UNIVERSITY OF NAPLES
"FEDERICO II"

Department of Mathematics and Applications:

Renato Caccioppoli

Ph.D. Program in "Matematica e Applicazioni"

Cycle XXXIV

DOMAIN DECOMPOSITION APPROACHES FOR MULTISCALE/MULTIPHYSICS DATA ASSIMILATION

Supervisor:
Prof.ssa Luisa D'Amore

Ph.D. Candidate:
Rosalba Cacciapuoti

Contents

Notations

DA	Data Assimilation
3DVAR	3D Variational DA
4DVAR	4D Variational DA
KF	Kalman Filter
EnKF	ensemble Kalman filter
ROMS	Regional Ocean Modeling System
PinT	Parallel in time
ASM	Additive Schwarz Method
Ω	spatial domain
Δ	time interval
N_p	number of nodes in Ω
N	number of instants of time in Δ
$\mathcal{M}_{l-1,l}$	mathematical model from t_{l-1} to t_l
$\mathcal{M}$	mathematical model from t_0 to t_N
$M_{l,l-1}$	discretization of a linear approximation of M_{t_{l-1},t_l}
M	discretization of a linear approximation of M
$\mathcal{H}$	non-linear observations mapping
H_l	linear approximation of $\mathcal{H}$
$\mathbf{R}$	covariance matrices of the errors on observations

$\mathbf{B}$	covariance matrices of the errors on background
R_l	covariance matrices of the errors on observations at time t_l
B_l	covariance matrices of the errors on background at time t_l
$u^{\mathcal{M}}$	state function of $\mathcal{M}$
u^M	solution of discrete model
x_l	KF solution at time t_l
$\mathbf{u^{DA}}$	solution of 3DVAR/4DVAR problem
y_l	observations at time t_l
$\mathbf{J}$	3DVAR/4DVAR functional

Chapter 1

Introduction

1.1 Data assimilation (DA): a large scale inverse ill posed problem

Predictive Science is the paradigm shift of the emerging CSE (Computational Science and Engineering) tightly integrating the numerical simulations of Computational Science and Engineering with Validation and Verification. The aim of Predictive Science is to not only to reproduce with high-fidelity an observed phenomenon, but also to predict the reality in situations for which the numerical simulation has not been specifically validated nor tested. To this end, reliable numerical predictions require complex non linear physical models as well as a systematic and comprehensive treatment of calibration and validation procedure, including the quantification of inherent uncertainties (Uncertainty Quantifcation (UQ) and Sensitivity Analysis (SA)). Data assimilation (DA) has long been playing a crucial role in the quantification of uncertainties in numerical weather prediction (NWP) and oceanography [57]-[102] and more in general, in data science; recently, DA started to be applied more widely to numerical simulations beyond geophysical applications [19], medicine and biological science [51] for improving the accuracy

and reliability of computational approaches. DA encompasses the entire sequence of operations that, starting from observations/measurements of physical quantities and from additional information - such as a mathematical model governing the evolution of these quantities - improve their estimate minimizing a suitable functional. In order to understand how such functional comes out, we start saying that from the mathematical viewpoint DA is an inverse and ill posed problem. Hence, regularization methods are used to introduce prior knowledge. Usually, the objective function measures the mismatch between the model predictions and the observed system states, weighted by the inverse of the error covariance matrices [25, 106, 128]. In this way, DA provides mathematical methods for finding an optimal trade–off between the current estimate of the model's state and the observations, knowing that both carry their own uncertainties. Due to the scale of the forecasting area and the number of state variables, DA are very large scale problems [92]. In operational DA the amount of observations is insufficient to fully describe the system and one cannot strongly rely on a data driven approach: the model is paramount. It is the model that fills the spatial and temporal gaps in the observational network: it propagates information from observed to unobserved areas. Thus, DA methods are designed to achieve the best possible use of a never sufficient (albeit constantly growing) amount of data, and to attain an efficient data model fusion, in a short period of time. This poses a formidable computational challenge, and makes DA an example of big data inverse problems [9, 7, 10, 33].

1.2 DA methods coupled with PDE-based predictive models: related works and contribution of the present work

There is a lot of DA algorithms. Two main approaches gained acceptance as powerful methods: variational approaches (namely 3DVAR and 4DVAR) and Kalman Filter (KF) [53, 71, 116, 77]. Variational approaches are based on the minimization of the objective function estimating the discrepancy between numerical results and observations. These approaches assume that the two

sources of information, forecast and observations, have errors that are adequately described by stationary error covariances. In contrast to variational methods, KF (and its variants) is a recursive filter solving the Euler-Lagrange equations. It uses a dynamic error covariance estimate evolving over a given time interval. The process is sequential, meaning that observations are assimilated in chronological order, and KF alternates a forecast step, when the covariance is evolved, with an analysis step in which covariance of the filtering conditional is updated. In both kind of methods the model is integrated forward in time and the result is used to reinitialize the model before the integration continues [72]. DA is dealing with the joint assimilation of observational data from different spatial scales and different data type, namely multiscale and multiphysics data. Multiscale and multiphysics DA [94, 61] refers to the assimilation of data obtained at a different resolution than the model resolution. Most of DA methods scale data and the simulation model, but with loss of important information. Multiscale and multiphysics DA leads itself to Domain Decomposition (DD) approaches such that data can be processed for distinct spatial scales. Most approaches for delivering parallel solutions of simulations based on DA methods integrated with Partial Differential Equations (PDEs)–based models essentially takes full advantage of existing parallel DD solvers where the solver is suitably modified to also handle the adjoint system [5, 87, 103, 122]. Further, these approaches follow the path *optimize–then–discretize* to build a discrete Lagrangian. At the heart of these schemes lies the solution of a linear system (the Karush–Kuhn–Tucker (KKT) system) which is finally solved using Schwarz preconditioners [15]. In contrast, when DA problems are posed as PDEs–constrained nonlinear optimization problem, their numerical solution uses nonlinear solvers such as Newton–Krylov methods or one its variants (such as Gauss-Newton, L-BFGS, Levenberg-Marquardt). Large scale nonlinear solvers basically rely on linear algebra solvers, such as Krylov–based iterative methods, and direct solvers, mainly based on QR and SVD factorizations [8, 30, 45, 41, 117]. These approaches follow the path *discretize–then–optimize* approach. A different approach is the combination of DD–methods in space and DA, where spatial domain–decomposed uncer-

tainty quantification approach performs DA at the local level by using Monte Carlo sampling [5, 1, 82]. The Parallel DA Framework [105] implements parallel ensemble–based Kalman Filters algorithms coupled within the PDE–model solver. A common drawback of such parallel algorithms is their limited scalability, due to the fact that the most computationally demanding components are adapted for parallel execution. Amdhal's law [2] clearly applied in these situations because the computational cost of the component that are not parallelized provides a lower bound on the execution time of the parallel algorithm[1]. On the contrary, the challenge is to consider parallelization from the beginning of the computational problem solving.

Most PDEs–based simulations contains time–stepping both in the mathematical modelling and in its numeric approximation. Several approaches have been proposed to reduce the overall time-to-solution. Among them, there are KF simplifications reducing computational complexity. Approximations are designed on the basis of a reduction in the order of the system model (usually the approximation is performed trough the use of the Empirical Orthogonal Functions (EOF)) [60, 111], or they are based on the Ensemble methods, where a prediction of the error at a future time is computed by integrating each ensemble state independently by the model. Integration is typically performed until observations are available. At this time, the information from the observations and the ensemble are combined by performing an analysis step based on KF [53]. However, the choice of the dimension of the reduced-state space or of the ensemble size giving an accurate approximation of KF still remains a delicate question [8]. Besides these variants, there are parallel approaches to KF algorithm. Traditional algorithms based on parallelized linear algebra implementations make little or no use of parallelism in the time domain: time-stepping is currently treated as a serial process. Approaches able to develop effective

[1] Speedup is defined as sequential execution time over parallel execution time in parallel processing. Let f be the portion of the workload that can be parallelized; when the number of processors increases to infinity, the speedup upper bound is $1/(1-f)$, where $1-f$ clearly represents the part of the algorithm that cannot be parallelized [2].

scalable algorithms, taking a step–change beyond traditional high perfromance computing approaches, are strongly recommended. A revolutionary approach for solving PDEs-based model is the Parallel in Time (PinT) strategy [11, 14, 84, 55, 85]. By introducing parallelization from the beginning of the computational problem solving, PinT approach overcomes the inherent bottleneck of time–marching solvers (such as those of traditional algorithms based on parallelized linear algebra). In order to introduce a consistent DD along the time direction, PinT methods share this general idea: they use a coarse/global/predictor propagator to obtain approximate initial values of local models on the coarse time-grid; a fine/local/corrector solver to obtain the solution of local models starting from the approximate initial values; an iterative procedure to smooth out the discontinuities of the global model on the overlapping domains. Nevertheless, one of the key limitation of any PinT-based methods is data dependencies of the local solvers from the coarse solver: the coarse solver must always be executed serially for the full duration of the simulation and local solver have to wait for the approximate initial values provided by the coarse solver. The strength of the DD approach we are going to present in this thesis is the exploitation of the coupling between the DA functional and the PDE model. The idea goes back to the work of Schwarz [114] on overlapping domains, nevertheless in contrast to Schwarz methods which uses as boundary conditions the approximation of the numerical solution computed on the interfaces between adjacent subdomains, here the proposed approach uses the DA model as a predictor for the local PDE-based model, providing the approximations needed for locally solving the initial value problems on each subinterval, concurrently.

The primary motivation of Schwarz based DD methods was the inherent parallelism arising from a flexible, adaptive and independent decomposition of the given problem into several subproblems, though they can also reduce the complexity of sequential solvers. DD–DA framework allows to employ a model reduction in space and time which is coherent with the filter localization. There is a quite different rationale behind such DD framework and the so called MOR methods, even though they are closely related each other. The primary motivation of

DD methods based on Schwarz DD methods was the inherent parallelism arising from a flexible, adaptive and independent decomposition of the given problem into several sub problems, though they can also reduce the complexity of sequential solvers. MOR techniques are based on projection of the full order model onto a lower dimensional space spanned by a reduced order basis. These methods has been used extensively in a variety of fields for efficient simulations of highly intensive computational problems. But all numerical issues concerning the quality of approximation still are of paramount importance [64]. As previously mentioned, DD–DA framework makes it natural to switch from a full scale solver to a model order reduction solver for solution of subproblems for which no relevant low-dimensional reduced space should be constructed. In the same way, DD–DA framework allows to employ a model reduction in space and time which is coherent with the filter localization. Main advantage of the DD–DA is to combine in one theoretical framework, model reduction, along the space and time directions, and filter localization, while providing a flexible, adaptive, reliable and robust decomposition.
Summarizing, we partition initial domain along space and time, then we extend each subdomain to overlap its neighbors by an amount; partitioning can be adapted according to the availability of measurements and data. Accordingly, we reduce dynamic model both in space and time. As initial and boundary values of local models, according to PinT approach, we employ estimates provided by 4DVAR and KF itself, as soon as these are available. In this way, concurrency of local models is achieved. On each subdomain we formulate a local 4DVAR and KF problem analogous to the original one, defined on local models. In order to enforce the matching of local solutions on overlapping regions, local 4DVAR and KF problems are slightly modified by adding a correction term; such a correction term, acting as a regularization constraint on local solutions, keeps track of contributions of adjacent domains to overlapping regions; localization excludes remote observations from each analyzed location, thereby improving the conditioning of the error covariance matrices. To the best of our knowledge, such a full decomposition of 4DVAR and KF in space and time has never been investigated before.

Any interest reader who wants to apply DD framework in a real-world application, i.e. with a (PDE-based) model state and an observation mapping, once the dynamic (PDE-based) model state has been discretized, he should rewrite the state estimation problem under consideration as a Constrained Least Square (CLS) model problem (cfr Section A.1) and then to apply DD algorithm. In other words, she/he should follow the discretize-then-optmize approach, common to most DA problems and state estimation problems, before employing the DD framework.
However, a static and or a priori DD strategy could not ensure a well balanced work load, while a way to re-partition the mesh so that the subdomains maintain a nearly equal number of observations plays an essential role in the success of any effective DD approach. There has been widespread interests in load balancing since the introduction of large scale multiprocessors. Applications requiring dynamic load balancing varies from the parallel solution of a PDE by finite elements on an unstructured grids [43] to parallelized particle simulations [80]. Load balancing is one of the central problems which have to be solved in parallel computing. Moreover, problems whose load changes during the computation or it depends on data layout which may be unknown a priori, will necessitate the redistribution of the data in order to retain efficiency. Such a strategy is known as dynamic load balancing. Algorithms for dynamic load balancing, as in [27, 20, 126, 125], are based on transferring an amount of work among processors to neighbours; the process is iterated until the load difference between any two processors is smaller than a specified value, consequently it will not provide a balanced solution immediately. One of disadvantages of this approach is its possible slow convergence. A multilevel diffusion method for dynamic load balancing, as in [65], is based on bisection of processor graph. The disadvantage is that can occur movement of data between non-neighbouring processors to ensure the connectivity of subgraphs and must be avoided. The mentioned algorithms do not take into account one important factor, namely that the data movement resulting from the load balancing schedule should be kept to minimum. We apply a dynamic load balancing to maintain a nearly equal number of observations between subdomains. Main feature of the this approach is the

Migration step, i.e. the shifting of the boundaries of adjacent subdomains in order to achieve a balanced load. The disadvantage is that the final balance between subdomains depends strongly on the degree of the vertices of processors graph, i.e. number of neighbouring subdomains for each subdomain.

In conclusion, in this thesis, we present a DD-based parallel framework for solution of large scale variational 3D, 4D DA and KF problems, involving decomposition of the physical domain, partitioning of the solution and modification of the regularization functional describing the variational DA problem. We address DA problems where the observations are non uniformly distributed, general sparse and its distribution changes during the time window, consequently, we present Dynamic Domain Decomposition (DyDD) and Dynamic Domain Decomposition in Space and Time (DyDDST) methods to employ a load balancing scheme involving an adaptive and dynamic workload redistribution both along Space and Time directions for solving Data Assimilation problems.
Main topics of DD–DA coupled with DyDD or DyDDST can be listed as follows.

1. DD step: we begin by partitioning along space and time the domain into subdomains and then extending each subdomain to overlap its neighbors by an amount. Partitioning can be adapted according to the availability of measurements and data.

2. DyDD or DyDDST: if workload is not well balanced, it involves an adaptive and dynamic repartitioning of load among spatial subdomains (DyDD) at each time interval (DyDDST).

3. Filter Localization and Model Order Reduction (MOR): on each subdomain we formulate a local DA problem analogous to the original one, combining filter localization and model order reduction approaches.

4. Regularization constraints: in order to enforce the matching of local solutions on the

overlapping regions, local DA problems are slightly modified by adding a correction term. Such a correction term balances localization errors and computational efforts, acting as a regularization constraint on local solutions. This is a typical approach for solving ill posed inverse problems (see for instance [40]).

5. Parallel in Time: as the dynamic model is coupled with DA operator, at each integration step we employ, as initial and boundary values of all reduced models, estimates provided by the DA model itself, as soon as these are available.

6. Conditioning: localization excludes remote observations from each analyzed location, thereby improving the conditioning of the error covariance matrices. We give a new definition of conditioning of DD-DA problem depending on reduced models and local covariance matrices. To the best of our knowledge, such ab-initio space and time decomposition of DA models has never been investigated before.

In the following we briefly resume our contribution concerning DD–DA [32, 33, 48, 35], DyDD[34] and DyDDST [47].

- DD–DA methods:
  - DD–3DVAR [32]: We prove that the functional decomposition of the 3DVAR DA operator, previously introduced in [9, 7, 8, 32, 30, 46, 45], is equivalent to apply Additive Schwarz Method (ASM) to the Euler–Lagrange equations resulting from the variational functional.
  - DD–4DVAR [33]: We present a space–and–time decomposition approach consisting of a decomposition of the whole domain, i.e. both along the spacial and temporal direction, and of a reduction of the whole operator, i.e. both the DA variational functional and the PDE–based model. We begin by partitioning the domain into subdomains and then extending each subdomain to overlap its neighbors by

an amount. On each subdomain we formulate a discrete 4DVAR problem analogous to the original decomposing both the variational functional and the model both in space and in time. In addition, in order to enforce the matching of local solutions on the overlapping regions, local problems are slightly modified by using a local correction. To the best of our knowledge, space and time decomposition of 4DVAR has never been investigated before. We provide a mathematical formulation of this approach and a feasibility analysis on Shallow Water Equations (SWEs) including convergence analysis and error propagation. We will describe the main components of DD–4DVAR method to Regional Ocean Modeling System (ROMS) (https://www.myroms.org) software, by highlighting the topics that we will address both on the mathematical problem underlying ROMS and the code implementation.

– DD–KF [48, 35]: DD–KF involves decomposition of the whole computational problem, partitioning of the solution and a slight modification of KF algorithm allowing a correction at run-time of local solutions. The resulted parallel algorithm consists of concurrent copies of KF algorithm, each one requiring the same amount of computations on each subdomain and an exchange of boundary conditions between adjacent subdomains. It partitions the whole problem, including filter equations and dynamic model along space and time directions. As a consequence, instead of solving one larger KF problem we solve local problems reproducing the original one at smaller dimensions. Also, sub problems could be solved in parallel. We derive and discuss DD–KF algorithm for solving CLS models, which underlie any data sampling and estimation problems arising from the so-called discretize-then-optimize approach, and DA problems. We start considering CLS model, seen as a prototype of DA model [56]. CLS is obtained combining two overdetermined linear systems, representing the state and the observation mapping, respectively. In this regards, in [35] we presented a feasibility analysis on CLS models of an innovative DD framework

for using CLS in large scale applications.

- DyDD [34] and DyDDST [47] to DD–DA: in many problems within the earth and environmental sciences observations are non uniformly distributed and its distribution change during time. DyDD and DyDDST algorithms are proposed to support real time application where load measurement is necessary to determine when load imbalance occurs. Firstly, we introduce DyDD method based on adaptive and dynamic redefining of the boundaries of initial DD in space, then, we introduce DyDDST by extending DyDD to DD in space and time. We apply DyDD and DyDDST on algorithm proposed in [67] in order to ensure a balanced distribution of load among processes.

  Main advantage of the DD–DA coupled with DyDD or DyDDST is to combine in the same theoretical framework model reduction, along the space and time directions, with filter localization, while providing a flexible, adaptive and dynamic decomposition. We present a revision of DD framework such that a DyDD and DyDDST algorithms allow for a minimal data movement restricted to the neighboring processors.
  This is achieved by considering a connected graph induced by the domain partitioning whose vertices represent a subdomain associated with a scalar representing the number of observations on that subdomain. Main step of DyDD and DyDDST framework can be listed as follows.

  – Scheduling step: DyDD and DyDDST compute the amount of observations needed for achieving the average load in each physical subdomain; this is performed by introducing a diffusion type algorithm derived by minimizing the Euclidean norm of the cost transfer.
  – Migration step: DyDD and DyDDST shift the boundaries of adjacent subdomains to achieve a balanced load.

  The most intensive kernel is the scheduling step which defines a schedule for comput-

ing the load imbalance (which we quantify in terms of number of observations) among neighbouring subdomains. Such quantity is then used to update the shifting the adjacent boundaries of subdomains (Migration step) which are finally re mapped to achieve a balanced decomposition. We are assuming that load balancing is restricted to the neighbouring domains so that we reduce the overhead processing time. Finally, following [67] we use a diffusion type scheduling algorithm minimizing the euclidean norm of data movement. The resulting constrained optimization problem leads to normal equations whose matrix is associated to the decomposition graph. The disadvantage is that the overhead time, due to the final balance between subdomains, strongly depends on the degree of the vertices of processors graph, i.e. on the number of neighbouring subdomains for each subdomain. Such overhead represents the surface-to-volume ratio whose impact on the overall performance of the parallel algorithm decreases as the problem size increases.

1.3 Outline of the work

The thesis is organized as follows.

- In Chapter 2 we firstly introduce DA problem. Then, we define 3DVAR and 4DVAR DA problem and introduce KF algorithm.

- In Chapter 3 we introduce the space and time decomposition approach. Firstly, we describe DD of physical domain $\Omega \times \Delta$, then we introduce DD approach to 3DVAR, 4DVAR DA problem and KF. We call them DD–3DVAR, DD–4DVAR and DD-KF, respectively. We perform sensitivity analysis of DD-DA. In particular we discuss consistency, convergence, stability and round-off error propagation of DD–4DVAR method, giving the definition of condition number of the method. Then, we introduce DD–KF for solving CLS models and DA problems and discuss its performance analysis and reliability assessment.

- In Chapter 4 we introduce dynamic domain decomposition in space and time (DyDD and DyDDST), since a static and or a priori DD strategy could not ensure a well balanced work load. DyDD and DyDDST ensure a well balanced workload with DA problems where observations are non uniformly distributed and general spare and its distribution change during time.

- In Chapter 5 we present validation analysis of the method. We prove the consistence, convergence and stability. We carry on performance analysis of DD–KF to CLS and SWEs problems. Then, we address performance and scalability of DD–4DVAR algorithm and address the role of overlapping region in DD–4DVAR algorithm. We prove that the experimental order of consistency of DD–4DVAR corresponds to the theoretical one derived in Chapter 3 and that DD–4DVAR with the initial boundary problem of SWEs one dimensional is well-conditioned according to the definition of condition number given in Chapter 3.

- In Chapter 6 we summarise the work in this thesis. We present the main conclusions and suggestions on further work that could be done in this area.

- In Appendix we introduce CLS, SWEs problem and its discrete formulation. Then, we present ROMS, an open-source, mature numerical framework used by both the scientific and operational communities to study ocean dynamics over 3D spatial domain and time interval. We describe the main components of DD–4DVAR method to ROMS model and code, highlighting the topics that we will address both on the mathematical problem underlying ROMS and the code implementation. Finally, we report MATLAB codes to perform validation analysis in Chapter 5.

Chapter 2

Data Assimilation methods

From the mathematical viewpoint the overall concept underpinning DA is an inverse modeling problem. The forward modelling seeks to predict output observables (such as magnitude and process of micro-seismic events at seismometer locations, or the ground motion at a millimetric precision by satellite images) given the parameters by solving the governing equations. The forward problem is usually well posed (the solution exists, is unique, and is stable to perturbations in inputs). The inverse modeling reverses this relationship, however, by seeking to determine parameter values that are consistent with particular measurements. Solving inverse modeling can be very challenging for the following reasons: (1) the mapping from observations (i.e., measurements) to parameters generally is not one to one, particularly when the number of parameters is large and the number of measurements is small; (2) small changes in the measurement value may lead to changes in many, or all parameters, particularly when the forward model is nonlinear, i.e., the problem is severely ill-conditioned; and (3) typically, the computational model approximately solves the forward problem taking into account only a limited number of physical processes. The popular approach to obtain a unique "solution" to the inverse problem in these circumstances is to formulate it as a variational problem minimizing the sum of two terms: the first is a combination of the residual between observed and predicted outputs (the

so-called misfit) in an appropriate norm, and the second is a regularization term that penalizes unwanted features of the parameters. The inverse problem thus leads to a nonlinear variational problem in which the forward simulation model is embedded in the residual term.

When the forward model takes the form of partial differential equations (PDEs) or some other expensive model, the result is a PDE-based variational problem that may be extremely large scale in the state variables, even when the number of inversion parameters is small. More generally, uncertain parameters can be taken from numbers on a continuum (such as initial or boundary conditions, heterogeneous material parameters, or heterogeneous sources) that, when discretized, result in an inverse problem that is very large scale in the inversion parameters as well. An estimation of parameters using the regularization approach to inverse problems as described above will yield an estimate of the "best" parameter values that minimize the combined misfit and penalty function. We will start from the high-level problem definition in terms of a set of mathematical equations subject to boundary and initial conditions over some space-time domain where the spatial domain is expressed in terms of a mathematical definition. From this point, a reasonable starting mesh for that space-and-time domain must be constructed, and a numerical discretization on the mesh must be solved. The resulting solution would be analyzed to determine if the combination of the mesh and its discretization yielded the desired accuracy.

2.1 The DA inverse problem

If $\Omega \subset \mathbb{R}^n$, $n \in \mathbb{N}$, is a spatial domain with a Lipschitz boundary and $\Delta := [0, T] \subset \mathbb{R}$, $T \in \mathbb{R}$, is time interval, let:

$$\begin{cases} u^{\mathcal{M}}(x, t + \Delta t) = \mathcal{M}_{t,t+\Delta t}[u(x, t)] & \forall x \in \Omega, t, t + \Delta t \in \Delta, (\Delta t > 0) \\ u^{\mathcal{M}}(x, t_0) = u_0(x) & t_0 \equiv 0, \;\; x \in \Omega \\ u^{\mathcal{M}}(x, t) = f(x) & x \in \partial\Omega, \forall t \in \Delta \end{cases}, \tag{2.1}$$

be a symbolic description of the DA model of interest where

$$u^{\mathcal{M}} : (x,t) \in \Omega \times \Delta \mapsto u(x,t) = [u^{\mathcal{M}}[1](x,t), u^{\mathcal{M}}[2](x,t), \ldots, u^{\mathcal{M}}[p_v](x,t)], \tag{2.2}$$

is the state function of $\mathcal{M}$ (i.e. model on $\Omega \times \Delta$) with $p_v \in \mathbb{N}$ the number of physical variables, f is a known function defined on the boundary $\partial\Omega$, and let

$$y : (x,t) \in \Omega \times \Delta \mapsto v(x,t),$$

be the observations function, and

$$\mathcal{H}_{t+h} : u^{\mathcal{M}}(x, t+h) \mapsto y(x, t+h), \quad \forall (x,t) \in \Omega \times \Delta,$$

denote the non-linear observations mapping. To simplify future treatments we assume $p_v \equiv 1$. The first equation in 2.1 is often an approximation of an evolutionary PDE, consequently the operator $\mathcal{M}_{t,t+\Delta t}$ is linear PDE model. While, the observations y are in general not direct measurements of state variable $u^{\mathcal{M}}$, the observation operator $\mathcal{H}_{t+h}$ allows to compare observations vector with the state vector.

In order to describe a numerical method for solving DA problems, we introduce the following discretization of domain $\Omega \times \Delta$.

Definition 1. *(Discretization of domain $\Omega \times \Delta$) Let*

$$\Omega_I \equiv \{x_{\tilde{i}}\}_{\tilde{i} \in I} \subset \Omega \tag{2.3}$$

be discretization of spatial domain Ω where

$$I = \{1, \ldots, N_p\} \text{ and } N_p = |I|^1 \tag{2.4}$$

are respectively the set of indices of nodes in Ω and its cardinality i.e. the number of inner nodes in Ω. Let

$$\Delta_K \equiv \{t_{\tilde{k}}\}_{\tilde{k} \in K} \subset \Delta$$

[1] We refer to $|I|$ as cardinality of set I.

be discretization of time interval Δ *where*

$$K = \{0, 1, \ldots, N-1\} \text{ and } N = |K| \tag{2.5}$$

are respectively the set of indices of time in Δ *and its cardinality i.e. number of instants of time in time interval* Δ*. Consequently, we define*

$$\Omega_I \times \Delta_K \equiv \{(x_{\tilde{i}}, t_{\tilde{k}})\}_{\tilde{i}\in I;\ \tilde{k}\in K} \subset \Omega \times \Delta \tag{2.6}$$

as the discrete domain.

We define the DA inverse problem in discrete form [36, 8].

Definition 2. *(The DA inverse problem in discrete form). DA problem concerns the computation of*

$$\mathbf{u^{DA}} := \{u^{DA}(j,l)\}_{j=1,\ldots,N_p; l=0,1,\ldots,N-1} \in \mathbb{R}^{N_p\times(N)}$$

such that

$$y = G \cdot \mathbf{u^{DA}}, \tag{2.7}$$

subject to the constraint that

$$\mathbf{u^{DA}}(\cdot, 0) = u_0.$$

where

- N_p*: is the number of nodes in* $\Omega \subset \mathbb{R}^n$ *defined in (2.4);*
- n_{obs}*: is the number of observations in* Ω*, where* $n_{obs} << N_p$*;*
- N*: is the number of instants of time in* Δ *defined in (2.5);*
- $u_0 = \{u_{0,j}\}_{j=1,\ldots,N_p} \equiv \{u_0(x_j)\}_{j=1,\ldots,N_p} \in \mathbb{R}^{N_p}$*: is the state at time* t_0*;*
- $y := \{y(z_j, t_l)\}_{j=1,\ldots,n_{obs}; l=0,1,\ldots,N-1} \in \mathbb{R}^{n_{obs}\times N}$*: representing observations in* $\Omega \times \Delta$*;*

- $H_l \in \mathbb{R}^{n_{obs} \times N_p}, \quad l = 0, \ldots, N-1$, *: is a linear approximation of observation mapping* $\mathcal{H}_{t_l}$*;*

- $G := G_{N-1} \in \mathbb{R}^{(N \times n_{obs}) \times N_p}$*:*

$$G_l = \begin{cases} \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_l \end{bmatrix} & l > 0 \\ \\ H_0 & l = 0 \end{cases} . \tag{2.8}$$

In the next section, we introduce the regularized DA inverse problems, i.e. the 3DVAR DA and 4DVAR DA problems.

