mathbf{W} \quad , \qquad \mathbf{b} = (\mathbf{F}^N)^T\mathbf{y} \tag{4.48}$$

Remark 4.5. *We will use the notation* $\boldsymbol{A}$ *interchangeably for the matrix of the exact and the approximated gradient and the context will be made clear depending on the linear system we will solve.*

4.4 Krylov subspace enhanced Parareal

The Krylov subspace enhanced Parareal method introduced in section 2.7 will be used in our experiments. In the 1D case, the very low (and not realistic) dimension of our toy problem will allow this method to be really efficient. More interest is to be found in the application of the Krylov subspace enhanced method to the 2D case and in this section we explain the ideas used for its implementation in a more efficient manner. This includes some important findings on the use or not the use of a coarse solver, the needed computations of the image of the Krylov basis vectors projection by the fine solver and a way to obtain orthogonal basis vectors at a sufficient accuracy.

4.4.1 Using (or not) a coarse solver

The Krylov subspace enhanced method [Gander and Petcu, 2008] constructs an improved coarse solver $\mathbf{K}$ based on previous fine solvers integrations and of an underlying coarse solver $\mathbf{G}$. In practice there can be several strategies of implementing the Krylov subspace enhanced Parareal. One of the ways is to completely remove the use of the underlying coarse solver $\mathbf{G}$ and just keep it in the beginning for obtaining the initial states at each time window (at the very first Parareal iteration). What then remains is a Parareal algorithm where all the computations are done by the fine solver and thus is fully parallelisable: the sequential step associated with the integration of the coarse solver is removed. If $\mathbf{P}^k$ is the projection on the subspace of fine evaluations at iteration k then (2.67) becomes,

$$\mathbf{K}(t_{i+1}, t_i, \mathbf{x}_i^k) = \mathbf{F}(t_{i+1}, t_i, \mathbf{P}^k \mathbf{x}_i^k) \tag{4.49}$$

We test this "fully parallel" Krylov subspace enhanced Parareal version in the 2D shallow water model. To show its dependency to the initial vector, we perform the experiments starting with two different initial condition vectors: one with the initial state $\mathbf{x}_0$ and the other with the first direction vector of the inexact CG (which is $-\mathbf{b}$). Fig. 4.13 starting from $\mathbf{x}_0$ and Fig. 4.14 starting from $-\mathbf{b}$ shows the evolution of the error according to the Parareal iterations for 1) the usual Parareal (i.e. without Krylov subspace enhanced), 2) this "fully parallel" version (Krylov subspace enhanced with no coarse solver) and 3) for the basic Krylov subspace enhanced version (i.e. with the use of the underlying coarse solver). The figure also includes one additional experiment where the underlying coarse solver is used only in the correction step but only for the first Parareal correction. With the initial state $\mathbf{x}_0$, we encountered that the error shoots up already after the first Parareal iteration and we were not able to converge. To fix this issue we got a nice convergence when instead we used the coarse solver only for the first Parareal iteration. The error plots of the usual Parareal, the Krylov subspace enhanced Parareal and the Krylov subspace enhanced Parareal with one coarse run are shown in Fig. 4.13.

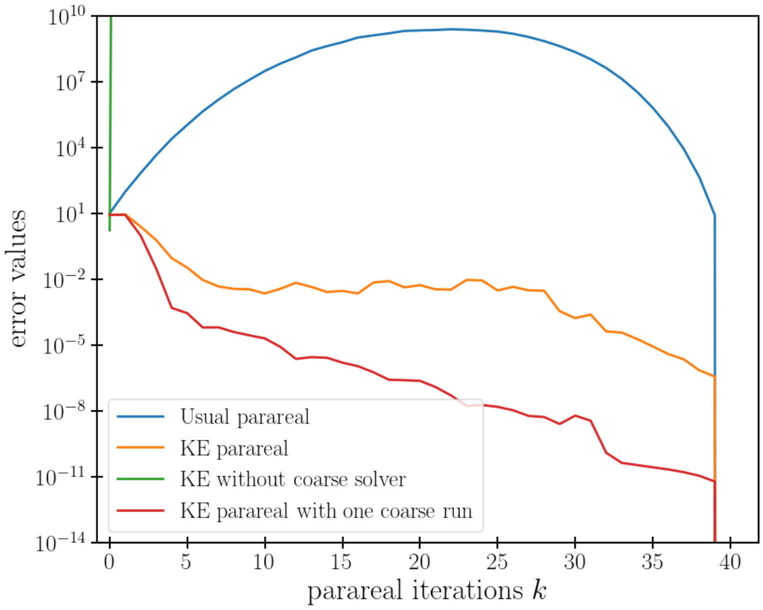


Figure 4.13: Error plots of different variants of Parareal using $\mathbf{x}_0$ as the initial condition

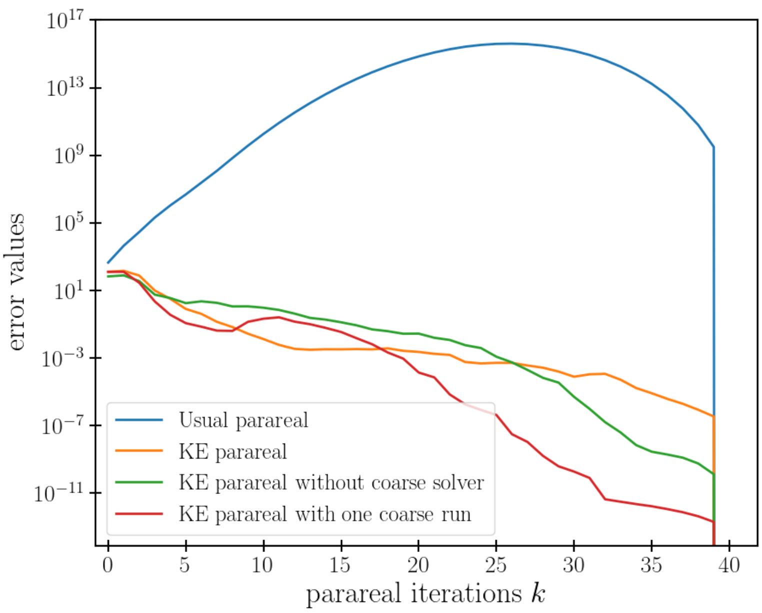


Figure 4.14: Error plots of different variants of Parareal using $-\mathbf{b}$ as the initial condition

For the initial direction vector $-\mathbf{b}$ we were able to converge using the fully parallel Krylov subspace enhanced Parareal without using the coarse solver. The corresponding error plots are shown in Fig. 4.14.

To conclude, the two figures illustrate that the answer to the question "Whether we should use the coarse solver?" is not easy. A closer look at the difference between the Parareal solutions after two successive iterations gives

$$\begin{aligned}
\mathbf{x}_{i+1}^{k+1} - \mathbf{x}_{i+1}^{k} &= \mathbf{F}\mathbf{P}^k\mathbf{x}_i^{k+1} + \mathbf{G}(\mathbf{I} - \mathbf{P}^k)\mathbf{x}_i^{k+1} - \mathbf{F}\mathbf{P}^{k-1}\mathbf{x}_i^k - \mathbf{G}(\mathbf{I} - \mathbf{P}^{k-1})\mathbf{x}_i^k \\
&= \mathbf{F}(\mathbf{P}^k\mathbf{x}_i^{k+1} - \mathbf{P}^{k-1}\mathbf{x}_i^k) - \mathbf{G}(\mathbf{P}^k\mathbf{x}_i^{k+1} - \mathbf{P}^{k-1}\mathbf{x}_i^k) \\
&\quad + \mathbf{G}(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k) \\
&= (\mathbf{F} - \mathbf{G})[\mathbf{P}^k\mathbf{x}_i^{k+1} - \mathbf{P}^{k-1}\mathbf{x}_i^k] + \mathbf{G}(\mathbf{x}_i^{k+1} - \mathbf{x}_i^k)
\end{aligned} \tag{4.50}$$

If we take norms on both the sides and then use the triangle inequality we get

$$\|\mathbf{x}_{i+1}^{k+1} - \mathbf{x}_{i+1}^{k}\| \leq \|\mathbf{F} - \mathbf{G}\|\|\mathbf{P}^k\mathbf{x}_i^{k+1} - \mathbf{P}^{k-1}\mathbf{x}_i^k\| + \|\mathbf{G}\|\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\| \tag{4.51}$$

The monitoring and approximating of the error requires a thorough study of the action of $\mathbf{F} - \mathbf{G}$ on the projection of the vectors (Parareal iterates) and weeding out the bad basis vectors which are causing the error to shoot. Also by playing with the accuracy of the coarse solver i.e. using an adaptive coarse solver it can be identified when and when not to use the coarse solver for Parareal. For now, we can say that the use of the coarse solver in all Parareal iterations is the safest choice.

4.4.2 Practical implementation of the Krylov subspace enhanced method

Computation of FPx

Here we show a practical implementation of the Krylov subspace enhanced Parareal and how we get the projections. Suppose we have a set $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{N_r}\}$ of N_r snapshots of the fine evaluation vectors of size n arranged in a matrix $\mathbf{S}$ of size $n \times N_r$. Let us also suppose that the image of these vectors by the application of the fine solver $\mathbf{F}$ is stored in the matrix $\mathbf{F_S} = (\mathbf{F}\mathbf{x}_1, \mathbf{F}\mathbf{x}_2, \cdots, \mathbf{F}\mathbf{x}_{N_r})$.

As required by the Krylov subspace enhanced method, the objective is to be able to compute $\mathbf{FPx}$ for a given vector $\mathbf{x}$, where $\mathbf{P}$ is the projection on $\mathbf{S}$. Then the enhanced coarse solver will then be approximated by

$$\mathbf{K} = \mathbf{F}\mathbf{P}\mathbf{x} + \mathbf{G}(\mathbf{I} - \mathbf{P})\mathbf{x} \tag{4.52}$$

If $\mathbf{U}$ is an orthonormal basis of $\mathbf{S}$ we can write $\mathbf{P} = \mathbf{U}\mathbf{U}^T$.

$$\mathbf{F}\mathbf{P}\mathbf{x} = (\mathbf{F}\mathbf{U})(\mathbf{U}^T\mathbf{x}) \tag{4.53}$$

Since $\mathbf{P}$ is the projection on $\mathbf{S}$, $\mathbf{S} = \mathbf{P}\mathbf{S} = \mathbf{U}\mathbf{U}^T\mathbf{S}$ and therefore taking the inverse of $\mathbf{U}^T\mathbf{S}$ on both the sides we get

$$\mathbf{U} = \mathbf{S}(\mathbf{U}^T\mathbf{S})^{-1} \tag{4.54}$$

and thus $\mathbf{F}\mathbf{U} = \mathbf{F}\mathbf{S}(\mathbf{U}^T\mathbf{S})^{-1} = \mathbf{F}_\mathbf{S}(\mathbf{U}^T\mathbf{S})^{-1}$.
The computation of $\mathbf{F}\mathbf{P}\mathbf{x}$ can thus be obtained by

$$\mathbf{F}\mathbf{P}\mathbf{x} = \mathbf{F}_\mathbf{S}(\mathbf{U}^T\mathbf{S})^{-1}(\mathbf{U}^T\mathbf{x}) \tag{4.55}$$

and note that this expression does not require the integration by $\mathbf{F}$ of the vectors of the orthonormal basis $\mathbf{U}$. We however mention an important point that we still need to store the vectors for the matrix $\mathbf{S}$ and $\mathbf{F}_\mathbf{S}$ and it was fine for our applications. But it can certainly become a bottleneck once we shift to more realistic applications and in that case one approach could be coupling this practical implementation with a reduced basis approach [Caldas Steinstraesser et al., 2021].

Remark 4.6. *From one CG iteration to another, we completely reset the Krylov basis for Parareal. The reuse of previous Krylov basis does not seem to produce improvements. It would be interesting to further explore this point in order to accelerate the convergence of Parareal. Note that this point is clearly intrinsically linked to the used minimisation method and may be more efficient with other methods than the conjugate gradient."*

Derivation of the orthonormal basis: Gram-Schmidt with reorthogonalisation

To obtain the orthogonal basis $\mathbf{U}$, the Gram-Schmidt orthogonalisation (GS) process is one of the most widely used orthogonalisation techniques. For a matrix

$\mathbf{S} = (\mathbf{x}_1, \cdots, \mathbf{x}_{N_r}) \in \mathbb{R}^{n \times N_r}$ of vectors with rank$(\mathbf{S}) = N_r$, GS produces a **QR** factorisation

$$\mathbf{S} = \mathbf{QR} \tag{4.56}$$

where $\mathbf{Q} = (\mathbf{q}_1, \cdots, \mathbf{q}_{N_r})$ is the orthogonal matrix and $\mathbf{R} \in \mathbb{R}^{N_r \times N_r}$ is an upper-triangular matrix. GS can be implemented in two ways: one is the Classical Gram-Schmidt (CGS) method and the other is the Modified Gram-Schmidt (MGS) method described in table 4.6.

The modified Gram-Schmidt procedure orthogonalises against the current estimation of the latest orthogonal basis vector $\mathbf{x}_j^{(1)}$. Although both algorithms are mathematically equivalent, MGS has better numerical properties than CGS. In fact CGS suffers from a bad reputation of lacking numerical stability due to the occurring of the roundoff errors when cancellation takes place by subtracting the orthogonal projection [Björck, 1994].

$$\mathbf{x}_j^{(1)} = \mathbf{x}_j^{(1)} - (\mathbf{x}_j, \mathbf{q}_k)\mathbf{q}_k \tag{4.57}$$

Classical GS algorithm	Modified GS algorithm
for $j = 1, \cdots, N_r$ $\quad \mathbf{x}_j^{(1)} = \mathbf{x}_j$ $\quad$ for $k = 1, \cdots, j-1$ $\qquad \mathbf{x}_j^{(1)} = \mathbf{x}_j^{(1)} - \langle \mathbf{x}_j, \mathbf{q}_k \rangle \mathbf{q}_k$ $\quad$ end $\quad \mathbf{q}_j = \mathbf{x}_j^{(1)} / \Vert \mathbf{x}_j^{(1)} \Vert$ end	for $j = 1, \cdots, N_r$ $\quad \mathbf{x}_j^{(1)} = \mathbf{x}_j$ $\quad$ for $k = 1, \cdots, j-1$ $\qquad \mathbf{x}_j^{(1)} = \mathbf{x}_j^{(1)} - \langle \mathbf{x}_j^{(1)}, \mathbf{q}_k \rangle \mathbf{q}_k$ $\quad$ end $\quad \mathbf{q}_j = \mathbf{x}_j^{(1)} / \Vert \mathbf{x}_j^{(1)} \Vert$ end

Table 4.6: The classical and modified Gram-Schmidt algorithm

As a result the produced set of vectors gradually lose their orthogonality and are far from orthogonal. Even if the modified GS algorithm performs better, it is still not accurate enough to lead to an efficient implementation for the Krylov subspace enhanced method.

As shown in Algorithm 12, the next step is to perform a reorthogonalisation step each time, i.e. "twice is enough" to get preserve the orthogonality of the basis vectors to the precision level [Giraud et al., 2005]. The cost is then doubled ($2nN_r$ floating point operations). Note however, that in comparison with the modified

Gram-Schmidt with reorthogonalisation the inner loop of the CGS is still fully parallelised. This can be seen from the table 4.6 where in the scalar products $\langle \mathbf{x}_j, \mathbf{q}_k \rangle$ computed in CGS do not depend on $\mathbf{x}_j^{(1)}$ whereas they do depend on $\mathbf{x}_j^{(1)}$ for MGS.

Algorithm 12 CGS METHOD WITH REORTHOGONALISATION STEP

Initialisation: $\mathbf{x}_j^{(0)} = \mathbf{x}_j$
for $j = 1, \ldots, N_r$ **do**
  for $i = 1, 2$ **do**
    $\mathbf{x}_j^{(i)} = \mathbf{x}_j^{(i-1)}$
    for $k = 1, \ldots, j-1$ **do**
      $\mathbf{x}_j^{(i)} = \mathbf{x}_j^{(i)} - \langle \mathbf{x}_j^{(i-1)}, \mathbf{q}_k \rangle, \mathbf{q}_k$
    end for
  end for
  $\mathbf{q}_j = \mathbf{x}_j^{(i)} / \|\mathbf{x}_j^{(i)}\|$
end for

To highlight the impact of the reorthogonalisation step, we ran two experiments of using the Krylov subspace enhanced Parareal where the orthogonal basis is obtained from one pass and two passes of the Classical Gram-Schmidt procedure. We check the impact in terms of the loss in orthogonality at CGS iteration step i defined as

$$\text{Loss of orthogonality} = \max_{1 \le j \le N, j \ne i} \{ |\langle \mathbf{x}_i, \mathbf{x}_j \rangle| \}$$

We again use the 2D shallow water model with the initial condition $\mathbf{x}_0$ and $-\mathbf{b}$. In Fig. 4.15 the solid lines and dashed lines indicate the loss of orthogonality when CGS and CGS with a reorthogonalisation step is used respectively. The blue colour is for the initial condition $\mathbf{x}_0$ and the red colour is for the initial condition $-\mathbf{b}$. There is a clear difference with the basis vectors losing their orthogonality rapidly when CGS is used with an even faster loss when $\mathbf{x}_0$ is used. The orthogonality is preserved when two passes of CGS are used which clearly asserts the statement that we have to use two passes of CGS for getting an orthogonal basis. When this CGS with reorthogonalisation algorithm is used in our 2D shallow water model, its cost is fully negligible in comparison to the costs of the fine/coarse solvers integrations.