2.1.1 The 3D and 4DVAR DA problems

In the following we introduce variational approaches i.e. 3DVAR and 4DVAR DA approaches based on the minimization of the objective function estimating the discrepancy between numerical results and observations. These approaches assume that the two sources of information, forecast and observations, have errors that are adequately described by error covariances. We first present the 4DVAR problem.

Definition 3. *(The 4DVAR DA problem). The 4DVAR DA problem concerns the computation of:*

$$\mathbf{u^{DA}} = argmin_{u \in \mathbb{R}^{N_p \times N}} \mathbf{J}(u), \tag{2.9}$$

with

$$\mathbf{J}(u) = \alpha \|u - u^M\|^2_{\boldsymbol{B}^{-1}} + \|Gu - y\|^2_{\boldsymbol{R}^{-1}} \tag{2.10}$$

where

- N_p: *is the number of nodes in* $\Omega \subset \mathbb{R}^n$ *defined in (2.4);*
- n_{obs}: *is the number of observations in* Ω, *where* $n_{obs} << N_p$;
- N: *is the number of instants of time in* Δ *defined in (2.5);*
- $u_0 = \{u_{j,0}\}_{j=1,\dots,N_p} \equiv \{u_0(x_j)\}_{j=1,\dots,N_p} \in \mathbb{R}^{N_p}$: *is the state at time* t_0;
- *the operator*

$$M_{l-1,l} \in \mathbb{R}^{N_p \times N_p}, \quad l = 1, \dots, N, \tag{2.11}$$

representing a discretization of a linear approximation of $\mathcal{M}_{t_{l-1},t_l}$ *from* t_{l-1} *to* t_l,

- *the operator*

$$M \in \mathbb{R}^{N_p \times N_p}, \tag{2.12}$$

representing a discretization of a linear approximation of $\mathcal{M}$ *from* t_0 *to* t_N;

- *the matrix*

$$u^M := \{u^M_{j,l}\}_{j=1,\dots,N_p;l=1,\dots,N} \equiv \{u^M(x_j,t_l)\}_{j=1,\dots,N_p;l=0,1,\dots,N-1} \in \mathbb{R}^{N_p \times N} \tag{2.13}$$

representing the solution of model M *i.e. the background;*

- α: *is the regularization parameter;*
- $y := \{y(z_j,t_l)\}_{j=1,\dots,n_{obs};l=0,1,\dots,N-1} \in \mathbb{R}^{n_{obs} \times N}$: *representing observations in* $\Omega \times \Delta$;
- $G := G_{N-1} \in \mathbb{R}^{(N \times n_{obs}) \times N_p}$: *where* G_{N-1} *is defined in (2.8);*
- $\boldsymbol{R} = diag(R_0, R_1, \dots, R_{N-1}) \in \mathbb{R}^{N \cdot n_{obs} \times N \cdot n_{obs}}$ *and* $\boldsymbol{B} = \boldsymbol{V}\boldsymbol{V}^T \in \mathbb{R}^{N_p \times N_p}$: *are respectively covariance matrices of the errors on observations and background where* $R_{k-1} \in \mathbb{R}^{n_{obs} \times n_{obs}}$, $\forall k = 1, \dots, N$.

The 3DVAR problem is stationary case of 4DVAR problem namely 3DVAR misses the time-dependency.

Definition 4. *(The 3DVAR DA problem) The 3DVAR DA problem concerns the computation of* $\mathbf{u}^{\mathbf{DA}}$ *defined in (2.9) at fixed time* t_l.

The 4DVAR algorithm can be considered as an extension of the 3DVAR to the time dimension by including the dynamical models to evolve in time the state [109]. 4DVAR algorithm consider the time distribution of observations differently from 3DVAR. The cost function is the same, provided that the observation operators are generalized to include a forecast model that will allow a comparison between the model state and the observations at the appropriate time.

Remark 1. *It is worth noting that here we are considering a linear approximation of the observation operator, hence a linear operator* G*, although this is not at all required, at least in the formulation of the 3D and 4DVAR problem. A more general approach for numerically linearize and solve 3D and 4DVAR DA problem consists in defining a sequence of local approximations of* $\mathbf{J}$ *where each member of the sequence is minimized by employing Newton's method or one its variants [42]. More precisely, two approaches could be employed:*

(a) by truncating Taylor's series expansion of $\mathbf{J}$ *at the second order, giving a quadratic approximation of* $\mathbf{J}$*, let us say* $\mathbf{J}^{QN}$*. Newton'methods (including LBFGS and Levenberg-Marquardt) use* $\mathbf{J}^{QD}$*. The minimum is computed solving the linear system involving the Hessian matrix* $\nabla^2\mathbf{J}$*, and the negative gradient* $-\nabla\mathbf{J}$.

(b) by truncating Taylor's series expansion of $\mathbf{J}$ *at the first order which gives a linear approximation of* $\mathbf{J}$*, let us say let us say* $\mathbf{J}^{TL}$*. Gauss-Newton's methods (including Truncated or Approximated Gauss-Newton uses* $\mathbf{J}^{TL}$*). The minimum is computed solving the normal equations arising from the local Linear Least Squares problem.*

Both approaches will employ the Tangent Linear Model (TLM) and the adjoint operator of the observation mapping and of the model of interest [18, 129].

2.1.2 Kalman Filter (KF)

In the past years Kalman Filter (KF) [71] has become a main component in satellite navigation, economics, or telecommunications and in the validation of the mathematical models used in meteorology, climatology, geophysics, geology and hydrology. Today, KF is one of the most important and common estimation algorithms. Its main strength is its recursive property: new measurements can be processed as they arrive. Nevertheless, the standard formulation of the KF becomes computationally intractable for solving large scale estimation problems due to matrix storage and inversion requirements. Consequently, several approaches have been proposed to reduce the overall time-to-solution. In this regards, in Sections 3.4 and 3.5, we presented an innovative DD framework for using KF in large scale applications. In the following we will refer to discretization of $\Omega \times \Delta$ in Definition 1, section (2.1).
We define KF state function of a dynamic system in $\Omega \times \Delta$ at time t_{l+1} as follows:

$$x_{l+1} \equiv x(t_{l+1}) := \{u^{\mathcal{M}}(x_{\tilde{i}}, t_{\tilde{k}})\}_{\tilde{i}\in I;\ \tilde{k}\in K}$$

where $u^{\mathcal{M}}$ and $\{(x_{\tilde{i}}, t_{\tilde{k}})\}_{\tilde{i}\in I;\ \tilde{k}\in K}$ are defined in (2.1) and Definition 1, respectively.
In the context of DA methods, KF aims to bring the state $x(t_{l+1})$ as close as possible to the measurements/observations $y(t_{l+1})$. One can do this by discretizing then optimize or first optimize and then discretize. Here, with the aim of making the best use of Schwarz and PinT methods in a linear algebra settings, we first discretize then optimize.
We will use the following discrete formulation of KF method [116]:

- N_p: number of inner nodes in Ω defined in (2.4);
- $N := r + 2$: number of instants of time in Δ defined in (2.5), where $r \in \mathbb{N}$;

- $x_l \equiv x(t_l) \in \mathbb{R}^{N_p}$: the state of system at time t_l, for $l = 0, 1, \ldots, r+1$;
- $\widehat{x}_0 \equiv x_0$: the state estimate at time $t_0 \equiv 0$;
- $\widehat{x}_l$: the state estimate at time t_l, for $l = 1, \ldots, r+1$;
- $w_l \in \mathbb{R}^{N_p}$ and $v_l \in \mathbb{R}^{n_{obs}}$: model and observation errors with normal distribution and zero mean such that $E[w_l v_j^T] = 0$, for $j, l = 0, 1, \ldots, r+1$, where $E[\cdot]$ denotes the expected value;
- $B_l \in \mathbb{R}^{N_p \times N_p}$ and $R_l \in \mathbb{R}^{n_{obs} \times n_{obs}}$: covariance matrices of the errors on the model and on the observations, respectively i.e.

$$B_l := E[w_l w_l^T] \quad R_l := E[v_l v_l^T] \quad \forall\, l = 0, 1, \ldots, r+1.$$

  These matrices are symmetric and positive definite.

KF method: KF method consists in calculating an estimate $\widehat{x}_{l+1}$, at time t_{l+1}, of the state $x_{l+1} \in \mathbb{R}^{N_p}$:

$$x_{l+1} = M_{l,l+1} x_l + w_l, \quad \forall l = 0, 1, \ldots, r \tag{2.14}$$

such that

$$y_{l+1} = H_{l+1} x_{l+1} + v_{l+1}, \quad \forall l = 0, 1, \ldots, r. \tag{2.15}$$

KF algorithm: Given $\widehat{x}_0 \in \mathbb{R}^{N_p}$ and $P_0 = O \in \mathbb{R}^{N_p \times N_p}$ a null matrix, for each $l = 0, 1, \ldots, r$ KF algorithm is made by:

- Predicted phase.
  - Compute predicted state estimate:

$$x_{l+1} = M_{l,l+1} \widehat{x}_l; \tag{2.16}$$

- Compute predicted covariance matrix:

$$P_{l+1} = M_{l,l+1} P_l M_{l,l+1}^T + B_l. \tag{2.17}$$

- Corrector phase.
  - Compute Kalman gain:

$$\mathbf{K}_{l+1} = P_{l+1} H_{l+1}^T (H_{l+1} P_{l+1} H_{l+1}^T + R_{l+1})^{-1}. \tag{2.18}$$

  - Update covariance matrix:

$$P_{l+1} = (I - \mathbf{K}_{l+1} H_{l+1}) P_{l+1}, \tag{2.19}$$

  - Update state estimate:

$$\widehat{x}_{l+1} = x_{l+1} + \mathbf{K}_{l+1}(y_{l+1} - H_{l+1} x_{l+1}). \tag{2.20}$$

When the measurement uncertainty R_l is large, then the $\mathbf{K}_{l+1}$ defined in (2.18) will be low, this means that the model data are more reliable than observations data. However, when the measurement uncertainty R_l is small, then the $\mathbf{K}_{l+1}$ defined in (2.18) will be high, this means that the observations data are more reliable than model data.

Chapter 3

The DA-driven Space and Time decomposition approach

Main approaches for delivering scalable solutions of simulations based on DA methods integrated with a PDE-based model essentially only takes full advantage of existing parallel PDE solvers, and in particular those based on Domain Decomposition (DD) methods in space, where the DD-solver is suitably modified to also handle the adjoint system. While this scheme is efficient, it has a limited scalability, due to the strong synchronization between the PDE integration and the DA solver. Time-parallel approaches provide a new avenue to achieve scaling on new generation computing environments. The core of our PinT-based algorithm is that PDE&DA models are tightly coupled so that, once the whole space-and-time domain is decomposed (by using PinT-based approaches for decomposing the PDE model and variational calculus for decomposing the DA functional), DA model acts as coarse/predictor of the local PDE model, by providing the background values as initial conditions of the local PDE model. Moreover, in contrast to other PinT-based approaches, in our DA-driven PinT-based approach, local solvers run concurrently, so that the resulting algorithm only requires exchange of boundary conditions between adjacent sub-domains. In this way it leads to an ”adaptive composition of local solvers”

in which, following a tree configuration manner, its customization varies from sub-problem to sub-problem. Finally, the proposed approach is non-intrusive, allowing the incremental transition of existing software (as for instance, the Regional Ocean Modelling System-ROMS). In this Chapter, we present a domain decomposition-based frameworks for solution of 3DVAR, 4DVAR DA and KF problems, involving decomposition of the physical domain, partitioning of the solution and modification of the regularization functional describing the variational DA problem. We call them DD–3DVAR, DD–4DVAR and DD–KF.

3.1 Domain Decomposition of physical domain

In the following we introduce the main modules of DD–DA algorithm:

Domain Decomposition of $\Omega \times \Delta$.

We describe DD of $\Omega \times \Delta$ and $\Omega_I \times \Delta_k$.

- DD of $\Omega \times \Delta$ consists of:
  - DD of Ω: decomposition of $\Omega \subset \mathbb{R}^n$ into a sequence of subdomains Ω_i such that:

$$\Omega = \bigcup_{i=1}^{N_{sub}} \Omega_i \tag{3.1}$$

    and definition of set of indices of subdomains adjacent to Ω_i and its cardinality, as follows

$$J_i \subset \{1, \ldots, N_{sub}\} \quad \text{and} \quad ad_i = |J_i| \tag{3.2}$$

    in particular, ad_i is the number of subdomains adjacent to Ω_i.

    For $i = 1, \ldots, N_{sub}$, definition of overlap regions Ω_{ij} as follows

$$\Omega_{ij} := \Omega_i \cap \Omega_j \neq \emptyset \quad \forall j \in J_i. \tag{3.3}$$

    Definition of interfaces Ω_i, for $i = 1, \ldots, N_{sub}$:

$$\Gamma_{ij} := \partial\Omega_i \cap \Omega_j \quad j \in J_i. \tag{3.4}$$

- DD of Δ: decomposition of time interval $\Delta \subset \mathbb{R}$ into a sequence of time interval Δ_k such that:

$$\Delta = \bigcup_{k=1}^{N_t} \Delta_k.$$

Consequently, we define

$$\{\Omega_i \times \Delta_k\}_{i=1,\dots,N_{sub};k=1,\dots,N_t} \tag{3.5}$$

as local domains.

- DD of $\Omega_I \times \Delta_K$ defined in (2.6) consists of:
  - DD of Ω_I:

    identification of inner nodes of subdomains $\{\Omega_i\}_{i=1,\dots,N_{sub}}$: for $i = 1, \dots, N_{sub}$

$$\Omega_{I_i} \equiv \{x_{\tilde{i}}\}_{\tilde{i} \in I_i} \subset \Omega_i$$

    are inner nodes of Ω_i where I_i is set defined as follows

$$I_i := \begin{cases} \left\{1, \dots, \frac{N_p}{N_{sub}} + \frac{\delta}{2}\right\} & \text{if } i = 1 \\ \left\{(i-1) \times \frac{N_p}{N_{sub}} - \frac{\delta}{2} + 1, \dots, i \times \frac{N_p}{N_{sub}} + \frac{\delta}{2}\right\} & \text{if } 1 < i < N_{sub} \\ \left\{(N_{sub}-1) \times \frac{N_p}{N_{sub}} - \frac{\delta}{2} + 1, \dots, N_p\right\} & \text{if } i = N_{sub} \end{cases} \tag{3.6}$$

    such that

$$I = \bigcup_{i=1}^{N_{sub}} I_i. \tag{3.7}$$

    with I set of indices of inner nodes in Ω defined in (2.4) and

$$\delta := |I_{i,j}| \tag{3.8}$$

    the number of inner nodes in overlap region Ω_{ij} defined in (3.3).

    Identification of inner nodes of overlap regions $\{\Omega_{ij}\}_{i=1,\dots,N_{sub},j \in J_i}$: for $i = 1, \dots, N_{sub}$

$$\{x_{\tilde{j}}\}_{\tilde{j} \in I_{ij}} \subset \Omega_{ij} \quad \forall j \in J_i$$

are inner nodes of Ω_{ij} where

$$I_{ij} := I_i \cap I_j \neq \emptyset \quad \forall j \in J_i \tag{3.9}$$

is the set of indices of nodes in overlap region. Consequently, for $i = 1, \ldots, N_{sub}$ we define the cardinality of I_i as follows:

$$N_{loc} := |I_i| = \frac{N_p}{N_{sub}} + \frac{\delta}{2} \tag{3.10}$$

i.e. number of inner nodes of spatial subdomain Ω_i.

– DD of Δ_K: identification of instants of time in time intervals $\{\Delta_k\}_{k=1,\ldots,N_t}$: for $k = 1, \ldots, N_t$

$$\Delta_{K_k} \equiv \{t_{\tilde{k}}\}_{\tilde{k} \in K_k} \subset \Delta.$$

are equispaced time instants in Δ_k where K_k is set defined as follows:

$$K_k := \left\{ (k-1) \times \frac{N}{N_t}, \ldots, k \times \frac{N}{N_t} \right\}$$

where

$$K_k \cap K_{k+1} = \left\{ k \times \frac{N}{N_t} \right\} \neq \emptyset \quad \forall k = 1, \ldots, N_t - 1$$

and

$$N_k := |K_k| = \frac{N}{N_t}$$

its cardinality of K_k i.e. number of instants of time in K_k, such that

$$K = \bigcup_{k=1}^{N_t} K_k. \tag{3.11}$$

with K set defined in (2.5).

Consistently with Definition 1, for $i = 1, \ldots, N_{sub}$, $k = 1, \ldots, N_t$

$$\Omega_{I_i} \times \Delta_{K_k} := \{(x_{\tilde{i}}, t_{\tilde{k}})\}_{\tilde{i} \in I_i; \tilde{k} \in K_k} \subset \Omega_i \times \Delta_k$$

is local discrete domain.

We are able to define the restriction and extension spatial operator.

Definition 5. *(Restriction Operator) For $i = 1, \dots, N_{sub}$ we define restriction matrices*

$$R_i : \ \Omega \times Y \to \Omega_i \times Y$$

on Ω_i and

$$R_{ij} : \ \Omega \times Y \to \Omega_{ij} \times Y$$

on Ω_i where $Y \subset \mathbb{N}$ and $|Y|$ its cardinality. Given $x \in \mathbb{R}^{N_p \times |Y|}$ and $z \in \mathbb{R}^{N_p \times N}$ we define restriction of x to Ω_i :

$$x/\Omega_i := R_i x = \{x(\tilde{i}, \tilde{k})\}_{\tilde{i} \in I_i, \tilde{k} \in |Y|} \in \mathbb{R}^{N_{loc} \times \frac{N}{N_t}} \tag{3.12}$$

$$x/\Omega_{ij} := R_{ij} x = \{x(\tilde{j}, \tilde{k})\}_{\tilde{j} \in I_{ij}, \tilde{k} \in K_k} \in \mathbb{R}^{\delta \times \frac{N}{N_t}}$$

and restriction of z to $\Omega_i \times \Delta_k$

$$z/(\Omega_i \times \Delta_k) := \{z(\cdot, \tilde{k})\}_{\tilde{k} \in K_k}/\Omega_i = R_i \cdot \{z(\cdot, \tilde{k})\}_{\tilde{k} \in K_k} = \{z(\tilde{i}, \tilde{k})\}_{\tilde{i} \in I_i, \tilde{k} \in K_k} \in \mathbb{R}^{N_{loc} \times N_k}$$

$$z/(\Omega_{ij} \times \Delta_k) := \{z(\cdot, \tilde{k})\}_{\tilde{k} \in K_k}/\Omega_{ij} = R_{ij} \cdot \{z(\cdot, \tilde{k})\}_{\tilde{k} \in K_k} = \{z(\tilde{i}, \tilde{k})\}_{\tilde{i} \in I_{ij}, \tilde{k} \in K_k} \in \mathbb{R}^{N_{loc} \times N_k}$$

where I_i and I_{ij} are respectively set of indices of inner nodes in Ω_i and Ω_{ij}, $\forall j \in J_i$.

Definition 6. *(Extension operator) We define the Extension Operator (EO). If $x \in \mathbb{R}^{N_{loc} \times N_k}$, it is*

$$EO(x) := R_i^T x = \begin{cases} x(\tilde{i}, \tilde{k}) & \text{if } (\tilde{i}, \tilde{k}) \in I_i \times K_k \\ 0 & elsewhere \end{cases}$$

where R_i^T is the transpose of R_i in (3.12) and $EO(x) \equiv x^{EO}$.

We underline that R_i is the restriction matrix to subdomain Ω_i differently from R_l that is covariance matrices of the error on the observations at time t_l. In the next sections we introduce the DD–DA methods: DD–3DVAR, DD–4DVAR and DD–KF.

3.2 Domain Decomposition of 3DVAR problem (DD–3DVAR)

The basic idea of of ASM is to divide the region into several overlapping subregions and then to solve the subproblems in the subregions alternatively with boundary information from the neighboring subregions. Let us review the main steps of ASM method applied to 3DVAR DA problem ([23]). It is composed of the following modules:

1. **DD of Ω:** as described in section 3.1 and definition of restriction matrices $R_i \equiv RO_i$ and extension matrices $R_i^T \equiv EO_i$, for $i = 1, ..., N_{sub}$ as follows:

$$R_i = \begin{array}{c} \\ \\ \\ \\ s_{i-1}+1 \\ \vdots \\ s_{i-1}+r_i \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{cccccccccc} & & & & & s_{i-1}+1 & \cdots & s_{i-1}+r_i & & \end{array} \\ \begin{pmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & & & 0 & \cdots & 0 \\ \vdots & & \vdots & & \ddots & & & & \\ 0 & \cdots & 0 & 0 & & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{pmatrix} \end{array}, \tag{3.13}$$

$$
R_i^T = \begin{array}{c} \\ \\ \\ \bar{s}_{i-1,i}+1 \\ \\ \bar{s}_{i-1,i}+r_i \\ \\ \\ \\ \end{array}
\begin{array}{c}
\begin{array}{ccccccccc} & & & \bar{s}_{i-1,i}+1 & \cdots & \bar{s}_{i-1,i}+r_i & & & \end{array} \\
\begin{pmatrix}
0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
0 & \cdots & & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 1 & & & 0\cdots & 0 & \\
\vdots & & \vdots & & \ddots & & & & \\
0 & \cdots & 0 & 0 & & 1 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\
0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0
\end{pmatrix}
\end{array}
\tag{3.14}
$$

where $s_{i,j} = r_i - C_{i,j}$, $\bar{s}_{i,j} = s_{i,j} + t_{ij}$, and r_i, $t_{i,j}$, $C_{i,j}$ points of subdomain Ω_i, interfaces Γ_{ij} and subdomain $\Omega_{ij} = \Omega_i \cap \Omega_j$, respectively.

In order to compute local estimates on local domains defined in (3.5) we introduce the following notation. For $i = 1, \ldots, N_{sub}$, we pose

$$z_i = \{z(\tilde{i})\}_{\tilde{i} \in I_i} \in \mathbb{R}^{N_{loc} \times 1}$$

i.e. a vector defined in local domain Ω_i.

For $n = 0, 1, 2, ...$ until convergence is reached.

2. **Model Reduction:** For $i = 1, \ldots, N_{sub}$, let u_i^{n+1} solution of local problem P_i^{n+1} defined as follows:

$$
\begin{aligned}
argmin_{u_i^{n+1} \in \mathbb{R}^{N_{loc}}} \mathbf{J}_i(u_i^{n+1}) = argmin_{u_i^{n+1} \in \mathbb{R}^{N_{loc}}} \Big[||H_i u_i^{n+1} - y_i||^2_{\mathbf{R}_i} + \\
\alpha \cdot \Big(||u_i^{n+1} - ({u_i}^M)||^2_{\mathbf{B}_i} + ||u_i^{n+1}/\Gamma_{ij} - u_k^n/\Gamma_{ij}||^2_{\mathbf{B}/\Gamma_{ij}} \Big) \Big] ;
\end{aligned}
\tag{3.15}
$$

$\mathbf{B}_i = R_i \mathbf{B} R_i^T$ is a covariance matrix, $\mathbf{B}/\Gamma_{ij} = R_{ij} \mathbf{B} R_{ij}^T$ is restriction of matrix $\mathbf{B}$ to Γ_{ij}; $H_i = R_i H R_i^T$, $\mathbf{R}_i = R_i \mathbf{R} R_i^T$ are restriction of matrices H and $\mathbf{R}$ to subdomain Ω_i, $u_i^b = R_i u^M$, $u_i^{n+1}/\Gamma_{ij} = R_{ij} u_i^{n+1}$, $u_j^n/\Gamma_{ij} = R_{ij} u_j^n$ are restriction of vectors u^M, u_i^{n+1}, u_j^n to subdomain Ω_i and interface Γ_{ij} for $\forall i, j = 1, 2, ..., N_{sub}$.

For simplicity of notations, in the following we pose parameter $\alpha = 1$.

The gradient of $\mathbf{J}_i$, by direct computation, reads as:

$$\nabla \mathbf{J}_i(\mathbf{w}_i^{n+1}) = \mathbf{w}_i^{n+1} + \mathbf{V}_i^T H_i^T \mathbf{R}_i^{-1}(H_i \mathbf{V}_i \mathbf{w}_i^{n+1} - \mathbf{d}_i) + \mathbf{V}_{ij}^T(\mathbf{V}_{ij} \mathbf{w}_i^{n+1} - \mathbf{V}_{ij} \mathbf{w}_j^n)$$

which can be written as follows

$$\nabla \mathbf{J}_i(\mathbf{w}_i^{n+1}) = (\mathbf{V}_i^T H_i^T \mathbf{R}_i^{-1} H_i \mathbf{V}_i + I_i + B/\Gamma_{ij})\mathbf{w}_i^{n+1} - \mathbf{c}_i + \mathbf{B}/\Gamma_{ij} \mathbf{w}_j^n, \tag{3.16}$$

where

$$c_i = (\mathbf{V}_i^T H_i^T \mathbf{R}_i^{-1} H_i \mathbf{V}_i \mathbf{d}_i), \quad \mathbf{d}_i = (y_i - H_i u_i^{n+1}) \tag{3.17}$$

$I_i \in \mathbb{R}^{N_{loc} \times N_{loc}}$ is the identity matrix, R_i is defined in (3.13) and $\mathbf{B} = \mathbf{V}\mathbf{V}^T$ is covariance matrix of the errors on background.

3. **ASM:** from (3.16), according to ASM method and considering the Euler-Lagrange equations corresponding to (3.15), for $i,\ j = 1, ..., N_{sub}$, the systems $(S_i^{ASM})^{n+1}$ defined as:

$$(S_i^{ASM})^{n+1} : \quad A_i^{ASM} \mathbf{w}_i^{n+1} = c_i - \sum_{j \neq i} A_{i,j} \mathbf{w}_j^n, \tag{3.18}$$

where

$$A_i^{ASM} = (\mathbf{V}_i^T H_i^T \mathbf{R}_i^{-1} H_i \mathbf{V}_i + I_i + \mathbf{B}/\Gamma_{ij}), \tag{3.19}$$

and $A_{ij} = \mathbf{B}/\Gamma_{ij}$.

4. **DD–3DVAR approximation in Ω_i:** computation of u_i^{n+1}, related to subdomain Ω_i as:

$$u_i^{n+1} = u_i^M + \mathbf{B}_i^{-1} \mathbf{V}_i \mathbf{w}_i^{n+1}, \quad \text{for } i = 1, ..., J. \tag{3.20}$$

5. **DD–3DVAR solution in Ω_i:** computation of $\widetilde{\mathbf{u}}^{\mathbf{DD-DA}}$, solution of 3DVAR DA problem in (2.9), by patching together vectors $\mathbf{u}_i^{n+1}$, i.e.:

$$\widetilde{\mathbf{u}}^{\mathbf{DD-DA}}(j) = \begin{cases} u_i^{\bar{n}}(j) & \text{se } x_j \in \Omega_i \\ u_k^{\bar{n}}(j) & \text{se } x_j \in \Omega_k \text{ o } x_j \in \Omega_i \cap \Omega_k, \end{cases}, \tag{3.21}$$

for $i, k = 1, ...N_{sub}$, and $\bar{n}$ corresponding iterations needed to stop of the iterative procedure.