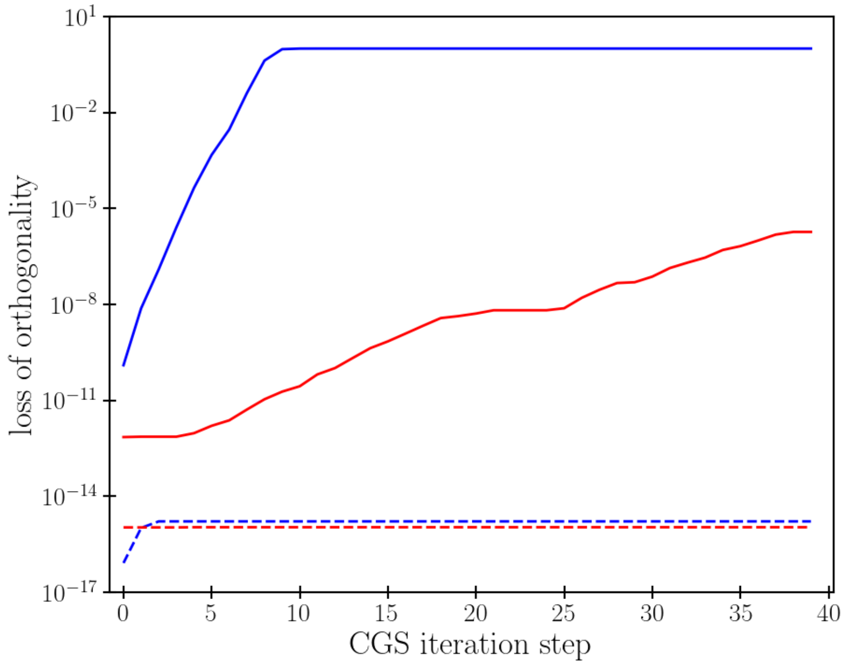


Figure 4.15: The loss of orthogonality in Krylov basis vectors. The solid lines represent when CGS is used while the dashed lines depict CGS used with a re-orthogonalisation step. Blue and red colour are for $\mathbf{x}_0$ and $-\mathbf{b}$ as initial conditions respectively

4.5 Results

We recapitulate so far what we have defined in our 1D and 2D problem.

- A linearised 1D shallow water model (4.8) leading to matrix $\mathbf{C}$ of the ODE (4.10) after semi-discretisation.

- Data assimilation cost function $J(\mathbf{x}_0)$ in (4.26) for the 1D case and its approximated gradient in (4.46) with the corresponding matrix $\mathbf{A} = (\mathbf{F}^N)^T\mathbf{P}(k) + \alpha\mathbf{Q}_2$.

- A linearised 2D shallow water model (4.16) leading to the ODE system (4.18) after semi-discretisation.

- Data assimilation cost function $J(\mathbf{x}_0)$ in (4.42) for the 2D case and its approximated gradient with the corresponding matrix $\mathbf{A} = (\mathbf{F}^N)^T\mathbf{P}(k) + \alpha\mathbf{W}$.

- Parareal propagators $\mathbf{F}$ and $\mathbf{G}$ constructed in (4.22) from the ODE (4.10) using an implicit theta scheme.

In this section we are going to apply the Parareal algorithm in a series of different conjugate gradient setup and analyse the results to see what differs at each step. The reader can check all the conjugate gradient versions which we will use from table 3.1 provided in chapter 3. We will follow a close presentation of the results for both the 1D and the 2D experiments.

First, for a reference experiment, we are going to minimise the cost function by the conjugate gradient (CG) method which uses the exact matrix-vector products for constructing the residuals and the search directions. The minimisation terminates for a given tolerance on based on the 2-norm of the residual $\|\mathbf{r}(\mathbf{x}_j)\|_2$. Second, we use the CG version where the matrix-vector products are obtained through the Parareal algorithm (CG_Para). The important point to consider here is that we don't know a priori what Parareal precision is exactly required to achieve the same performance level as the CG with exact matrix-vector product. We run CG with arbitrary Parareal tolerances and note the corresponding Parareal iterations. This will be used to check the actual required Parareal tolerance when we use the inexact conjugate gradient.

The next step is to mimic this result by now using the inexact conjugate gradient (ICG_Para) method with the original (using the value of $\|\mathbf{E}\|_{\mathbf{A}^{-1},\mathbf{A}}$) and the modified (using the value of $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$) Parareal stopping criterion for the 1D case. The 2D case however is only tested with the modified Parareal stopping criterion. We reiterate from Remark 3.3 that the computed residuals are no longer exact and hence not orthogonal to each other which requires a reorthogonalisation step at each minimisation iteration.

In these sets of experiments we use the exact values of all the norms we require and we do not use the inaccuracy budget presented in chapter 3 (section 3.3). We go then one step further, with the (ICG_Para_Approx) experiment, by just using the approximate value of $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ given chapter 3, subsection 3.4.3 while still using exact estimates for the other norms.

Finally we present the inexact conjugate gradient method which uses the inaccuracy budget and all the practical approximations (ICG_Para_Prac) to see the impact it has on the total number of Parareal iterations. For the 1D model, we use all the approximations suggested by Gratton et al. from chapter 3. Since for

2D model in our case, we do not have a direct access to the matrix $\mathbf{A}$, we propose another feasible alternatives in particular to $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ and $\|\mathbf{p}_j\|_{\mathbf{A}}$ in this chapter.

The original CG and the inexact CG use different stopping criteria. The first is based on the 2-norm of the residual while the other is based on the $\mathbf{A}^{-1}$ norm. To ensure that the results obtained from both methods follow the same stopping criterion we see from (3.50) that the inexact conjugate gradient terminates when, for a given value of ϵ_{icg},

$$\|\mathbf{r}_j\|_{\mathbf{A}^{-1}} \leq \frac{\sqrt{\epsilon_{\text{icg}}}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}} \tag{4.58}$$

If we rearrange the above terms for the equality we obtain

$$\epsilon_{\text{icg}} = \left(\frac{2\,\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}}{\|\mathbf{b}\|_{\mathbf{A}^{-1}}}\right)^2 \tag{4.59}$$

To find the value of ϵ_{icg} which corresponds to the stopping tolerance of CG we just need to store the value of $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ at the last CG iteration to deduce ϵ_{icg} from (4.59).

4.5.1 Analysis for the 1D case

4.5.1.1 Conjugate gradient (CG)

To begin our analysis we first check how our data assimilation problem performs with the shallow water model when CG is used. That is, we use the exact matrix-vector multiplication and therefore we have $\mathbf{A} = (\mathbf{F}^N)^T\mathbf{F}^N + \alpha\mathbf{Q}_2$. Recall that the CG stopping criterion for a given tolerance ϵ_{cg} requires that

$$\|\mathbf{r}(\mathbf{x}_j)\|_2 \leq \epsilon_{\text{cg}} \tag{4.60}$$

where the quantity $\|\mathbf{r}(\mathbf{x}_j)\|_2$ is the 2-norm of the residual vector for solution guess $\mathbf{x}_j$ at the jth iteration. Fig. 4.16 below shows the evolution of $\|\mathbf{r}(\mathbf{x}_j)\|_2$ (blue line) according to the CG iterations. The tolerance ϵ_{cg} (black dashed line) was here set to the value of 10^{-4}. The residual norm eventually decreases and the stopping criterion is satisfied after 24 minimisation iterations.

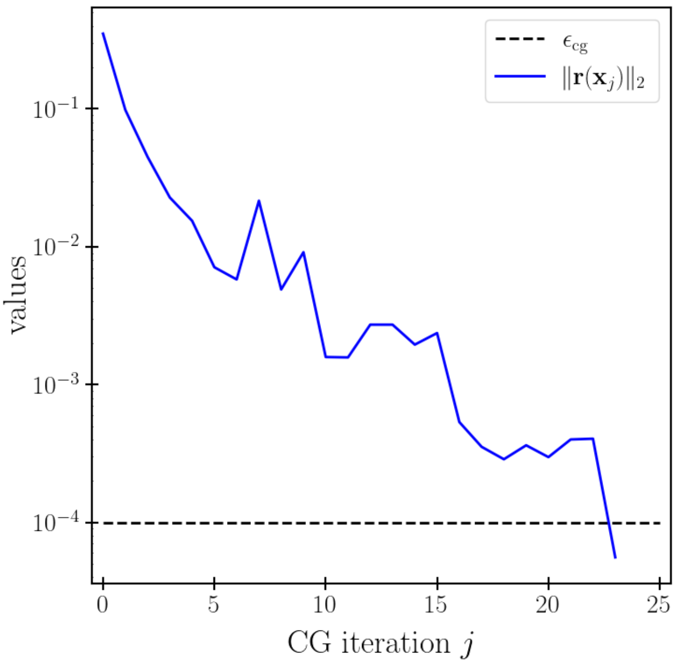


Figure 4.16: CG without using Parareal

4.5.1.2 Conjugate gradient with Parareal

We now run CG where the matrix-vector product comes through the Parareal (CG_Para) with a fixed Parareal stopping tolerance ϵ_{p}. The problem is then to fix the value of ϵ_{p}. Here we compare different choices for ϵ_{p}. In the first one, we choose $\epsilon_{\mathrm{p}} = 10^{-6}$. This corresponds to the minimum of the values $\{\xi_j\}_j$ which are produced, at each CG iterations, by the inexact CG algorithm as a stopping criterion for Parareal (see next subsection 4.5.1.4). Obviously, this choice is only for comparison since this value is not accessible without running the inexact CG. Results are shown in Fig. 4.17 following algorithm 13.

Algorithm 13 CG WITH PAREREAL

Given: Right hand side vector $\mathbf{b} \in \mathbb{R}^n$, CG tolerance ϵ_{cg}, Parareal tolerance ϵ_{p}
Set $\mathbf{x}_0 = 0$, $\mathbf{r}_0 = \mathbf{b}$, $\mathbf{p}_0 = -\mathbf{b}$
$j = 0$
while true **do**
    Run Parareal with tolerance ϵ_{p} and initial condition $\mathbf{p}_j$ to give $\mathbf{P}(k)\mathbf{p}_j$.
    Compute the matrix-vector product $\mathbf{A}\mathbf{p}_j = (\mathbf{F}^N)^T\mathbf{P}(k)\mathbf{p}_j$.
    $\alpha_j = \mathbf{r}_j^T\mathbf{r}_j/\mathbf{p}_j\mathbf{A}\mathbf{p}_j$

```
    x_{j+1} = x_j + α_j p_j
    r_{j+1} = r_j − α_j A p_j
    if ||r_{j+1}||_2 ≤ ε_cg then
        break
    end if
    p_{j+1} = −r_{j+1} + ( r_{j+1}^T r_{j+1} / r_j^T r_j )
    j = j + 1
end while
```

For the same CG tolerance $\epsilon_{\text{cg}} = 10^{-4}$ we perform again 24 minimisation iterations, which means that the stopping criterion for parareal was indeed small enough and we end up using 191 Parareal iterations in total. We recall (see table 4.3) that we have a total number of time windows N equal to 20 in these 1D experiments. On average 7.96 Parareal iterations per CG iteration are performed which will be shown to be slightly higher than the average of 6.25 Parareal iteration when using the inexact CG. This emphasises the need for an *adaptive* Parareal in the framework of inexact CG.

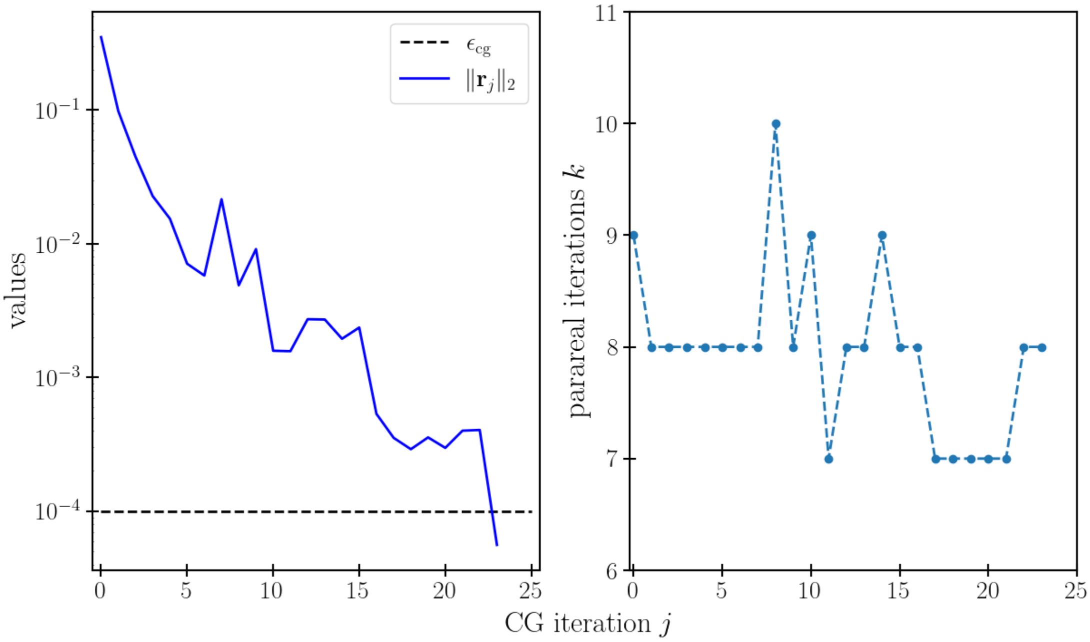


Figure 4.17: CG with matrix-vector product from Parareal, $\epsilon_{\text{p}} = 10^{-6}$

To check how CG performs in terms of the relative error in the 2-norm of the residual and the total number of Parareal iterations we show in Fig. 4.18 the results of CG when $\epsilon_p = 10^{-5}, 10^{-6}$ and 10^{-7} are used. The relative error in the 2-norm of the residual is given as

$$\text{Relative error in the 2-norm of residual} = \frac{|\, \|\mathbf{r}_j\|_2 - \|\mathbf{r}(\mathbf{x}_j)\|_2 \,|}{|\, \|\mathbf{r}(\mathbf{x}_j)\|_2 \,|}$$

where $\|\mathbf{r}(\mathbf{x}_j)\|_2$ is the 2-norm of the residual obtained when CG is used. Initially we see that the relative error is low and almost the same for all the 3 cases. After a few CG iterations the relative error starts to vary and is smaller for a smaller ϵ_{p}. The relative error also gradually increases with the CG iteration suggesting that a lower Parareal precision is needed near the end of the minimisation. Also as one would expect a smaller ϵ_{p} implies more Parareal iterations and vice-versa.

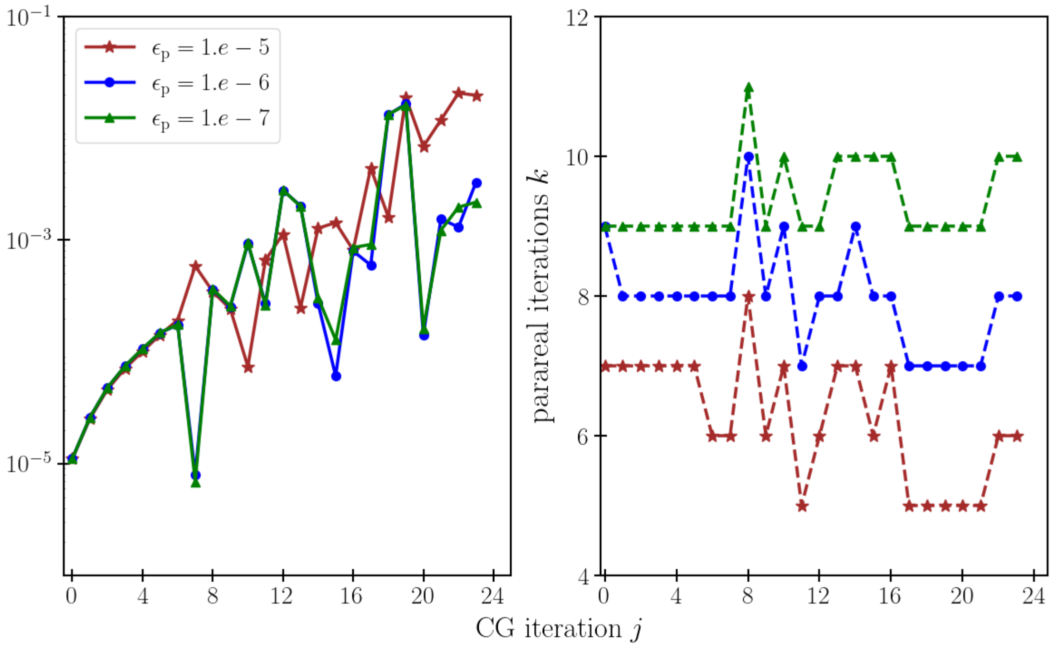


Figure 4.18: Relative error in the 2-norm of the residual (left) and the corresponding Parareal iterations k (right) for different fixed values of ϵ_{p} when using CG

4.5.1.3 Inexact conjugate gradient using Parareal

Now we keep the result from CG as the reference for performing similar experiments with the inexact CG. As mentioned previously, the stopping criterion for inexact CG is deduced from the one from CG using (4.59) leading to a value of $\epsilon_{\text{icg}} = 1.12 \times 10^{-7}$.

We first show the results of the ICG_Para when the primal-dual norm of the perturbation matrix $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$ bounded by ω_j in (3.48) is used as the stopping criterion for the Parareal. Algorithm 6 describes in detail the way to implement this version of the inexact CG. Fig. 4.19 below demonstrates the result.