The DD method introduced in [30] provides the minimum of functional **J** in (2.10) (defined on the entire domain) as a piecewise function by collecting the minimum of each local functional $\mathbf{J}_i$ defined in (3.15) by adding a local constraint about the entire overlap region. The DD method in [30] is composed of the following steps:

i. DD of Ω as described in section 3.1;

ii. definition of restriction RO and extension EO operators;

iii. for $n = 0, 1, 2, ...$, solution of N_{sub} subproblems $\left(P_i^{DD-DA}\right)^{n+1}$ where

$$\begin{aligned} \left(P_i^{DD-DA}\right)^{n+1} \; argmin_{u_i^{n+1} \in \mathbb{R}^{N_{loc}}} \mathbf{J}_i^{DD-DA}(u_i^{n+1}) = \\ argmin_{u_i^{n+1} \in \mathbb{R}^{N_{loc}}} (||H_i u^{RO} - y^{RO}||^2_{\mathbf{R}_i} + \lambda \cdot ||u^{RO} - \left(u^M\right)^{RO}||^2_{\mathbf{B}_i} + \\ \beta \cdot ||u^{RO}/\Omega_{ij} - u^{RO}/\Omega_{ij}||^2_{\mathbf{B}_{ij}}), \end{aligned} \tag{3.22}$$

where $\mathbf{B}_{ij}$ is restriction of matrix **B** to overlap region Ω_{ij}, and H_i, $\mathbf{B}_i$ restriction of matrices **H**, **B** to subdomain Ω_i for $i = 1, ..., N_{sub}$, according the description in [30]. Finally β is non negative parameter and *RO* is restricted operator of DD method in [8].
From (3.22) the Euler-Lagrange equations give rise to the following systems $\left(S_i^{DD-DA}\right)^{n+1}$ [30]:

$$\left(S_i^{DD-DA}\right)^{n+1} : \quad A_i^{DD-DA} \mathbf{w}_i^{n+1} = c_i^{DD-DA}, \tag{3.23}$$

to solve for $n = 0, 1, ...$, while the vectors c_i^{DD-DA} and matrices A_i^{DD-DA} are defined as follows:

$$c_i^{DD-DA} = (\mathbf{V}_i^T H_i^T \mathbf{R}_i^{-1} H_i \mathbf{V}_i \mathbf{d}_i), \tag{3.24}$$

$$A_i^{DD-DA} = (\mathbf{V}_i^T H_i^T \mathbf{R}_i^{-1} H_i \mathbf{V}_i + I_i) \tag{3.25}$$

and I_i is the identity matrix, for $i = 1, ..., N_{sub}$.

iv. computation of $\mathbf{u}_i^{n+1}$, as in (3.20), for $i = 1, ..., N_{sub}$.

v. computation of $\widetilde{\mathbf{u}}^{\mathbf{DD-DA}}$, solution of problem 3DVAR DA in (2.9) obtained by gathering $u_i^{\bar{n}}$ solution of $(P_i^{DD-DA})^{\bar{n}}$ defined in (3.22), where $\bar{n}$ is latest iteration.

In order to prove equivalence between the DD method in [30] and ASM for solving the DA problem, firstly we note that the following equivalence holds on:

$$EO \equiv R_i, \quad RO \equiv R_i^T,$$

where RO and EO are restriction and extension operator in Definitions 3.116 and 6. Then if we let $A \in \mathbb{R}^{NP \times NP}$ and consider N_{loc} points of Ω_i, it is

$$RO(A) \equiv R_i A R_i^T,$$

for $i = 1, ..., N_{sub}$. We now prove the following result.

Proposition 1. *Let $\widetilde{\mathbf{u}}^{\mathbf{DD-DA}}$ in (3.21) be the solution 3DVAR DA problem in (2.9) obtained applying the DD method in [8], that is, by solving for $n = 0, 1, ...$ systems $(S_i^{DD-DA})^{n+1}$ in (3.23). Similarly, the ASM, provides $\bar{\mathbf{u}}^{\mathbf{DD-DA}}$ by solving for $n = 0, 1, ...$ systems $(S_i^{ASM})^{n+1}$ in (3.18). We prove that systems in (3.18) and (3.23) are equivalent.*

Proof. Let us assume that $J = 2$. We consider subdomains Ω_1, Ω_2 and interfaces $\Gamma_{12} := \partial\Omega_1 \cap \Omega_2$, $\Gamma_{21} := \partial\Omega_2 \cap \Omega_1$.

By using the DD method in [8] it follows that:

$$(S_1^{DD-DA})^{n+1}: \quad A_1^{DD-DA}\mathbf{w}_1^{n+1} = c_1^{DD-DA} \longrightarrow \mathbf{w}_1^{n+1} = (A_1^{DD-DA})^{-1}c_1^{DD-DA} \tag{3.26}$$

and

$$(S_2^{DD-DA})^{n}: \quad A_2^{DD-DA}\mathbf{w}_1^{n} = c_1^{DD-DA} \longrightarrow \mathbf{w}_2^{n} = (A_2^{DD-DA})^{-1}c_2^{DD-DA}$$

by using ASM we get

$$\begin{aligned} (S_1^{ASM})^{n+1}: \quad & A_1^{ASM}\mathbf{w}_1^{n+1} = c_1 + A_{12}\mathbf{w}_2^{n} \\ (S_2^{ASM})^{n+1}: \quad & A_2^{ASM}\mathbf{w}_2^{n+1} = c_2 + A_{21}\mathbf{w}_1^{n}. \end{aligned} \tag{3.27}$$

We prove the equivalence between $(S_1^{DD-DA})^{n+1}$ in (3.26) and $(S_1^{ASM})^{n+1}$ in (3.27), i.e. we prove that solutions obtained from $(S_1^{DD-DA})^{n+1}$ and $(S_2^{DD-DA})^{n}$ in (3.26) satisfy $(S_1^{ASM})^{n+1}$ in (3.27).

Replacing $\mathbf{w}_1^{n+1}$ by $(A_1^{DD-DA})^{-1}c_1^{DD-DA}$ in (3.27), it follows that:

$$A_1^{ASM}(A_1^{DD-DA})^{-1}c_1^{DD-DDA} = c_1 + A_{12}\mathbf{w}_2^{n}; \tag{3.28}$$

matrix A_1^{ASM} in (3.19) can be rewritten as

$$A_1^{ASM} := (\mathbf{V}_1^T H_1^T \mathbf{R}_1^{-1} H_1 \mathbf{V}_1 + I_1 + \mathbf{B}/\Gamma_{12}) = A_1^{DD-DA} + A_{12},$$

where A_1^{DD-DA} is defined in (3.25). Then, the (3.28) becomes:

$$(A_1^{DD-DA} + A_{12})(A_1^{DD-DA})^{-1}c_1^{DD-DA} = c_1 + A_{12}\mathbf{w}_2^{n}. \tag{3.29}$$

Replacing $\mathbf{w}_2^{n}$ with $(A_2^{DD-DA})^{-1}c_2^{DD-DA}$ in (3.29), it follows that:

$$c_1^{DD-DA} + A_{12}((A_1^{DD-DA})^{-1}c_1^{DD-DA})|_{\Gamma_{12}} = c_1 + A_{12}((A_2^{DD-DA})^{-1}c_2^{DD-DA})|_{\Gamma_{12}}$$

then, we obtain

$$c_1^{DD-DA} = c_1. \tag{3.30}$$

From (3.17) and (3.24) we have

$$c_1^{DD-DA} = (\mathbf{V}_i^T H_i^T \mathbf{R}_i^{-1} H_i \mathbf{V}_i \mathbf{d}_i) = c_1,$$

and the proof is complete. □

3.3 Domain Decomposition of 4DVAR problem (DD–4DVAR)

The proposed DD method consists in decomposing the domain of computation $\Omega \times \Delta$ into subdomains in space and time and solving reduced forecast models and local 4DVAR DA problems. The modules of DD method are: domain decomposition of $\Omega \times \Delta$, model reduction, ASM method based on Additive Schwarz method idea [44], DD–4DVAR local solution and DD–4DVAR global solution. The method is made of two nested loops: the outer loop (over index n) defines approximations of the reduced models, while the inner loop (over index r) solve reduced 4DVAR DA problems. The schematic description of DD–4DVAR algorithm is reported in Figure 3.1.

1. **DD of Ω:** as described in section 3.1.

   In order to compute local approximation in local domains defined in (3.5) we introduce the following notation. For $i = 1, \ldots, N_{sub}$, $k = 1, \ldots, N_t$, we pose

   $$z_{i,k} = \{z(\tilde{i}, \tilde{k})\}_{\tilde{i} \in I_i;\ \tilde{k} \in K_k} \in \mathbb{R}^{N_{loc} \times N_k}$$

   i.e. a vector defined in local domain $\Omega_i \times \Delta_k$.

   We introduce an **outer-loop**. The outer-loops solve the nonlinear aspects of the assimilation problem which for DD–4DVAR, in general for 4DVAR techniques, includes the

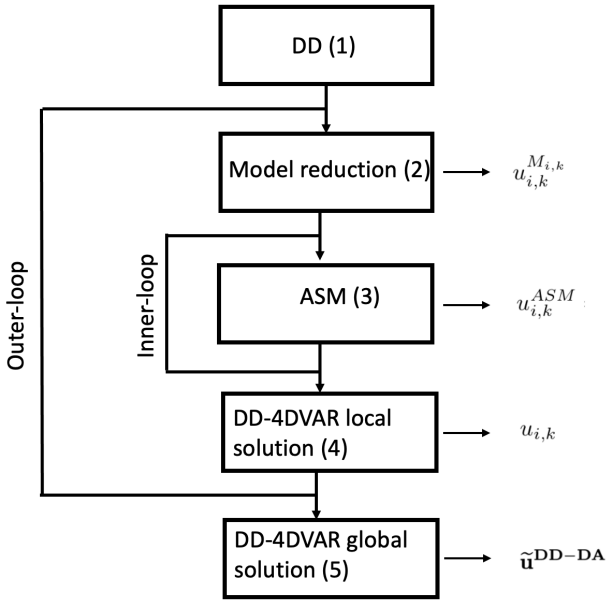


Figure 3.1: Schematic description of DD–4DVAR algorithm. The modules of DD–4DVAR: DD, Model Reduction, ASM, DD–4DVAR local solution and DD–4DVAR global solution are identified and the Arabic numbers in parentheses refer to the corresponding module in section 3.3. For each module we report the corresponding solution.

integration of the nonlinear model.

For $n = 0, 1, \ldots, \bar{n}$, do

2. **Model reduction:** for $i = 1, \ldots, N_{sub}$; $k = 1, \ldots, N_t$, posed $u^0_{i,k} := \{u^M(x_{\tilde{i}}, t_{\tilde{k}})\}_{\tilde{i} \in I_{ij}, \tilde{k} \in K_k}$ i.e. by using the background as local initial values, let $u^{M_{i,k},n+1}_{i,k}$ be the solution of the local model $(P^{M_{i,k},n}_{i,k})_{i=1\ldots,N_{sub},k=1,\ldots,N_t}$:

$$(P^{M_{i,k},n}_{i,k})_{i=1\ldots,N_{sub},k=1,\ldots,N_t} \; : \quad \begin{cases} u^{M_{i,k},n}_{i,k} = M_{i,k} \cdot u^n_{i,k-1} + b^n_{i,k}, \\ u^n_{i,k-1} = u^{M_{i,k},n}_{i,k-1} \\ u^n_{i,k}/\Gamma_{ij} = u^n_{j,k}/\Gamma_{ij}, \quad j \in J_i \end{cases} \tag{3.31}$$

where $u^M_{i,k}$, $b^n_{i,k}$ are respectively the background in $\Omega_i \times \Delta_k$, the vector accounting boundary conditions of Ω_i and

$$M_{i,k} := M_k/\Omega_i \tag{3.32}$$

the restriction to Ω_i of the matrix

$$M_k \equiv M_{\bar{s}_{k-1},\bar{s}_k} := M_{\bar{s}_{k-1},\bar{s}_{k-1}+1} \cdots M_{\bar{s}_k-1,\bar{s}_k}. \tag{3.33}$$

where

$$\bar{s}_k := \sum_{j=1}^{k-1} N_j - (N-1) \quad \text{and} \quad \bar{s}_0 := 0 \tag{3.34}$$

are respectively the first index of Δ_{K_K} and Δ_{K_1} and $b^n_{i,k}$ is vector computed from information in boundary of Ω_i at interval time Δ_k that depends on discretization scheme used.

Let:

$$(P^n_{i,k})_{i=1,\ldots,N_{sub},k=1,\ldots,N_t} \; : u^{ASM,n}_{i,k} = \arg\min_{u^n_{i,k}} \; \mathbf{J}_{i,k}(u^n_{i,k}) \tag{3.35}$$

be the local 4DVAR DA model with

$$\mathbf{J}_{i,k}(u^n_{i,k}) = \mathbf{J}(u^n_{i,k})/(\Omega_i \times \Delta_k) + \mathcal{O}_{ij}. \tag{3.36}$$

We let

$$\mathcal{O}_{ij} = \sum_{j \in J_i} \beta_j \cdot \|u^n_{i,k}/\Omega_{ij} - u^n_{j,k}/\Omega_{ij}\|^2_{\mathbf{B}^{-1}_{ij}} \tag{3.37}$$

be the overlapping operator in Ω_{ij}, and

$$\mathbf{J}_{i,k}(u^n_{i,k})/(\Omega_i \times \Delta_k) = \alpha_{i,k} \cdot \|u^n_{i,k} - u^{M_{i,k},n}_{i,k}\|_{\mathbf{B}^{-1}_i} + \|G_{i,k}u^n_{i,k} - y_{i,k}\|^2_{\mathbf{R}^{-1}_i}$$

be the restriction of $\mathbf{J}$ in $\Omega_i \times \Delta_k$. Parameters $\alpha_{i,k}$ and β_j in (3.37) denotes the regularization parameters. Following, we let $\alpha_{i,k} = \beta_j = 1$, $j \in J_i$.

3. **ASM**: according to ASM [44] we compute solution of $(P^n_{i,k})_{i=1,\ldots,N_{sub},k=1,\ldots,N_t}$. Gradient of $\mathbf{J}_{i,k}$ is [8]:

$$\begin{aligned}\nabla \mathbf{J}_{i,k}(\mathbf{w}^n_{i,k}) = \quad & (\mathbf{V}^T_i (G_{i,k})^T (\mathbf{R}_{i,k})^{-1} G_{i,k}\mathbf{V}_i + I_i + ad_i \cdot \mathbf{B}_{ij})\mathbf{w}^n_{i,k} \\ & -\mathbf{c}_i + \textstyle\sum_{j \in J_i} \mathbf{B}_{ij}\mathbf{w}^n_{j,k},\end{aligned}$$

where

$$\mathbf{w}^n_{i,k} = \mathbf{V}^{-1}_i (u^n_{i,k} - u^{M_{i,k},n}_{i,k}), \tag{3.38}$$

$$\mathbf{d}_i = (\mathbf{v}_i - G_{i,k}u^{M_{i,k},n}_{i,k}) \quad \mathbf{c}_i = (\mathbf{V}^T_i (G_{i,k})^T (\mathbf{R}_{i,k})^{-1}\mathbf{d}_i)$$

and I_i the identity matrix, ad_i number of subdomains adjacent to Ω_i defined in (3.2). Solution of $(P^n_{i,k})_{i=1,\ldots,N_{sub},k=1,\ldots,N_t}$ is obtained by requiring $\nabla \mathbf{J}_{i,k}(\mathbf{w}^n_{i,k}) = 0$. This gives rise to the linear system:

$$A_{i,k}\mathbf{w}^n_{i,k} = c_i - \sum_{j \in J_i} \mathbf{B}_{ij}\mathbf{w}^n_{j,k}, \tag{3.39}$$

where

$$A_{i,k} = (\mathbf{V}^T_i (G_{i,k})^T \mathbf{R}^{-1}_{i,k} G_{i,k}\mathbf{V}_i + I_i + ad_i \cdot \mathbf{B}_{ij}). \tag{3.40}$$

(3.1) For each iteration n, r.h.s. of (3.39) depends on unknown value $\mathbf{w}^n_{j,k}$ defined in

adjacent subdomains Ω_{ij} of Ω_i, $j \in J_i$. Hence, for solving the system in (3.39).

We introduce an inner-loops related to minimization of local 4DVAR model defined in (3.35). For $r = 0, 1, \ldots, \bar{r}$

$$A_{i,k}\mathbf{w}_{i,k}^{r+1,n} = c_i - \sum_{j \in J_i} \mathbf{B}_{ij}\mathbf{w}_{j,k}^{r,n}, \tag{3.41}$$

where at each step $r+1$, Ω_i *receives* $\mathbf{w}_{i,k}^{r,n}$ from Ω_{ij}, $j \in J_i$ for computing the r.h.s. of (3.41) then it computes $\mathbf{w}_{i,k}^{r+1,n}$ by using Conjugate Gradient (CG) method and finally it *sends* $\mathbf{w}_{i,k}^{r+1,n}$ to Ω_{ij}, $j \in J_i$ for updating the r.h.s. of (3.41) needed to the next iteration. $\mathbf{w}_{i_j,k}^{0,n}$ is an arbitrary initial value. Finally, we pose

$$\mathbf{w}_{i,k}^{n} \equiv \mathbf{w}_{i,k}^{\bar{r},n},$$

consequently

$$u_{i,k}^{ASM,n} := u_{i,k}^{ASM,\bar{r}}. \tag{3.42}$$

4. **DD–4DVAR approximation in $\Omega_i \times \Delta_k$:** final solution update, using (A.40) and (3.38):

$$u_{i,k}^{n+1} \ = u_{i,k}^{M_{i,k},n+1} + \mathbf{V}_i\mathbf{w}_{i,k}^{n} = u_{i,k}^{M_{i,k},n+1} + [u_{i,k}^{ASM,n} - u_{i,k}^{M_{i,k},n}]. \tag{3.43}$$

Endfor n

5. **DD–4DVAR solution in $\Omega \times \Delta$:** computation of DD–4DVAR approximation in $\Omega \times \Delta$: let

$$\widetilde{\mathbf{u}}^{\mathbf{DD-DA,n}} := \sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} (u_{i,k}^{n})^{EO}. \tag{3.44}$$

be DD–4DVAR approximation at iteration n where $(u_{i,k}^{n})^{EO}$ is extension to $\Omega \times \Delta$ of DD–4DVAR approximations in $\Omega_i \times \Delta_k$, we define

$$\widetilde{\mathbf{u}}^{\mathbf{DD-DA}} := \widetilde{\mathbf{u}}^{\mathbf{DD-DA},\bar{\mathbf{n}}} \tag{3.45}$$

as DD–4DVAR solution in $\Omega \times \Delta$.

Note that $\mathbf{B}_i = R_i \mathbf{B} R_i^T$ and $\mathbf{B}_{ij} := \mathbf{B}/\Gamma_{ij} = R_i \mathbf{B} R_{ij}^T$ are the restrictions of covariance matrix $\mathbf{B}$, respectively, to subdomain Ω_i and interface Γ_{ij} in (3.4), $G_{i,k}$, $\mathbf{R}_{i,k}$ are the restriction of matrices $G_k := G_{\bar{s}_k}$ and $\mathbf{R}_k := diag(\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_{\bar{s}_k})$ to Ω_i, and finally $u_{i,k}^M = R_i u_k^M$, $u_{i,k}^{n+1}/\Gamma_{ij} = R_{ij} u_{i,k}^{n+1}$, $u_{j,k}^n/\Gamma_{ij} = R_{ij} u_k^n$ are the restriction of vectors u_k^b, $u_{i,k}^{n+1}$, $u_{j,k}^n$ to Ω_i and Γ_{ij}, for $i = 1, 2, \ldots, N_{sub}$, $j \in J_i$.

It is worth noting that each local functional $\mathbf{J}_{i,k}$ is obtained starting from a restriction of the global functional $\mathbf{J}$ and adding a local term (3.37) defined in the overlapping regions. In addition, regarding the decomposition in time direction, we use DA as predictor operator for the local PDE-based model, providing the approximations needed for solving the initial value problem in each sub interval.

We prove that the minimum of $\mathbf{J}$ can be obtained by patching together all the local solution obtained as minimum of local function $\mathbf{J}_{i,k}$. We are able to prove the following result.

Theorem 1. *If* $\mathbf{J}$ *is convex:*

$$\mathbf{J}(\mathbf{u}^{\mathbf{DA}}) = \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA}}). \tag{3.46}$$

Proof. As $\mathbf{u}^{\mathbf{DA}}$ is the global minimum of $\mathbf{J}$ it follows that:

$$\mathbf{J}(\mathbf{u}^{\mathbf{DA}}) \leq \mathbf{J}\left(\sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} (\mathbf{u}_{i,k}^{\bar{n}})^{EO}\right), \quad \forall\, i, k \tag{3.47}$$

then, from the (3.45) it follows that

$$\mathbf{J}(\mathbf{u}^{\mathbf{DA}}) \leq \mathbf{J}\left(\widetilde{\mathbf{u}}^{\mathbf{DD-DA}}\right) \quad . \tag{3.48}$$

Now we prove that if $\mathbf{J}$ is convex, then

$$\mathbf{J}(\mathbf{u}^{\mathbf{DA}}) = \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA}})$$

by reduction to the absurd. Assume that

$$\mathbf{J}(\mathbf{u^{DA}}) < \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA}}). \tag{3.49}$$

In particular,

$$\mathbf{J}(\mathbf{u^{DA}}) < \mathbf{J}\left(\sum_{i=1}^{N_{sub}}\sum_{k=1}^{N_t}\widetilde{\mathbf{u}}_{\mathbf{i,k}}\right) \leq \sum_{i=1}^{N_{sub}}\sum_{k=1}^{N_t}\mathbf{J}_{\mathbf{i,k}}(u_{i,k}^{\bar{n}}) \quad . \tag{3.50}$$

This means that

$$\mathbf{J}(\mathbf{u^{DA}}) < \sum_{i=1}^{N_{sub}}\sum_{k=1}^{N_t}\mathbf{J}_{\mathbf{i,k}}(u_{i,k}^{\bar{n}}) \quad . \tag{3.51}$$

From the (3.51) and the (3.50), it is:

$$\nabla\mathbf{J}(\mathbf{u}^{DA}) < \nabla\sum_{i=1}^{N_{sub}}\sum_{k=1}^{N_t}\mathbf{J}_{\mathbf{i,k}}(u_{i,k}^{\bar{n}}) = \sum_{i=1}^{N_{sub}}\sum_{k=1}^{N_t}\nabla\mathbf{J}_{\mathbf{i,k}}(u_{i,k}^{\bar{n}}) \tag{3.52}$$

$\mathbf{u}^{DA}$ is global minimum of J i.e. $\nabla\mathbf{J}(\mathbf{u}^{DA}) = 0$, from (3.52) it is

$$\nabla\mathbf{J}_{\mathbf{i,k}}(u_{i,k}) > 0 \quad \forall(i,k). \tag{3.53}$$

The (3.53) is an absurd as the values of $u_{i,k}^{\bar{n}}$ are the minimum for $\mathbf{J}_{\mathbf{i,k}}$ for $\forall(i,k)$. Hence, the (3.46) is proved.

□

In the next section we will show that this formulation leads to local numerical solutions convergent to the numerical solution of the global problem.

3.3.1 Algorithm

In Table 3.1 we report DD procedure for solving 4DVAR DA problems in $\Omega\times\Delta$. For a fixed subdomain in space and time, i.e. $\Omega_i \times \Delta_{k+1}$, the procedure is made of two nested loops: the outer

loop over index n, defines an approximation of local models by calling the Predictor–Update procedure (Table 3.2), while the inner loop over index r solves local 4DVAR DA problems calling the ASM Procedure (Table 3.3).

Iterative schemes are stopped when *convergence* is satisfied, i.e. at the minimum value of n and r, respectively, such that

$$E^{r,\Omega_i\times\Delta_k} = \|\mathbf{w}_{i,k}^{r+1} - \mathbf{w}_{i,k}^{r}\| < tol_{sub}, \quad i = 1,\ldots,N_{sub}, k = 1,..,N_t \tag{3.54}$$

and

$$E^{n,\Omega_i\times\Delta_k} = \|u_{i,k}^{n+1} - u_{i,k}^{n}\| < tol_{t}, \quad i = 1,\ldots,N_{sub}, k = 1,..,N_t \tag{3.55}$$

where tol_{sub} and tol_t are fixed. Note that, for each value of n, convergence of ASM in [44] involves convergence of $(\mathbf{w}_{i,k}^{r})_{r\in\mathbb{N}_0}$ i.e.

$$\mathbf{w}_{i,k}^{r} = \mathbf{V}_i^{-1}(u_{i,k}^{ASM,r} - u_{i,k}^{M_{i,k},n}) \xrightarrow[r\to\infty]{} \mathbf{V}_i^{-1}(u_{i,k}^{n} - u_{i,k}^{M_i,n}) =: \mathbf{w}_{i,k}$$

Table 3.1: DD algorithm in $\Omega \times \Delta$.

Procedure DD-Procedure(in:$N_t, N_{sub}, M, v, u^M, u_0, \mathbf{B}, \mathbf{R}, G;\ out : \widetilde{\mathbf{u}}^{\mathbf{DD-DA}}$)

Begin of DD Step:

$\Omega \leftarrow (\Omega_1, \Omega_2, \ldots, \Omega_{N_{sub}})$ % decomposition of spatial domain and

Define ad_i % number of adjacent subdomains (module 1, section 3.3).

Define Γ_{ij} % interfaces of domain Ω_i % (module 2, section 3.3).

$\Delta \leftarrow (\Delta_1, \Delta_2, \ldots, \Delta_{N_t})$% decomposition of the spatial domain (module 3, section 3.3)

For $i = 1,\ldots,N_{sub}$

  For $k = 1,\ldots,N_t$

    $u_{i,k}^0 \leftarrow u_{i,k}^M$

  EndFor

End of DD step

$n \leftarrow 0$

While *convergence* **do** % the outer loop (module 4, section 3.3).

For $k = 1, \ldots, N_t$

$u^n_{i,0} \leftarrow u^M_{i,0}$

% restriction procedure

Call $\mathcal{R}^{\Omega_i \times \Delta_k}$ (in:$M, \mathbf{R}, \mathbf{B}, G, y, u^M, u_0$; out:$M_{i,k+1}, \mathbf{R}_{i,k+1}, \mathbf{B}_i, \mathbf{B}_{i,1}$, $\ldots, \mathbf{B}_{i,ad_i}, G_{i,k+1}, y^k_i, u^M_{i,k}, u_{i,0}$)

$u^n_{i,0} \leftarrow u_{i,0}$

Receive *interfaces conditions from adjacent subdomains*

Compute $b_{i,k}$ % compute the vector accounting of information received from adjacent subdomains.

Send *interfaces conditions to adjacent subdomains*

% procedure solving and updating the solution of local models (modules 5-8, section 3.3).

Call Predictor-Corrector (in:$as_i, u^n_{i,k-1}, b_{i,k}, M_{i,k}, M_{j,k}, y_{i,k}, \mathbf{B}_i, \mathbf{R}_{i,k}, G_{i,k}$; out:$u^{n+1}_{i,k}$)

Endfor

$n \leftarrow n + 1$

EndWhile

$\bar{n} \leftarrow n$

EndFor

% gather of local solutions (module 9, section 3.3).

Gather of $u_{i,k}$: $\widetilde{\mathbf{u}}^{\mathbf{DD-DA}} = \{u^{\bar{n}}_{i,k}\}_{i=1,\ldots,N_{sub},k=1,\ldots,N_t}$

EndProcedure

Table 3.2: Predictor–Corrector procedure in $\Omega_i \times \Delta_k$.

Procedure Predictor–Corrector (in:$ad_i, u^n_{i,k-1}, b_{i,k}, M_{i,k}, M_{j,k}, y_{i,k}, \mathbf{B}_i, \mathbf{R}_{i,k}, G_{i,k}$; out:$u^{n+1}_{i,k}$)

% Procedure solving and updating the solution of local models (modules 5–8, section 3.3).

Compute $u^{M_{i,k},n+1}_{i,k} \leftarrow M_{i,k}u^n_{i,k-1} + b^n_{i,k}$

% procedure solving local 4DVAR DA problems (module 7, section 3.3).

Call ASM–Procedure(in:$ad_i, u^n_{i,k}, M_{i,k}, M_{j,k}, y_{i,k}, \mathbf{B}_i, \mathbf{R}_{i,k}, G_{i,k}$; out:$\mathbf{w}^{\bar{r}}_{i,k}$)

Define $\mathbf{w}^n_i \leftarrow \mathbf{w}^{\bar{r}}_i$

Compute $u^{n+1}_{i,k} \leftarrow u^{M_{i,k},n+1}_{i,k} + \mathbf{V}_i\mathbf{w}^n_{i,k}$

EndProcedure

Table 3.3: ASM algorithm in $\Omega_i \times \Delta_k$ with $ad_i \in \mathbb{N}$ adjacent subdomains.

Procedure ASM–Procedure(in:$ad_i, u^n_{i,k}, M_{i,k}, M_{j,k}, y_{i,k}, \mathbf{B}_i, \mathbf{R}_{i,k}, G_{i,k}$; out:$\mathbf{w}^{\bar{r}}_{i,k}$)

% Procedure for solving local 4DVAR DA problems (module 7, section 3.3).

$r \leftarrow 0$

Factorization $\mathbf{V}_i = chol(\mathbf{B}_i)$ % Cholesky factor of $\mathbf{B}_i$

Initialize $\mathbf{w}^r_{i,k}$, $\mathbf{w}^r_{j,k}$

While *convergence* **do** % the inner loop that solve local 4DVAR DA problems (module 7, section 3.3).