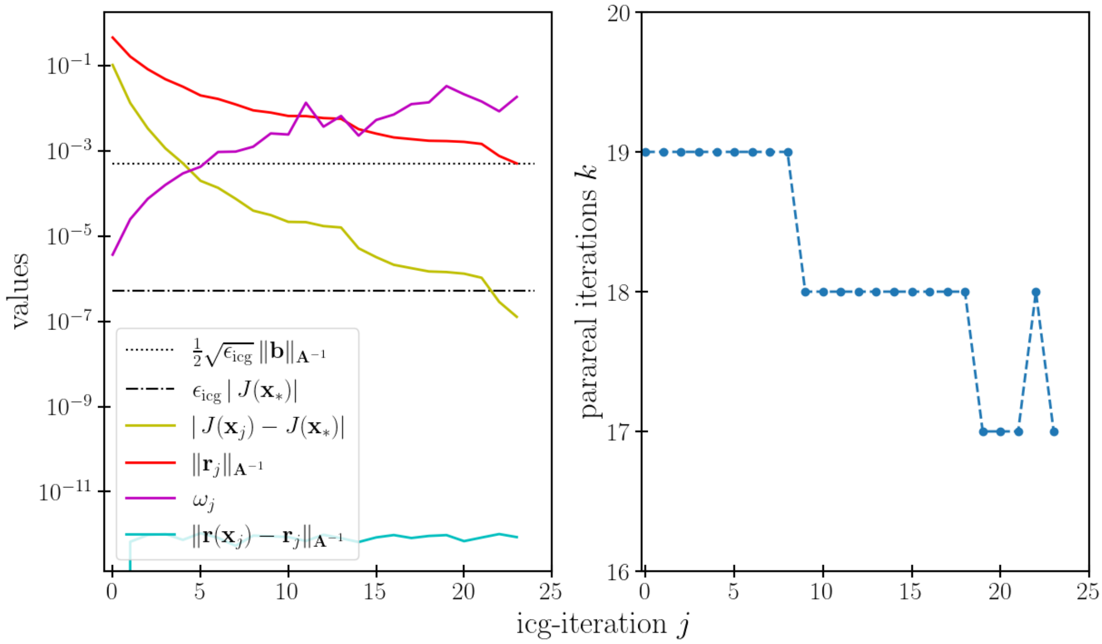


Figure 4.19: Inexact CG with the Parareal stopping criterion $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}} \leq \omega_j$

On the left image, the solid red line depicts the residual norm $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ at icg-iteration j and the black dotted line represents the inexact CG stopping tolerance $(\frac{1}{2}\sqrt{\epsilon_{\text{icg}}}\,\|\mathbf{b}\|_{\mathbf{A}^{-1}})$ which $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ has to satisfy. The quantity ω_j in the violet is the bound for the error norm $\|\mathbf{E}\|_{\mathbf{A}^{-1},\mathbf{A}}$; the cyan solid line is the residual gap norm $\|\mathbf{r}(\mathbf{x}_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}}$ and the yellow solid line is the cost function change $|J(\mathbf{x}_j) - J(\mathbf{x}_*)|$ which is bounded by the quantity $\epsilon_{\text{icg}}|J(\mathbf{x}_*)|$ in the black dashdot line. The right image shows the evolution of the corresponding Parareal iterations k with the icg-iterations j in a blue dashed line with a blue circle marker.

Remark 4.7. *Throughout the manuscript we are going to follow the same colour coding for quantities used in Fig. 4.19 for the different inexact CG configurations.*

As a reminder from Theorem 3.1 the inexact CG stops when both

$$\|\mathbf{r}_j\|_{\mathbf{A}^{-1}} \leq \frac{\sqrt{\epsilon_{\text{icg}}}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}} \quad \text{and} \quad \|\mathbf{r}(\mathbf{x}_j) - \mathbf{r}_j\|_{\mathbf{A}^{-1}} \leq \frac{\sqrt{\epsilon_{\text{icg}}}}{2}\|\mathbf{b}\|_{\mathbf{A}^{-1}} \tag{4.61}$$

and we have $|J(\mathbf{x}_j) - J(\mathbf{x}_*)| \leq \epsilon_{\text{icg}} |J(\mathbf{x}_*)|$. We can observe that from Fig 4.19 that at the end of the minimisation the conditions of Theorem 3.1 are well satisfied.

In this experiment, the minimisation ends after 24 icg-iterations at the expense of a total of 437 Parareal iterations. This means that on an average we use 18.2 Parareal iterations per icg-iteration which is almost equal to the total number of time windows $N = 20$. This means that we are not gaining any speedup out of it since we are only able to converge to the solution after the second-last or the last iteration. This emphasises the fact that making use of an estimate of the $\|\mathbf{E}\|_{\mathbf{A}^{-1},\mathbf{A}}$ norm when deriving the stopping criterion leads to a suboptimal algorithm. The only positive point of using inexact CG in this case can be seen by observing the values of ω_j which gradually increase and the values of $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ decrease at the same time as we proceed with the icg-iterations.

The next result in Fig. 4.20 shows the impact of using the modified Parareal stopping criterion (3.61) based on $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ within ICG_Para on the number of Parareal iterations.

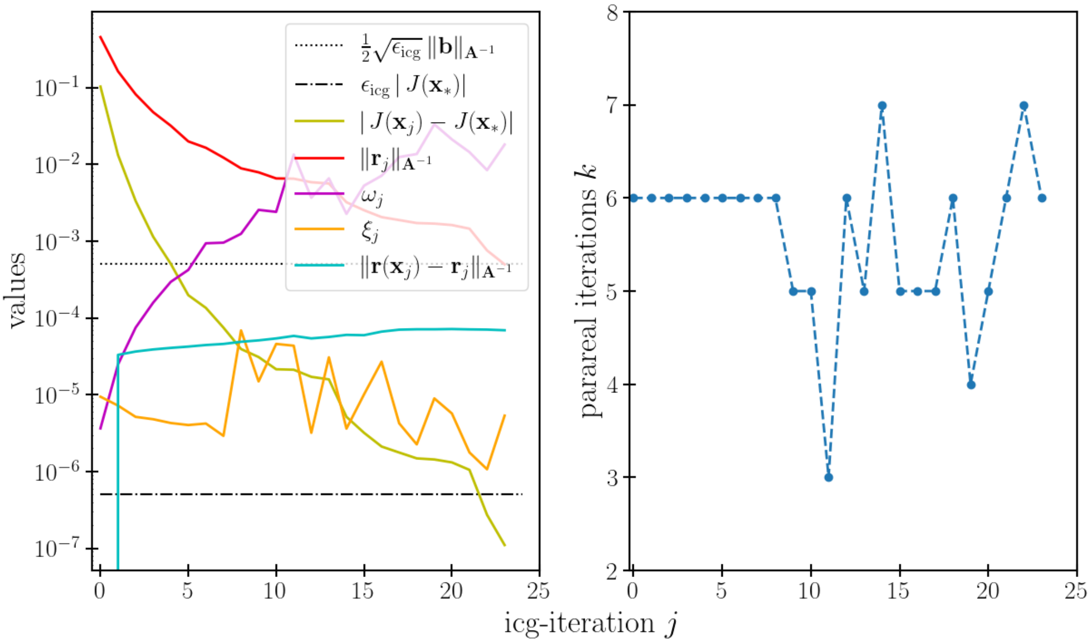


Figure 4.20: Inexact CG with the Parareal stopping criterion $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \xi_j$

Algorithm 7 shows the implementation of this method which is same as algorithm 6 except that the Parareal stopping criterion is different. The new quantity in the orange solid line represents ξ_j which is the bound for $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$. The minimisation again ends in 24 iterations and the total Parareal iterations goes significantly down to 134 resulting in an average of 5.5 Parareal iterations per icg-iteration.

The use of ICG_Para allows us to use Parareal in an adaptive sense since its tolerance ω_j or ξ_j is changing with every icg-iteration j. We still keep the quantity ω_j in Fig. 4.20 as it helps us to see if the permissible error in the matrix-vector product is gradually increasing. Looking back at the Parareal iteration evolution we can say that $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ provides a much more efficient Parareal stopping criterion than $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$.

4.5.1.4 Inexact conjugate gradient with practical estimates

We now use an estimate for $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ instead of an exact computation in the previous paragraph. In the chapter 3, section 3.4.3 we found a way to have a feasible approximation for $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ as the difference between the successive Parareal iterates in the 2-norm. We state the approximation here for reference,

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \approx \|\mathbf{P}(k+1)\mathbf{p}_j - \mathbf{P}(k)\mathbf{p}_j\|_2 \tag{4.62}$$

We show in Fig. 4.21 the results of using this approximation for ICG_Para_Approx. The algorithm is given below. After closely examining the two plots of ICG_Para in Fig. 4.20 and ICG_Para_Approx in Fig. 4.21 we see almost no difference on the left side within the various quantities and only a marginal difference on the right side where ICG_Para_Approx takes one less Parareal iteration (133 in total) at the last minimisation iteration. From this we can conclude that the we have a very good and valid approximation to $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$.

Finally we show the main result by using the practical estimates for all the remaining norms used in subsection 3.3.2, the inaccuracy budget in subsection 3.3.3. We briefly explain how we get an estimate for the termination criterion (3.50) following [Gratton et al., 2021]. The $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ is estimated as

$$\frac{1}{2}\|\mathbf{r}_j\|^2_{\mathbf{A}^{-1}} \approx J(\mathbf{x}_j) - J(\mathbf{x}_*) \approx J(\mathbf{x}_{j-d}) - J(\mathbf{x}_j) \tag{4.63}$$

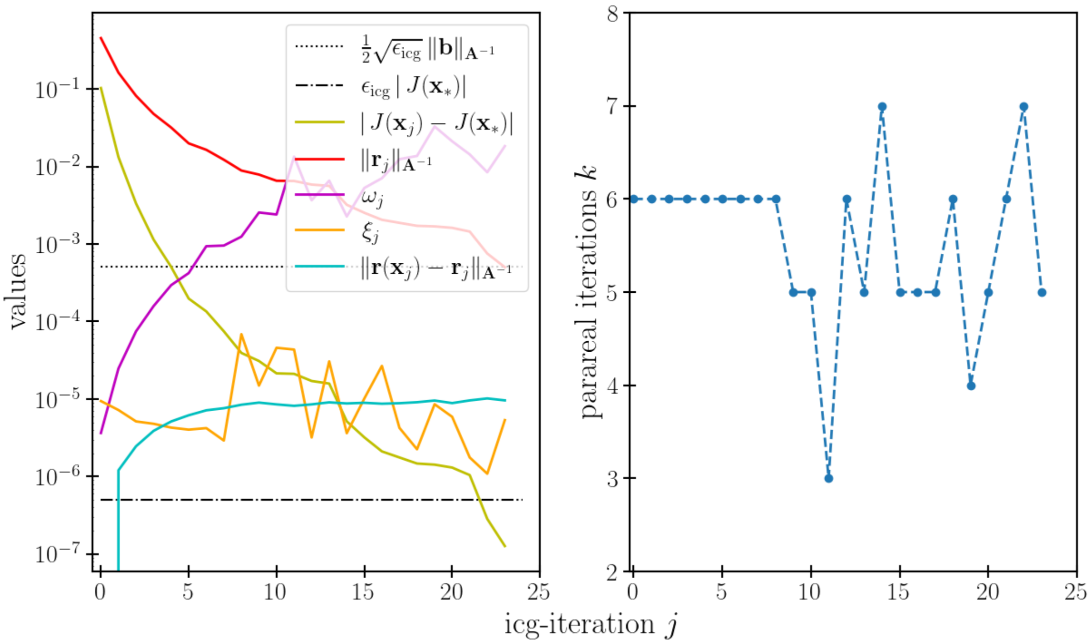


Figure 4.21: Inexact CG with the approximated $\|\mathbf{E}_j \mathbf{p}_j\|_{\mathbf{A}^{-1}}$

where d is any integer, $d < j$. Using (3.50) to replace $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ and then subsequently (3.53), (3.52) in the above equation we get

$$
\begin{aligned}
J(\mathbf{x}_{j-d}) - J(\mathbf{x}_j) &\leq \frac{\epsilon_{\text{icg}}}{8}\,\|\mathbf{b}\|^2_{\mathbf{A}^{-1}} \\
\implies J_{j-d} - J_j &\leq \frac{\epsilon_{\text{icg}}}{4}|J_j|
\end{aligned}
\tag{4.64}
$$

We require to store the values of the approximate quadratic J_j to use them later for the termination of the ICG_Para_Prac starting from $j = d$. For this case we choose $d = 2$ and plot the results in Fig. 4.22 following algorithm 9. What we observe is that we have a very gradual decrease in the Parareal iterations even after using different approximations. In fact ICG_Para_Prac gives similar results as seen before with the previous results of the conjugate gradient variants. The minimisation takes 24 iterations using a total of 126 Parareal iterations. On an average, the number of Parareal iterations per icg-iteration slightly decreases to 6.25. The theoretical speed up is given by

$$
S = \frac{\text{number of time windows}}{\text{average Parareal iterations per icg-iteration}} = \frac{20}{6.25} \approx 3.2 \tag{4.65}
$$

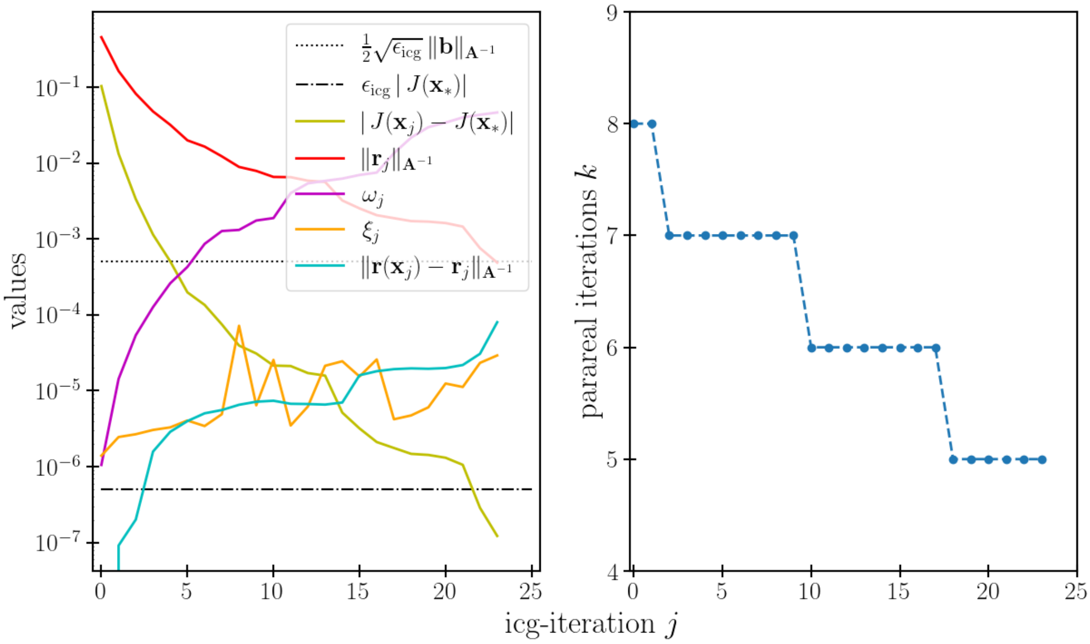


Figure 4.22: Inexact CG with all practical approximations

4.5.1.5 Unsuitability of Krylov subspace enhanced Parareal for 1D case

We run the previous experiment of inexact CG with practical estimates using the same parameter values except that we now use the Krylov subspace enhanced Parareal version. As a result the total number of Parareal iterations drops significantly to 70 with an average of 2.69 Parareal iterations per icg-iteration (see Fig. 4.23). This extremely quick convergence is the small size of the 1D problem. Now the result in Fig. 4.23 may suggest that Krylov subspace enhanced Parareal should be used for the 1D problem. But this is misleading because the total time to compute the Krylov basis is quite more than the time to actually solve the minimisation problem.

At Parareal iteration k, the size of subspace $\mathbf{S}$ containing the fine evaluations used to build the projection matrix $\mathbf{P}^k$ is equal to kN, N being the number of time windows. A simple computation shows that the dimension of the Krylov basis (assuming linearly independent vectors in $\mathbf{S}$) becomes rapidly larger than the dimension of the problem itself. Our problem size is $N_x = 120$ and only 3 Parareal iterations are needed for the size of the basis (120) to match the problem size. Thus, $\mathbf{P}^k$ is rapidly close to the identity.

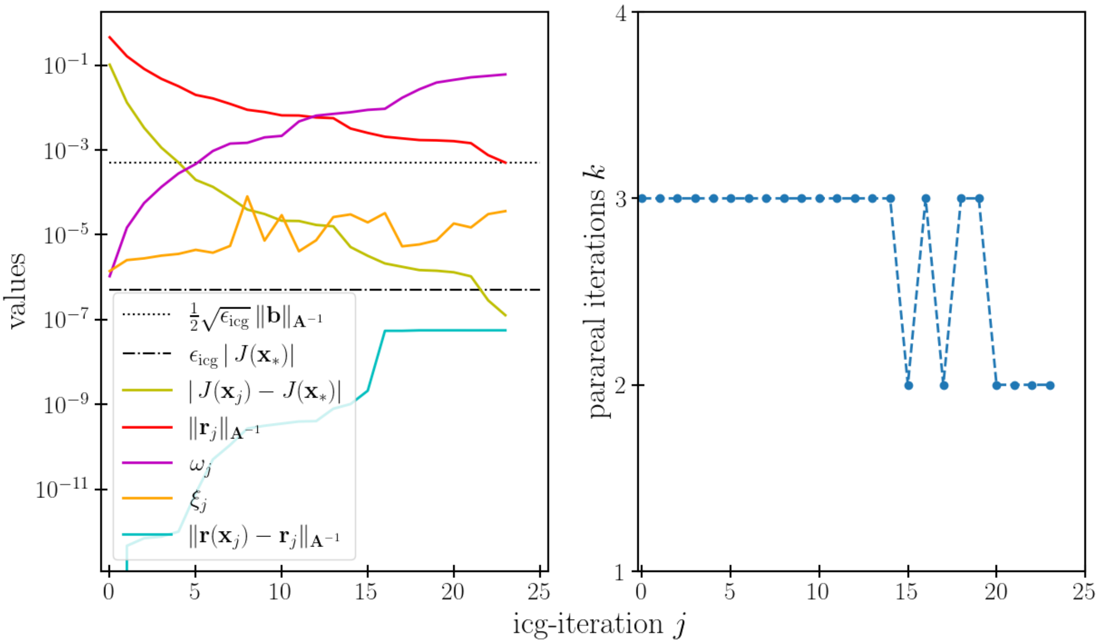


Figure 4.23: Modified practical inexact CG with Krylov subspace enhanced Parareal

4.5.2 Analysis for 2D model

We now turn our attention to the case of the 2D shallow water model. In comparison to the 1D case, the model incorporates new kind of waves but also has a much higher dimension which will result in the need for additional approximations. All the tests performed in this subsection uses the Krylov subspace enhanced method, the practical implementation of which has been described in Section 4.4.

4.5.2.1 Conjugate gradient (CG)

For CG the stopping criterion depends on the 2-norm of the residual $\|\mathbf{r}(\mathbf{x}_j)\|_2$ and for a tolerance of $\epsilon_{\text{cg}} = 1.$, the minimisation process takes 11 iterations (Fig. 4.24). This gives a starting point or a reference to how our coupling of Parareal with the inexact CG is done.

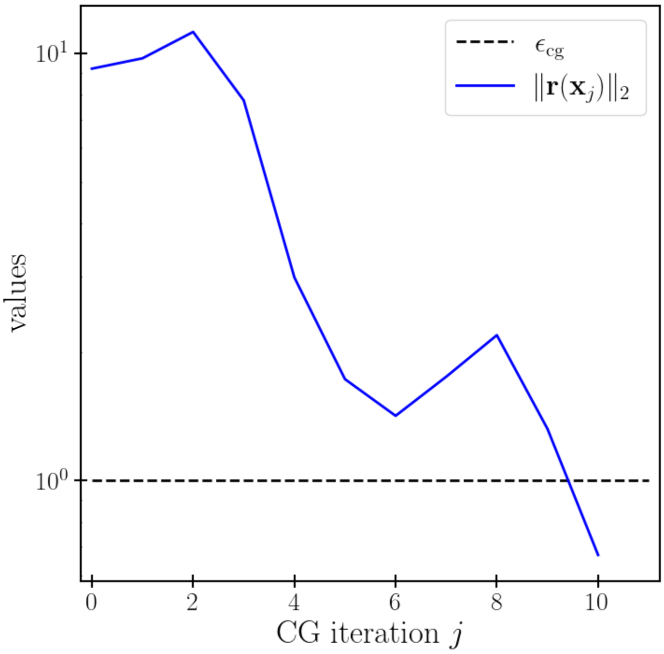


Figure 4.24: Exact CG without using Parareal

4.5.2.2 Conjugate gradient with Parareal

Having calculated the number of minimisation iterations we need for CG, we now use CG with Parareal (CG_Para) with the only modification that the matrix-vector multiplication uses the Parareal algorithm for the forward model. Again the crucial point here is to chose a fixed stopping criterion tolerance for Parareal iterations. Just like in subsection 4.5.1.2 we compare here different choices for ϵ_{p}. We choose $\epsilon_{\mathrm{p}} = 1.d-1$ which is the minimum of the values $\{\xi_j\}_j$ which are produced, at each CG iterations, by the inexact CG algorithm as a stopping criterion for Parareal (see next subsection 4.5.2.4).

Fig. 4.25 shows that CG_Para takes 11 minimisation iterations with the total of 82 Parareal iterations. That means CG with Parareal requires on an average of 7.45 Parareal iterations per CG iteration. In this case the choice of ϵ_{p} is arbitrary, we know after hit and trial that the value of ϵ_{p} is optimal and it does not affect the convergence of CG. Actually this particular value of ϵ_{p} is fully dependent on the configuration of our problem. The value can definitely change if we make any changes to the current configuration by for instance changing the value of N_x or the gravity g.

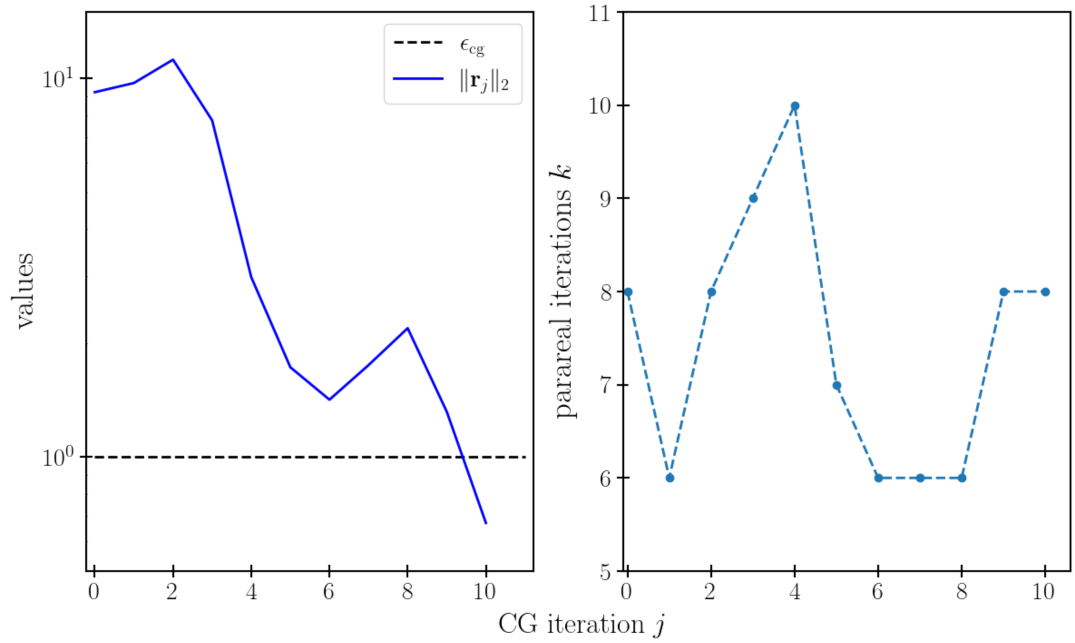


Figure 4.25: Exact CG with matrix-vector product from Parareal, $\epsilon_{\mathrm{p}} = 1.d-1$

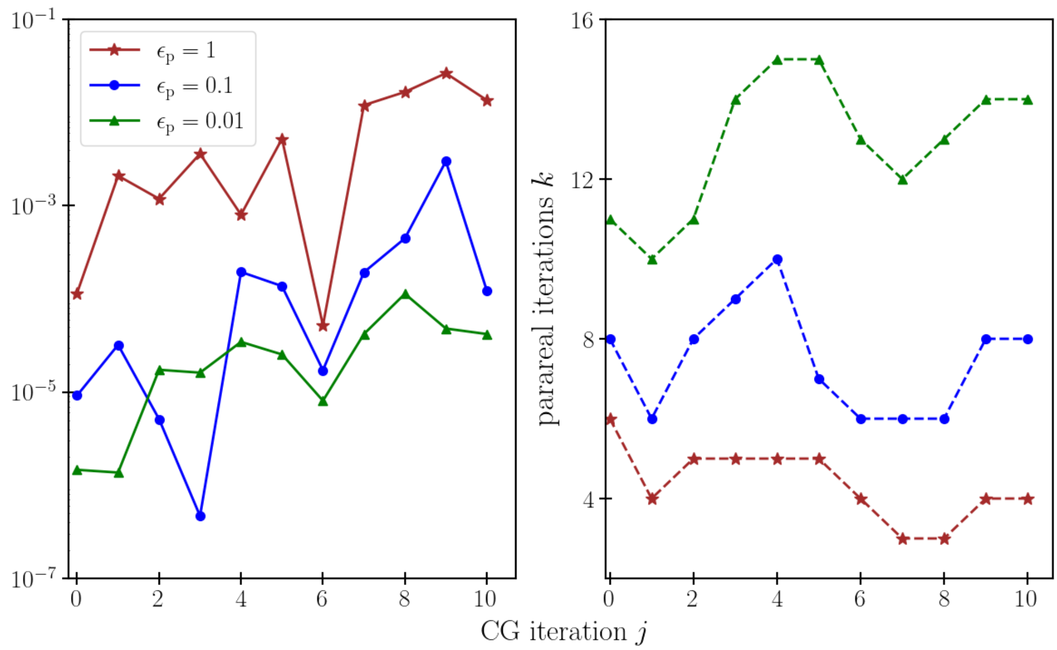


Figure 4.26: Relative error in the 2-norm of the residual (left) and the corresponding Parareal iterations k (right) for different fixed values of ϵ_{p} when using CG

Since we do not know in advance the tolerance ϵ_{p} we perform numerical experiments for 3 different values of ϵ_{p}. The idea is to see the difference in the number of Parareal iterations and the relative error in the 2-norm of the residual at each CG iteration. Fig. 4.26 is about the plot of CG with parareal for the values of ϵ_{p} to be $1.d0, 1.d-1$ and $1.d-2$. Evidently a smaller value of ϵ_{p} implies a more accurate residual value and more Parareal iterations and vice versa. The slight increase in the relative error with the CG_Para iterations is an indication of a higher Parareal precision required in the beginning than in the end of CG_Para. This will reflect the performance of the inexact CG in coming sub-subsections where we will be able to see inexact CG adapt the Parareal's precision.

4.5.2.3 Inexact conjugate gradient with Parareal

Now we run the ICG_Para and we present the results only for the case where the Parareal stopping criterion is based on $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ (see Fig. 4.27), whose values are here computed exactly. Relation (4.59) leads to a stopping criterion for the inexact CG of $\epsilon_{\mathrm{icg}} = 1.075 \times 10^{-2}$.

As a reminder we state again the colour coding for the quantities used in the

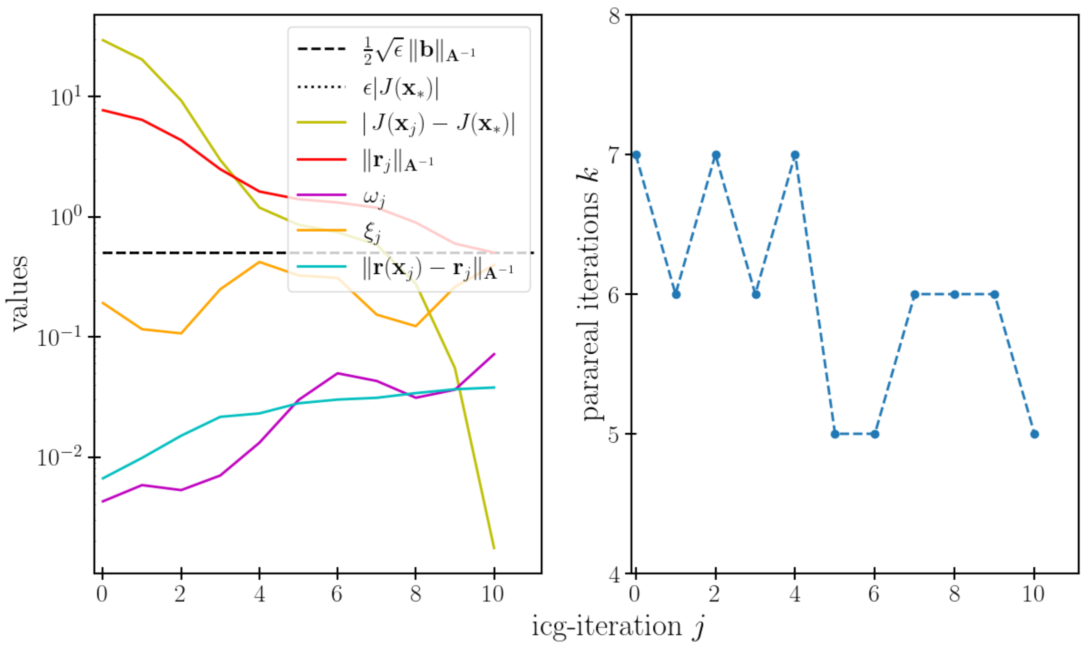


Figure 4.27: Inexact CG with the Parareal stopping criterion $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \xi_j$

plot. The black dashed and the black dotted line are the stopping tolerance for the minimisation and the bound for the quadratic change respectively. For this particular case we can see only one of them because they have almost the same value. The red solid line represents the $\|\mathbf{r}_j\|_{\mathbf{A}^{-1}}$ which ends the minimisation when its value is less than than the stopping tolerance. The yellow solid line is the quadratic change $|J(\mathbf{x}_j) - J(\mathbf{x}_*)|$. The violet solid line is the level of permissible error in the matrix vector product measured by $\|\mathbf{E}_j\|_{\mathbf{A}^{-1},\mathbf{A}}$. The cyan solid line represents the residual gap norm $\|\mathbf{r}(\mathbf{x})_j - \mathbf{r}_j\|_{\mathbf{A}^{-1}}$ which is supposed to stay below the black dashed line.

The quantity ξ_j is assigned the orange solid line which depicts the stopping tolerance for the Parareal for each minimisation iteration. Values of ξ_j for the inexact CG can be compared with the fixed ϵ_{p} in the CG case.

From Fig. 4.27 we see it takes the same number of minimisation iterations (11) as in the above two cases of CG and CG_Para but there is a substantial decrease in the total number of Parareal iterations (66) which is nearly by a factor of 2. As a result the average Parareal iterations per icg-iteration for ICG_Para is 6 which gives a speedup of $40/6 = 6.67$. This is some significant improvement given the fact that the use of ICG_Para with a varying ϵ_{p} is almost twice as more efficient in terms of the total Parareal iterations as the use of CG_Para with a fixed ϵ_{p}.

The next step now is to replace the theoretical estimates with the approximations from subsection 3.3.2 and use our approximations for $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ and check whether the same performance levels can be replicated.

4.5.2.4 Inexact conjugate gradient with practical estimates

As a reminder from chapter 3 that for a practical implementation of our modified inexact CG we need approximations for the various norms used. In realistic applications we don't have access to the matrix $\mathbf{A}$ but only to its product with a vector $\mathbf{Ax}$. To be able to apply the modified inexact CG for a more complicated 2D model, we reconsider the previous approximations from chapter 3 subsection 3.3.2 by providing some more reasonable alternatives for $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ and $\|\mathbf{p}_j\|_{\mathbf{A}}$. Moreover if we recall from section 3.4, equation (3.62) we have

$$\xi_j = \frac{\sqrt{\epsilon}\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_j\|_{\mathbf{A}}^2}{2\phi_{j+1}\|\mathbf{r}_j\|_2^2 + \sqrt{\epsilon}\|\mathbf{b}\|_{\mathbf{A}^{-1}}\|\mathbf{p}_j\|_{\mathbf{A}}} \tag{4.66}$$

The above definition clearly indicates that we also need to know a priori the estimates for $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ and $\|\mathbf{p}_j\|_{\mathbf{A}}$ for estimating ξ_j. We are going to talk about this in the following sub-subsections.

4.5.2.5 A computable estimate for $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$

The approximation for the $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ is done in the same way as in the preceding chapter (Ch. 3, Subsec. 3.4.3) by using the difference from the last Parareal iterate in the 2-norm. That is,

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} = \|\mathbf{P}(k)\mathbf{p}_j - \mathbf{F}^N\mathbf{p}_j\|_2 \approx \|\mathbf{P}(k+1)\mathbf{p}_j - \mathbf{P}(k)\mathbf{p}_j\|_2 \tag{4.67}$$

We verify our estimates for the 2D case by plotting the exact and the approximate $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ for given icg-iterations in in Fig. 4.28. Just like in the 1D case, both the quantities become reasonably close to each other after a few Parareal iterations.

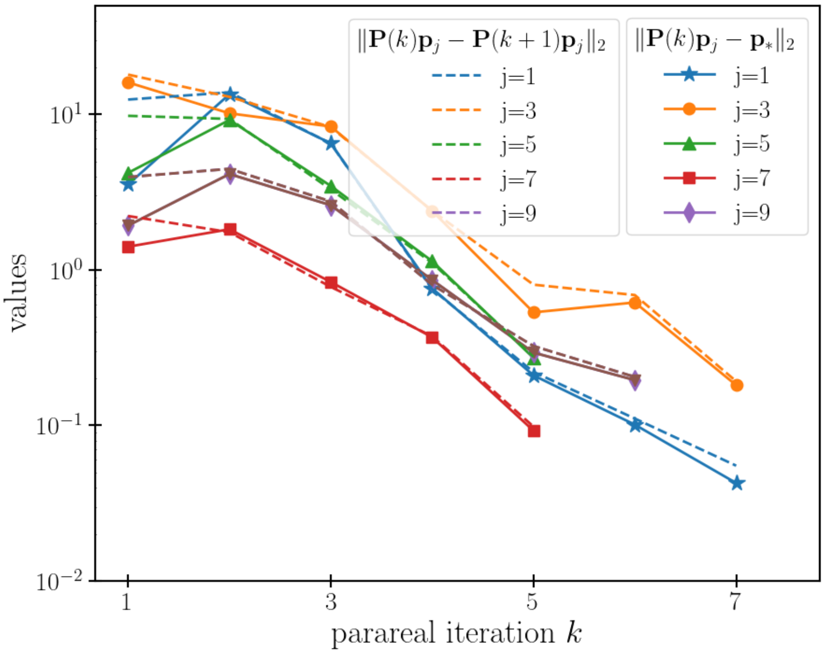


Figure 4.28: Comparison of the exact $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ (plain line) with its approximation (dashed line) by using the last Parareal iterate

4.5.2.6 The $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$

The approximation for $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ as per subsection 3.3.2 Gratton et al. is given as

$$\|\mathbf{b}\|_{\mathbf{A}^{-1}} \approx \begin{cases} \dfrac{\|\mathbf{b}\|_2}{\sqrt{\mu_{\max}(\mathbf{A})}} & \text{for } j = 0 \\ \sqrt{2\,|J_j|}, & \text{for } j = 1, \cdots j_{\max} \end{cases} \tag{4.68}$$

with $J_j = -\frac{1}{2}\mathbf{b}^T\mathbf{x}_j$ being the approximate quadratic $q(\mathbf{x}_j)$ and $\mu_{\max}(\mathbf{A})$ being the maximum eigenvalue of $\mathbf{A}$. Finding the approximation of $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ is troublesome only at the beginning of the minimisation i.e. at icg-iteration $j = 0$. From equation (4.68), we see that the estimate at $j = 0$ requires itself an estimate of the maximum eigenvalue $\mu_{\max}(\mathbf{A})$ of the matrix $\mathbf{A}$ which is difficult to obtain when we do not have access to the full matrix $\mathbf{A}$. The simplest solution that we will take in our experiments is just to use the exact matrix-vector multiplications during the first minimisation iteration. This means we make no use of Parareal and we instead run the fine solver $\mathbf{F}$ sequentially to only exploit the space parallelisation here. Only, the subsequent icg-iterations $j = 1, \cdots, j_{\max}$ will be based on Parareal.