Compute $\mathbf{d}_i \leftarrow y_{i,k} - G_{i,k}u^M_{i,k}$

Compute $A_{i,k} \leftarrow \mathbf{V}^T_i(G_{i,k})^T\mathbf{R}^{-1}_{i,k}\mathbf{V}_iG_{i,k} + I_i + n_i \cdot \mathbf{B}_{ij}$

Compute $\mathbf{c}_i \leftarrow \mathbf{V}^T_i(G^k_i)^T(\mathbf{R}_{i,k})^{-1}\mathbf{d}_i$

Compute $\mathbf{c}_i - \sum_{j=1}^{ad_i}\mathbf{B}_{ij}\mathbf{w}^r_{i_j,k}$

% CG method for solving the linear system $A_{i,k}\mathbf{w}^{r+1}_{i,k} = c_i - \sum_{j=1}^{ad_i}\mathbf{B}_{ij}\mathbf{w}^r_{i_j,k}$

Call CG(in: $A_{i,k}, c_i - \sum_{j=1}^{n_i}\mathbf{B}_{ij}\mathbf{w}^r_{i_j,k}; out : \mathbf{w}^{r+1}_{i,k}$)

$r \leftarrow r + 1$

EndWhile

Define $\bar{r} \leftarrow r$

EndProcedure

Hence, convergence of $(\mathbf{w}_{i,k}^r)_{r\in\mathbb{N}_0}$ and Theorem 2 justifies *convergence* conditions in (3.54) and (3.55) of loops over index n and r.

Note that, for each value of n, local 4DVAR DA problems in $\Omega_i \times \Delta_{k+1}$ are independent on each other, hence, they can be solved in parallel.

3.3.2 Sensitivity analysis

In the following, we analyze error propagation. To this aim, we introduce the following Lemma [100]. We pose $\|\cdot\| = \|\cdot\|_2$[1].

Lemma 1. *Let $N_t \in \mathbb{N}$ and $R > 0$, $H \geq 0$. If for $k = 1, \ldots, N_t$ we have that:*

$$|E_k| \leq (1+R)|E_{k-1}| + H \quad \textit{for } k = 1, 2, \ldots, N_t$$

then it holds that

$$|E_k| \leq e^{N_t R}|E_1| + \frac{e^{N_t R} - 1}{R} H \quad \textit{for } k = 1, 2, \ldots, N_t.$$

We consider round off errors propagation.

Definition 7. *Let $u_{i,k}^{n+1}$ and $\tilde{u}_{i,k}^{n+1}$ be respectively the numerical solution at iteration $(n+1)$ in (3.43) and the corresponding floating point representation. Fixed iteration n, for $i = 1, \ldots, N_{sub}, k = 1, \ldots, N_t$ let*

$$R_{i,k}^{n+1} := u_{i,k}^{n+1} - \tilde{u}_{i,k}^{n+1} \tag{3.56}$$

be global round-off error in $\Omega_i \times \Delta_k$.

[1] We refer to $\|z_{i,k}\|_2 = \|\{z_{i,k}(\bar{i},\bar{k})\}_{\bar{i}\in I_i, \bar{k}\in K_k}\|_2$ where $z \in R^{N_p \times N}$ and $z_{i,k} := z/(\Omega_i \times \Delta_k)$ and I_i and K_k are defined in (3.7) and (3.7), respectively.

From (3.31) the numerical solution at iteration $(n+1)$ can be written as follows

$$\begin{aligned} u_{i,k}^{n+1} &= (M_{i,k}u_{k-1}^{n+1} + b_{i,k}^{n+1}) + [u_{i,k}^{ASM,n} - (M_{i,k}u_{i,k-1}^{n} + b_{i,k}^{n})] \\ &= M_{i,k}u_{i,k-1}^{n+1} + \delta(u_{i,k}^{n}) \end{aligned}$$

and consequently the corresponding floating point representation is

$$\tilde{u}_{i,k}^{n+1} = M_{i,k}\tilde{u}_{i,k-1}^{n+1} + \delta(\tilde{u}_{i,k}^{n}) + \rho_k^{n+1}$$

where $\delta(u_{i,k}^n) := (u_{i,k}^{ASM,n} - M_{i,k}u_{i,k-1}^n) + (b_{i,k}^{n+1} - b_{i,k}^n)$ and ρ_k^{n+1} is *local round-off error*.

Fixed $(n+1)$, we have that

$$\begin{aligned} \|R_{i,k}^{n+1}\| &= \|u_{i,k}^{n+1} - \tilde{u}_{i,k}^{n+1}\| = \|M_{i,k}u_{i,k-1}^{n+1} + \delta(u_{i,k}^n) - M_{i,k}\tilde{u}_{i,k-1}^{n+1} - \delta(\tilde{u}_{i,k}^n) - \rho_k^{n+1}\| \\ &\leq \|M_{i,k}u_{i,k-1}^{n+1} - M_{i,k}\tilde{u}_{i,k-1}^{n+1}\| + \|\delta(u_{i,k}^n) - \delta(\tilde{u}_{i,k}^n)\| + |\rho_k^{n+1}| \\ &\leq \|M_{i,k}\|\|u_{k-1}^{n+1} - \tilde{u}_{k-1}^{n+1}\| + \|u_k^{ASM,n} - \tilde{u}_k^{ASM,n}\| \\ &+\|M_{i,k}u_{i,k-1}^n + b_{i,k}^n - (M_{i,k}\tilde{u}_{i,k-1}^n + \tilde{b}_{i,k}^n)\| + \|b_k^{n+1} - \tilde{b}_{i,k}^{n+1}\| \\ &+\|b_{i,k}^n - \tilde{b}_{i,k}^n\| + |\rho_k^{n+1}| \end{aligned}$$

from (3.31) we have

$$\|b_{i,k}^{n+1} - \tilde{b}_{i,k}^{n+1}\| = \|M_{i,k}u_{i,k-1}^{n+1} - M_{i,k}\tilde{u}_{i,k-1}^{n+1}\|$$

and as in [8] we let the condition index $\mu(M_{i,k}) \geq \|M_{i,k}\|$ then

$$\begin{aligned} \|R_{i,k}^{n+1}\| &\leq \mu(M_i^k)\|u_{k-1}^{n+1} - \tilde{u}_{k-1}^{n+1}\| + \|u_k^{ASM,n} - \tilde{u}_k^{ASM,n}\| + \|u_{i,k}^{M_{i,k},n} - \tilde{u}_{i,k}^{M_{i,k},n}\| \\ &+\mu(M_{i,k})\|u_{k-1}^{n+1} - \tilde{u}_{k-1}^{n+1}\| + \mu(M_{i,k})\|u_{i,k-1}^n - \tilde{u}_{i,k-1}^n\| + |\rho_i^k| \end{aligned}$$

and in compact form

$$\|R_{i,k}^{n+1}\| \leq 2\mu(M_{i,k})\|R_{i,k-1}^{n+1}\| + \|R_{i,k}^{ASM,n}\| + \|R_{i,k}^{M,n}\| + \mu(M_{i,k})\|R_{i,k-1}^n\| + |\rho_i^k|$$

where

$$R_{i,k}^{ASM,n} := u_{i,k}^{ASM,n} - \tilde{u}_{i,k}^{ASM,n} \quad R_{i,k}^{M_{i,k},n} := u_{i,k}^{M_{i,k},n} - \tilde{u}_{i,k}^{M_{i,k},n}. \tag{3.57}$$

From Lemma 1, with $R = 2\mu(M_{i,k}) - 1$ and $H = R_{i,k}^{ASM,n} + R_{i,k}^{M_{i,k},n+1} + \mu(M_{i,k})R_{i,k-1}^{n} + \rho_i^k$ it follows that:

$$\|R_{i,k}^{n+1}\| \leq e^{N_t R}\|R_{i,1}^{n+1}\| + \frac{e^{N_t R} - 1}{R_{\mu(M_{i,k})}} \left[\|R_{i,k}^{ASM,n}\| + \|R_{i,k}^{M_{i,k},n}\| + \mu(M_{i,k})\|R_{k-1}^{n}\| + \rho_i^k\right].$$

then, it is

$$\|R_{i,k}^{n+1}\| \leq e^{N_t R}\|R_{i,1}^{n+1}\| + \frac{e^{N_t R} - 1}{R} \left[\|R_{i,k}^{ASM,n}\| + \|R_{i,k}^{M_{i,k},n}\| + \|R_{i,k-1}^{n}\|\right] + \frac{e^{N_t R} - 1}{R}\rho_i^k. \tag{3.58}$$

The upper bound in (3.58) is made of three terms: the first one represents the propagation of the error introduced on the first time interval, the second represents the propagation of the error introduced at previous step and the last term depends on the local round-off error. In particular, we note that, as expected, round-off error propagation grows with N_t, i.e. the number of subdomains of Δ.

3.3.3 Consistency, convergence and stability

We first prove convergence results (see Figure 3.1). We pose $\|\cdot\| = \|\cdot\|_2$[2].

Theorem 2. *(Convergence of outer loop) DD-4DVAR is a convergent method i.e.*

$$lim_{n\to\infty}\|\tilde{\mathbf{u}}^{\mathbf{DD-DA,n+1}} - \tilde{\mathbf{u}}^{\mathbf{DD-DA,n}}\| = 0 \tag{3.59}$$

where $\tilde{\mathbf{u}}^{\mathbf{DD-DA,n}}$ *is defined in (3.44).*

Proof. From (3.46) and (3.45), it is

$$\mathbf{J}(\mathbf{u}^{\mathbf{DA}}) = J(\widetilde{\mathbf{u}}^{\mathbf{DD-DA}}) = \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\bar{\mathbf{n}}}) \tag{3.60}$$

[2]We refer to $\|z_{i,k}\|_2 = \|\{z_{i,k}(\bar{i},\bar{k})\}_{\bar{i}\in I_i,\bar{k}\in K_k}\|_2$ where $z \in R^{N_p\times N}$ and $z_{i,k} := z/(\Omega_i \times \Delta_k)$ and I_i and K_k are defined in (3.7) and (3.7), respectively.

where $\bar{n}$ is the latest iteration of the outer loop (see Figure 3.1). By minimum properties of $\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\bar{\mathbf{n}}}$, we get

$$\mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\bar{\mathbf{n}}}) \leq \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\bar{\mathbf{n}}-\mathbf{1}}). \tag{3.61}$$

From (3.61), by recurring we obtain

$$0 \leq \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\bar{\mathbf{n}}}) \leq \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}\bar{-}\mathbf{1}}) \leq \ldots \leq \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{1}}) \leq \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{0}})$$

then $\{\mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}})\}_{\mathbf{n}\in\mathbb{N}}$ is monotonically decreasing and bounded from below by zero, then it is convergent i.e.

$$lim_{n\to\infty}\left[\mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n+1}}) - \mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}})\right] = 0. \tag{3.62}$$

Concerning iteration n of the outer loop, we get

$$\mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}}) = \|\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}} - u^M\|^2_{\mathbf{B}^{-1}} + \|G\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}} - v\|^2_{\mathbf{R}^{-1}} \geq \|\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}} - u^M\|_{\mathbf{B}^{-1}} \tag{3.63}$$

from the triangle inequality we get

$$\mathbf{J}(\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}}) \geq |\|\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}}\|_{\mathbf{B}^{-1}} - \|u^M\|_{\mathbf{B}^{-1}}| \tag{3.64}$$

where u^M is defined in (2.13) and does not depend on n then $\{\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}}\}_{\mathbf{n}\in\mathbb{N}}$ is convergent respect to $\|\cdot\|_{\mathbf{B}^{-1}}$, consequently respect to $\|\cdot\|$. Then we get the thesis in (3.59). □

The convergence of ASM is proved in [44].

We also analyse the convergence of DD-4DVAR modules in terms of local truncation errors $E^{M_{i,k}}_{i,k}$, $E^{ASM}_{i,k}$, $E_{i,k}$ and E_g reported in Figure 3.2. Similar to the differential equations case [29], local truncation errors $E^{M_{i,k}}_{i,k}$, $E^{ASM}_{i,k}$, $E_{i,k}$ in $\Omega_i \times \Delta_k$ and E_g in $\Omega \times \Delta$ are defined as the remainder after solutions $u^{\mathcal{M}}$ of (2.1) and $\mathbf{u}^{\mathbf{DA}}$ of 4D-DA problem (2.9) are substituted into the discrete models. To this aim, we give the following definitions.

Definition 8. *We define*

$$u^{\mathcal{M}\to M} := M \cdot \{u^{\mathcal{M}}(x_i, t_k)\}_{\tilde{i}\in I;\ \tilde{k}\in K} \tag{3.65}$$

approximation of $u^{\mathcal{M}}$ in (2.1) in $\Omega \times \Delta$ obtained by replacing $u^{\mathcal{M}}$ evaluated in $\Omega_I \times \Delta_K$ defined in Definition 1, into M defined in (2.12).

Definition 9. *(Local truncation errors in $\Omega_i \times \Delta_k$) $\forall (i,k), i = 1, \ldots, N_{sub}$ and $k = 1, \ldots, N_t$, at at iteration $\bar{n}$ we define*

$$E_{i,k}^{M_{i,k},\bar{n}} := \left\| u^{\mathcal{M}\to M}/(\Omega_i \times \Delta_k) - u_{i,k}^{M_{i,k},\bar{n}} \right\| \tag{3.66}$$

as local truncation error of the numerical model restricted to $\Omega_i \times \Delta_k$;

$$E_{i,k}^{ASM,\bar{n}} := \left\| \mathbf{u^{DA}}/(\Omega_i \times \Delta_k) - u_{i,k}^{ASM,\bar{n}} \right\| \tag{3.67}$$

as local truncation error of ASM restricted to $\Omega_i \times \Delta_k$;

$$E_{i,k}^{\bar{n}} := \left\| \mathbf{u^{DA}}/(\Omega_i \times \Delta_k) - u_{i,k}^{\bar{n}} \right\| \tag{3.68}$$

as local truncation error of DD–4DVAR method restricted to $\Omega_i \times \Delta_k$;

$$E_g := \left\| \mathbf{u^{DA}} - \widetilde{\mathbf{u}}^{\mathbf{DD-DA}} \right\| \tag{3.69}$$

as local truncation error of DD–4DVAR method in $\Omega \times \Delta$.

DD-4DVAR method needs few iterations of DD–4DVAR outer loop on n to update DD-4DVAR approximation in (3.43). Consequently, in the next, we neglect the dependency on $\bar{n}$ of $u_{i,k}^{M_{i,k},\bar{n}}$, $u_{i,k}^{ASM,\bar{n}}$ and $u_{i,k}^{n}$ defined respectively in (3.31), (3.35) and (3.43) and $E_{i,k}^{M_{i,k},\bar{n}}$, $E_{i,k}^{ASM,\bar{n}}$ and $E_{i,k}^{\bar{n}}$ in $\Omega_i \times \Delta_k$ defined respectively in (3.66), (3.67) and (3.68).

For (i,k), $i = 1, \ldots, N_{sub}$ and $k = 1, \ldots, N_t$, we pose

$$\widetilde{\mathbf{u}}_{\mathbf{i,k}} := u_{i,k}^{\bar{n}}. \tag{3.70}$$

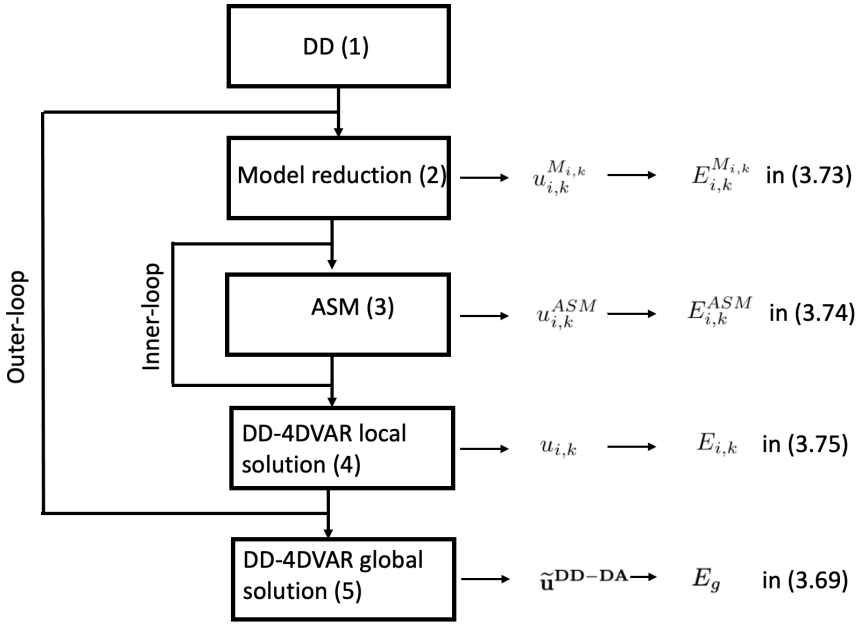


Figure 3.2: Local truncation errors related to each module of DD–4DVAR method.

From (3.70), the DD–4DVAR approximation in $\Omega \times \Delta$ defined in (3.45) becomes

$$\widetilde{\mathbf{u}}^{\mathbf{DD-DA}} = \sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} \widetilde{\mathbf{u}}_{\mathbf{i,k}}^{EO}. \tag{3.71}$$

For $i = 1, \ldots, N_{sub}$ and $k = 1, \ldots, N_t$, we pose

$$\begin{aligned} u_{i,k}^{ASM} &= u_{i,k}^{ASM,\bar{n}}; \\ u_{i,k}^{M_{i,k}} &= u_{i,k}^{M_{i,k},\bar{n}}; \\ u_{i,k} &= u_{i,k}^{\bar{n}}. \end{aligned} \tag{3.72}$$

Consequently, from (3.72) we pose

$$E_{i,k}^{M_{i,k}} = E_{i,k}^{M_{i,k},\bar{n}}, \tag{3.73}$$

as local model truncation error in $\Omega_i \times \Delta_k$,

$$E_{i,k}^{ASM} = E_{i,k}^{ASM,\bar{n}}, \tag{3.74}$$

as local ASM truncation error in $\Omega_i \times \Delta_k$ in (3.67),

$$E_{i,k} = E_{i,k}^{\bar{n}}. \tag{3.75}$$

as local truncation error $\Omega_i \times \Delta_k$ in (3.68).

We introduce the definition of consistency of DD–4DVAR method.

Definition 10. *(Consistency of DD–4DVAR method) Let E_g be local truncation error of DD–4DVAR in $\Omega \times \Delta$ defined in (3.69). The DD–4DVAR method is said to be consistent, if $E_g \to 0$ as $\Delta x,\ \Delta t \to 0$, where*

- $\Delta x := max_{i=1,\ldots,N_{sub}}(\Delta x)_i$, *where $\{(\Delta x)_i\}_{i=1,\ldots,N_{sub}}$ are spatial step sizes of $M_{i,k}$ defined in (3.32);*
- $\Delta t := max_{k=1,\ldots,N_t}(\Delta t)_k$, *where $\{(\Delta t)_k\}_{k=1,\ldots,N_t}$ are temporal step sizes of $M_{i,k}$ defined in (3.32).*

In order to prove the consistency of DD–4DVAR method, we perform the analysis of local truncation errors $E_{i,k}^{M_{i,k}}$, $E_{i,k}^{ASM}$, $E_{i,k}$ and E_g defined respectively in (3.73), (3.74), (3.75) and (3.69) related to Module Reduction, ASM, DD–4DVAR local solution and DD–4DVAR solution modules as described in Figure 3.3.

Assumption 1. *(Local truncation error of Model Reduction in $\Omega_i \times \Delta_k$) Let*

$$E_{i,k}^{M_{i,k}} = \mathcal{O}((\Delta x)_i^p + (\Delta t)_k^q), \quad \forall (i,k) \in \{1,\dots,N_{sub}\} \times \{1,\dots,N_t\} \tag{3.76}$$

be local truncation error defined in (3.73) where $(\Delta x)_i$ and $(\Delta t)_k$ are spatial and temporal step sizes of $M_{i,k}$ defined in (3.32) and p and q are the order of convergence in space and in time.

In the experimental analysis (see section 5), in order to discretize the SWEs model we consider Lax–Wendroff scheme [81]. Hence, in that case, $p = q = 2$.

Lemma 2. *(Local truncation error of ASM in $\Omega_i \times \Delta_k$) Let us consider*

- *σ_0^2: be observational error variance;*
- *$\mathbf{B}_i = \mathbf{V}_i \mathbf{V}_i^T$: be restriction of covariance matrices of the error on background to Ω_i;*
- *$G_{i,k}$: be restriction to $\Omega_i \times \Delta_k$ of matrix G defined in (2.8);*
- *ad_i: be number of subdomains adjacent to Ω_i, defined in (3.2);*
- *$\mathbf{B}_{ij} = \mathbf{V}_{ij} \mathbf{V}_{ij}^T$: be restriction to Ω_{ij} of variance matrix of the error on background;*
- *$M_{i,k}$: be matrix defined in (3.33), restricted to $\Omega_i \times \Delta_k$;*
- *$\mu(\mathbf{V}_i)$, $\mu(G_{i,k})$, $\mu(M_{i,k})$ and $\mu(\mathbf{V}_{ij})$: be condition number of matrices $\mathbf{V}_i, G_{i,k}$, $M_{i,k}$ and $\mathbf{V}_{ij}$, respectively.*

Then, $\forall i = 1, \ldots, N_{sub}; k = 1, \ldots, N_t$*, it holds that:*

$$E_{i,k}^{ASM} \leq \mu_{i,k}^{DD-DA} \cdot E_{i,1}^{ASM} \tag{3.77}$$

where

$$\mu_{i,k}^{DD-DA} := \left[1 + \frac{1}{\sigma_0^2}\mu^2(\mathbf{V}_i)\mu^2(G_{i,k}) + ad_i \cdot \mu^2(\mathbf{V}_{ij})\right] \mu(M_{i,k}). \tag{3.78}$$

Proof. By applying error approximation in [8], we have

$$\|\mathbf{u}^{DA}/(\Omega_i \times \Delta_k) - u_{i,k}^{ASM}\| \leq \mu(\mathbf{J}_{i,k})\mu(M_{i,k}) \cdot \|\mathbf{u}^{\mathbf{DA}}/(\Omega_i \times \Delta_1) - u_{i,1}^{ASM}\| \tag{3.79}$$

where $\|\mathbf{u}^{\mathbf{DA}}/(\Omega_i \times \Delta_1) - u_{i,1}^{ASM}\|$ is the error in $\Omega_i \times \Delta_1$. As proved in [8], it is

$$\mu(\mathbf{J}_{i,k}) = \mu(A_{i,k}) \tag{3.80}$$

where $\mathbf{J}_{i,k}$ and $A_{i,k}$ are respectively defined in (3.36) and (3.40) and form the triangle inequality, it is

$$\mu(A_{i,k}) \leq 1 + \frac{1}{\sigma_0^2}\mu^2(\mathbf{V}_i)\mu^2(G_{i,k}) + ad_i \cdot \mu(\mathbf{B}_{ij}) \tag{3.81}$$

$$\leq 1 + \frac{1}{\sigma_0^2}\mu^2(\mathbf{V}_i)\mu^2(G_{i,k}) + ad_i \cdot \mu^2(\mathbf{V}_{ij}), \tag{3.82}$$

From (3.79) and (3.81), the (3.77) follows. □

Lemma 3. *(Local truncation error of DD-4DVAR method in* $\Omega_i \times \Delta_k$*)* $\forall (i,k), i = 1, \ldots, N_{sub}; k = 1, \ldots, N_t$*, it holds that:*

$$E_{i,k} \leq \mu_{i,k}^{DD-DA} \cdot E_{i,1}^{ASM} + 2 \cdot E_{i,k}^{M_{i,k}} \tag{3.83}$$

where $\mu_{i,k}^{DD-DA}$ *is defined in (3.78).*

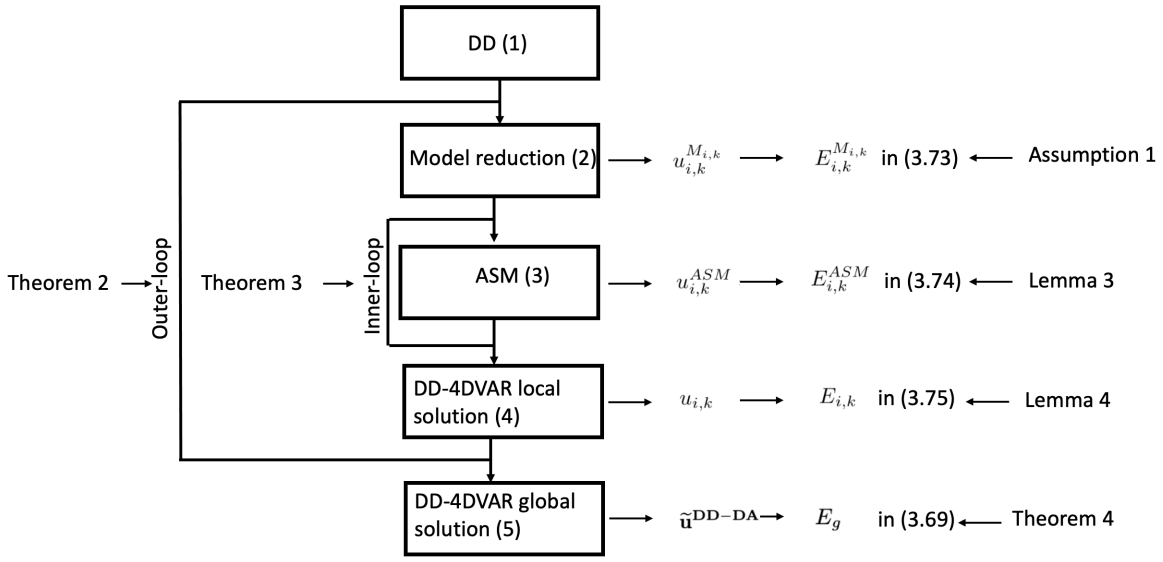


Figure 3.3: Assumption, Lemmas, Theorems related to each module of DD-4DVAR method.

Proof. From (3.72) and (3.43), $E_{i,k}$ defined in (3.75) can be rewritten as follows

$$\begin{aligned} E_{i,k} &:= \|\mathbf{u^{DA}}/(\Omega_i \times \Delta_k) - u_{i,k}\| \\ &= \|\mathbf{u^{DA}}/(\Omega_i \times \Delta_k) - u_{i,k}^{M_{i,k},\bar{n}} - (u_{i,k}^{ASM,\bar{n}-1} - u_{i,k}^{M_{i,k},\bar{n}-1})\| \end{aligned}$$

from the triangle inequality

$$E_{i,k} \leq \|\mathbf{u^{DA}}/(\Omega_i \times \Delta_k) - u_{i,k}^{ASM,\bar{n}}\| + \|u_{i,k}^{M_{i,k},\bar{n}-1} - u_{i,k}^{M_{i,k},\bar{n}}\| \tag{3.84}$$

as consequence of Lemma 2 and (3.74) we have

$$E_{i,k} \leq \mu_{i,k}^{DD-DA} \cdot E_{i,1}^{ASM} + \|u_{i,k}^{M_{i,k},\bar{n}-1} - u_{i,k}^{M_{i,k},\bar{n}}\| \tag{3.85}$$

where $E_{i,1}^{ASM}$ is defined in (3.74) and $\mu_{i,k}^{DD-DA}$ is defined in (3.78).
In particular, by adding and subtracting $u^{\mathcal{M}\to M}/(\Omega_i \times \Delta_k)$ in $\|u_{i,k}^{M_{i,k},\bar{n}-1} - u_{i,k}^{M_{i,k},\bar{n}}\|$ we get:

$$\begin{aligned} \|u_{i,k}^{M_{i,k},\bar{n}-1} - u_{i,k}^{M_{i,k},\bar{n}}\| = \|&(u_{i,k}^{M_{i,k},\bar{n}-1} - u^{\mathcal{M}\to M}/(\Omega_i \times \Delta_k)) \\ &+ (u^{\mathcal{M}\to M}/(\Omega_i \times \Delta_k) - u_{i,k}^{M_{i,k},\bar{n}})\| \end{aligned}$$

and from the triangle inequality

$$\begin{aligned}\|u_{i,k}^{M_{i,k},\bar{n}-1} - u_{i,k}^{M_{i,k},\bar{n}}\| &\leq \|u_{i,k}^{M_{i,k},\bar{n}-1} - u^{\mathcal{M}\to M}/(\Omega_i \times \Delta_k)\| \\ &+ \|u^{\mathcal{M}\to M}/(\Omega_i \times \Delta_k) - u_{i,k}^{M_{i,k},\bar{n}}\| \\ &= E_{i,k}^{M_{i,k},\bar{n}} + E_{i,k}^{M_{i,k},\bar{n}-1}\end{aligned} \tag{3.86}$$

From Theorem 2, we get that $\{\widetilde{\mathbf{u}}^{\mathbf{DD-DA},\mathbf{n}}\}_{n\in\mathbb{N}}$ is a convergent sequence, then it is a Cauchy sequence. From (3.44), we get that $\{u_{i,k}^{M_{i,k},n}\}_{n\in\mathbb{N}}$ is also a Cauchy sequence, i.e.

$$\forall\, \epsilon > 0\ \exists\, \mathbf{N} > 0\ :\ \ \|u_{i,k}^{M_{i,k},n} - u_{i,k}^{M_{i,k},m}\| \leq \epsilon \quad \forall n, m > \mathbf{N}. \tag{3.87}$$

In particular, (3.87) is true for $n = \bar{n}$ and $m = \bar{n} - 1$, assuming that $\bar{n}$ is large enough. Consequently, we can neglect the dependency on outer loop in $E_{i,k}^{M_{i,k},\bar{n}}$ and $E_{i,k}^{M_{i,k},\bar{n}-1}$ in (3.86) i.e.

$$\|u_{i,k}^{M_{i,k},\bar{n}-1} - u_{i,k}^{M_{i,k},\bar{n}}\| \leq 2 \cdot E_{i,k}^{M_{i,k}} \tag{3.88}$$

where $E_{i,k}^{M_{i,k}}$ is defined in (3.73). From (3.85), (3.86) and (3.88) we get the thesis in (3.83). □

Lemma 4. *Under Assumption (1),if e_0, the error on initial condition of M in (2.12) is equal to zero i.e.*

$$e_0 = 0 \tag{3.89}$$

then

$$E_{i,k} \leq c \cdot ((\Delta x)_i^p + \cdot(\Delta t)_k^q) \tag{3.90}$$

where $E_{i,k}$ is the local truncation error defined in (3.75) and c is positive constant independent on DD.