Fig 4.29 shows the comparison of the approximate and the exact values of $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ with respect to the icg-iterations j. The crude estimates for $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ during the first few icg-iterations can be attributed to the less accurate solution guess $\mathbf{x}_j$ which is used to calculate J_j in (4.68). We also take the opportunity to visualise how the approximation made to $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ alone affects the calculation of ξ_j, i.e. we keep the exact value of $\|\mathbf{p}_j\|_{\mathbf{A}}$. To do so we plot the relative error $|\widetilde{\xi}_j - \xi_j|/|\xi_j|$ in Fig. 4.30 where $\widetilde{\xi}_j$ is the approximated ξ_j.

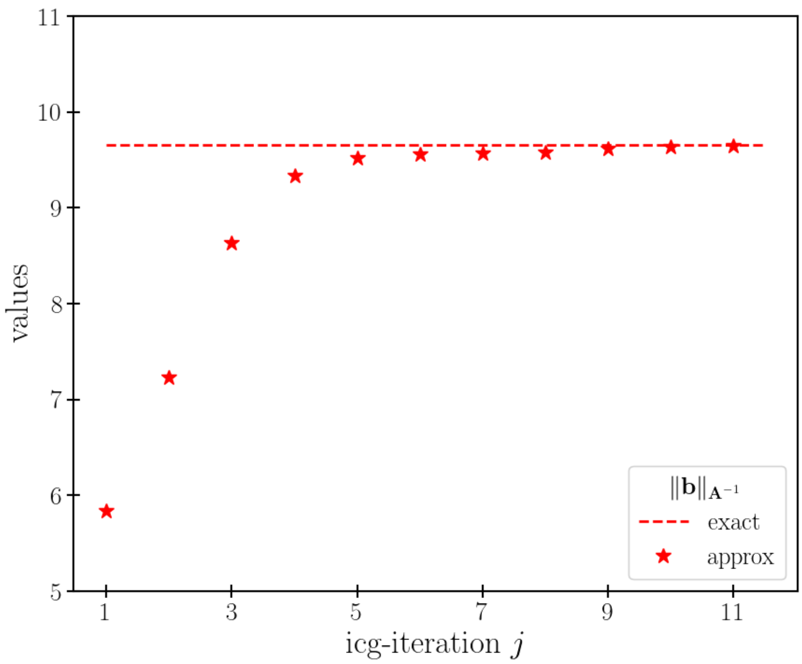


Figure 4.29: Comparison of exact $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ with its approximation for inexact CG iterations j

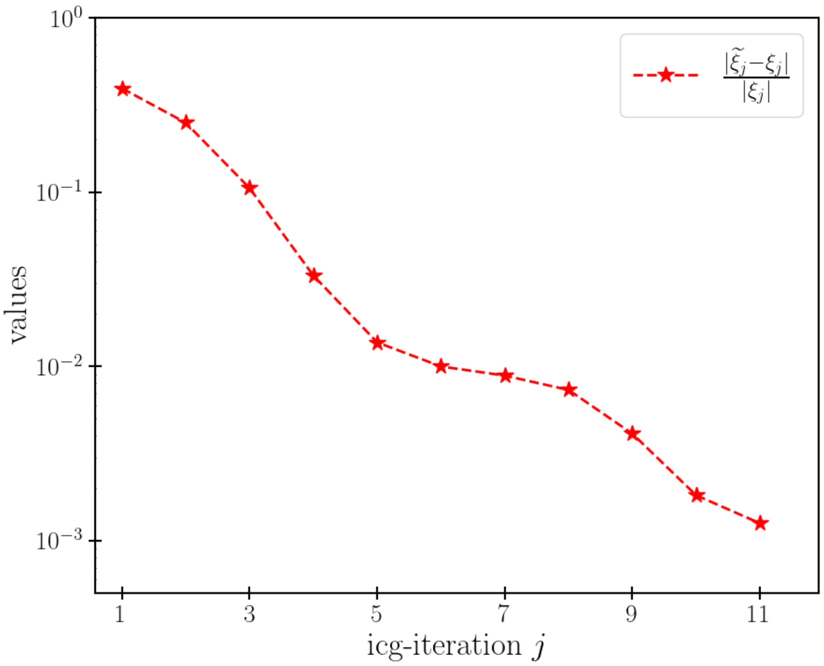


Figure 4.30: Relative error in ξ_j as a function of Parareal iteration k. The approximate and exact values of $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ and $\|\mathbf{p}_j\|_{\mathbf{A}}$ are used respectively to compute $\widetilde{\xi}_j$

We see that we have a higher relative error during the starting icg-iterations as compared to the relative error in the last icg-iterations. This pattern can be justified by the same argument made for Fig. 4.29 that the accuracy of the $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ approximation improves with the succeeding icg-iterations.

We now turn our attention to providing an estimate to $\|\mathbf{p}_j\|_{\mathbf{A}}$.

4.5.2.7 A computable estimate for $\|\mathbf{p}_j\|_{\mathbf{A}}$

Once more from subsection 3.3.2, the approximation suggested for the $\|\mathbf{p}_j\|_{\mathbf{A}}$ requires the information about the trace of $\mathbf{A}$, $\mathrm{Tr}(\mathbf{A})$ given below

$$\|\mathbf{p}_j\|_{\mathbf{A}} \approx \sqrt{\frac{1}{n}\,\mathrm{Tr}(\mathbf{A})}\,\|\mathbf{p}_j\|_2 \tag{4.69}$$

which is not accessible without having access to the full matrix $\mathbf{A}$. As a consequence of not knowing the trace note that we cannot acquire an estimate for ξ_j since it is only obtained if we have access to $\|\mathbf{p}_j\|_{\mathbf{A}}$.

To get around this issue, we propose an alternative by approximating $\|\mathbf{p}_j\|_{\mathbf{A}}$

with the help of the Parareal operator $\mathbf{P}(k)$. By definition $\|\mathbf{p}_j\|_\mathbf{A}$ is given by

$$\|\mathbf{p}_j\|_\mathbf{A} = \sqrt{\mathbf{p}_j^T \mathbf{A} \mathbf{p}_j} \tag{4.70}$$

We see that evaluating $\|\mathbf{p}_j\|_\mathbf{A}$ implicitly involves the dot product of $\mathbf{p}_j$ with the matrix-vector product $\mathbf{A}\mathbf{p}_j$. Since $\mathbf{p}_j$ is the initial condition for the Parareal during icg-iteration j, we can make use of this Parareal run to approximate $\|\mathbf{p}_j\|_\mathbf{A}$ without any extra cost. Indeed the moment when the stopping condition for the Parareal (here for the approximated $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$)

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \approx \|\mathbf{P}(k)\mathbf{p}_j - \mathbf{P}(k+1)\mathbf{p}_j\|_2 \leq \xi_j \tag{4.71}$$

is satisfied, we can use the latest Parareal iterate $\mathbf{P}(k+1)\mathbf{p}_j$ such that

$$\begin{aligned} \|\mathbf{p}_j\|_\mathbf{A} &= \sqrt{\mathbf{p}_j^T \mathbf{A} \mathbf{p}_j} \\ &= \sqrt{\mathbf{p}_j^T [(\mathbf{F}^N)^T \mathbf{F}^N + \alpha \mathbf{W}] \mathbf{p}_j} \\ &= \sqrt{(\mathbf{F}^N \mathbf{p}_j)^T (\mathbf{F}^N \mathbf{p}_j) + \alpha \mathbf{p}_j^T \mathbf{W} \mathbf{p}_j} \\ &\approx \sqrt{[\mathbf{P}(k+1)\mathbf{p}_j]^T [\mathbf{P}(k+1)\mathbf{p}_j] + \alpha \mathbf{p}_j^T \mathbf{W} \mathbf{p}_j} \end{aligned} \tag{4.72}$$

We emphasize the fact that in this formulation the computation of $\|\mathbf{p}_j\|_\mathbf{A}$ and ξ_j are interlinked. Let us denote the approximation for $\|\mathbf{p}_j\|_\mathbf{A}$ and ξ_j by $\widetilde{\|\mathbf{p}_j\|_\mathbf{A}}$ and $\widetilde{\xi_j}$ respectively. For a given icg-iteration j, with each Parareal iteration k the value of $\widetilde{\|\mathbf{p}_j\|_\mathbf{A}}(k)$ and thereafter the value of $\widetilde{\xi_j}(k)$ are updated till the Parareal stopping criterion is satisfied. We refer to Algorithm 14 which provides the required steps to correctly compute the approximate $\|\mathbf{p}_j\|_\mathbf{A}$.
To verify our estimates the relative errors in $\|\mathbf{p}_j\|_\mathbf{A}$ and ξ_j are plotted as a function of the Parareal iteration k in figures Fig. 4.31 and Fig. 4.32 respectively. The relative error in ξ_j is shown only using the $\|\mathbf{p}_j\|_\mathbf{A}$ estimate i.e. the exact value of $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ is used. The values are plotted for the icg-iterations $j = 1, 3, 5, 7, 9$ and 11.

Looking at the two figures we see that the both relative errors follow an almost similar decrease with the increasing Parareal iterations which one would normally expect. The relative error in ξ_j at the last Parareal iteration in Fig. 4.32 (the one which we actually use) is low and we can say that ξ_j is well approximated when exact $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ is used. Again if we look at Fig. 4.30 the relative error in ξ_j is

Algorithm 14 APPROXIMATION FOR $\|\mathbf{p}_j\|_{\mathbf{A}}$

1: **Given**: icg-iteration $j = 1, \cdots, j_{\max}$
2: **for** $k = 1, \cdots, k_{\max} - 1$ **do**
3: Run Parareal for $k+1$ iterations
4: Find $\widetilde{\|\mathbf{p}_j\|_{\mathbf{A}}} = \sqrt{[\mathbf{P}(k+1)\mathbf{p}_j]^T[\mathbf{P}(k+1)\mathbf{p}_j] + \alpha \mathbf{p}_j^T \mathbf{W} \mathbf{p}_j}$
5: Get the corresponding $\widetilde{\xi}_j$ using (4.66)
6: **if** $\|\mathbf{P}(k+1)\mathbf{p}_j - \mathbf{P}(k)\mathbf{p}_j\|_2 \le \widetilde{\xi}_j$ **then**
7: **break**
8: **else**
9: $k = k+1$
10: **end if**
11: **end for**

high in the initial icg-iterations and this can affect $\widetilde{\xi}_j$ when the approximations for $\|\mathbf{p}_j\|_{\mathbf{A}}$ and $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ are used. To see the impact of using both $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ and $\|\mathbf{p}_j\|_{\mathbf{A}}$ approximations on $\widetilde{\xi}_j$ we plot again the relative error in ξ_j in Fig. 4.33.

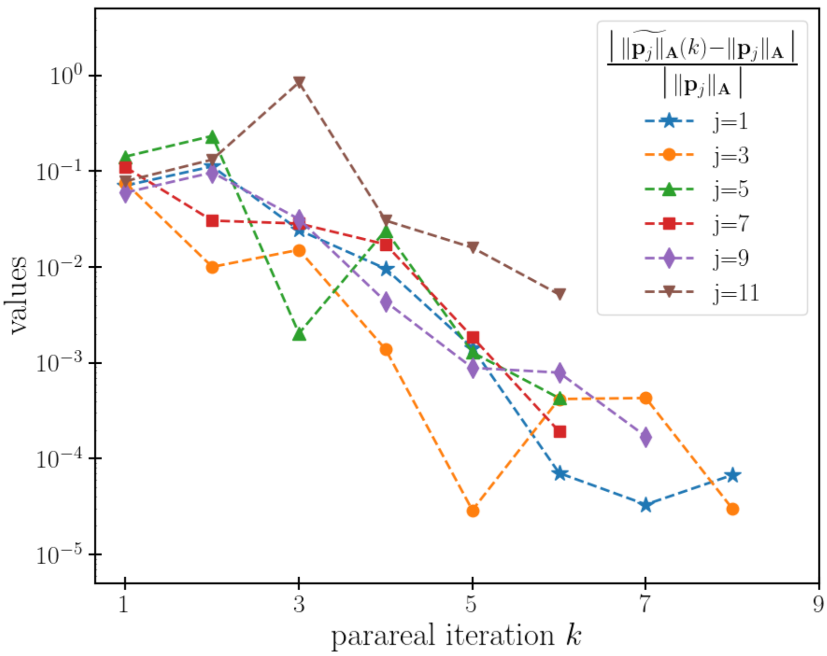


Figure 4.31: Relative error in $\|\mathbf{p}_j\|_{\mathbf{A}}$ as a function of Parareal iteration k

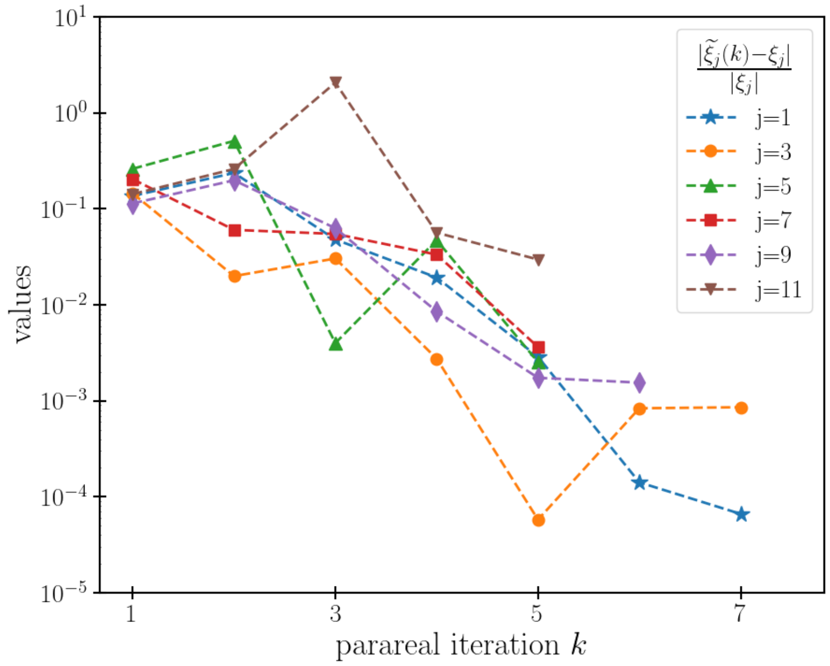


Figure 4.32: Relative error in ξ_j as a function of Parareal iteration k. The approximate $\|\mathbf{p}_j\|_\mathbf{A}$ and exact $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ are used to compute $\widetilde{\xi}_j$

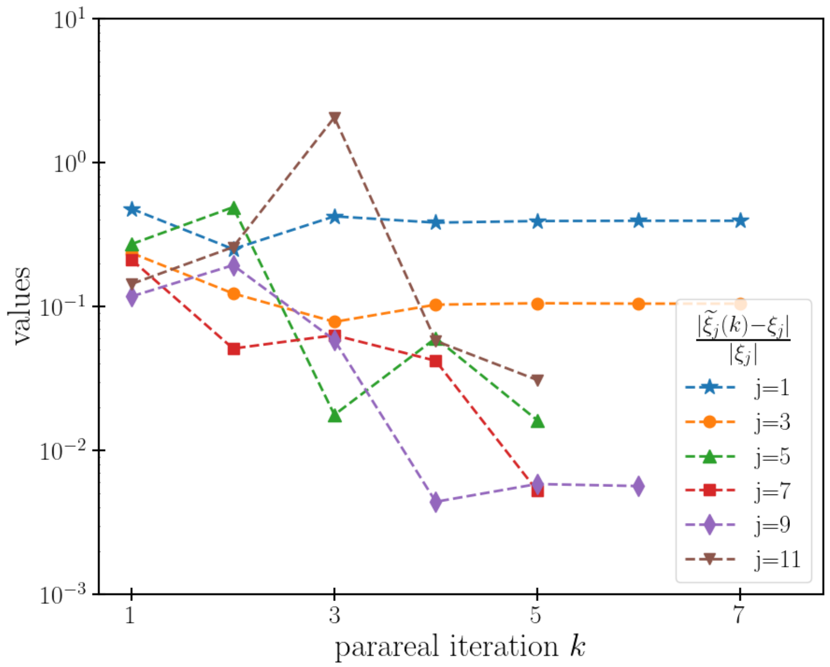


Figure 4.33: Relative error in ξ_j as a function of Parareal iteration k. Both approximate $\|\mathbf{b}\|_{\mathbf{A}^{-1}}$ and $\|\mathbf{p}_j\|_\mathbf{A}$ are used to compute $\widetilde{\xi}_j$

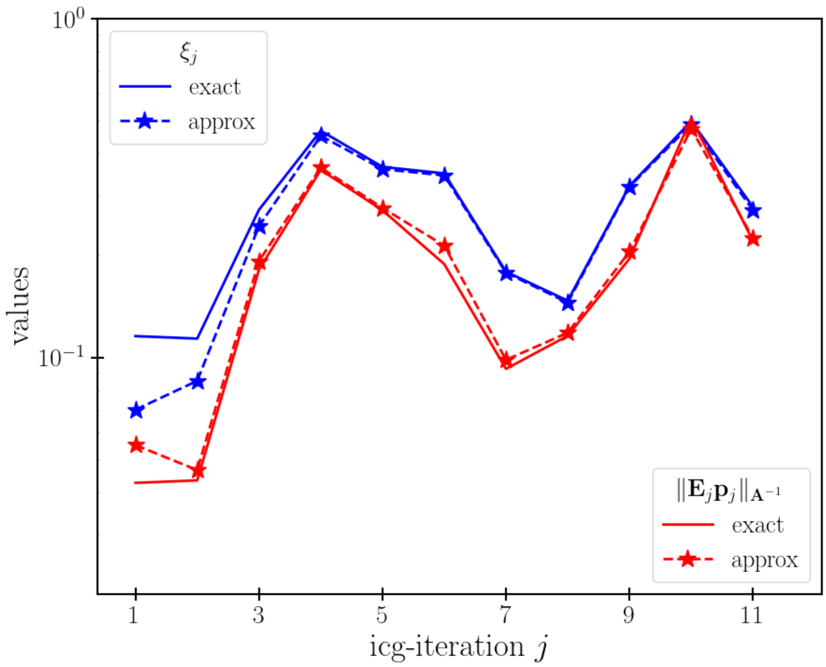


Figure 4.34: Exact and approximate values of $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ as opposed to the exact and approximate values of ξ_j

In Fig. 4.34 above the exact and approximate values of ξ_j are plotted with the exact and approximate values of $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ and we see how it is consistently lower than ξ_j.

Finally we present our result after combining all the practical estimates for the inexact CG by using the ICG_Para_Prac algorithm outlined in Algorithm 15. This is illustrated in Fig. 4.35 which shows the plot of the minimisation quantities on the left hand side and a comparison of Parareal iteration evolution of ICG_Para_Prac (in blue) with that of ICG_Para (in green) from Fig. 4.27 on the right hand side. We show the approximations starting from icg-iteration 1 since we run the icg-iteration 0 with exactly using the fine solver. Note that the practical minimisation criterion (3.54)

$$(J_{j-d} - J_j) \le \frac{1}{4}\epsilon_{\text{icg}}|J_j| \tag{4.73}$$

requires an integer d which is used to measure the quadratic difference between the d successive icg-iteration steps. We choose $d = 2$ and the minimisation terminates at icg-iteration 11 with a total of 76 Parareal iterations. Thus, on an average it takes 6.91 Parareal iterations per icg-iteration giving us a speedup of $40/6.91 = 5.788$.

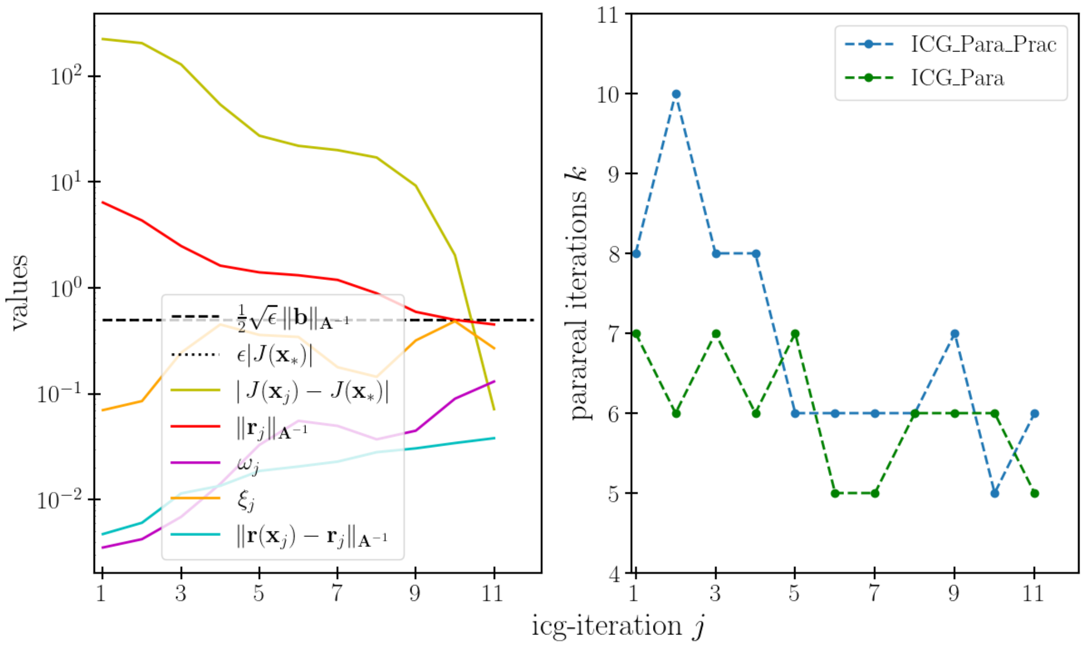


Figure 4.35: Minimisation results using ICG_Para_Prac (on left) and a comparison of the Parareal iterations between ICG_Para and ICG_Para_Prac (on right)

By a comparison of the Parareal iteration evolution on the right image our motive is to show the impact of using practical estimates in ICG_Para_Prac on the Parareal iterations as opposed to using the exact values of all the quantities in ICG_Para. We draw attention to the fact that by approximating $\mathbf{p}_*$ using the last Parareal iterate in ICG_Para_Prac we end up using one more Parareal iteration than needed (see equation (4.67)). Consequently, there is an addition to the total number of Parareal iterations by 1 for each minimisation iteration.

We observe from Fig. 4.35 that if we remove the additional 1 Parareal iteration from each icg-iteration we get a total of 65 Parareal iterations which is almost the same when we use ICG_Para (66).

Algorithm 15 ICG WITH ALL APPROXIMATIONS FOR 2D-SWE

1: **Given**: Symmetric positive definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, right hand side vector $\mathbf{b} \in \mathbb{R}^n$, tolerance ϵ_{icg}
2: Set $\mathbf{x}_0 = 0$, $\mathbf{r}_0 = -\mathbf{b}$, $\mathbf{p}_0 = \mathbf{r}_0$, $\beta_0 = \|\mathbf{b}\|_2^2$, $\mathbf{u}_1 = \mathbf{b}/\beta_0$, $\phi_0 = j_{\max}$, $\Phi_0 = 1$, reorth = True
3: **for** $j = 0, \ldots j_{\max}$ **do**
4:    **if** $j = 0$ **then**

Call the exact matrix-vector subroutine and set $\mathbf{c}_j = \mathbf{A}\mathbf{p}_j$
else
$J_j = \frac{1}{2}\mathbf{b}^T\mathbf{x}_j$
$\|\mathbf{b}\|_{\mathbf{A}^{-1}} = \sqrt{2|J_j|}$
for $k = 1, \cdots, N-1$ **do**
Run Parareal with $k+1$ iterations
Find the approximation $\widetilde{\|\mathbf{p}_j\|_{\mathbf{A}}}(k)$ using (4.72)
Compute the corresponding $\widetilde{\xi_j}(k)$ from (4.66)
$e = \|\mathbf{P}(k+1)\mathbf{p}_j - \mathbf{P}(k)\mathbf{p}_j\|_2$
if $e < \widetilde{\xi_j}(k)$ **then**
Set $\widehat{\xi_j} = e$
$\mathbf{c}_1 = (\mathbf{F}^N)^T\mathbf{P}(k+1)\mathbf{p}_j$
break
end if
end for
Compute $\hat{\phi}_j$ from $\hat{\omega}_j$
$\Phi_{j+1} = \Phi_j - \hat{\phi}_j^{-1}$
if $j < j_{\max}$ **then**
$\phi_{j+1} = (j_{\max} - j)/\Phi_{j+1}$
else
$\phi_{j+1} = \phi_j$
end if
end if
$\alpha_j = \beta_j/\mathbf{p}_j^T\mathbf{c}_j$
$\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{p}_j$
$\mathbf{r}_{j+1} = \mathbf{r}_j + \alpha_j\mathbf{c}_j$
if $(J_{j+1-d} - J_{j+1}) \leq \frac{1}{4}\,\epsilon_{\text{icg}}|J_{j+1}|$ **then**
break
end if
if (reorth) **then**
for $i = 1, \cdots, j$ **do**
$\mathbf{r}_{j+1} = \mathbf{r}_{j+1} - (\mathbf{u}_i^T\mathbf{r}_{j+1})\mathbf{u}_i$
end for
$\beta_{j+1} = \mathbf{r}_{j+1}^T\mathbf{r}_{j+1}$
$\mathbf{u}_{j+1} = \mathbf{r}_{j+1}/\sqrt{\beta_{j+1}}$
else
$\beta_{j+1} = \mathbf{r}_{j+1}^T\mathbf{r}_{j+1}$
end if
$\mathbf{p}_{j+1} = -\mathbf{r}_{j+1} + (\beta_{j+1}/\beta_j)\mathbf{p}_j$
end for

4.6 Using multiple observations

In the entire manuscript our idea has been to devise a strategy for running a time-parallel incremental 4D-Var. The methodology and experiments have been shown for a simple data assimilation problem where the system observes the true state at the last time window.

To generalise our approach for multiple observations independently of the model we add a test case where we use more than one observation scattered at different times. We observe the full state vector and keep the same total period of integration. In practice this should improve the performance. If we have observations $\mathbf{y}_i$ at time t_i we have the modified cost function

$$J(\mathbf{x}_0) = \frac{1}{2p}\sum_{i=1}^{p} \|\mathbf{F}^{N_i}\mathbf{x}_0 - \mathbf{y}_i\|_2^2 \tag{4.74}$$

and the gradient

$$\nabla J(\mathbf{x}_0) = \frac{1}{p}\sum_{i=1}^{p} (\mathbf{F}^{N_i})^T(\mathbf{F}^{N_i}\mathbf{x}_0 - \mathbf{y}_i) \tag{4.75}$$

Again we approximate the gradient by doing the forward integration using the Parareal algorithm, but now the Parareal operator $\mathbf{P}^i(k)$ is the solution of $\mathbf{P}(k)$ at the end of the time window N_i, $N_i \leq N$.

$$\nabla J(\mathbf{x}_0) \approx \frac{1}{p}\sum_{i=1}^{p} (\mathbf{F}^{N_i})^T(\mathbf{P}^i(k)\mathbf{x}_0 - \mathbf{y}_i) \tag{4.76}$$

The linear system we are supposed to solve as a result of setting the gradient to zero is

$$\mathbf{A}\mathbf{x}_0 = \mathbf{b} \tag{4.77}$$

where $\mathbf{A} = \frac{1}{p}\sum_{i=1}^{p} (\mathbf{F}^{N_i})^T\mathbf{P}^i(k)$ and $\mathbf{b} = \frac{1}{p}\sum_{i=1}^{p}(\mathbf{F}^{N_i})^T\mathbf{y}_i$. And now for the $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ and its approximation we have

$$\begin{aligned}\mathbf{E}_j\mathbf{p}_j &= \frac{1}{p}\sum_{i=1}^{p}(\mathbf{F}^{N_i})^T(\mathbf{F}^{N_i} - \mathbf{P}^i(k))\mathbf{p}_j \\ &= \frac{1}{p}\sum_{i=1}^{p}\mathbf{E}^i\mathbf{p}_j\end{aligned} \tag{4.78}$$

where $\mathbf{E}^i = (\mathbf{F}^{N_i})^T(\mathbf{F}^{N_i} - \mathbf{P}^i(k))$. Taking the $\mathbf{A}^{-1}$ norm on both the sides and using the triangle inequality we obtain

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \frac{1}{p}\sum_{i=1}^{p}\|\mathbf{E}^i\mathbf{p}_j\|_{\mathbf{A}^{-1}} \tag{4.79}$$

We know from (4.67) that in similar manner for each i, $\|\mathbf{E}^i\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ can be approximated as

$$\|\mathbf{E}^i\mathbf{p}_j\|_{\mathbf{A}^{-1}} = \|\mathbf{P}^i(k)\mathbf{p}_j - \mathbf{F}^{N_i}\mathbf{p}_j\|_2 \approx \|\mathbf{P}^i(k+1)\mathbf{p}_j - \mathbf{P}^i(k)\mathbf{p}_j\|_2 \tag{4.80}$$

Putting the approximations back into (4.79) we find

$$\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \frac{1}{p}\sum_{i=1}^{p}\|\mathbf{P}^i(k+1)\mathbf{p}_j - \mathbf{P}^i(k)\mathbf{p}_j\|_2 \tag{4.81}$$

Now if we impose the summation term on right hand side of (4.81) with the condition

$$\sum_{i=1}^{p}\|\mathbf{P}^i(k+1)\mathbf{p}_j - \mathbf{P}^i(k)\mathbf{p}_j\|_2 \leq p\,\xi_j \tag{4.82}$$

then we have $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \xi_j$ and so the Parareal's accuracy is controlled by the quantity ξ_j for a given icg-iteration j.

Apart from the change in the $\|\mathbf{p}_j\|_{\mathbf{A}}$ estimate in particular for the 2D model provided in sub-subsection 4.5.2.7 which we explain next, the approximations for the other norms remain unchanged. Following along the lines of (4.72) we have

$$\begin{aligned}
\|\mathbf{p}_j\|_{\mathbf{A}} &= \sqrt{\mathbf{p}_j^T\left(\frac{1}{p}\sum_{i=1}^{p}(\mathbf{F}^{N_i})^T\mathbf{F}^{N_i} + \alpha\mathbf{W}\right)\mathbf{p}_j} \\
&= \sqrt{\frac{1}{p}\sum_{i=1}^{p}(\mathbf{F}^{N_i}\mathbf{p}_j)^T(\mathbf{F}^{N_i}\mathbf{p}_j) + \alpha\mathbf{p}_j^T\mathbf{W}\mathbf{p}_j} \\
&\approx \sqrt{\frac{1}{p}\sum_{i=1}^{p}[\mathbf{P}^i(k+1)\mathbf{p}_j]^T[\mathbf{P}^i(k+1)\mathbf{p}_j] + \alpha\mathbf{p}_j^T\mathbf{W}\mathbf{p}_j}
\end{aligned} \tag{4.83}$$

We run a test case with the 1D model by running the ICG_Para using the same parameter values from sub-subsection 4.5.1.3 except that we observe the full state vector at time windows $N_i, i = 4, 8, 12, 16$ and 20. The results of the test run are plotted in Fig. 4.36 and are compared with Fig. 4.20 which contains the plot of ICG_Para with one observation. When multiple observations are used the minimisation ends after 16 minimisation iterations and uses an average of 4 Parareal iterations per icg-iteration as compared to the 24 icg-iterations and an average of 5.58 Parareal iterations per icg-iteration when only one observation is used. The relatively rapid convergence of the inexact CG and parareal is expected

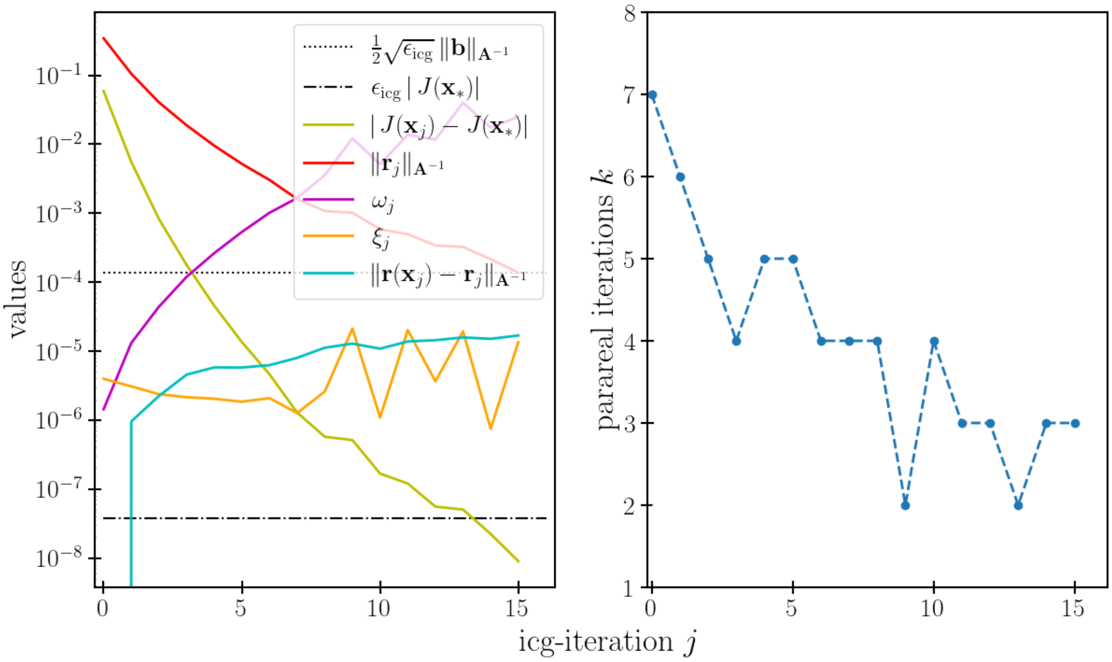


Figure 4.36: Inexact CG with Parareal stopping criterion $\|\mathbf{E}_j\mathbf{p}_j\|_{\mathbf{A}^{-1}} \leq \xi_j$ when multiple observations are used

due to the presence of some observations at times which are closer to the initial time. Theoretically the Parareal error is supposed to be smaller at the initial times compared to the error at the last time window. Therefore the weight of the term $\|\mathbf{E}^i\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ from the initial times is smaller than that of the last time window.