Proof. By applying the Lemma 3.74 to local ASM truncation error $E_{i,1}^{ASM}$ in $\Omega_i \times \Delta_1$, we get

$$E_{i,1}^{ASM} \leq \mu_{i,1}^{DD-DA} \cdot e_0/\Omega_i. \tag{3.91}$$

where $\mu_{i,1}^{DD-DA}$ is defined in (3.78) and e_0/Ω_i is the restriction of e_0 to Ω_i. From the assumptions it is $e_0/\Omega_i = 0$ and by replacing it in (3.91) we have

$$E_{i,1}^{ASM} = 0 \tag{3.92}$$

consequently local ASM truncation error in $\Omega_i \times \Delta_k$ is

$$E_{i,k}^{ASM} = 0. \tag{3.93}$$

From (3.93), (3.76) and (3.83) we get the thesis in (3.90). □

Now, we are able to prove the following result on global truncation error.

Theorem 3. *(Local truncation error in $\Omega \times \Delta$) Let assume the assumption of Lemma 4 in (3.89). The local truncation error in $\Omega \times \Delta$ is*

$$E_g \leq c \cdot (N_{sub} \cdot N_t) \cdot [(\Delta x)^p + (\Delta t)^q], \tag{3.94}$$

where

- $\Delta x := max_{i=1,\dots,N_{sub}}(\Delta x)_i$*: where* $\{(\Delta x)_i\}_{i=1,\dots,N_{sub}}$ *are spatial step sizes of* $M_{i,k}$ *defined in (3.32);*
- $\Delta t := max_{k=1,\dots,N_t}(\Delta t)_k$*: where* $\{(\Delta t)_k\}_{k=1,\dots,N_t}$ *are temporal step sizes of* $M_{i,k}$ *defined in (3.32);*
- $c > 0$*: is positive constant independent on DD.*

Proof. From (3.71) E_g defined in (3.69) can be rewritten as follows

$$E_g := \left\|\mathbf{u^{DA}} - \tilde{\mathbf{u}}^{\mathbf{DD-DA}}\right\| = \left\|\mathbf{u^{DA}} - \sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} \widetilde{\mathbf{u}}_{\mathbf{i,k}}\right\|$$

by applying the restriction and extension operator (Definitions 5 and 6) to $\mathbf{u^{DA}}$, we get

$$E_g = \left\| \sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} \left[(\mathbf{u^{DA}}/(\Omega_i \times \Delta_k))^{EO} - \widetilde{\mathbf{u}}_{\mathbf{i,k}} \right] \right\|$$

from triangle inequality, it is

$$\begin{aligned} E_g &= \left\| \sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} \left[(\mathbf{u^{DA}}/(\Omega_i \times \Delta_k))^{EO} - \widetilde{\mathbf{u}}_{\mathbf{i,k}} \right] \right\| \\ &\leq \sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} \left\| (\mathbf{u}_{\mathbf{i,k}}^{\mathbf{DA}})^{EO} - \widetilde{\mathbf{u}}_{\mathbf{i,k}} \right\| = \sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} E_{i,k} \end{aligned} \tag{3.95}$$

where $E_{i,k}$ is defined in (3.75). From Lemma 4 we have

$$\begin{aligned} \sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} E_{i,k} &\leq c \cdot \sum_{i=1}^{N_{sub}} \sum_{k=1}^{N_t} ((\Delta x)_i^p + \cdot (\Delta t)_k^q) \\ &= c \cdot \left[N_t \cdot \sum_{i=1}^{N_{sub}} (\Delta x)_i^p + N_{sub} \sum_{k=1}^{N_t} (\Delta t)_k^q \right] \end{aligned} \tag{3.96}$$

consequently

$$E_g \leq c \cdot \left[N_t \cdot \sum_{i=1}^{N_{sub}} (\Delta x)_i^p + N_{sub} \sum_{k=1}^{N_t} (\Delta t)_k^q \right].$$

By defining

$$\Delta x := max_{i=1,\ldots,N_{sub}} (\Delta x)_i$$

$$\Delta t := max_{k=1,\ldots,N_t} (\Delta t)_k$$

we get

$$\begin{aligned} E_g &\leq c \cdot \left[N_t \cdot \sum_{i=1}^{N_{sub}} (\Delta x)_i^p + N_{sub} \sum_{k=1}^{N_t} (\Delta t)_k^q \right] \\ &\leq c \cdot \left[N_t \cdot \sum_{i=1}^{N_{sub}} (\Delta x)^p + N_{sub} \sum_{k=1}^{N_t} (\Delta t)^q \right] \\ &\leq c \cdot (N_{sub} \cdot N_t) \cdot [(\Delta x)^p + (\Delta t)^q]. \end{aligned}$$

Hence, the (3.94) is proved.

□

In Figure 3.4, we show the schematic description of results obtained from local truncation analysis.

Now, we prove the stability by studying the propagation of the error from each time interval to the next, assuming that the predictive model is stable.

Assumption 2. *(Stability of discrete model) The discrete model applied to $\mathcal{M}$ in (2.1) is stable i.e. $\exists D > 0$ such that*

$$\|u^M - v^M\| \leq D \cdot e_0, \tag{3.97}$$

where

- *u^M: is solution of M in (2.12);*
- *v^M: is solution of $\bar{M}$, where $\bar{M}$ is obtained by considering initial error e_0 on initial condition;*
- *e_0: is initial error on initial condition of M in (2.12).*

Definition 11. *(Propagation error from Δ_{k-1} to Δ_k) Let us consider*

- *$\tilde{\mathbf{v}}^{\mathbf{DD-DA}}$: be the solution in $\Omega \times \Delta$ computed by adding a perturbation e_k to initial condition of $P_{i,k}^{M_{i,k},n}$ defined in (3.31).*

We define

$$\bar{E}_k := \|\tilde{\mathbf{u}}^{\mathbf{DD-DA}}/\Delta_k - \tilde{\mathbf{v}}^{\mathbf{DD-DA}}/\Delta_k\| \tag{3.98}$$

the propagation error from Δ_{k-1} to Δ_k.

Theorem 4. *(Stability of DD–4DVAR) Under Assumption 2, if e_0, the error on initial condition of M in (2.12), is equal to zero i.e.*

$$e_0 = 0 \tag{3.99}$$

then, $\forall k = 1, \dots, N_t \; \exists \, C_k > 0$ *such that*

$$\bar{E}_k \leq C_k \cdot \bar{e}_k$$

where

- $\bar{E}_k$*: is the error propagated from* Δ_{k-1} *to* Δ_k *and defined in (3.98);*
- C_k*: is a constant depending on the model reduction and on the ASM;*
- $\bar{e}_k$*: perturbation on initial condition of* $P_{i,k}^{M_{i,k}}$ *defined in (3.31).*

Proof. To simplify the notations in the proof, we consider $\bar{e}_k = \bar{e}, \; \forall k = 1, \dots, N_t$. From (3.43), (3.72) and triangle inequality, we get

$$\begin{aligned} \bar{E}_k := \|\tilde{\mathbf{u}}^{\mathbf{DD-DA}}/\Delta_k - \tilde{\mathbf{v}}^{\mathbf{DD-DA}}/\Delta_k\| \leq & \|(u_{i,k}^{M_{i,k},\bar{n}})^{EO}/\Delta_k - (v_{i,k}^{M_{i,k},\bar{n}})^{EO}/\Delta_k\| \\ & + \|(u_{i,k}^{M_{i,k},\bar{n}-1})^{EO}/\Delta_k - (v_{i,k}^{M_{i,k},\bar{n}-1})^{EO}/\Delta_k\| \\ & + \|(u_{i,k}^{ASM,\bar{n}})^{EO}/\Delta_k - (v_{i,k}^{ASM,\bar{n}})^{EO}/\Delta_k\|. \end{aligned} \tag{3.100}$$

From (3.87) and (3.74), we can neglect the dependency on outer loop iteration $\bar{n}$ i.e.

$$\bar{E}_k \leq 2 \cdot \|(u_{i,k}^{M_{i,k}})^{EO}/\Delta_k - (v_{i,k}^{M_{i,k}})^{EO}/\Delta_k\| + \|(u_{i,k}^{ASM})^{EO}/\Delta_k - (v_{i,k}^{ASM})^{EO}/\Delta_k\|. \tag{3.101}$$

From Assumption 2, we get that $\exists \bar{D} > 0$ such that

$$\|(u_{i,k}^{M_{i,k}})^{EO}/\Delta_k - (v_{i,k}^{M_{i,k}})^{EO}/\Delta_k\| \leq \bar{D} \cdot e_0. \tag{3.102}$$

where e_0 is the error on initial condition of M in (2.12). By adding and subtracting $\mathbf{u}^{\mathbf{DA}}/\Delta_k$ to $\left[(u_{i,k}^{ASM})^{EO}/\Delta_k - (v_{i,k}^{ASM})^{EO}/\Delta_k\right]$ and from triangle inequality, it is

$$\begin{aligned} \|(u_{i,k}^{ASM})^{EO}/\Delta_k - (v_{i,k}^{ASM})^{EO}/\Delta_k\| \leq & \|\mathbf{u}^{\mathbf{DA}}/\Delta_k - (u_{i,k}^{ASM})^{EO}/\Delta_k\| \\ & + \|\mathbf{u}^{\mathbf{DA}}/\Delta_k - (v_{i,k}^{ASM})^{EO}/\Delta_k\|. \end{aligned} \tag{3.103}$$

From (3.103) and (3.77), we get

$$\|(u_{i,k}^{ASM})^{EO}/\Delta_k - (v_{i,k}^{ASM})^{EO}/\Delta_k\| \leq \mu_k^{DD-DA} \cdot E_1^{ASM} + \bar{\mu}_k^{DD-DA} \cdot \bar{E}_1^{ASM} \tag{3.104}$$

where

$$\begin{aligned} E_1^{ASM} &= \|\mathbf{u^{DA}}/\Delta_1 - (u_{i,1}^{ASM})^{EO}/\Delta_1\| \\ \bar{E}_1^{ASM} &= \|\mathbf{u^{DA}}/\Delta_1 - (v_{i,1}^{ASM})^{EO}/\Delta_1\|. \end{aligned} \tag{3.105}$$

and

$$\begin{aligned} \mu_k^{DD-DA} &:= \left[1 + \frac{1}{\sigma_0^2}\mu^2(\mathbf{V})\mu^2(G/\Delta_k)\right]\mu(M/\Delta_k) \\ \bar{\mu}_k^{DD-DA} &:= \left[1 + \frac{1}{\sigma_0^2}\mu^2(\mathbf{V})\mu^2(G/\Delta_k)\right]\mu(\bar{M}/\Delta_k) \end{aligned} \tag{3.106}$$

with σ_0 observational error variance, $\mathbf{B} = \mathbf{V}\mathbf{V}^{\mathbf{T}}$ covariance matrix of the error on background to Ω, G matrix defined in (2.8), M defines in (2.12) and $\bar{M}$ is discrete model obtained by considering initial error e_0 on initial condition of M. By applying (3.77) to E_1^{ASM} and $\bar{E}_1^{ASM}$ in (3.105), we get

$$\|(u_{i,k}^{ASM})^{EO}/\Delta_k - (v_{i,k}^{ASM})^{EO}/\Delta_k\| \leq \mu_k^{DD-DA} \cdot \mu_1^{DD-DA} e_0 + \bar{\mu}_k^{DD-DA} \cdot \bar{\mu}_k^{DD-DA} \bar{e}. \tag{3.107}$$

From (3.101), (3.102) and (3.104), it is

$$\bar{E}_k \leq 2 \cdot \bar{D} \cdot e_0 + \mu_k^{DD-DA} \cdot \mu_1^{DD-DA} e_0 + \bar{\mu}_k^{DD-DA} \cdot \bar{\mu}_1^{DD-DA} \bar{e}$$

and from hypothesis in (3.99), we get

$$\bar{E}_k \leq \bar{\mu}_k^{DD-DA} \cdot \bar{\mu}_1^{DD-DA} \bar{e} \tag{3.108}$$

Consequently, for $k = 1, \ldots, N_t$ we get that $\exists\, C_k > 0$ such that

$$\bar{E}_k \leq C_k \cdot \bar{e} \tag{3.109}$$

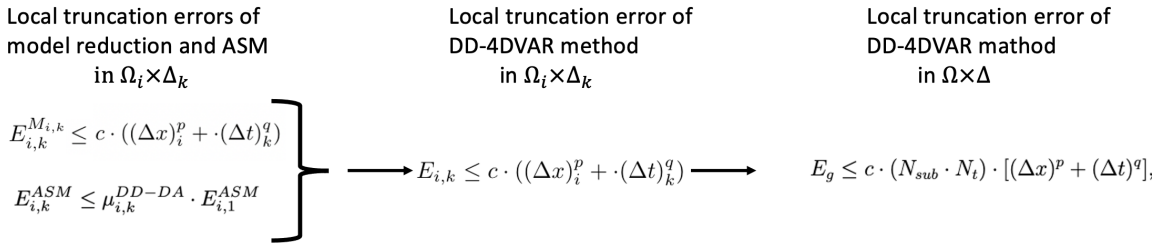


Figure 3.4: Schematic description of DD–4DVAR local truncation analysis.

where

$$C_k := \bar{\mu}_k^{DD-DA} \cdot \bar{\mu}_1^{DD-DA}. \tag{3.110}$$

The thesis is proved.

From Theorem 4, we get the stability of DD–4DVAR method.

□

Remark 2. *(**Condition number of DD–4DVAR method**) We note that in Theorem 4, to prove the stability of DD–4DVAR method, we study the propagation error according to the so-called forward error analysis. In particular, we get that the constant* C_k, $\forall k = 1, \ldots, N_t$ *in (3.110) depends on the quantity* $\bar{\mu}_k^{DD-DA}$ *defined in (3.106), which is in turn depends on condition numbers of* M *in (2.12),* G *in (2.8) and* $\mathbf{B} = \mathbf{V}\mathbf{V}^{\mathbf{T}}$ *covariance matrix of the error on background in* Ω*. As a consequence of the forward error analysis, we may say that the quantity* $\bar{\mu}_k^{DD-DA}$ *can be regarded as the condition number of DD–4DVAR method.*

3.4 DD-KF-CLS: Domain Decomposition of KF on CLS problem

The standard formulation of the KF becomes computationally intractable for solving large scale estimation problems due to matrix storage and inversion requirements. Then, several approaches have been proposed to reduce the overall time-to-solution. Among them, there are KF simplifications reducing computational complexity. Approximations are designed on the basis of a reduction in the order of the system model (usually the approximation is performed trough the use of the Empirical Orthogonal Functions (EOF)) [60, 111], or they are based on the Ensemble methods, where a prediction of the error at a future time is computed by integrating each ensemble state independently by the model. Integration is typically performed until observations are available. At this time, the information from the observations and the ensemble are combined by performing an analysis step based on KF [53]. However, the choice of the dimension of the reduced-state space or of the ensemble size giving an accurate approximation of KF still remains a delicate question [8].
Besides these variants, there are parallel approaches to KF algorithm.

- **KF-DD**: most approaches for delivering parallel solutions of simulations based on KF algorithm integrated with PDEs-based models essentially takes full advantage of existing parallel PDEs solvers; in particular, it is worth mention those based on DD [54]. Nevertheless, a common drawback of such parallel algorithms is their limited scalability, due to the fact that parallelism is achieved adapting the most computationally demanding components for parallel execution. Amdhal's law clearly applied in these situations because the computational cost of the components that are not parallelized provides a lower bound on the execution time of the parallel algorithm. On the contrary, the challenge is to consider parallelization from the beginning of the computational problem solving.

- **DD–KF**: involves decomposition of the whole computational problem, i.e. data and model decomposition, partitioning of the solution and a slight modification of KF algorithm allowing a correction at run-time of local solutions, through its exchange between adjacent subdomains. The analysis carried out in [48] highlighted the reliability of the proposed framework mainly in terms of the accuracy of local solutions with respect to the global solution. In order to analyze the benefits arising from decomposition we firstly present scalability analysis of DD–KF algorithm measuring the scale-up factor which expresses the performance gain of the algorithm in terms of reduction of its time complexity [39]. As the numerical model underlying variational Data Assimilation problems arising from the so-called discretize-then-optime approach is the well known Constrained Least Squares (CLS) defined in Appendix A.1, we will use CLS as a reference state estimation problem and we will employ DD–KF on CLS problems. In Figure 3.5 we give a schematic picture of these approaches showing how they arise. We see that KF-DD-CLS is obtained by following the path on the right, while on the left we get DD-KF-CLS.

3.4.1 KF-CLS: KF algorithm solving CLS problems

We prove that solution of CLS problem in (A.4), can be obtained by applying KF to S in (A.2). To this end, regarding (A.2) as an inverse ill posed problem [8, 50], we rewrite KF as a Variational problem, the so-called VAR-KF formulation, obtained minimizing the sum of the weighted Euclidean norm $||\cdot||_{B_l}$ of the model error $w_l = x_{l+1} - M_{l,l+1}x_l$ and the weighted Euclidean norm $||\cdot||_{R_{l+1}}$ of the observation error $v_{l+1} = y_{l+1} - H_{l+1}x_{l+1}$.

VAR-KF method: VAR-KF method computes, for $k = 0, 1, \ldots, r$:

$$\begin{aligned}\widehat{x}_{l+1} &= argmin_{x_{l+1}\in\mathbb{R}^{N_p}} J_{l+1}(x_{l+1}) \\ &= argmin_{x_{l+1}\in\mathbb{R}^{N_p}} \left\{ ||x_{l+1} - M_{l,l+1}\widehat{x}_l||^2_{B_l} + ||y_{l+1} - H_{l+1}x_{l+1}||^2_{R_{l+1}} \right\}.\end{aligned}$$

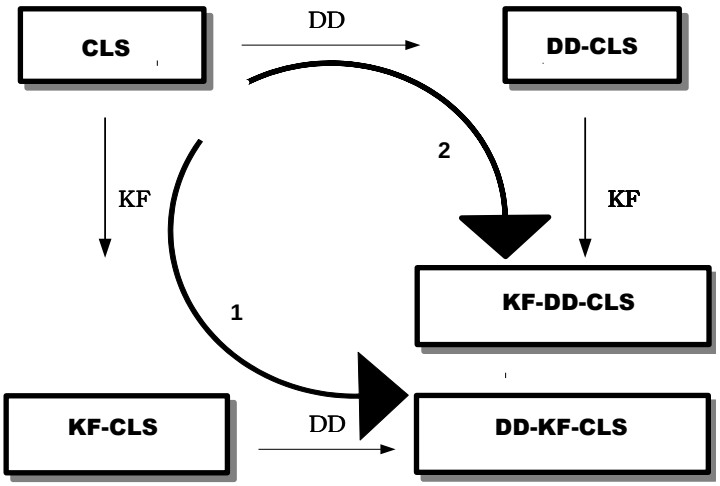


Figure 3.5: Schematic description of the framework we introduce for using DD in KF.

Remark 3. *Besides covariance matrices of the errors, main components of KF algorithm are dynamic model and observation mapping. These are two main components of any variational Data Assimilation operator and state estimation problem, too. In this regard, in the following, as proof of concept of the DD–KF framework, we start considering Constrained Least Squares (CLS) (cfr Appendix A.1) model as a prototype of a variational Data Assimilation model. CLS is obtained combining two overdetermined linear systems, representing the state and the observation mapping, respectively. Then, we introduce the VAR-KF method as reference data sampling method solving CLS model problem. VAR-KF will be decomposed by using the proposed DD framework. That said, anyone wants to apply DD framework in a real-world application, i.e. with a (PDE-based) model state and an observation mapping, once the dynamic (PDE-based) model state has been discretized, should rewrite the state estimation problem under consideration as a CLS model problem (cfr Appendix A.1) and then to apply KF-CLS algorithm (cfr Section 3.4). In other words, she/he should follow the discretize-then-optmize approach, com-*

mon to most Data Assimilation problems and state estimation problems, before employing the DD–KF framework (cfr Sections 3.4.1, 3.1, 3.4.2, 3.4.3).

By using linear algebra results we proved that [48]:

Proposition 2. *(KF-CLS) Let S be the overdetermined linear system in (A.2) with $A \in \mathbb{R}^{(m_0+m_1)\times N_p}$, $b \in \mathbb{R}^{m_0+m_1}$ defined in (A.3). Let us consider*

- *for $l = 0, 1$, $H_l \in \mathbb{R}^{m_l \times N_p}$ and $y_l \in \mathbb{R}^{m_l}$ with $m_0 > N_p$ and $m_1 > 0$;*
- *$M = (H_0^T R_0 H_0)^{-1} H_0^T R_0 \in \mathbb{R}^{N_p \times m_0}$ with $R_0 \in \mathbb{R}^{m_0 \times m_0}$, $R_1 \in \mathbb{R}^{m_1 \times m_1}$ and $R = diag(R_0, R_1) \in \mathbb{R}^{(m_0+m_1)\times(m_0+m_1)}$ weight matrices;*
- *$\widehat{x}_0 = My_0 \in \mathbb{R}^{N_p}$, least squares solution of system in (A.1);*
- *$\widehat{x} = (A^T RA)^{-1} A^T Rb \in \mathbb{R}^{N_p}$ least squares solution of S in (A.2).*

We pose:

$$\begin{aligned} M_{0,1} &\equiv I_{N_p,N_p} \in \mathbb{R}^{N_p \times N_p}, \\ Q_0 &\equiv O_{N_p,N_p} \in \mathbb{R}^{N_p \times N_p} \\ P_0 &\equiv (H_0^T R_0 H_0)^{-1} \in \mathbb{R}^{N_p \times N_p}; \end{aligned} \quad , \tag{3.111}$$

where I_{N_p,N_p} is the identity matrix and O_{N_p,N_p} is the null matrix, then by using KF algorithm 2.1.2, for $k = 0$ we obtain KF estimate $\widehat{x}_1$ in (2.20) such that

$$\widehat{x} \equiv \widehat{x}_1.$$

Remark 4. *By adding $(r+1) \cdot m$ equations, with $r \geq 0$, $m > 0$, to the system in (A.1) and posing, for $l = 0, 1, \ldots, r$,*

$$\begin{aligned} M_{l,l+1} &:= I_{N_p,N_p} \in \mathbb{R}^{N_p \times N_p}, \\ Q_l &:= O_{N_p,N_p} \in \mathbb{R}^{N_p \times N_p} \\ P_0 &:= (H_0^T R_0 H_0)^{-1} \in \mathbb{R}^{N_p \times N_p} \end{aligned} \quad ,$$

and $R \in \mathbb{R}^{(r+2)\cdot m\times(r+2)\cdot m}$ the weight matrix, KF procedure 2.1.2 can be applied to solve the overdetermined system

$$Mz = p \tag{3.112}$$

where

$$M = \begin{bmatrix} H_0 \\ H_1 \\ \vdots \\ H_{r+1} \end{bmatrix} \in \mathbb{R}^{(r+2)\cdot m\times N_p};\ p = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{r+1} \end{bmatrix} \in \mathbb{R}^{(r+2)\cdot n_{obs}},\ z \in \mathbb{R}^{N_p}$$

and $H_l \in \mathbb{R}^{m\times N_p}$, $y_l \in \mathbb{R}^m$, where $m_0 \equiv m$. This means, as proved in Proposition 2, that KF estimate $\widehat{z}_{l+1}$ at step $l = r$ coincides with $\widehat{z} = (M^T RM)^{-1}M^T Rp$, i.e. least squares solution of (3.112).

3.4.2 DD-CLS problems: DD of CLS model

We apply DD approach for solving system S in (A.2). Here, we consider $N_{sub} = 2$ spatial subdomains i.e. the set $I \in \mathbb{N}$ is decomposed as $I = I_1 \cup I_2$.

Definition 12. *(DD-CLS model) Let S be the overdetermined linear system in (A.2) and $A \in \mathbb{R}^{(m_0+m_1)\times N_p}$, $b \in \mathbb{R}^{m_0+m_1}$ the matrix and the vector defined in (A.3) and $R_0 \in \mathbb{R}^{m_0\times m_0}$, $R_1 \in \mathbb{R}^{m_1\times m_1}$, $\boldsymbol{R} = diag(R_0, R_1) \in \mathbb{R}^{(m_0+m_1)\times(m_0+m_1)}$ be the weight matrices with $n^0_{obs} > N_p$ and $n^1_{obs} > 0$. Let us consider the index set of columns of A, $I = \{1, ..., N_p\}$. DD-CLS model consists in:*

- ***DD** step: DD of I*

$$I_1 = \{1, \ldots, n_1\}, \quad I_2 = \{n_1 - \frac{\delta}{2} + 1, \ldots, N_p\}, \tag{3.113}$$

where $\delta \geq 0$ is the number of indexes in common, $|I_1| = n_1 > 0$, $|I_2| = n_2 > 0$, and the overlap sets

$$I_{1,2} = \{n_1 - \frac{\delta}{2} + 1, \ldots, n_1\}, \tag{3.114}$$

If $\delta = 0$, then I is decomposed without using the overlap, i.e. $I_1 \cap I_2 = \emptyset$ and $I_{1,2} \neq \emptyset$, instead if $\delta > 0$ i.e. I is decomposed using overlap, i.e. $I_1 \cap I_2 \neq \emptyset$ and $I_{1,2} = \emptyset$.

- *restrictions of A to I_1 and I_2 defined in (3.113)*

$$A_1 = A|_{I_1} \in \mathbb{R}^{(m_0+m_1)\times n_1}, \quad A_2 = A|_{I_2} \in \mathbb{R}^{(m_0+m_1)\times n_2}, \tag{3.115}$$

- *DD-CLS step:*

  - ***Model Reduction**: let us consider $A \in \mathbb{R}^{(m_0+m_1)\times N_p}$, $b \in \mathbb{R}^{m_0+m_1}$, the matrix and the vector defined in (A.3), $I_1 = \{1, \ldots, n_1\}$, $I_2 = \{1, \ldots, n_2\}$ with $n_1, n_2 > 0$ and the vectors $x \in \mathbb{R}^{N_p}$. Let*

$$J|_{(I_i,I_j)} : (x|_{I_i}, x|_{I_j}) \longmapsto J|_{(I_i,I_j)}(x|_{I_i}, x|_{I_j}) \quad \forall i, j = 1, 2$$

  denote the reduction of J defined in (A.5). It is defined as

$$J|_{(I_i,I_j)}(x|_{I_i}, x|_{I_j}) = ||H_0|_{I_i} x|_{I_i} - (y_0 + H_0|_{I_j} x|_{I_j})||^2_{R_0} + ||H_1|_{I_i} x|_{I_i} - (y_1 + H_1|_{I_j} x|_{I_j})||^2_{R_1}, \tag{3.116}$$

  for $i, j = 1, 2$.

  - ***ASM**: given $x_2^0 \in \mathbb{R}^{n_2}$, according to the ASM in [56], DD-CLS approach consists in solving for $n = 0, 1, 2, \ldots$ the following overdetermined linear systems:*

$$S_1^{n+1} : \ A_1 x_1^{n+1} = b - A_2 x_2^n; \quad S_2^{n+1} : \ A_2 x_2^{n+1} = b - A_1 x_1^{n+1}, \tag{3.117}$$

  by employing a regularized VAR-KF model. It means that DD-CLS consists of a sequence of two subproblems:

$$\begin{aligned} P_1^{n+1} : \ \widehat{x}_1^{n+1} &= argmin_{x_1^{n+1} \in \mathbb{R}^{n_1}} J_1(x_1^{n+1}, x_2^n) \\ &= argmin_{x_1^{n+1} \in \mathbb{R}^{n_1}} \left[J|_{(I_1,I_2)}(x_1^{n+1}, x_2^n) + \mu \cdot \mathcal{O}_{1,2}(x_1^{n+1}, x_2^n) \right] \end{aligned} \tag{3.118}$$

$$\begin{aligned} P_2^{n+1}: \ \widehat{x}_2^{n+1} &= argmin_{x_2^{n+1}\in\mathbb{R}^{n_2}} J_2(x_2^{n+1}, x_1^{n+1}) \\ &= argmin_{x_2^{n+1}\in\mathbb{R}^{n_2}} \left[J|_{(I_2,I_1)}(x_2^{n+1}, x_1^{n+1}) + \mu \cdot \mathcal{O}_{1,2}(x_2^{n+1}, x_1^{n+1})\right] \end{aligned} \tag{3.119}$$

where I_i is defined in (3.113) and $J|_{I_i,I_j}$ is defined in (3.116), $\mathcal{O}_{1,2}$ is the overlapping operator and $\mu > 0$ is the regularization parameter.

Regarding the operator $\mathcal{O}_{1,2}$, we consider $x_1 \in \mathbb{R}^{n_1}$ and $x_2 \in \mathbb{R}^{n_2}$, and we pose

$$\mathcal{O}_{1,2}(x_i, x_j) = \frac{1}{2} \cdot \|EO_{I_i}(x_i|_{I_{1,2}}) - EO_{I_i}(x_j|_{I_{1,2}})\|^2, \ i, j = 1, 2$$

with $EO_{I_i}(x_1|_{I_{1,2}})$, $EO_{I_i}(x_2|_{I_{1,2}})$ be the extension to I_i, of restriction to $I_{1,2}$ in (3.114) of $x_1 \in \mathbb{R}^{n_1}$ and $x_2 \in \mathbb{R}^{n_2}$, respectively. Operator $\mathcal{O}_{1,2}$ represents the exchange of data on the overlap $I_{1,2}$ in (3.114).

Remark 5. *If I is decomposed without using the overlap (i.e. $\delta = 0$) then $\widehat{x}_1^{n+1} \in \mathbb{R}^{n_1}$ and $\widehat{x}_2^{n+1} \in \mathbb{R}^{n_2}$ can be written in terms of normal equations as follows*

$$\begin{aligned} \tilde{S}_1^{n+1}: \ (A_1^T R A_1)\widehat{x}_1^{n+1} &= A_1^T R(b - A_2 x_2^n) \quad &\Rightarrow \widehat{x}_1^{n+1} &= (A_1^T R A_1)^{-1} A_1^T R b_1^n \\ \tilde{S}_2^{n+1}: \ (A_2^T R A_2)\widehat{x}_2^{n+1} &= A_2^T R(b - A_1 x_1^{n+1}) \quad &\Rightarrow \widehat{x}_2^{n+1} &= (A_2^T R A_2)^{-1} A_2^T R b_2^{n+1}, \end{aligned} \tag{3.120}$$

where $b_1^n = b - A_2 x_2^n$ and $b_2^{n+1} = b - A_1 x_1^{n+1}$.

Remark 6. *DD-CLS gives rise to the sequences $\{x^{n+1}\}_{n\in\mathbb{N}_0}$:*

$$x^{n+1} = \begin{cases} \widehat{x}_1^{n+1}|_{I_1\setminus I_{1,2}} & \text{on } I_1 \setminus I_{1,2} \\ \frac{\mu}{2}(\widehat{x}_2^{n+1}|_{I_{1,2}} + \widehat{x}_1^{n+1}|_{I_{1,2}}) & \text{on } I_{1,2} \\ \widehat{x}_2^{n+1}|_{I_2\setminus I_{1,2}} & \text{on } I_2 \setminus I_{1,2} \end{cases}, \tag{3.121}$$

where I_1, I_2 are defined in (3.113) and $I_{1,2}$ in (3.114).

3.4.3 DD–KF solving DD–CLS problems

In the same setting of the previous Section, we aim to find an estimate of $\widehat{x} = (A^T R A)^{-1} A^T R b \in \mathbb{R}^{N_p}$, i.e. the least squares solution of S in (A.2), by solving DD-CLS by KF algorithm. To this

end, for $n = 0, 1, 2, \ldots$, we prove that $\widehat{x}_1^{n+1} \in \mathbb{R}^{n_1}$, $\widehat{x}_2^{n+1} \in \mathbb{R}^{n_2}$, solutions of P_1^{n+1} and P_2^{n+1} in (3.118) and (3.119) respectively are equal to KF estimates $\widehat{x}_{1,1}^{n+1}$, $\widehat{x}_{2,1}^{n+1}$ obtained by applying KF to P_1^{n+1}, P_2^{n+1}.

To apply KF algorithm to P_1^{n+1} and P_2^{n+1} in (3.118) and (3.119), we need $\widehat{x}_{1,0} \in \mathbb{R}^{n_1}$ and $\widehat{x}_{2,0} \in \mathbb{R}^{n_2}$ (i.e. reduction of $\widehat{x}_0 = (H_0^T R H_0)^{-1} H_0^T R y_0$). This vectors are calculated as follows, in the case without or with overlap:

1. DD of I without overlap (i.e. $\delta = 0$).

   Theorem 5. *Let us consider $H_0 \in \mathbb{R}^{m_0 \times N_p}$, $y_0 \in \mathbb{R}^{m_0}$ and $R_0 \in \mathbb{R}^{m_0 \times m_0}$, where $m_0 > N_p$, $m_1 > 0$, system $H_0 x_0 = y_0$ and the set I. Let us consider the following steps:*

   - ***DD** of I:*

   $$I_1 = \{1, ..., n_1\} \text{ and } I_2 = \{n_1 + 1, ..., N_p\}. \tag{3.122}$$

   - *reduction of H_0:*

   $$H_{1,0} = H_0|_{I_1} \in \mathbb{R}^{m_0 \times n_1} \text{ and } H_{2,0} = H_0|_{I_2} \in \mathbb{R}^{m_0 \times n_2}. \tag{3.123}$$

   - *computation of $P_{i,0} \in \mathbb{R}^{n_i \times n_i}$:*

   $$P_{i,0} = (H_{i,0}^T R_0 H_{i,0})^{-1}, \quad i=1,2. \tag{3.124}$$

   - *computation of $P_{H_{i,0}} \in \mathbb{R}^{m_0 \times m_0}$:*

   $$P_{H_{i,0}} = R_0 - R_0 H_{i,0} (P_{i,0}) H_{i,0}^T R_0, \quad i=1,2. \tag{3.125}$$

   - *computation of $x_1 \in \mathbb{R}^{n_1}$ and $x_2 \in \mathbb{R}^{N_p - n_1}$:*

   $$x_1 = (H_{1,0}^T P_{H_{2,0}} H_{1,0})^{-1} H_{1,0}^T P_{H_{2,0}} y_0, \quad x_2 = (H_{2,0}^T P_{H_{1,0}} H_{2,0})^{-1} H_{2,0}^T P_{H_{1,0}} y_0. \tag{3.126}$$

Then $\widehat{x}_0 = (H_0^T R_0 H_0)^{-1} H_0^T R_0 y_0 \in \mathbb{R}^{N_p}$*, which is the least squares solution of* $H_0 x_0 = y_0$*, is obtained as follows:*

$$\widehat{x}_0|_{I_1} = x_1, \quad \widehat{x}_0|_{I_2} = x_2, \tag{3.127}$$

where $\widehat{x}_0|_{I_i}$ *is the restriction of* $\widehat{x}_0$ *to* I_i*, for* $i = 1, 2$*.*

2. **DD** with overlap i.e. $\delta \neq 0$ and overlap set $I_{1,2}$ be as in (3.114).

$$\widehat{x}_0|_{I_1} = x_1 \in \mathbb{R}^{n_1} \quad \widehat{x}_0|_{I_2} = \begin{cases} x_1|_{I_{1,2}} \in \mathbb{R}^s & \text{on } I_{1,2} \\ x_2 \in \mathbb{R}^{n_2-\delta} & \text{on } I_2 \setminus I_{1,2} \end{cases} \in \mathbb{R}^{n_2}, \tag{3.128}$$

where $x_1 \in \mathbb{R}^{n_1}$ and $x_2 \in \mathbb{R}^{n_2}$ are defined in (3.126) and $n_1 = |I_1|$, $n_2 = |I_2|$, $\delta = |I_{1,2}|$.

Theorem 6. *(DD-KF)[48] Let us consider the overdetermined linear system in (A.2),* $H_0 \in \mathbb{R}^{m_0 \times N_p}$*,* $R_0 \in \mathbb{R}^{m_0 \times m_0}$*,* $M = (H_0^T R_0 H_0)^{-1} H_0^T R_0 \in \mathbb{R}^{N_p \times m_0}$*,* $y_0 \in \mathbb{R}^{m_0}$ *and* $\widehat{x}_0 = M y_0 \in \mathbb{R}^{N_p}$*, where* $m_0 > N_p$ *and* $m_1 > 0$*. DD-KF-CLS approach is composed by the following steps.*

- ***DD*** *step. Decomposition of* $I = \{1, ..., N_p\}$*, i.e. columns index set of* $A \in \mathbb{R}^{(m_0+m_1)\times N_p}$ *of* S *in (A.2) into*

$$I_1 = \{1 \ldots, n_1\} \text{ and } I_2 = \{n_1 + 1, \ldots, N_p\}, \tag{3.129}$$

  with $|I_1| = n_1$ *and* $|I_2| = n_2$*.*

- ***KF-CLS step****. Computation of* $\widehat{x}_{1,0} \equiv \widehat{x}_0|_{I_1} \in \mathbb{R}^{n_1}$ *and* $\widehat{x}_{2,0} \equiv \widehat{x}_0|_{I_2} \in \mathbb{R}^{n_2}$*, restrictions of* $\widehat{x}_0 \in \mathbb{R}^{N_p}$ *as in (3.127) to* I_1 *and* I_2 *in (3.129).*

Given $\widehat{x}_{2,0}^0 \in \mathbb{R}^{n_2}$*, we consider*

$$\begin{array}{llll} M_{0,1}^1 & = I_{n_1,n_1} \in \mathbb{R}^{n_1 \times n_1}, & M_{0,1}^2 & = I_{n_2,n_2} \in \mathbb{R}^{n_2 \times n_2}, \\ B_0^1 & = O_{n_1,n_1} \in \mathbb{R}^{n_1 \times n_1}, & B_0^2 & = O_{n_2,n_2} \in \mathbb{R}^{n_2 \times n_2} \\ P_{1,0} & = (H_0^T|_{I_1} R_0 H_0|_{I_1})^{-1} \in \mathbb{R}^{n_1 \times n_1} & P_{2,0} & = (H_0^T|_{I_2} R_0 H_0|_{I_2})^{-1} \in \mathbb{R}^{n_2 \times n_2} \end{array}, \tag{3.130}$$

where I_{N_p,N_p} is the identity matrix and O_{N_p,N_p} the null matrix.

For $n = 0, 1, 2, \ldots$ and for $l = 0$, by applying KF algorithm described in (2.1.2) to P_1^{n+1} and P_2^{n+1} in (3.118) and (3.119), we obtained as predicted estimates

$$\begin{cases} x_{1,1} &= M_{0,1}^1 \widehat{x}_{1,0} &= \widehat{x}_{1,0} \\ x_{2,1} &= M_{0,1}^2 \widehat{x}_{2,0} &= \widehat{x}_{2,0} \end{cases}, \tag{3.131}$$

predicted covariance matrices

$$\begin{cases} P_{1,1} &= M_{0,1}^1 P_{1,0} (M_{0,1}^1)^T + Q_0^1 &= P_{1,0} \\ P_{2,1} &= M_{0,1}^2 P_{2,0} (M_{0,1}^2)^T + Q_0^2 &= P_{2,0} \end{cases}, \tag{3.132}$$

Kalman gains

$$\begin{cases} \boldsymbol{K}_{1,1} &= P_{1,1} H_1|_{I_1}^T (H_1|_{I_1} P_{1,1} H_1^T|_{I_1} + R_1)^{-1} \\ \boldsymbol{K}_{2,1} &= P_{2,1} H_1|_{I_2}^T (H_1|_{I_2} P_{2,1} H_1^T|_{I_2} + R_1)^{-1} \end{cases}, \tag{3.133}$$

the Kalman covarinace matrices

$$\begin{cases} P_{1,1} = (I - \boldsymbol{K}_{1,1} H_1|_{I_1}) P_{1,1}, \\ P_{2,1} = (I - \boldsymbol{K}_{2,1} H_1|_{I_2}) P_{2,1}, \end{cases}, \tag{3.134}$$

and the matrices

$$\begin{cases} \boldsymbol{K}_{1,2} &= P_{1,1} H_0|_{I_1}^T R_0 \\ \boldsymbol{K}_{2,1} &= P_{2,1} H_0|_{I_2}^T R_0 \end{cases}. \tag{3.135}$$

So, for $n = 0, 1, 2, \ldots$ the Kalman estimates are

$$\begin{aligned} \widehat{x}_{1,1}^{n+1} &= \widehat{x}_{1,0} + \boldsymbol{K}_{1,1} \left[(y_1 - H_1|_{I_2} \widehat{x}_{2,1}^n) - H_1|_{I_1} \widehat{x}_{1,0} \right] + \mathcal{S}_{I_1 \leftrightarrow I_2}(\widehat{x}_{2,1}^n) \\ \widehat{x}_{2,1}^{n+1} &= \widehat{x}_{2,0} + \boldsymbol{K}_{2,1} \left[(y_1 - H_1|_{I_1} \widehat{x}_{1,1}^{n+1}) - H_0|_{I_2} \widehat{x}_{2,0} \right] + \mathcal{S}_{I_1 \leftrightarrow I_2}(\widehat{x}_{1,1}^{n+1}) \end{aligned} \tag{3.136}$$

where for $i, j = 1, 2$

$$\mathcal{S}_{I_1 \leftrightarrow I_2}(\widehat{x}_{i,1}^n) := \boldsymbol{K}_{i,j} \left[H_0|_{I_i} (\widehat{x}_{i,0} - \widehat{x}_{i,1}^n) \right] \tag{3.137}$$

represents the exchange of data between the sets I_1, I_2.

Then, for each $n = 0, 1, 2, \ldots$, we obtain KF estimates $\widehat{x}_{1,1}^{n+1} \in \mathbb{R}^{n_1}$ and $\widehat{x}_{2,1}^{n+1} \in \mathbb{R}^{n_2}$ in (3.136)

such that

$$\begin{aligned}\widehat{x}_{1,1}^{n+1} &= \widehat{x}_1^{n+1}\\ \widehat{x}_{2,1}^{n+1} &= \widehat{x}_2^{n+1}\end{aligned}$$

where $\widehat{x}_1^{n+1}$, $\widehat{x}_2^{n+1}$ *are least squares solutions of systems* $\tilde{S}_1^{n+1}$ *and* $\tilde{S}_2^{n+1}$ *defined in (3.120).*

Remark 7. *DD with overlap i.e.* $I_1 = \{1, \ldots, n_1\}$, $I_2 = \{n_1 - \frac{\delta}{2} + 1, \ldots, N_p\}$ *and* $I_{1,2} = \{n_1 - \frac{\delta}{2} + 1, \ldots, n_1\}$ *with* $\delta \neq 0$ *is similarly obtained by considering the initial estimates as in (3.128). Furthermore, for* $n = 0, 1, 2, \ldots$, *we add operator* $\mathcal{O}_{1,2}$ *to* P_1^{n+1} *and* P_2^{n+1} *in (3.118) and (3.119). It means that it is*

$$\begin{aligned}\widehat{x}_{1,1}^{n+1} &\equiv \widehat{x}_{1,1}^{n+1} + \mu P_{1,1} \nabla \mathcal{O}_{1,2}(EO_{I_1}(\widehat{x}_{1,1}^n|_{I_{1,2}}), EO_{I_1}(\widehat{x}_{2,1}^n|_{I_{1,2}}))\\ \widehat{x}_{2,1}^{n+1} &\equiv \widehat{x}_{2,1}^{n+1} + \mu P_{2,1} \nabla \mathcal{O}_{1,2}(EO_{I_2}(\widehat{x}_{1,1}^{n+1}|_{I_{1,2}}), EO_{I_1}(\widehat{x}_{2,1}^{n+1}|_{I_{1,2}}))\end{aligned} \tag{3.138}$$

where

$$\nabla \mathcal{O}_{1,2}(EO_{I_i}(\widehat{x}_{i,1}^n|_{I_{1,2}}), EO_{I_i}(\widehat{x}_{j,1}^n|_{I_{1,2}})) = \left[EO_{I_1}(\widehat{x}_{j,1}^n|_{I_{1,2}}) - EO_{I_1}(\widehat{x}_{i,1}^n|_{I_{1,2}})\right], \tag{3.139}$$

and $EO_{I_i}(\widehat{x}_{1,0}^n|_{\tilde{I}_{1,2}})$ *and* $EO_{I_i}(\widehat{x}_{2,0}^n|_{\tilde{I}_{1,2}})$, *for* $i = 1, 2$ *are the extensions to* I_i *of restrictions to* $\tilde{I}_{1,2}$ *of* $\widehat{x}_{1,0}^n$ *and* $\widehat{x}_{2,0}^n$ *and* μ *is the regularization parameter.*

Finally, in [48] we proved that the sequence of KF estimates $\{\widehat{x}_{1,1}^{n+1}\}_{n\in\mathbb{N}_0}$, $\{\widehat{x}_{2,1}^{n+1}\}_{n\in\mathbb{N}_0}$ in (3.138) converge to $\widehat{x}_1|_{I_1}$, $\widehat{x}_1|_{I_2}$ restrictions of KF estimate $\widehat{x}_1$ in (2.20) to I_1 and I_2, respectively.

Theorem 7. *[48] Let* S *be the over determined linear system in (A.2),* $A \in \mathbb{R}^{(m_0+m_1)\times N_p}$, $b \in \mathbb{R}^{m_0+m_1}$ *defined in (A.3) with* $m_0 > N_p$, $m_1 > 0$, $\widehat{x}_1 \in \mathbb{R}^{N_p}$ *be the Kalman estimate in (2.20) of the* $\widehat{x}$ *as in (A.4) and* $\widehat{x}_{1,1}^{n+1}$, $\widehat{x}_{2,1}^{n+1}$ *in (3.138) Kalman estimates of* P_1^{n+1}, P_2^{n+1} *in (3.118), (3.119), then*

$$\widehat{x}_{1,1}^{n+1} \to \widehat{x}_1|_{I_1}, \quad \widehat{x}_{2,1}^{n+1} \to \widehat{x}_1|_{I_2}.$$

3.4.4 DD–KF-CLS: Performance Analysis

In Table 3.5, we summarize time complexity of KF algorithm. The total time complexity is:

$$T_{KF}(m_1, N_p) = O(N_p^3 + 3N_p^2 m_1 + N_p^2 + N_p m_1^2 + 2N_p m_1 + N_p + m_1^3 + m_1^2 + m_1). \quad (3.140)$$

For $i, j = 1, 2$ and $i \neq j$, in Table 3.6 we report time complexity of DD–KF procedure needed to solve, at each $n = 0, 1, 2, \ldots$, P_1^{n+1} in (3.118) and P_2^{n+1} in (3.119). Then, time complexity of DD–KF is:

$$\begin{aligned} T_{DD-KF}(m_1, n_i, n_j) &= O(n_i^3(1+d) + 3n_i^2 m_1 + (2+d)n_i^2 + m_1^2 n_i + 3n_i m_1 + n_i m_0 \\ &\quad +(3+2d)n_i + n_j + m_1^3 + m_1^2 + 2m_1), \end{aligned} \quad (3.141)$$

where $d = 0$ or $d = 1$ if we perform DD without or with overlap, respectively.

Table 3.4: KF algorithm. Time complexity needed to calculate $\widehat{x}_0 = (H_0^T R_0^{-1} H_0)^{-1} H_0^T R_0^{-1} y_0 \in \mathbb{R}^{N_p}$ solution (in the least squares sense) in (A.8) of the overdetermined linear system $H_0 x_0 = y_0$ in (A.1), i.e. predicted state estimate of KF algorithm in (2.16), $P_0 \in \mathbb{R}^{N_p \times N_p}$ in (3.111), i.e. predicted covariance matrix of KF algorithm in (2.17), where $N_p, m_0 \in \mathbb{N}$ is the number of columns and rows of matrix $H_0 \in \mathbb{R}^{m_0 \times N_p}$, with $m_0 > N_p$; $R_0 \in \mathbb{R}^{m_0 \times m_0}$ the weight matrix.

matrix/vector	operations	cost
R_0^{-1} in (A.8)	matrix inversion	$O(m_0^3)$
$(H_0^T R_0^{-1}) H_0$ in (A.8)	2 matrix-matrix products	$O(m_0^2 N_p + N_p^2 m_0)$
P_0 in (3.111)	matrix inversion	$O(N_p^3)$
$H_0^T (R_0 y_0)$ in (A.8)	2 matrix-vector products	$O(m_0^2 + m_0 N_p)$
$\widehat{x}_0$ in (A.8)	matrix-vector product	$O(N_p^2)$

Table 3.5: KF algorithm. Time complexity of each operation of KF in (2.18)-(2.20) for solving CLS in (A.4), where $m_1 \in \mathbb{N}$ is the number of equations added to the overdetermined linear system $H_0 x_0 = y_0$ in (A.1); $H_1 \in \mathbb{R}^{m_1 \times N_p}$ is the matrix of the coefficients of the equations added to the overdetermined linear system $H_0 x_0 = y_0$ in (A.1); $R_1 \in \mathbb{R}^{m_1 \times m_1}$ is the weight matrix.

matrix/vector	operations	cost
K_1 in (2.18)	2 matrix-matrix products	$O(m_1^2 N_p + N_p^2 m_1)$
	and sum of matrices	$O(m_1^2)$
	a matrix inversion	$O(m_1^3)$
	2 matrix-matrix products	$O(N_p^2 m_1 + m_1^2 N_p)$
P_1 in (2.19)	matrix-matrix product,	$O(N_p^2 m_1)$
	subtraction of matrices,	$O(N_p^2)$
	product of matrices	$O(N_p^3)$
$\widehat{x}_1$ in (2.20)	matrix-vector product	$O(N_p m_1)$
	subtraction vectors	$O(m_1)$
	matrix-vector product	$O(m_1 N_p)$
	sum of vectors	$O(N_p)$

Table 3.6: DD–KF algorithm. For $i = 1, 2$ we report time complexity of each operation of DD-Setup algorithm in (3.123)-(3.126) needed to compute $\widehat{x}_{i,0} = (H_{i,0}^T P_{H_{i,0}} H_{i,0})^{-1} H_{i,0}^T P_{H_{i,0}} y_0 \in \mathbb{R}^{n_i}$ in (3.126), i.e. predicted state estimate of DD–KF algorithm in (3.131), $P_{i,0} \in \mathbb{R}^{N_p \times N_p}$ in (3.124), i.e. predicted covariance matrix of DD–KF algorithm in (3.132), where $N_p \in \mathbb{N}$ is the size of columns index set I of matrix $H_0 \in \mathbb{R}^{m_0 \times N_p}$; $n_i \in \mathbb{N}$ is the size of $I_i \subseteq I$, where $n_i \leq N_p$; $H_{i,0} \equiv H_0|_{I_i} \in \mathbb{R}^{m_0 \times n_i}$ as in (3.123).

matrix/vector	operations	cost
$P_{i,0}$ in (3.124)	matrix-matrix product	$O(m_0^2 n_i)$

	matrix-matrix product	$O(n_i^2 m_0)$
	matrix inversion	$O(n_i^3)$
$P_{H_{i,0}}$ in (3.125)	matrix-matrix product	$O(m_0^2 n_i)$
	matrix-matrix product	$O(n_i^2 m_0)$
	matrix-matrix product	$O(n_i m_0^2)$
	subtraction of matrices	$O(m_0^2)$
$(H_{i,0}^T P_{H_{i,0}}) H_{i,0}$ in (3.126)	2 matrix-matrix products	$O(m_0^2 n_i + n_i^2 m_0)$
$H_{i,0}^T (P_{H_{i,0}} y_0)$ in (3.126)	2 matrix-vector products	$O(m_0^2 + m_0 n_i)$
$\widehat{x}_{i,0}$ in (3.126)	matrix-vector product	$O(n_i^2)$

Table 3.7: Time complexity needed to compute $\widehat{x}_0 = (H_0^T R_0^{-1} H_0)^{-1} H_0^T R_0^{-1} y_0$ in (A.8) and for $i = 1, 2$, $\widehat{x}_{i,0} = (H_{i,0}^T P_{H_{i,0}} H_{i,0})^{-1} H_{i,0}^T P_{H_{i,0}} y_0 \in \mathbb{R}^{n_i}$ in (3.126) where $N_p \in \mathbb{N}$ is the size of columns index set I of matrix $H_0 \in \mathbb{R}^{m_0 \times N_p}$; $n_i \in \mathbb{N}$ is the size of the set $I_i \subseteq I$, where $n_i \leq N_p$; $m_0 \in \mathbb{N}$ is the number of rows of matrix $H_0 \in \mathbb{R}^{m_0 \times N_p}$ and $H_{i,0} \in \mathbb{R}^{m_0 \times N_p}$, where $m_0 > N_p$.

vectors	cost
$\widehat{x}_0$	$O(N_p^3 + N_p^2 + n^2 m_0 + N_p m_0^2 + m_0 N_p + 2m_0^2)$
$\widehat{x}_{i,0}$	$O(n_i^3 + n_i^2 + 2n_i^2 m_0 + 3n_i m_0^2 + 2m_0 n_i + 2m_0^2)$

Table 3.8: For $i, j = 1, 2$, time complexity of each operation of DD–KF algorithm in (3.133)-(3.138) solving DD-CLS models in (3.118) and (3.119), where $N_p \in \mathbb{N}$ is the size of columns index set I of matrix $H_0 \in \mathbb{R}^{m_0 \times N_p}$; $n_i \in \mathbb{N}$ is the size of the set $I_i \subseteq I$, where $n_i \leq N_p$; $m_1 \in \mathbb{N}$ is the number of equations added to the over determined linear system $H_0|_{I_i} x_i = y_0$; $H_1|_{I_i} \in \mathbb{R}^{m_1 \times n_i}$ is the matrix of the coefficients of the equations added to the over determined linear system $H_0|_{I_i} x_i = y_0$; $R_1 \in \mathbb{R}^{m_1 \times m_1}$ the weight matrix.

matrix/vector	operations	cost

$K_{i,1}$ in (3.133)	2 matrix-matrix product	$O(m_1^2 n_i + n_i^2 m_1)$
	and sum of matrices	$O(m_1^2)$
	product of matrices	$O(n_i^3)$
	matrix inversion	$O(m_1^3)$
	matrix-matrix product	$O(n_i^2 m_1)$
$P_{i,1}$ in (3.134)	matrix-matrix product	$O(n_i^2 m_1)$
	subtraction of matrices	$O(n_i^2)$
$K_{i,j}$ in (3.135)	matrix-matrix product	$O(n_i^3)$
$\mathcal{S}_{I_1 \leftrightarrow I_2}$ in (3.137)	subtraction of vectors	$O(n_j)$
	product matrix-vector	$O(n_i^2)$
$\nabla \mathcal{O}_{1,2}(EO_{I_1}(\widehat{x}_{1,1}\|_{I_{1,2}}), EO_{I_1}(\widehat{x}_{2,1}\|_{I_{1,2}}))$ in (3.139)	subtraction of vectors	$O(n_i)$
$\widehat{x}_{i,1}$ in (3.138)	matrix-vector product	$O(n_i^2)$
	matrix-vector product	$O(n_i m_1)$
	2 subtraction of vectors	$O(2m_1)$
	2 matrix-vector products	$O(2m_1 n_i)$
	3 sum of vectors	$O(3n_i)$
	subtraction/sum vectors	$O(n_i)$

Table 3.9: Time complexity of KF and DD–KF, where $N_p \in \mathbb{N}$ is the size of columns index set I of matrix $H_0 \in \mathbb{R}^{m_0 \times N_p}$; $n_i \in \mathbb{N}$ is the size $I_i \subseteq I$, and $n_i \leq N_p$; $m_1 \in \mathbb{N}$ is the number of equations added to the overdetermined linear system $H_0 x_0 = y_0$; $d = 0$ or $d = 1$: if we consider a decomposition without or with overlap, respectively.

algorithm	cost
KF	$O(N_p^3 + 4N_p^2 m_1 + N_p^2 + N_p m_1^2 + 2N_p m_1 + N_p + m_1^3 + m_1^2 + m)$

DD–KF	$O(n_i^3(1+d)+3n_i^2m_1+(2+d)n_i^2+m_1^2n_i+3n_im_1+(3+2d)n_i+n_j$ $+m_1^3+m_1^2+2m_1)$.