To demonstrate this, the evolution of $\|\mathbf{E}^i\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ with respect to the icg-iteration j is plotted in Fig. 4.37 where the norm values are smaller when the observations are closer to the initial time than when they are at the end of the integration time.

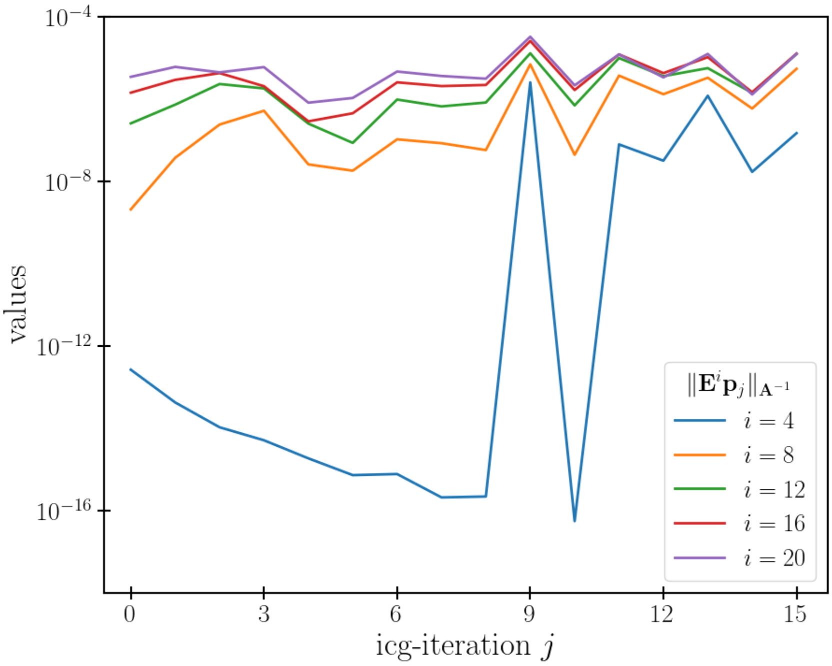


Figure 4.37: Evolution of $\|\mathbf{E}^i\mathbf{p}_j\|_{\mathbf{A}^{-1}}$ when observations are at time windows N_i as a function of the inexact CG iteration j

Conclusion and further remarks

Conclusion

The objective of this PhD thesis is to explore the idea of introducing time parallelisation to the data assimilation algorithms. In this manuscript we present a way of coupling a parallel-in-time method (Parareal) with variational data assimilation (incremental 4D-Var). An overview of the state of the art for variational data assimilation is provided in chapter 1 and for time parallel methods in chapter 2.

Our presentation focusses on time parallelising the inner loop of the incremental 4D-Var by utilising Parareal for the tangent-linear model integration. The study is done in two phases: by first establishing a methodology in chapter 3 and then validating it through numerical experiments in chapter 4. When Parareal is used for minimising the inner loop cost function it results in an approximate linear system. The system is solved by using an inexact version of the conjugate gradient (CG) method proposed by Gratton et al [Gratton et al., 2021]. The advantage of using the proposed method over the usual CG is that it allows us to adaptively control the required Parareal's accuracy. The usage of the inexact CG with the settings as in Gratton et al paper showed that the original precision criterion is not sharp enough. We proposed a modified precision criterion (3.61) based on the original criterion which significantly reduces the required number of required Parareal iterations. Also in theory the inexact CG employs the norms of the quantities (including the modified stopping criterion) which are not easy to obtain in practice. To deliver a feasible implementation of the inexact CG, we used some of the easy to compute approximations suggested in [Gratton et al., 2021] and proposed our own approximations for some of the other quantities.

Next we performed a series of numerical experiments on the linearised 1D and 2D shallow water equations. These equations are widely used for modelling important geophysical phenomena. Also the 1D and 2D model provide a challenge to Parareal in terms of the level of complexity and the number of physical parameters to play with. For the 2D model, a variant of the usual Parareal called the Krylov subspace enhanced Parareal is used which is known for accelerating the convergence of hyperbolic problems. For the experiments we used different versions of CG and inexact CG (table 3.1) for both models. We compared the performances by noting the total number of Parareal iterations, the average number of Parareal iterations per CG/inexact CG iterations and the parallel speedup. On one hand for running CG with Parareal we had to resort to a hit-and-trial approach for finding a fixed Parareal tolerance. On the other hand the different inexact CG versions using Parareal automatically provided a Parareal tolerance using the original (only for 1D case), the modified and the approximated modified precision criterion. In the end we presented the crux of our analysis by running the inexact CG version comprising all the approximations and modifications discussed in the manuscript.

We conclude by mentioning the most important point that of our study. Using our modified inexact CG with all the approximations outperforms CG with Parareal (with the correct fixed Parareal tolerance) for both 1D and 2D model by achieving higher parallel speedups.

Perspectives

There are still a few gaps and unexplored ideas which could lead to further research directions. In the manuscript all the results have been shown by the inexact CG method proposed by Gratton et al. [Gratton et al., 2021] which adaptively controls the Parareal's accuracy. One idea has been to reuse the information from the Parareal iterates of the previous icg-iterations. One instance is that when we use the Krylov subspace enhanced Parareal we build our Krylov subspace basis all over again after every icg-iteration. Reusing the existing previous Krylov basis vectors for building the next Krylov subspace basis proved to be unsuccessful. This is specific to CG since we know that it has a property that all the direction vectors $\mathbf{p}_j$ are $\mathbf{A}$-conjugate. However, the problem remain to solve remain quite similar from one minimisation iteration to next and so it should be possible to extract useful information from pervious Krylov bases if a different minimisation method is used. This leads to another point that we have not investigated other minimisation algorithms. We can adapt the results from the Gratton et al. paper to another minimisation algorithms where we can take more benefit from the coupling of Parareal and data assimilation. Moreover the linear system which we solve for the inner loop using inexact CG is non-symmetric and is the reason why we have to use reorthogonalisation at the end of each icg-iteration. There are more robust methods present in the literature for dealing with non-symmetric matrices such as the GMRES, for which we would need to derive a similar inexact method.

Another point of interest is the introduction of time parallelism in the adjoint direction where we already discussed some preliminary ideas in chapter 3. What remains as an obstacle is implementing the inexact CG when all approximations are used. The existing approximation for $\|\mathbf{E}_j \mathbf{p}_j\|_{\mathbf{A}^{-1}}$ is no longer valid since it requires that we use the exact adjoint. But once the estimate is found the implementation should be straightforward. A natural extension to this idea is making use of the asynchronous Parareal iterations [Magoulès et al., 2018, Magoulès and Gbikpi-Benissan, 2018] where the algorithm has no synchronisation point. In that case the forward and adjoint parallel runs are done simultaneously and they do not wait for sharing any information. This way there is no communication overhead and we obtain an even higher parallel speedup.

One of the biggest blocking factors in improving the Parareal's speedup is the cost of the coarse solver which justifies the future research directions for the development of *adaptive coarse solvers*. In section 4.4 we tried to use the Krylov enhanced Parareal without using the coarse solver. We did not get any concrete results and what we observed in (4.50) is that the Parareal error was more dependent on the initial condition used. The understanding of the behaviour of the error in Krylov enhanced Parareal would require somehow being able to compute (at least) an estimate of $(\mathbf{F} - \mathbf{G})$ applied to the current vector $\mathbf{x}$.

From the perspective of the parallel architecture performance we have not talked about the scaling properties of our coupling. Scaling usually refers to the ability of a hardware or a software to handle more work as the size of the problem or the number of cores are varied. The scaling for a certain application can be studied and tested using two ways: by doing the strong and the weak scaling. In strong scaling the problem size is kept constant and the number of cores are increased. The total workload per core gets reduced and as a result an important question can be asked: If the number of cores are doubled, would the total computation time be cut in half? Amdahl [Amdahl, 1967] pointed out that the reduction in the execution time has an upper limit and it depends on how much the part of the code is parallelisable. This is known as the Amdahl's law and for an application with p as the parallel part, $1-p$ as the serial part and N as the number of cores it is written as

$$S = \frac{1}{\left(1 - p + \frac{p}{N}\right)} \tag{4.84}$$

Strong scaling becomes important in our case since we are adding more cores to our fixed problem through time parallelisation. But it is hard to achieve due to the fact that the communication overhead becomes large enough and the classical domain decomposition methods reach saturation. But as discussed above the use of an adaptive coarse solver together with the asynchronous Parareal iterations can help us to remove the communication overhead and to perform strong scaling.

The applications where our method can be applied is restricted to seasonal forecasts where the model is integrated for short to medium range of time (a few weeks to a few months). The seasonal prediction is an initial condition problem which makes use of the data assimilation algorithms. An extension to this can be towards applications in climate modelling where the time period of integration is for several decades. In that case we can no longer control the initial condition as it is forgotten by the system and it will not have any impact on long time integrations. A different strategy needs to be applied by controlling the parameters of the numerical methods for instance or by controlling the tendencies, taking averages over a period.

Bibliography

[Amdahl, 1967] Amdahl, G. M. (1967). Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485.

[Andersson et al., 1998] Andersson, E., Haseler, J., Undén, P., Courtier, P., Kelly, G., Vasiljevic, D., Brankovic, C., Gaffard, C., Hollingsworth, A., Jakob, C., et al. (1998). The ecmwf implementation of three-dimensional variational assimilation (3d-var). iii: Experimental results. *Quarterly Journal of the Royal Meteorological Society*, 124(550):1831–1860.

[Arakawa and Lamb, 1977] Arakawa, A. and Lamb, V. R. (1977). Computational design of the basic dynamical processes of the ucla general circulation model. *General circulation models of the atmosphere*, 17(Supplement C):173–265.

[Asch et al., 2016] Asch, M., Bocquet, M., and Nodet, M. (2016). *Data assimilation: methods, algorithms, and applications*. SIAM.

[Aubanel, 2011] Aubanel, E. (2011). Scheduling of tasks in the parareal algorithm. *Parallel Computing*, 37(3):172–182.

[Axelsson, 1996] Axelsson, O. (1996). *Iterative solution methods*. Cambridge university press.

[Baffico et al., 2002] Baffico, L., Bernard, S., Maday, Y., Turinici, G., and Zérah, G. (2002). Parallel-in-time molecular-dynamics simulations. *Physical Review E*, 66(5):057701.

[Bal, 2003] Bal, G. (2003). Parallelization in time of (stochastic) ordinary differential equations. *Math. Meth. Anal. Num.(submitted)*.

[Bal, 2005] Bal, G. (2005). On the convergence and the stability of the parareal algorithm to solve partial differential equations. In *Domain decomposition methods in science and engineering*, pages 425–432. Springer.

[Bal and Maday, 2002] 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 *Recent developments in domain decomposition methods*, pages 189–202. Springer.

[Bannister, 2017] Bannister, R. (2017). A review of operational methods of variational and ensemble-variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 143(703):607–633.

[Bellen and Zennaro, 1989] 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.

[Berry et al., 2012] Berry, L. A., Elwasif, W., Reynolds-Barredo, J. M., Samaddar, D., Sanchez, R., and Newman, D. E. (2012). Event-based parareal: A data-flow based implementation of parareal. *Journal of Computational Physics*, 231(17):5945–5954.

[Björck, 1994] Björck, Å. (1994). Numerics of gram-schmidt orthogonalization. *Linear Algebra and Its Applications*, 197:297–316.

[Björck, 1996] Björck, Å. (1996). *Numerical methods for least squares problems.* SIAM.

[Bouras and Frayssé, 2005] Bouras, A. and Frayssé, V. (2005). Inexact matrix-vector products in krylov methods for solving linear systems: a relaxation strategy. *SIAM Journal on Matrix Analysis and Applications*, 26(3):660–678.

[Bouttier and Courtier, 2002] Bouttier, F. and Courtier, P. (2002). Data assimilation concepts and methods march 1999. *Meteorological training course lecture series. ECMWF*, 718:59.

[Burrage, 1995] Burrage, K. (1995). *Parallel and sequential methods for ordinary differential equations.* Clarendon Press.

[Caldas Steinstraesser et al., 2021] Caldas Steinstraesser, J. G., Guinot, V., and Rousseau, A. (2021). Modified parareal method for solving the two-dimensional nonlinear shallow water equations using finite volumes. *The SMAI journal of computational mathematics*, 7:159–184.

[Carrassi et al., 2018] Carrassi, A., Bocquet, M., Bertino, L., and Evensen, G. (2018). Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535.

[Chartier and Philippe, 1993] Chartier, P. and Philippe, B. (1993). A parallel shooting technique for solving dissipative ode's. *Computing*, 51(3):209–236.

[Chen et al., 2014] Chen, F., Hesthaven, J. S., and Zhu, X. (2014). On the use of reduced basis methods to accelerate and stabilize the parareal method. In *Reduced Order Methods for modeling and computational reduction*, pages 187–214. Springer.

[Cortial and Farhat, 2009] 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.

[Courtier et al., 1998] Courtier, P., Andersson, E., Heckley, W., Vasiljevic, D., Hamrud, M., Hollingsworth, A., Rabier, F., Fisher, M., and Pailleux, J. (1998). The ecmwf implementation of three-dimensional variational assimilation (3d-var). i: Formulation. *Quarterly Journal of the Royal Meteorological Society*, 124(550):1783–1807.

[Courtier and Talagrand, 1990] Courtier, P. and Talagrand, O. (1990). Variational assimilation of meteorological observations with the direct and adjoint shallow-water equations. *Tellus A: Dynamic Meteorology and Oceanography*, 42(5):531–549.

[Courtier et al., 1994] Courtier, P., Thépaut, J.-N., and Hollingsworth, A. (1994). A strategy for operational implementation of 4d-var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120(519):1367–1387.

[Cushman-Roisin and Beckers, 2011] Cushman-Roisin, B. and Beckers, J.-M. (2011). *Introduction to geophysical fluid dynamics: physical and numerical aspects*. Academic press.

[Dai and Maday, 2013] Dai, X. and Maday, Y. (2013). Stable parareal in time method for first-and second-order hyperbolic systems. *SIAM Journal on Scientific Computing*, 35(1):A52–A78.

[Daley, 1993] Daley, R. (1993). *Atmospheric data analysis*. Number 2. Cambridge university press.

[Daley, 1997] Daley, R. (1997). Atmospheric data assimilation (gtspecial issueltdata assimilation in meteology and oceanography: Theory and practice). *Journal of the Meteorological Society of Japan. Ser. II*, 75(1B):319–329.

[Dolean et al., 2015] Dolean, V., Jolivet, P., and Nataf, F. (2015). *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. SIAM.

[Durran, 2013] Durran, D. R. (2013). *Numerical methods for wave equations in geophysical fluid dynamics*, volume 32. Springer Science & Business Media.

[Eghbal et al., 2017] 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.

[Errico, 1997] Errico, R. M. (1997). What is an adjoint model? *Bulletin of the American Meteorological Society*, 78(11):2577–2592.

[Eyre, 1991] Eyre, J. (1991). A fast radiative transfer model for satellite sounding systems. *ECMWF Tech. Memo 176*.

[Farhat and Chandesris, 2003] 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 et al., 2006] 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(5):697–724.

[Fischer et al., 2005] Fischer, P. F., Hecht, F., and Maday, Y. (2005). A parareal in time semi-implicit approximation of the navier-stokes equations. *Lecture Notes in Computational Science and Engineering*, 40:433–440.

[Fisher and Gürol, 2017] Fisher, M. and Gürol, S. (2017). Parallelization in the time dimension of four-dimensional variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 143(703):1136–1147.

[Fisher et al., 2012] Fisher, M., Tremolet, Y., Auvinen, H., Tan, D., and Poli, P. (2012). *Weak-constraint and long-window 4D-Var*. ECMWF Reading, UK.

[Freund et al., 1992] Freund, R. W., Golub, G. H., and Nachtigal, N. M. (1992). Iterative solution of linear systems. *Acta numerica*, 1:57–100.

[Gander and Petcu, 2008] Gander, M. and Petcu, M. (2008). Analysis of a krylov subspace enhanced parareal algorithm for linear problems. In *ESAIM: Proceedings*, volume 25, pages 114–129. EDP Sciences.

[Gander, 2015] Gander, M. J. (2015). 50 years of time parallel time integration. In *Multiple shooting and time domain decomposition methods*, pages 69–113. Springer.

[Gander and Hairer, 2008] Gander, M. J. and Hairer, E. (2008). Nonlinear convergence analysis for the parareal algorithm. In *Domain decomposition methods in science and engineering XVII*, pages 45–56. Springer.

[Gander et al., 2020] Gander, M. J., Kwok, F., and Salomon, J. (2020). Paraopt: A parareal algorithm for optimality systems. *SIAM Journal on Scientific Computing*, 42(5):A2773–A2802.

[Gander et al., 2022] Gander, M. J., Lunet, T., Ruprecht, D., and Speck, R. (2022). A unified analysis framework for iterative parallel-in-time algorithms. *arXiv preprint arXiv:2203.16069*.

[Gander and Vandewalle, 2007] Gander, M. J. and Vandewalle, S. (2007). Analysis of the parareal time-parallel time-integration method. *SIAM Journal on Scientific Computing*, 29(2):556–578.

[Gear, 1988] Gear, C. W. (1988). Parallel methods for ordinary differential equations. *Calcolo*, 25(1):1–20.

[Ghil and Malanotte-Rizzoli, 1991] Ghil, M. and Malanotte-Rizzoli, P. (1991). Data assimilation in meteorology and oceanography. In *Advances in geophysics*, volume 33, pages 141–266. Elsevier.

[Giering and Kaminski, 1998] Giering, R. and Kaminski, T. (1998). Recipes for adjoint code construction. *ACM Transactions on Mathematical Software (TOMS)*, 24(4):437–474.

[Gill, 1982] Gill, A. E. (1982). *Atmosphere-ocean dynamics*, volume 30. Academic press.

[Giraud et al., 2005] Giraud, L., Langou, J., and Rozloznik, M. (2005). The loss of orthogonality in the gram-schmidt orthogonalization process. *Computers & Mathematics with Applications*, 50(7):1069–1075.

[Golub and Van Loan, 2013] Golub, G. H. and Van Loan, C. F. (2013). *Matrix computations*. JHU press.

[Golub and Ye, 1999] Golub, G. H. and Ye, Q. (1999). Inexact preconditioned conjugate gradient method with inner-outer iteration. *SIAM Journal on Scientific Computing*, 21(4):1305–1320.

[Gratton et al., 2007] Gratton, S., Lawless, A. S., and Nichols, N. K. (2007). Approximate gauss–newton methods for nonlinear least squares problems. *SIAM Journal on Optimization*, 18(1):106–132.

[Gratton et al., 2021] Gratton, S., Simon, E., Titley-Peloquin, D., and Toint, P. L. (2021). Minimizing convex quadratics with variable precision conjugate gradients. *Numerical Linear Algebra with Applications*, 28(1):e2337.

[Greenbaum, 1997] Greenbaum, A. (1997). *Iterative methods for solving linear systems*. SIAM.

[Griffies, 2018] Griffies, S. (2018). *Fundamentals of ocean climate models*. Princeton university press.

[Griffith and Nichols, 2000] Griffith, A. K. and Nichols, N. K. (2000). Adjoint methods in data assimilation for estimating model error. *Flow, turbulence and combustion*, 65:469–488.

[Harris, 2018] Harris, C. (2018). Coupled atmosphere-ocean modelling. *New Frontiers in Operational Oceanography*, pages 445–464.

[He, 2010] He, L. (2010). The reduced basis technique as a coarse solver for parareal in time simulations. *Journal of Computational Mathematics*, pages 676–692.

[Hestenes et al., 1952] Hestenes, M. R., Stiefel, E., et al. (1952). Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409–436.

[Holton, 1973] Holton, J. R. (1973). An introduction to dynamic meteorology. *American Journal of Physics*, 41(5):752–754.

[Ide et al., 1997] Ide, K., Courtier, P., Ghil, M., and Lorenc, A. C. (1997). Unified notation for data assimilation: Operational, sequential and variational (gtspecial issueltdata assimilation in meteology and oceanography: Theory and practice). *Journal of the Meteorological Society of Japan. Ser. II*, 75(1B):181–189.

[Ipsen and Meyer, 1998] Ipsen, I. C. and Meyer, C. D. (1998). The idea behind krylov methods. *The American mathematical monthly*, 105(10):889–899.

[Isaksen, 2012] Isaksen, L. (2012). Data assimilation on future computer architectures. In *Proc. ECMWF Seminar on Data Assimilation for Atmosphere and Ocean*, pages 301–322.

[Janjić et al., 2018] Janjić, T., Bormann, N., Bocquet, M., Carton, J., Cohn, S., Dance, S. L., Losa, S., Nichols, N. K., Potthast, R., Waller, J. A., et al. (2018). On the representation error in data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 144(713):1257–1278.

[Kalnay, 2003] Kalnay, E. (2003). *Atmospheric modeling, data assimilation and predictability*. Cambridge university press.

[Keller, 2018] Keller, H. B. (2018). *Numerical methods for two-point boundary-value problems*. Courier Dover Publications.

[Kelley, 1999] Kelley, C. T. (1999). *Iterative methods for optimization.* SIAM.

[Khalaf and Hutchinson, 1992] Khalaf, B. and Hutchinson, D. (1992). Parallel algorithms for initial value problems: parallel shooting. *Parallel computing*, 18(6):661–673.

[Kiehl, 1994] Kiehl, M. (1994). Parallel multiple shooting for the solution of initial value problems. *Parallel computing*, 20(3):275–295.

[Klinker et al., 2000] Klinker, E., Rabier, F., Kelly, G., and Mahfouf, J.-F. (2000). The ecmwf operational implementation of four-dimensional variational assimilation. iii: Experimental results and diagnostics with operational configuration. *Quarterly Journal of the Royal Meteorological Society*, 126(564):1191–1215.

[Lawless et al., 2005] Lawless, A., Gratton, S., and Nichols, N. (2005). An investigation of incremental 4d-var using non-tangent linear models. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(606):459–476.

[Lawless et al., 2003] Lawless, A. S., Nichols, N., and Ballard, S. (2003). A comparison of two methods for developing the linearization of a shallow-water model. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 129(589):1237–1254.

[Le Dimet and Talagrand, 1986] Le Dimet, F.-X. and Talagrand, O. (1986). Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus A: Dynamic Meteorology and Oceanography*, 38(2):97–110.

[Lewis and Derber, 1985] Lewis, J. M. and Derber, J. C. (1985). The use of adjoint equations to solve a variational adjustment problem with advective constraints. *Tellus A*, 37(4):309–322.

[Liesen and Strakos, 2013] Liesen, J. and Strakos, Z. (2013). *Krylov subspace methods: principles and analysis.* Oxford University Press.

[Lions et al., 2001] 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.

[Lorenc et al., 2000] Lorenc, A., Ballard, S., Bell, R., Ingleby, N., Andrews, P., Barker, D., Bray, J., Clayton, A., Dalby, T., Li, D., et al. (2000). The met. office global three-dimensional variational data assimilation scheme. *Quarterly Journal of the Royal Meteorological Society*, 126(570):2991–3012.

[Lorenc, 1986] Lorenc, A. C. (1986). Analysis methods for numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 112(474):1177–1194.

[Lorenc and Rawlins, 2005] Lorenc, A. C. and Rawlins, F. (2005). Why does 4d-var beat 3d-var? *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(613):3247–3257.

[Lorenz, 1963] Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141.

[Maday, 2008] Maday, Y. (2008). The parareal in time algorithm.

[Maday and Mula, 2020] Maday, Y. and Mula, O. (2020). An adaptive parareal algorithm. *Journal of computational and applied mathematics*, 377:112915.

[Maday et al., 2013] Maday, Y., Riahi, M.-K., and Salomon, J. (2013). Parareal in time intermediate targets methods for optimal control problems. *Control and optimization with PDE constraints*, pages 79–92.

[Maday et al., 2007] Maday, Y., Salomon, J., and Turinici, G. (2007). Monotonic parareal control for quantum systems. *SIAM Journal on Numerical Analysis*, 45(6):2468–2482.

[Maday and Turinici, 2002] Maday, Y. and Turinici, G. (2002). A parareal in time procedure for the control of partial differential equations. *Comptes Rendus Mathematique*, 335(4):387–392.

[Maday and Turinici, 2003] Maday, Y. and Turinici, G. (2003). Parallel in time algorithms for quantum control: Parareal time discretization scheme. *International journal of quantum chemistry*, 93(3):223–228.

[Madec et al., 2017] Madec, G., Bourdallé-Badie, R., Bouttier, P.-A., Bricaud, C., Bruciaferri, D., Calvert, D., Chanut, J., Clementi, E., Coward, A., Delrosso, D., et al. (2017). Nemo ocean engine.

[Magoulès and Gbikpi-Benissan, 2018] Magoulès, F. and Gbikpi-Benissan, G. (2018). Asynchronous parareal time discretization for partial differential equations. *SIAM Journal on Scientific Computing*, 40(6):C704–C725.

[Magoulès et al., 2018] Magoulès, F., Gbikpi-Benissan, G., and Zou, Q. (2018). Asynchronous iterations of parareal algorithm for option pricing models. *Mathematics*, 6(4):45.

[Mahfouf and Rabier, 2000] Mahfouf, J.-F. and Rabier, F. (2000). The ecmwf operational implementation of four-dimensional variational assimilation. ii: Experimental results with improved physics. *Quarterly Journal of the Royal Meteorological Society*, 126(564):1171–1190.

[Marshall and Plumb, 1989] Marshall, J. and Plumb, R. A. (1989). *Atmosphere, ocean and climate dynamics: an introductory text*. Academic Press.

[Mathew et al., 2010] Mathew, T. P., Sarkis, M., and Schaerer, C. E. (2010). Analysis of block parareal preconditioners for parabolic optimal control problems. *SIAM Journal on Scientific Computing*, 32(3):1180–1200.

[Matricardi et al., 2004] Matricardi, M., Chevallier, F., Kelly, G., and Thépaut, J.-N. (2004). An improved general fast radiative transfer model for the assimilation of radiance observations. *Quarterly Journal of the Royal Meteorological Society*, 130(596):153–173.

[Minion, 2011] Minion, M. (2011). A hybrid parareal spectral deferred corrections method. *Communications in Applied Mathematics and Computational Science*, 5(2):265–301.

[Miranker and Liniger, 1967] Miranker, W. L. and Liniger, W. (1967). Parallel methods for the numerical integration of ordinary differential equations. *Mathematics of Computation*, 21(99):303–320.

[Navon, 2009] Navon, I. M. (2009). Data assimilation for numerical weather prediction: a review. *Data assimilation for atmospheric, oceanic and hydrologic applications*, pages 21–65.

[Nielsen, 2012] Nielsen, A. S. (2012). Feasibility study of the parareal algorithm. *Technical University of Denmark*.

[Nievergelt, 1964] Nievergelt, J. (1964). Parallel methods for integrating ordinary differential equations. *Communications of the ACM*, 7(12):731–733.

[Nocedal and Wright, 1999] Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer.

[Ong and Schroder, 2020] Ong, B. W. and Schroder, J. B. (2020). Applications of time parallelization. *Computing and Visualization in Science*, 23:1–15.

[Palmer, 1993] Palmer, T. N. (1993). Extended-range atmospheric prediction and the lorenz model. *Bulletin of the American Meteorological Society*, 74(1):49–66.

[Palmer and Anderson, 1994] Palmer, T. N. and Anderson, D. L. T. (1994). The prospects for seasonal forecasting—a review paper. *Quarterly Journal of the Royal Meteorological Society*, 120(518):755–793.

[Persson, 1998] Persson, A. (1998). How do we understand the coriolis force? *Bulletin of the American Meteorological Society*, 79(7):1373–1386.

[Rabier, 2005] Rabier, F. (2005). Overview of global data assimilation developments in numerical weather-prediction centres. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(613):3215–3233.

[Rabier et al., 2000] Rabier, F., Järvinen, H., Klinker, E., Mahfouf, J.-F., and Simmons, A. (2000). The ecmwf operational implementation of four-dimensional variational assimilation. i: Experimental results with simplified physics. *Quarterly Journal of the Royal Meteorological Society*, 126(564):1143–1170.

[Rabier et al., 1998] Rabier, F., McNally, A., Andersson, E., Courtier, P., Undén, P., Eyre, J., Hollingsworth, A., and Bouttier, F. (1998). The ecmwf implementation of three-dimensional variational assimilation (3d-var). ii: Structure functions. *Quarterly Journal of the Royal Meteorological Society*, 124(550):1809–1829.

[Randall, 1994] Randall, D. A. (1994). Geostrophic adjustment and the finite-difference shallow-water equations. *Monthly Weather Review*, 122(6):1371–1377.

[Rawlins et al., 2007] Rawlins, F., Ballard, S., Bovis, K., Clayton, A., Li, D., Inverarity, G., Lorenc, A., and Payne, T. (2007). The met office global four-dimensional variational data assimilation scheme. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 133(623):347–362.

[Reynolds, 1895] Reynolds, O. (1895). Iv. on the dynamical theory of incompressible viscous fluids and the determination of the criterion. *Philosophical transactions of the royal society of london.(a.)*, (186):123–164.

[Reynolds-Barredo et al., 2012] Reynolds-Barredo, J. M., Newman, D. E., Sánchez, R., Samaddar, D., Berry, L. A., and Elwasif, W. R. (2012). Mechanisms for the convergence of time-parallelized, parareal turbulent plasma simulations. *Journal of Computational Physics*, 231(23):7851–7867.

[Ruprecht, 2014] Ruprecht, D. (2014). Convergence of parareal with spatial coarsening. *PAMM*, 14(1):1031–1034.

[Ruprecht, 2018] Ruprecht, D. (2018). Wave propagation characteristics of parareal. *Computing and Visualization in Science*, 19(1):1–17.

[Ruprecht and Krause, 2012] Ruprecht, D. and Krause, R. (2012). Explicit parallel-in-time integration of a linear acoustic-advection system. *Computers & Fluids*, 59:72–83.

[Saad, 2003] Saad, Y. (2003). *Iterative methods for sparse linear systems*. SIAM.

[Saad and Schultz, 1986] Saad, Y. and Schultz, M. H. (1986). Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869.

[Samaddar et al., 2010] Samaddar, D., Newman, D. E., and Sánchez, R. (2010). Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm. *Journal of Computational Physics*, 229(18):6558–6573.

[Sasaki, 1970] Sasaki, Y. (1970). Some basic formalisms in numerical variational analysis. *Monthly Weather Review*, 98(12):875–883.

[Saunders et al., 1999] Saunders, R., Matricardi, M., and Brunel, P. (1999). An improved fast radiative transfer model for assimilation of satellite radiance observations. *Quarterly Journal of the Royal Meteorological Society*, 125(556):1407–1425.

[Schlatter, 2000] Schlatter, T. W. (2000). Variational assimilation of meteorological observations in the lower atmosphere: A tutorial on how it works. *Journal of Atmospheric and Solar-Terrestrial Physics*, 62(12):1057–1070.

[Shewchuk, 1994] Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain.

[Simoncini and Szyld, 2003] Simoncini, V. and Szyld, D. B. (2003). Theory of inexact krylov subspace methods and applications to scientific computing. *SIAM Journal on Scientific Computing*, 25(2):454–477.

[Simoncini and Szyld, 2007] Simoncini, V. and Szyld, D. B. (2007). Recent computational developments in krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59.

[Staff and Rønquist, 2005] Staff, G. A. and Rønquist, E. M. (2005). Stability of the parareal algorithm. In *Domain decomposition methods in science and engineering*, pages 449–456. Springer.

[Steiner et al., 2015] Steiner, J., Ruprecht, D., Speck, R., and Krause, R. (2015). Convergence of parareal for the navier-stokes equations depending on the reynolds number. In *Numerical Mathematics and Advanced Applications-ENUMATH 2013: Proceedings of ENUMATH 2013, the 10th European Conference on Numerical Mathematics and Advanced Applications, Lausanne, August 2013*, pages 195–202. Springer.

[Talagrand, 1997] Talagrand, O. (1997). Assimilation of observations, an introduction (gtspecial issueltdata assimilation in meteology and oceanography: Theory and practice). *Journal of the Meteorological Society of Japan. Ser. II*, 75(1B):191–209.

[Talagrand and Courtier, 1987] Talagrand, O. and Courtier, P. (1987). Variational assimilation of meteorological observations with the adjoint vorticity equation. i: Theory. *Quarterly Journal of the Royal Meteorological Society*, 113(478):1311–1328.

[Thépaut, 2003] Thépaut, J.-N. (2003). Satellite data assimilation in numerical weather prediction: An overview. In *Proceedings of ECMWF Seminar on Recent Developments in Data Assimilation for Atmosphere and Ocean, ECMWF, Reading, UK*, pages 8–12.

[Thepaut and Courtier, 1991] Thepaut, J.-N. and Courtier, P. (1991). Four-dimensional variational data assimilation using the adjoint of a multilevel primitive-equation model. *Quarterly Journal of the Royal Meteorological Society*, 117(502):1225–1254.

[Thépaut et al., 1993] Thépaut, J.-N., Hoffman, R. N., and Courtier, P. (1993). Interactions of dynamics and observations in a four-dimensional variational assimilation. *Monthly Weather Review*, 121(12):3393–3414.

[Toselli and Widlund, 2004] Toselli, A. and Widlund, O. (2004). *Domain decomposition methods-algorithms and theory*, volume 34. Springer Science & Business Media.

[Trémolet, 2004] Trémolet, Y. (2004). Diagnostics of linear and incremental approximations in 4d-var. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 130(601):2233–2251.

[Trémolet, 2006] Trémolet, Y. (2006). Accounting for an imperfect model in 4d-var. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 132(621):2483–2504.

[Trémolet, 2007] Trémolet, Y. (2007). Incremental 4d-var convergence study. *Tellus A: Dynamic Meteorology and Oceanography*, 59(5):706–718.

[Trindade and Pereira, 2004] Trindade, J. and Pereira, J. (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.

[Vallis, 2017] Vallis, G. K. (2017). *Atmospheric and oceanic fluid dynamics*. Cambridge University Press.

[Van Den Eshof and Sleijpen, 2004] Van Den Eshof, J. and Sleijpen, G. L. (2004). Inexact krylov subspace methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 26(1):125–153.

[Van der Vorst, 2003] Van der Vorst, H. A. (2003). *Iterative Krylov methods for large linear systems*. Number 13. Cambridge University Press.

[Wilcox et al., 1998] Wilcox, D. C. et al. (1998). *Turbulence modeling for CFD*, volume 2. DCW industries La Canada, CA.

[Wu and Zhou, 2017] Wu, S.-L. and Zhou, T. (2017). Fast parareal iterations for fractional diffusion equations. *Journal of Computational Physics*, 329:210–226.

[Wu and Zhou, 2018] Wu, S.-L. and Zhou, T. (2018). Parareal algorithms with local time-integrators for time fractional differential equations. *Journal of Computational Physics*, 358:135–149.

[Xu et al., 2015] Xu, Q., Hesthaven, J. S., and Chen, F. (2015). A parareal method for time-fractional differential equations. *Journal of Computational Physics*, 293:173–183.