In Table 3.9 we compare time complexity of KF and DD–KF algorithms. We note that $n_i < N_p$, so performance gain by using DD-KF is significantly better than using KF, in particular it could still be improved if m_1 i.e. the number of equations added to the system in (A.1) is such that $m_1 \leq min(n_1, n_2)$.

Definition 13. ***(Scale-up factor [30, 39])*** *Let $N_p, m_1 \in \mathbb{N}$ and $N_{sub} \geq 2$ the number of subsets I_i of $I = \{1, \ldots, N_p\}$ such that $n_i \equiv |I_i|$, $i = 1, \ldots, N_{sub}$. Let $T_{KF}(m_1, N_p)$, $T_{DD-KF}(m_1, n_i)$, $i = 1, \ldots, N_{sub}$ be time complexity of KF algorithm applied to CLS model in (A.4) and DD-KF algorithm applied at each step $n = 0, 1, 2, \ldots$ to P_1^{n+1}, P_2^{n+1} problems in (3.118) and (3.119), respectively. We introduce the following ratio:*

$$Sc^f_{N_{sub},1}(m_1, N_p) = \frac{T_{KF}(m_1, N_p)}{N_{sub} \cdot T_{DD-KF}(m_1, N_p/N_{sub})}. \tag{3.142}$$

Following result allows us to analyze the behaviour of the scale up factor.

Proposition 3. *Let $N_p, m \in \mathbb{N}$ and $r \equiv N_p/N_{sub}$. Then it holds that*

$$Sc^f_{N_{sub},1}(m, N_p) = \alpha(N_p, m, N_{sub})N_{sub}^2$$

where

$$\alpha(N_p, m, N_{sub}) = \frac{a_3 + a_2\frac{1}{N_p} + a_1\frac{1}{N_p^2} + a_0\frac{1}{N_p^3}}{b_3 + b_2\frac{1}{r} + b_1\frac{1}{r^2} + b_0\frac{1}{r^3}}. \tag{3.143}$$

Proof. $T_{KF}(m, N_p) = O(p_1(N_p))$, $T_{DD-KF}(m, r) = O(p_2(r))$ are polynomials of degree 3 i.e. $p_1, p_2 \in \Pi_3$. We let

$$\begin{aligned} &a_3 = 3 \quad a_2 = 2m+3 \quad a_1 = 2m^2+2m+1 \quad a_0 = \tfrac{2}{3}m^3+m^2+m \\ &b_3 = 12 \quad b_2 = 4m+6 \quad b_1 = 3m^2+10m+2 \quad b_0 = \tfrac{2}{3}m^3+5m^2+m-\tfrac{\delta}{2} \end{aligned} \tag{3.144}$$

and we write $T_{KF}(m, N_p)$, $T_{DD-KF}(m, r)$ as follows

$$\begin{aligned} T_{KF}(m, N_p) &= a_3 N_p^3 + a_2 N_p^2 + a_1 N_p + a_0 \\ T_{DD-KF}(m, N_p) &= b_3 r^3 + b_2 r^2 + b_1 r + b_0 \end{aligned} . \tag{3.145}$$

It holds

$$\begin{aligned} Sc^f_{N_{sub},1}(m, N_p) &= \frac{a_3 N_p^3 + a_2 N_p^2 + a_1 N_p + a_0}{N_{sub}(b_3 r^3 + b_2 r^2 + b_1 r + b_0)} \cdot r^3 \cdot \frac{1}{r^3} = \frac{a_3 N_{sub}^3 + a_2 \frac{n_{sub}^3}{N_p} + a_1 \frac{N_{sub}^3}{N_p^2} + a_0 \frac{N_{sub}^3}{N_p^3}}{N_{sub}(b_3 + b_2 \frac{1}{r} + b_1 \frac{1}{r^2} + b_0 \frac{1}{r^3})} \\ &= \frac{N_{sub}^3 \cdot (a_3 + a_2 \frac{1}{N_p} + a_1 \frac{1}{N_p^2} + a_0 \frac{1}{N_p^3})}{N_{sub}(b_3 + b_2 \frac{1}{r} + b_1 \frac{1}{r^2} + b_0 \frac{1}{r^3})} = \frac{a_3 + a_2 \frac{1}{N_p} + a_1 \frac{1}{N_p^2} + a_0 \frac{1}{N_p^3}}{b_3 + b_2 \frac{1}{r} + b_1 \frac{1}{r^2} + b_0 \frac{1}{r^3}} \cdot N_{sub}^2 \end{aligned} \tag{3.146}$$

and

$$Sc^f_{N_{sub},1}(m, N_p) = \alpha(N_p, m, N_{sub}) N_{sub}^2$$

where

$$\alpha(N_p, m, N_{sub}) = \frac{a_3 + a_2 \frac{1}{N_p} + a_1 \frac{1}{N_p^2} + a_0 \frac{1}{N_p^3}}{b_3 + b_2 \frac{1}{r} + b_1 \frac{1}{r^2} + b_0 \frac{1}{r^3}}. \tag{3.147}$$

□

Recalling that $r = \frac{N_p}{N_{sub}}$ and $Sc^f_{N_{sub},1}(m, N_p) = \alpha(N_p, m, N_{sub}) N_{sub}^2$, this result says that the performance gain of DD-KF algorithm in terms of reduction of time complexity, scales as the number of subdomains N_{sub} squared, where the scaling factor depends on parameter $\alpha(N_p, m, N_{sub})$.

3.5 Domain Decomposition of KF (DD–KF)

In this Section we extend DD-KF method introduced in Section 3.4 on CLS problem to a DA problem. Let $\Omega \subset \mathbb{R}^n$ be spatial domain and $\Delta \subset \mathbb{R}$ be the time winterval. Let us assume: Ω_I discretization of Ω defined in (2.3), $\partial\Omega_I$ boundary of Ω_I and $\Omega_I \times \Delta_K$ discrete local domain defined in Definition (1). We get:

$$x_{l+1} \equiv x(t_{l+1}) := \{u(x_j, t_{l+1})\}_{(x_j, t_{l+1}) \in \Omega_I \times \Delta_K} \in \mathbb{R}^{N_p}$$

and if we let $x_{l+1}^{\partial\Omega} \in \mathbb{R}^{b_c}$ to be the state at t_{l+1} on $\partial\Omega$, it is

$$x_{l+1}^{\partial\Omega} \equiv x^{\partial\Omega}(t_{l+1}) := \{u(z_p, t_{l+1})\}_{(z_p,t_{l+1})\in\partial\Omega_I\times\Delta_k} \in \mathbb{R}^{b_c}. \tag{3.148}$$

where

$$b_c := |\partial\Omega_I| \tag{3.149}$$

number of nodes in $\partial\Omega_I$.

In sections 3.5.1 and 3.5.2 we consider respectively a DD of spatial domain Ω into $N_{sub} = 2$ and $N_{sub} > 2$ subdomains with a generic DD of time interval Δ into $Nt > 1$ time intervals.

3.5.1 DD-KF method in $\{\Omega_i \times \Delta_k\}_{i=1,2;k=1,...,Nt}$

In the following, for simplicity of notation, and without loss of generality [3], we consider $N_{sub} = 2$ spatial subdomains.

DD step: as described in section 3.1, we consider DD of $\Omega \times \Delta$. Let Ω_1, $\Omega_2 \subset \mathbb{R}^n$ be subsets of $\Omega \subset \mathbb{R}^n$ such that

$$\Omega = \Omega_1 \cup \Omega_2 \tag{3.150}$$

where

$$\Omega_{I_1} = \{x_i\}_{i\in I_1} \quad , \quad \Omega_{I_2} = \{x_i\}_{i\in I_2} \tag{3.151}$$

and $n_1 = |I_1|$, $n_2 = |I_2|$, where I_i is defined in 3.6 and $n_i < N_p$; let

$$\Omega_{1,2} := \Omega_1 \cap \Omega_2, \tag{3.152}$$

be the overlap region where

$$\Omega_{I_{12}} := \{x_i\}_{i\in I_{1,2}} \tag{3.153}$$

[3] The general case involving more than two subdomains will be briefly described in section 3.5.2

denote the discretization of $\Omega_{1,2}$ and

$$I_1 = \{1, \dots, n_1\}, \quad I_2 = \{n_1 - \frac{\delta}{2} + 1, \dots, N_p\}, \quad I_{1,2} = \{n_1 - \frac{\delta}{2} + 1, \dots, n_1\} \tag{3.154}$$

are the corresponding index sets, with $\delta \in \mathbb{N}_0$.

Concerning DD in time, let Δ_k be a generic subset of Δ such that

$$\Delta = \cup_{k=1}^{N_t} \Delta_k \tag{3.155}$$

with $D_{s_k}(\Delta_k) \subset D_{r+2}(\Delta)$ such that

$$\Delta_{K_k} = \{t_l\}_{l=\bar{s}_{k-1},\dots,\bar{s}_{k-1}+s_k} \tag{3.156}$$

and

$$\Delta_{k-1,k} := \Delta_{k-1} \cap \Delta_k, \tag{3.157}$$

with

$$\Delta_{K_{k-1,k}} = \Delta_{K_{k-1}} \cap \Delta_{K_k}$$

where in (3.156)

$$\bar{s}_{k-1} := \sum_{l=1}^{k-1} (s_k - s_{k-1,k}), \quad s_{k-1,k} \in \mathbb{N}_0$$

is the number of elements in common between subsets Δ_{k-1} and Δ_k, and it is such that $s_0 := 0$, $s_{0,1} := 0$ and $\bar{s}_{N_t-1} + s_{N_t} := r + 1$.

The second step of DD-KF is given in the following

Definition 14. *(**Model Reduction** step) For $k = 1, \dots, N_t$, let us indicate by $\widehat{x}_{l+1}^{\Delta_k}$, the KF estimate $\widehat{x}_{l+1} \in \mathbb{R}^{N_p}$ in $t_{l+1} \in \Delta_k$, such that:*

$$\widehat{x}_{l+1}^{\Delta_k} = M_{l,l+1} \widehat{x}_l^{\Delta_k} + b_l + w_l, \quad \forall l = \bar{s}_{k-1}, \dots, \bar{s}_{k-1} + s_k - 1 \tag{3.158}$$

where

$$\widehat{x}_{\bar{s}_{l-1}}^{\Delta_k} = \widehat{x}_{\bar{s}_{l-1}}^{\Delta_{l-1}}, \tag{3.159}$$

is the initial value (according to PinT compatibility conditions), such that (according to DA condition):

$$y_{l+1} = H_{l+1}\widehat{x}_{l+1}^{\Delta_j} + v_{l+1}, \tag{3.160}$$

in particular it is $\widehat{x}_0^{\Delta_0} := x_0$.

Model Reduction step consists of the block decomposition of $M := M_{l,l+1} \in \mathbb{R}^{N_p \times N_p}$ *and* $H_{l+1} \in \mathbb{R}^{m\cdot d \times N_p}$ *as follows:*

$$M = \left[\begin{array}{c|c} M_1 & M_{1,2} \\ \hline M_{2,1} & M_2 \end{array}\right], \tag{3.161}$$

and, if $\delta = 0$

$$H_{l+1} = \left[H_{l+1}^1 \; H_{l+1}^2\right]. \tag{3.162}$$

or, if $\delta \neq 0$

$$H_{l+1} = \left[H_{l+1}^{1,1} \; H_{l+1}^{1,2} \; H_{l+1}^{2,2}\right] \tag{3.163}$$

where

- *if* $\delta = 0$*, for* $i, j = 1, 2$*:*

$$M_i := M|_{I_i \times I_i} \in \mathbb{R}^{n_i \times n_i} \quad M_{i,j} := M|_{I_i \times I_j} \in \mathbb{R}^{n_i \times n_j}, \tag{3.164}$$

  and

$$H_{l+1}^1 := H_{l+1}|_{I_1} \in \mathbb{R}^{m\cdot d \times n_1}, \quad H_{l+1}^2 := H_{l+1}|_{I_2} \in \mathbb{R}^{m\cdot d \times n_2}, \tag{3.165}$$

- *if* $\delta \neq 0$*, we let:*

$$\begin{aligned}
\mathbb{M}_{1,1} &:= M|_{\tilde{I}_1 \times \tilde{I}_1} \in \mathbb{R}^{(n_1-s)\times(n_1-s)}, & \mathbb{M}_{1,2} &:= M|_{\tilde{I}_1 \times I_{1,2}} \in \mathbb{R}^{(n_1-s)\times s}, \\
\mathbb{M}_{1,3} &:= M|_{\tilde{I}_1 \times \tilde{I}_2} \in \mathbb{R}^{(n_1-s)\times(n_2-s)}, & \mathbb{M}_{2,1} &:= M|_{I_{1,2} \times \tilde{I}_1} \in \mathbb{R}^{s\times(n_1-s)}, \\
\mathbb{M}_{2,2} &:= M|_{I_{1,2} \times I_{1,2}} \in \mathbb{R}^{s\times s}, & \mathbb{M}_{2,3} &:= M|_{I_{1,2} \times \tilde{I}_2} \in \mathbb{R}^{s\times(n_2-s)}, \\
\mathbb{M}_{3,1} &:= M|_{\tilde{I}_2 \times \tilde{I}_1} \in \mathbb{R}^{(n_2-s)\times(n_1-s)}, & \mathbb{M}_{3,2} &:= M|_{\tilde{I}_2 \times I_{1,2}} \in \mathbb{R}^{(n_2-s)\times s}, \\
\mathbb{M}_{3,3} &:= M|_{\tilde{I}_2 \times \tilde{I}_2} \in \mathbb{R}^{(n_2-s)\times(n_2-s)},
\end{aligned}$$

and:

$$H_{l+1}^{1,1} := H_{l+1}|_{\tilde{I}_1}, \quad H_{l+1}^{2,2} := H_{l+1}|_{\tilde{I}_2}, \quad H_{l+1}^{1,2} := H_{l+1}|_{I_{1,2}}, \tag{3.166}$$

where

$$\tilde{I}_1 = I_1 \setminus I_{1,2}, \quad \tilde{I}_2 = I_2 \setminus I_{1,2}, \tag{3.167}$$

and $n_1 - \frac{\delta}{2} = |\tilde{I}_1|$, $n_2 - \frac{\delta}{2} = |\tilde{I}_2|$. *Then*

$$\begin{aligned} M_1 &:= \begin{bmatrix} \mathbb{M}_{1,1} & \mathbb{M}_{1,2} \\ \mathbb{M}_{2,1} & \mathbb{M}_{2,2} \end{bmatrix}, \quad M_{1,2} := \begin{bmatrix} 0 & \mathbb{M}_{1,3} \\ 0 & \mathbb{M}_{2,3} \end{bmatrix}, \\ M_{2,1} &:= \begin{bmatrix} \mathbb{M}_{2,1} & 0 \\ \mathbb{M}_{3,1} & 0 \end{bmatrix}, \quad M_2 := \begin{bmatrix} \mathbb{M}_{2,2} & \mathbb{M}_{2,3} \\ \mathbb{M}_{3,2} & \mathbb{M}_{2,3} \end{bmatrix}. \end{aligned} \tag{3.168}$$

and

$$H_{k+1}^1 := \begin{bmatrix} H_{k+1}^{1,1} & \frac{1}{2}H_{k+1}^{1,2} \end{bmatrix} \quad H_{k+1}^2 := \begin{bmatrix} \frac{1}{2}H_{k+1}^{1,2} & H_{k+1}^{2,2} \end{bmatrix}. \tag{3.169}$$

As initial value of reduced models on adjacent time subsets, DD-KF uses local KF estimates computed at previous step. In particular, at $t_0 \equiv 0$, DD-KF in $\Delta_1 := [t_0, t_{s_1}]$ employs as initial value $x_0 \equiv \widehat{x}_0$ which is the initial condition of the dynamic model appropriately restricted to Ω_1 or Ω_2. Hence, we get both boundary and initial conditions of reduced model by using KF method, as given in the following definition. That said, we now consider local KF in $\Delta_k \subset \Delta$ at time $t_{l+1} \in \Delta_k$, $\forall\, l = \bar{s}_{k-1}, \ldots, \bar{s}_{k-1} + s_k - 1$.

Definition 15. *(**Local DD-KF** problems) DD-KF problems in* $\Delta_k \times \Omega_1$ *and* $\Delta_k \times \Omega_2$ *are:*

$$P_{\Omega_1}^{\Delta_k} : \begin{cases} x_{1,l+1}^{\Delta_k} = M_1 x_{1,l}^{\Delta_k} + b_l|_{I_1} + b_{1,l} + w_l|_{I_1} \\ y_{l+1} = H_{l+1}^1 x_{1,l+1}^{\Delta_k} + H_{l+1}^2 x_{2,l+1}^{\Delta_k} + v_{l+1} \end{cases} \tag{3.170}$$

and

$$P_{\Omega_2}^{\Delta_k} : \begin{cases} x_{2,l+1}^{\Delta_k} = M_{2,1} x_{1,l}^{\Delta_k} + b_l|_{I_2} + b_{2,l} + w_l|_{I_2} \\ y_{l+1} = H_{l+1}^1 x_{1,l+1}^{\Delta_k} + H_{l+1}^2 x_{2,l+1}^{\Delta_k} + v_{l+1} \end{cases}, \tag{3.171}$$

with initial states (according to PinT approach)

$$x_{1,\bar{s}_{k-1}}^{\Delta_k} = x_{1,\bar{s}_{k-1}}^{\Delta_{k-1}} \quad x_{2,\bar{s}_{k-1}}^{\Delta_k} = x_{2,\bar{s}_{k-1}}^{\Delta_{k-1}} \tag{3.172}$$

in particular, $x_{1,\bar{s}_0}^{\Delta_0} := x_0|_{I_1}$, $x_{2,\bar{s}_0}^{\Delta_0} := x_0|_{I_2}$ *and boundary conditions (according to Schwarz DD)*

$$\begin{aligned} x_{1,l}^{\Delta_k}|_{\partial\Omega_1\setminus\Omega_2} &= x_l^{\partial\Omega_1\setminus\Omega_2} & x_{2,l}^{\Delta_k}|_{\partial\Omega_2\setminus\Omega_1} &= x_l^{\partial\Omega_2\setminus\Omega_1} \\ x_{1,l}^{\Delta_k}|_{\Gamma_1} &= x_{2,l}|_{\Gamma_1} & x_{2,l}^{\Delta_k}|_{\Gamma_2} &= x_{1,l}|_{\Gamma_2} \end{aligned} \tag{3.173}$$

where

$$\Gamma_1 := \partial\Omega_1 \cap \Omega_2 \quad , \quad \Gamma_2 := \partial\Omega_2 \cap \Omega_1 \tag{3.174}$$

and

$$b_{1,k} := \begin{bmatrix} \mathbb{M}_{1,3} \\ \mathbb{M}_{2,3} \end{bmatrix} x_{2,l}^{\Delta_k}|_{\Gamma_1} \quad b_{2,l} := \begin{bmatrix} \mathbb{M}_{2,1} \\ \mathbb{M}_{3,1} \end{bmatrix} x_{1,l}^{\Delta_k}|_{\Gamma_2}. \tag{3.175}$$

$H_l^1 \in \mathbb{R}^{m\cdot d\times n_1}$ *and* $H_l^2 \in \mathbb{R}^{m\cdot d\times n_2}$ *are defined in (3.169) and, , for* $i = 1, 2$, $x_{\bar{s}_{l-1}}^{\Delta_{k-1}}|_{I_i}$, $w|_{I_i}$ *are reductions to* I_i *of* $\widehat{x}_0 \in \mathbb{R}^{N_p}$ *and* $w \in \mathbb{R}^{N_p}$.

Remark 8. *(**Filter Localization**): We underline that we will consider the errors*

$$e_{1,l+1} := (x_{1,l+1} - x_{1,l}) \in \mathbb{R}^{n_1}$$

defined in Ω_1 *and*

$$e_{2,k+1} := (x_{2,l+1} - x_{1,l}) \in \mathbb{R}^{n_2}$$

defined in Ω_2 *and consequently, we get to the local covariance matrices*

$$P_{i,j} := Cov(e_{i,l+1}, e_{j,l+1})$$

defined in Ω_i *and* Ω_j, $i, j = 1, 2$ *with* $i \neq j$ *and* $\forall l = \bar{s}_{k-1}, \ldots, \bar{s}_{k-1} + s_k - 1$.

Furthermore, we will consider the matrices

$$\begin{aligned} B_l &= diag(B_{1,l}, B_{2,l}) \in \mathbb{R}^{N_p\times N_p} \\ R_l &= D_l \end{aligned} \tag{3.176}$$

which are local covariance matrices of errors on model and on observations where, $\forall l = \bar{s}_{k-1}, \ldots, \bar{s}_{k-1} + s_k - 1$, $B_{1,l} \in \mathbb{R}^{n_1 \times n_1}$ *and* $B_{2,l} \in \mathbb{R}^{n_2 \times n_2}$. *Finally,* $D_l \in \mathbb{R}^{m \cdot d \times m \cdot d}$ *is a diagonal matrix.*

Now, we describe DD–KF solving $P_{\Omega_1}^{\Delta_k}$ and $P_{\Omega_2}^{\Delta_k}$ in (3.170) and (3.171), respectively.
In the following we let Δ_k be fixed, then for simplicity of notation, we refer to $\widehat{x}_{i,l}^{\Delta_k}$ omitting superscript Δ_k. DD–KF at $t_{l+1} \in \Delta_k$, $\forall l = \bar{s}_{k-1}, \ldots, \bar{s}_{k-1} + s_k - 1$ consists of:

- Predictor phase. Computation of state estimates:

$$\begin{aligned} x_{1,l+1} &= M_1 \widehat{x}_{1,l} + b_l|_{I_1} + b_{1,l} + w_k|_{I_1} \\ x_{2,l+1} &= M_2 \widehat{x}_{2,l} + b_l|_{I_2} + b_{2,l} + w_k|_{I_2} \end{aligned} ; \tag{3.177}$$

  where

$$b_{1,l} := \begin{bmatrix} \mathbb{M}_{1,3} \\ \mathbb{M}_{2,3} \end{bmatrix} \widehat{x}_{2,l}|_{\Gamma_1} \quad b_{2,l} := \begin{bmatrix} \mathbb{M}_{2,1} \\ \mathbb{M}_{3,1} \end{bmatrix} \widehat{x}_{1,l}|_{\Gamma_2} \tag{3.178}$$

  computation of local error covariance matrices between Ω_1 and Ω_2.

$$\begin{aligned} P_{1,2} &= M_1 P_{1,2} M_2^T + C_{\Omega_1 \leftrightarrow \Omega_2} \\ P_{2,1} &= M_2 P_{2,1} M_1^T + C_{\Omega_1 \leftrightarrow \Omega_2}^T \end{aligned} . \tag{3.179}$$

  and

$$\begin{aligned} P_1 &= M_1 P_1 M_1^T + P_{\Omega_1 \leftrightarrow \Omega_2} + Q_{1,k} \\ P_2 &= M_2 P_2 M_2^T + P_{\Omega_2 \leftrightarrow \Omega_1} + Q_{2,k} \end{aligned} . \tag{3.180}$$

  where

$$C_{\Omega_1 \leftrightarrow \Omega_2} = M_1 P_1 M_{2,1}^T + M_{1,2} P_{2,1} M_{2,1}^T + M_{1,2} P_2 M_2^T \ .$$

  and

$$\begin{aligned} P_{\Omega_1 \leftrightarrow \Omega_2} &= M_{1,2} P_{2,1} M_1^T + M_1 P_{1,2} M_{1,2}^T + M_{1,2} P_2 M_{1,2}^T \\ P_{\Omega_2 \leftrightarrow \Omega_1} &= M_{2,1} P_{1,2} M_2^T + M_2 P_{2,1} M_{2,1}^T + M_{2,1} P_1 M_{2,1}^T \end{aligned} ; \tag{3.181}$$

keep track of contribution of Ω_1 and Ω_2 to overlapping region.

- Corrector phase. Update of DD–KF gains:

$$\begin{aligned} \mathbf{K}_1 &= (P_1 H_{l+1}|_{I_1}^T + P_{1,2} H_{l+1}|_{I_2}^T) \cdot F \\ \mathbf{K}_2 &= (P_2 H_{l+1}|_{I_2}^T + P_{2,1} H_{l+1}|_{I_1}^T) \cdot F \end{aligned} ; \tag{3.182}$$

where

$$F = (H_{l+1}|_{I_1} P_1 H_{l+1}|_{I_1}^T + H_{l+1}|_{I_2} P_2 H_{l+1}|_{I_2}^T + R_{1,2} + R_{l+1})^{-1} \tag{3.183}$$

and

$$R_{1,2} = (H_{l+1}|_{I_2} P_{2,1} H_{l+1}|_{I_1}^T + H_{l+1}|_{I_1} P_{1,2} H_{l+1}|_{I_2}^T); \tag{3.184}$$

keep track of contribution of Ω_1 and Ω_2 to overlapping region;

update of local covariance matrices:

$$\begin{aligned} P_1 &= (I - \mathbf{K}_1 H_{l+1}|_{I_1}) P_1 - \mathbf{K}_1 H_{l+1}|_{I_2} P_{2,1} \\ P_2 &= (I - \mathbf{K}_2 H_{l+1}|_{I_2}) P_2 - \mathbf{K}_2 H_{l+1}|_{I_1} P_{1,2} \end{aligned} ; \tag{3.185}$$

update of local covariance matrices between $e_1 \in \mathbb{R}^{n_1}$ and $e_2 \in \mathbb{R}^{n_2}$.

$$\begin{aligned} P_{1,2} &= (I - \mathbf{K}_1 H_{l+1}|_{I_1}) P_{1,2} - \mathbf{K}_1 H_{l+1}|_{I_2} P_2 \\ P_{2,1} &= (I - \mathbf{K}_2 H_{l+1}|_{I_2}) P_{2,1} - \mathbf{K}_2 H_{l+1}|_{I_1} P_1 \end{aligned} . \tag{3.186}$$

Finally, we get to the update of DD-KF estimates:

$$\begin{aligned} \widehat{x}_{1,l+1} &= x_{1,k+1} + \mathbf{K}_1 \left[y_{l+1} - (H_{l+1}|_{I_1} x_{1,l} + H_{l+1}|_{I_2} x_{2,l}) \right] \\ \widehat{x}_{2,l+1} &= x_{2,l+1} + \mathbf{K}_2 \left[y_{l+1} - (H_{l+1}|_{I_1} x_{1,l} + H_{l+1}|_{I_2} x_{2,l}) \right] \end{aligned} . \tag{3.187}$$

We underline that if observations are concentrated in spatial subdomains such that the first m_1 and the last m_2 are in Ω_1 and Ω_2, respectively, then at step $k+1$ these observations influence the

first n_1 and the last n_2 components of DD–KF estimates $\widehat{x}_{1,l+1}$ and $\widehat{x}_{2,l+1}$, respectively. Consequently, matrices $H_{l+1}|_{I_i}$ have $m - m_i$ null rows and then they can be reduced to $m_i \times n_i$ matrices, with $i = 1, 2$.

Remark 9. *(Filter Localization): P. L. Houtekamer and H. L. Mitchell [66] made a modification to EnKF to localize covariances matrices and consequently the Kalman gain using a Schur product* [4]*. By using suitably correlation functions ρ_1 and ρ_2 at each step $l + 1$, KF gains $K_1 = [(\rho_1 \circ (P_{l+1}H_{l+1}^T)][\rho_1 \circ (H_{l+1}P_{l+1}H_{l+1}^T) + R_{l+1}]$ and $K_2 = [(\rho_2 \circ (P_{l+1}H_{l+1}^T)][\rho_2 \circ (H_{l+1}P_{l+1}H_{l+1}^T) + R_{l+1}]$ are equivalent to DD-KF gains $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ in (3.182). More precisely, if we choose $\rho_1 := \mathbb{1}_{\Omega_1}$ and $\rho_2 := \mathbb{1}_{\Omega_2}$, i.e. the indicator functions of Ω_1 and Ω_2 respectively, it follows $K_1 \equiv \boldsymbol{K}_1$ and $K_2 \equiv \boldsymbol{K}_2$, i.e. DD-KF is mathematically equivalent to KF with localization.*

3.5.2 DD-KF method in $\{\Omega_i \times \Delta_k\}_{i=1,...,N_{sub};k=1,...,Nt}$

DD-KF can be generalised to $N_{sub} > 2$ spatial subdomains. Starting from the decomposition step in space and time given in Section 3.3, with $N_{sub} > 2$ spatial subdomains consecutively arranged along a one dimensional direction[5], and assuming that $M_{k,k+1}$ is block tridiagonal (such the matrix in (A.20) arising from discretization of SWEs described in section 5) Model Reduction step, as given in Definition 1, still holds where instead of (3.161) and (3.162) we will

[4]Schur product, also named Hadamard product, of two matrices having the same dimensions is denoted $A = B \circ C$ and consists of the element-wise product such that $A_{i,j} = B_{i,j} \circ C_{i,j}$. If B and C are covariance matrices, then so is A

[5]Such assumption allows to simplify the management of the contributions of adjacent domains to overlapping regions. In this case each subdomain overlaps with two domains and two compatibility conditions needs to be satisfied. As more than two overlapping regions occurs as more compatibility conditions should be satisfied.

have respectively:

$$M_{l,l+1} = \begin{bmatrix} M_1 & M_{1,2} & & & 0 \\ M_{2,1} & M_2 & M_{2,3} & & \\ & & & \ddots & \\ & 0 & & & \\ & & & M_{N_{sub},N_{sub}-1} & M_{N_{sub}} \end{bmatrix} \tag{3.188}$$

and

$$H_{l+1} = \left[H_{l+1}^1 \; H_{l+1}^2 \; \cdots \; H_{l+1}^{N_{sub}} \right]. \tag{3.189}$$

In the same way of (3.170) and (3.171), we get N_{sub} local problems $P_{\Omega_i}^{\Delta_k}$, in the i-th subdomain, for $i = 1, \ldots, N_{sub}$. DD-KF step at $t_{l+1} \in \Delta_k$, $\forall l = \bar{s}_{k-1}, \ldots, \bar{s}_{k-1} + s_k - 1$ provides solutions of local problems in the following steps.

Computation of state estimates:

$$x_{i,l+1} = M_h \widehat{x}_{i,l} + b_l|_{I_i} + b_{i,l} + w_k|_{I_i}$$

where

$$b_{i,l} = \begin{cases} M_{i,i+1}\widehat{x}_{i+1,l}|_{\Gamma_i} & \text{if } i = 1 \\ M_{i,i-1}\widehat{x}_{i-1,k}|_{\Gamma_i} & \text{if } i = N_{sub} \\ M_{i,i+1}\widehat{x}_{i+1,k}|_{\Gamma_i} + M_{i,i-1}\widehat{x}_{i-1,k}|_{\Gamma_i} & \text{otherwise} \end{cases};$$

for $h = 1, \ldots, N_{sub}$, $h \neq i$ and

$$\begin{array}{lll} j = i+1 & if & i = 1 \\ j = i-1 & if & i = N_{sub} \\ j = \{i+1, i-1\} & if & 1 < i < N_{sub} \end{array} ; \tag{3.190}$$

computation of local error covariance matrices between Ω_i and Ω_h

$$P_{i,h} = M_i P_{i,h} M_h^T + C_{\Omega_i \leftrightarrow \Omega_h},$$

and

$$P_i = M_i P_i M_i^T + P_{\Omega_i \leftrightarrow \Omega_j} + B_{i,k};$$

where

$$C_{\Omega_i \leftrightarrow \Omega_h}{}^{6} = \begin{cases} (M_i P_{i,h-1} + M_{i,i+1} P_{i+1,h-1}) M_{h,h-1}^T + M_{i,i+1} P_{i+1,h} M_h^T & \text{if } i = 1;\ h = N_{sub} \\ (M_i P_{i,h-1} + M_{i,i+1} P_{i+1,h-1}) M_{h,h-1}^T + M_{i,i+1} P_{i+1,h} M_h^T & \\ +(M_i P_{i,h+1} + M_{i,i+1} P_{i+1,h+1}) M_{h,h+1}^T & \text{if } i = 1;\ h \neq N_{sub} \\ (M_i P_{i,h+1} + M_{i,i-1} P_{i-1,h+1}) M_{h,h+1}^T + M_{i,i-1} P_{i-1,h} M_h^T & \text{if } i = N_{sub};\ h = 1 \\ (M_i P_{i,h+1} + M_{i,i-1} P_{i-1,h+1}) M_{h,h+1}^T + M_{i,i-1} P_{i-1,h} M_h^T & \\ +(M_i P_{i,h-1} + M_{i,i+1} P_{i-1,h-1}) M_{h,h-1}^T & \text{if } i = N_{sub};\ h \neq 1 \\ (M_i P_{i,h+1} + M_{i,i-1} P_{i-1,h+1} + M_{i,i+1} P_{i+1,h+1}) M_{h,h+1}^T & \\ +(M_{i,i+1} P_{i+1,h} + M_{i,i-1} P_{i-1,h}) M_h^T & 1 < i < N_{sub};\ h = 1 \\ (M_i P_{i,h-1} + M_{i,i-1} P_{i-1,h-1} + M_{i,i+1} P_{i+1,h-1}) M_{h,h-1}^T & \\ +(M_{i,i+1} P_{i+1,h} + M_{i,i-1} P_{i-1,h}) M_h^T & 1 < i < N_{sub};\ h = N_{sub} \\ (M_i P_{i,h+1} + M_{i,i-1} P_{i-1,h+1} + M_{i,i+1} P_{i+1,h+1}) M_{h,h+1}^T & \\ +(M_{i,i+1} P_{i+1,h} + M_{i,i-1} P_{i-1,h}) M_h^T & \text{otherwise} \\ +(M_i P_{i,h-1} + M_{i,i-1} P_{i-1,h-1} + M_{i,i+1} P_{i+1,h-1}) M_{h,h-1}^T & \end{cases}$$

and

$$P_{\Omega_i \leftrightarrow \Omega_j} = \begin{cases} M_{i,i+1} P_{i+1,i} M_i^T + M_i P_{i,i+1} M_{i,i+1}^T + M_{i,i+1} P_{i+1} M_{i,i+1}^T & \text{if } i = 1 \\ M_{i,i-1} P_{i-1,i} M_i^T + M_i P_{i,i-1} M_{i,i-1}^T + M_{i,i-1} P_{i-1} M_{i,i-1}^T & \text{if } i = N_{sub} \\ (M_{i,i+1} P_{i+1,i} + M_{i,i-1} P_{i-1,i}) M_i^T + M_i P_{i,i-1} M_{i,i-1}^T & \\ +M_i P_{i,i+1} M_{i,i+1}^T + (M_{i,i+1} P_{i+1,i-1} + M_{i,i-1} P_{i-1}) M_{i,i-1}^T & \text{otherwise} \\ +(M_{i,i+1} P_{i+1} + M_{i,i-1} P_{i-1,i+1}) M_{i,i+1}^T & \end{cases}$$

[6] In computation of matrix $C_{\Omega_i \leftrightarrow \Omega_h}$, we refer to covariance matrix P_i instead of $P_{i,i}$.

are the matrices keeping track of contributions of adjacent domains Ω_i and Ω_h to overlapping region; update of DD-KF gains:

$$\mathbf{K}_i = \sum_{j=1 j\neq i}^{N_{sub}} (P_i H_{l+1}^T|_{I_i} + P_{i,j} H_{l+1}^T|_{I_j}) \cdot F;$$

where

$$F = (\sum_{i=1}^{N_{sub}} H_{l+1}|_{I_i} P_i H_{l+1}|_{I_i}^T + R_{1,\dots,N_{sub}} + R_{k+1})^{-1}$$

and

$$R_{1,\dots,N_{sub}} = \sum_{i=1}^{N_{sub}} (\sum_{j=1 j\neq i}^{N_{sub}} H_{l+1}|_{I_j} P_{j,i}) H_{l+1}|_{I_i}^T;$$

update of local covariance matrices:

$$P_i = (I_i - \mathbf{K}_i H_{l+1}|_{I_i}) P_i - \sum_{j=1 j\neq i}^{N_{sub}} (\mathbf{K}_i H_{l+1}|_{I_j} P_{j,i});$$

update of local covariance matrices between $e_i \in \mathbb{R}^{n_i}$ and $e_h \in \mathbb{R}^{n_h}$.

$$P_{i,h} = (I_i - \mathbf{K}_i \cdot H_{l+1}|_{I_i}) P_{i,h} - \sum_{j=1 j\neq i}^{N_{sub}} (\mathbf{K}_i H_{k+1}|_{I_j} P_{j,h}), \quad h = 1, \dots, N_{sub} \quad h \neq i;$$

finally, we get to the update of local estimates:

$$\widehat{x}_{i,l+1} = x_{i,k+1} + \mathbf{K_i} \cdot (y_{l+1} - \sum_{j=1 j\neq i}^{N_{sub}} H_{l+1}|_{I_j} x_{j,l}). \tag{3.191}$$

3.5.3 Algorithm

DD-KF algorithm is described below.

Table 3.10: DD-KF procedure.

procedure DD-KF(in: $H_0, \ldots, H_{lmax}, Q_0, \ldots, Q_{lmax-1}$,$R_0, \ldots, R_{lmax}, y_0, \ldots,$ y_{lmax},μ, δ, out: $\widehat{x}_i$)

Set index j of I_j, i.e. the set adjacent to I_i

if (*floor*$(i/2) \times 2 == i$) **then**

$j := i - 1$ % i is even

else

$j := i + 1$ % i is odd

end

Call DD-Setup (in: $y_0, H_0|_{I_i}, H_0|_{I_j}, R_0$,out: $\widehat{x}_{i,0}, P_{i,0}$)

Call Local-KF (in: y_l, $H_{l-1}|_{I_i}, H_k|_{I_i}, B_{l-1}|_{I_i}, R_l, P_{i,0}$,$\mu$, δ, $\widehat{x}_{i,l-1},$ out: $\widehat{x}_{i,l}^n$)

Set $\widehat{x}_i := \widehat{x}_{i,l}^n$ % DD-KF estimate in I_i

endprocedure

Table 3.11: DD-Setup procedure.

procedure DD-Setup(in: $y_0, H_0|_{I_i}, H_0|_{I_j}, R_0$, out: $\widehat{x}_{i,0}, P_{i,0}$)

Set up reduced matrices: $H_{i,0} := H_0|_{I_i}$, $H_{j,0} := H_0|_{I_j}$ as in (3.123)

Compute predicted covariance matrix $P_{i,0} = (H_{i,0}^T R_0 H_{i,0})^{-1}$ as in (3.124)

Compute $P_{H_{j,0}}$ as in (3.125)

Compute Kalman estimate: $\widehat{x}_{i,0} = (H_{i,0}^T P_{H_{j,0}} H_{i,0})^{-1} H_{i,0}^T P_{H_{j,0}} y_0$ as in (3.126)

Table 3.12: Local-KF procedure.

procedure Local-KF(in: y_k, $H_{l-1}|_{I_i}, H_l|_{I_i}, B_{l-1}|_{I_i}, R_l, P_{i,0}$,$\mu$, δ, $\widehat{x}_{i,l-1}$; out: $\widehat{x}_{i,l}^n$)

for $l = 1, lmax$ %loop over KF steps

$n := 0$, $\widehat{x}_{i,l}^{n+1} := 0$

repeat

$n := n + 1$

Set up predicted covariance matrix $P_{i,l}$ as in (3.132)

Compute Kalman gains $K_{i,l}$ as in (3.133)

Update covariance matrix $P_{i,l}$ as in (3.134)

Compute matrices $K_{i,j}$ as in (3.135)

Send and Receive boundary conditions among adjacent sets

Exchange of data among $S_{I_i \leftrightarrow I_j}(\widehat{x}^n_{i,l})$ as in (3.137)

Compute Kalman estimate at step $n + 1$

$\widehat{x}^{n+1}_{i,l} = \widehat{x}_{i,l-1} + \mathbf{K}_{i,l}\left[(y_l - H_l|_{I_j}\widehat{x}^n_{j,l}) - H_l|_{I_i}\widehat{x}_{i,l-1}\right]$

$+\mathcal{S}_{I_i \leftrightarrow I_j}(\widehat{x}^n_{j,l})$ as in (3.136)

if($s \neq 0$) **then** % decomposition with overlap

Set up of the extensions on I_i of $\widehat{x}^{n+1}_{i,l}$ given on $I_{i,j}$:

$EO_{I_i}(\widehat{x}^n_{i,l}|_{I_{i,j}})$, $EO_{I_i}(\widehat{x}^n_{j,l}|_{I_{i,j}})$

Exchange data on the overlap set $I_{i,j}$:

$\nabla \mathcal{O}_{i,j}(EO_{I_i}(\widehat{x}^n_{i,l}|_{I_{i,j}}), EO_{I_i}(\widehat{x}^n_{j,l}|_{I_{i,j}}))$ as in (3.139)

Update Kalman estimate at step $n + 1$:

$\widehat{x}^{n+1}_{i,k} \leftarrow \widehat{x}^{n+1}_{i,l} + \mu \cdot P_{i,k}\nabla \mathcal{O}_{i,j}(EO_{I_i}(\widehat{x}^n_{i,l}|_{I_{i,j}}), EO_{I_i}(\widehat{x}^n_{j,l}|_{I_{i,l}}))$ as in (3.138)

endif

until ($\|\widehat{x}^{n+1}_{i,l} - \widehat{x}^n_{i,l}\| < TOL$)

endfor % end of the loop over KF steps

end procedure

3.5.4 Reliability assessment

In this section we assess reliability of the proposed method. In particular, Theorem 8 proves that state estimates $\widehat{x}_{1,l+1} \in \mathbb{R}^{n_1}$, $\widehat{x}_{2,l+1} \in \mathbb{R}^{n_2}$, as in (3.187), are equal to reductions of KF estimate $\widehat{x}_{l+1} \in \mathbb{R}^{N_p}$ to I_1 and I_2, respectively. This means that the DD not modify the analytical results of the conventional KF method. Same result can be obtained for $N_{sub} > 2$ spatial subdomains considering DD-KF estimates defined in (3.191) and proceeding as in proof of Theorem 8.

Theorem 8. *Let $\widehat{x}_{l+1} \in \mathbb{R}^{N_p}$, $\forall l = \bar{s}_{k-1}, \dots, \bar{s}_{k-1} + s_k - 1$ be KF estimate in (2.20) and $\widehat{x}_{1,l+1} \in \mathbb{R}^{n_1}$, $\widehat{x}_{2,l+1} \in \mathbb{R}^{n_2}$ be the DD-KF estimates in (3.187). Then it holds that*

$$\widehat{x}_{1,l+1} \equiv \widehat{x}_{l+1}|_{I_1}, \quad \widehat{x}_{2,l+1} \equiv \widehat{x}_{l+1}|_{I_2}. \tag{3.192}$$

Proof. For $l = \bar{s}_{k-1}, \dots, \bar{s}_{k-1} + s_k - 1$, we prove that DD-KF gains $\mathbf{K}_1 \in \mathbb{R}^{n_1 \times m \cdot d}$, $\mathbf{K}_2 \in \mathbb{R}^{n_2 \times m \cdot d}$ in (3.182) are

$$\mathbf{K}_1 \equiv \mathbf{K}_{l+1}|_{I_1} \quad \mathbf{K}_2 \equiv \mathbf{K}_{l+1}|_{I_2}, \tag{3.193}$$

where $\mathbf{K}_{l+1} \in \mathbb{R}^{N_p \times m \cdot d}$ is the KF gain in (2.18). We first consider DD-KF without overlapping. KF gain can be written as follows

$$\begin{aligned} \mathbf{K}_{l+1} &\equiv P_{l+1} H_{l+1}^T \cdot (H_{l+1} P_{k+1} H_{l+1}^T + R_{l+1})^{-1} \\ &= \begin{bmatrix} P_{l+1}|_{I_1 \times I_1} & P_{l+1}|_{I_1 \times I_2} \\ P_{l+1}|_{I_2 \times I_1} & P_{l+1}|_{I_2 \times I_2} \end{bmatrix} \begin{bmatrix} H_{l+1}|_{I_1}^T \\ H_{l+1}|_{I_2}^T \end{bmatrix} \\ &\cdot \left(\begin{bmatrix} H_{l+1}|_{I_1} & H_{l+1}|_{I_2} \end{bmatrix} \begin{bmatrix} P_{l+1}|_{I_1 \times I_1} & P_{l+1}|_{I_1 \times I_2} \\ P_{l+1}|_{I_2 \times I_1} & P_{l+1}|_{I_2 \times I_2} \end{bmatrix} \begin{bmatrix} H_{l+1}|_{I_1}^T \\ H_{l+1}|_{I_2}^T \end{bmatrix} + R_{l+1} \right)^{-1} \end{aligned}, \tag{3.194}$$

where $P_{l+1} \in \mathbb{R}^{N_p \times N_p}$ is the predicted covariance matrix, which is defined in (2.17). We define the matrix

$$R_{1,2} := (H_{l+1}|_{I_2} P_{l+1}|_{I_2 \times I_1} H_{l+1}|_{I_1}^T + H_{l+1}|_{I_1} P_{l+1}|_{I_1 \times I_2} H_{l+1}|_{I_2}^T); \tag{3.195}$$

and we obtain that

$$\begin{aligned}\mathbf{K}_{l+1}|_{I_1} &= (P_{l+1}|_{I_1\times I_1}H_{l+1}|_{I_1}^T + P_{l+1}|_{I_1\times I_2}H_{l+1}|_{I_2}^T)\\ &\cdot(H_{l+1}|_{I_1}P_{l+1}|_{I_1\times I_1}H_{l+1}|_{I_2}^T + H_{l+1}|_{I_2}P_{l+1}|_{I_2\times I_2}H_{l+1}|_{I_2}^T + R_{1,2} + R_{l+1})^{-1}\\ \mathbf{K}_{l+1}|_{I_2} &= (P_{l+1}|_{I_2\times I_2}H_{l+1}|_{I_2}^T + P_{l+1}|_{I_2\times I_1}H_{l+1}|_{I_1}^T)\\ &\cdot(H_{l+1}|_{I_1}P_{l+1}|_{I_1\times I_1}H_{l+1}|_{I_2}^T + H_{l+1}|_{I_2}P_{l+1}|_{I_2\times I_2}H_{l+1}|_{I_2}^T + R_{1,2} + R_{l+1})^{-1}\end{aligned} \quad . \tag{3.196}$$

We similarly obtain that

$$\begin{aligned}P_{l+1}|_{I_1\times I_1} \equiv P_1 \quad P_{l+1}|_{I_1\times I_2} \equiv P_{1,2}\\ P_{l+1}|_{I_2\times I_2} \equiv P_2 \quad P_{l+1}|_{I_2\times I_1} \equiv P_{2,1}\end{aligned} \tag{3.197}$$

where P_1, P_2 are defined in (3.185) and $P_{1,2}$, $P_{1,2}$ in (3.186). From (3.196) and (3.182) we obtain the equivalence in (3.192). We consider the predicted estimate x_{l+1} in (2.16) so, KF estimate $\widehat{x}_{l+1} \in \mathbb{R}^n$ in (2.20) can be written as follows

$$\begin{bmatrix}\widehat{x}_{l+1}|_{I_1}\\ \widehat{x}_{l+1}|_{I_2}\end{bmatrix} = \begin{bmatrix}x_{l+1}|_{I_1}\\ x_{l+1}|_{I_2}\end{bmatrix} + \begin{bmatrix}\mathbf{K}_1\\ \mathbf{K}_2\end{bmatrix}\left(y_{l+1} - \begin{bmatrix} H_{l+1}|_{I_1} & H_{l+1}|_{I_2}\end{bmatrix}\begin{bmatrix}x_l|_{I_1}\\ x_l|_{I_2}\end{bmatrix}\right). \tag{3.198}$$

It is simply to prove that

$$x_{l+1}|_{I_1} \equiv x_{1,l+1} \quad x_{l+1}|_{I_2} \equiv x_{2,l+1},$$

where $x_{1,l+1} \in \mathbb{R}^{n_1}$ and $x_{2,k+1} \in \mathbb{R}^{n_2}$ are the predicted estimates in (3.187), so we get the thesis in (3.192).

In case of decomposition of Ω with overlap, DD-KF estimates in (3.187) can be written as follows: for $i = 1, 2$

$$\widehat{x}_{i,l+1} = x_{i,l+1} + \mathbf{K}_i[y_{l+1} - (H_{l+1}|_{\tilde{I}_1}x_{1,l}|_{\tilde{I}_1} + \frac{1}{2}H_{l+1}|_{I_{1,2}}x_{1,l}|_{\tilde{I}_{1,2}} + \frac{1}{2}H_{l+1}|_{I_{1,2}}x_{2,l}|_{\tilde{I}_{1,2}} \tag{3.199}$$

$$+ H_{l+1}|_{\tilde{I}_2}x_{2,l})] \tag{3.200}$$

Since

$$x_{l+1}|_{I_{1,2}} = x_{1,l+1}|_{I_{1,2}} = x_{2,l+1}|_{I_{1,2}}$$

(3.199) becomes

$$\widehat{x}_{i,l+1} \quad = x_{i,l+1} + \mathbf{K}_i \left[y_{l+1} - (H_{l+1}|_{\tilde{I}_1} x_{1,l}|_{\tilde{I}_1} + H_{l+1}|_{I_{1,2}} x_l|_{I_{1,2}} + H_{l+1}|_{\tilde{I}_2} x_{2,l}) \right], \; i = 1, 2 \; . \tag{3.201}$$

Noting that:

$$H_{l+1} = \left[H_{l+1}|_{\tilde{I}_1} \; H_{l+1}|_{I_{1,2}} \; H_{l+1}|_{\tilde{I}_2} \right],$$

(3.199) becomes:

$$\widehat{x}_{l+1}|_{I_i} = x_{l+1}|_{I_i} + \mathbf{K}_i \left[y_{l+1} - (H_{l+1}|_{\tilde{I}_1} x_l|_{\tilde{I}_1} + H_{l+1}|_{I_{1,2}} x_l|_{I_{1,2}} + H_{l+1}|_{\tilde{I}_2} x_l|_{I_2}) \right] \tag{3.202}$$

where I_1, $I_{1,2}$, I_2 and $\tilde{I}_1$, $\tilde{I}_2$ are defined in (3.154) and (3.167), respectively. Moreover, equivalence in (3.193) can be similarly obtained. Supposing (3.192) is true for l, from (3.201) and (3.202), we obtain (3.192) for $l + 1$. □

Chapter 4

Parallel Domain Decomposition

4.1 Dynamic Domain Decomposition in Space (DyDD)

For effective parallelization of DD based algorithms, domain partitioning into subdomains must satisfy certain conditions. Firstly the computational load assigned to subdomains must be equally distributed. Usually, computational cost is proportional to the amount of data entities assigned to partitions. Good quality partitioning also requires the volume of communication during calculation to be kept at its minimum. We employ a dynamic load balancing scheme based on adaptive and dynamic redefining of initial DD aimed to balance workload between processors. Redefining of initial partitioning is performed by shifting the boundaries of neighbouring domains (this step is referred to as Migration step).

DyDD algorithm we implement is described by procedure DyDD shown in Table 4.1. To the aim of giving a clear and immediate view of DyDD algorithm, in the following figures (Figures 4.1-4.4) we outline algorithm workout on a reference initial DD configuration made of eight subdomains. We assume that at each point of the mesh we have the value of numerical simulation result (the so called background) while the circles denote observations. DyDD framework consists in four steps:

1. DD step: starting from the initial partition of Ω provided by DD-DA framework as in Section 3.1), DyDD performs a check of the initial partitioning. If a subdomain is empty, it decomposes subdomain adjacent to that domain which has maximum load (decomposition is performed in 2 subdomains). See Figure 4.1.

2. Scheduling step: DyDD computes the amount of observations needed for achieving the average load in each subdomain; this is performed by introducing a diffusion type algorithm (by using the connected graph G associated to the DD) derived by minimizing the Euclidean norm of the cost transfer. Solution of the laplacian system associated to the graph G gives the amount of data to migrate. See Figure 4.2.

3. Migration step: DyDD shifts the boundaries of adjacent subdomains to achieve a balanced workload. See Figure 4.3.

4. Update step: DyDD redefines subdomains such that each one contains the number of observations computed during the scheduling step and it redistributes subdomains among processors grids. See Figure 4.4.

Scheduling step is the computational kernel of DyDD algorithm. In particular, it requires definition of laplacian matrix and load imbalance associated to initial DD and its solution. Let us give a brief overview of this computation. Generic element L_{ij} of laplacian matrix is defined as follows[67]:

$$L_{ij} = \begin{cases} -1 & i \neq j \, and \, edge\,(i,j) \in G \\ deg(i) & i = j, \\ 0 & otherwise \end{cases} \tag{4.1}$$

and the load imbalance of vertex i, $b(i) = \left(l\,(i) - \overline{l}\right)$, where $deg(i)\,(i)$ is the degree of vertex i, $l\,(i)$ and $\overline{l}$ are the number of observations and the average workload of all vertices, respectively. Hence, as more edges are in G (as the number of subdomains which are adjacent to each other increases) as more non zero elements are in L.

Laplacian system $L\lambda = b$, related to the example of Figure 4.2 described below, is the following:

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 3 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 2 \end{bmatrix} \tag{4.2}$$

while the right hand side is the vector whose i-th component is given by the load imbalance, computed with respect to the average load $\bar{l} = 4$ as follows:

$$b(i) = \begin{bmatrix} 1 \\ 0 \\ 2 \\ -2 \\ 1 \\ -1 \\ 1 \\ -2 \end{bmatrix}.$$

In this example, solution of the laplacian system gives

$$\lambda = (3.43, 2.81, 3.05, 1.96, 2.00, 0., 0., -1)$$

so that the amount of load (rounded to the nearest integer) which should be migrated from Ω_i to Ω_j is

$$\delta_{1,2} = 1; \delta_{1,3} = 0; \delta_{2,4} = 1; \delta_{3,2} = 0; \delta_{3,4} = 1; \delta_{3,5} = 1;$$
$$\delta_{5,4} = 0; \delta_{5,6} = 2; \delta_{6,7} = 0; \delta_{6,8} = 1; \delta_{7,8} = 1;$$

i.e. $\delta_{i,j}$ is the nearest integer of $(\lambda_i - \lambda_j)$.

Table 4.1: Procedure DyDD.

Procedure DyDD-Dynamic Load Balancing(in: p, Ω, out: l_1,...,l_p)

%Procedure DyDD allows to balance observations between adjacent subdomains

% Domain Ω is decomposed in p subdomains and some of them may be empty.

% DyDD procedure is composed by: DD step, Scheduling step and Migration Step.

% DD step partitions Ω in subdomains and if some subdomains have not any observations, partitions adjacent subdomains with maximum load

%in 2 subdomains and redefines the subdomains.

% Scheduling step computes the amount of observations needed for shifting boundaries of neighbouring subdomains

%Migration step decides which subdomains should be reconfigured to achieve a balanced load.

% Finally, the Update step redefines the DD.

DD step

% DD step partitions Ω in $(\Omega_1, \Omega_2, \ldots, \Omega_p)$

Define n_i: the number of adjacent subdomains of Ω_i

Define l_i: the amount of observations in Ω_i

repeat for any i

% identification of Ω_m, the adjacent subdomain of Ω_i with the maximum load

Compute m such that $l_m = max_{j=1,\ldots,n_i}\ (l_j)$: the maximum amount of observations

Decompose Ω_m in 2 subdomains: $\Omega_m \leftarrow (\Omega_m^1, \Omega_m^2)$

until ($l_i \neq 0$)

end of DD Step

Begin Scheduling step

Define G: the graph associated with initial partition: vertex i corresponds to Ω_i

Distribute the amount of observations l_i on Ω_i

Define $deg(i) = n_i$, the degree of node i of G:

repeat

Compute the average load: $\bar{l} = \frac{\sum_{i=1}^{p} l_i}{p}$

Compute load imbalance: $b = (l_i - \bar{l})_{i=1,\dots,p}$

Compute L, Laplacian matrix of G

Call solve(in:L, b, out:λ) % algorithm solving the linear system $L\lambda = b$

Compute $\delta_{i,j}$, the load increment between the adjacent subdomains Ω_i and Ω_j. $\delta_{i,j}$ is the nearest integer of $(\lambda_i - \lambda_j)$

Define n_{s_i}, n_{r_i}, number of those subdomains whose configuration has to be updated

Update graph G

Update amount of observations of Ω_i: $l_i = l_i - \sum_{j=1}^{n_{s_i}} \delta_{i,j} + \sum_{j=1}^{n_{r_i}} \delta_{j,i}, \forall i$

until $(max_i \|l_i - \bar{l}\| == \frac{deg(i)}{2})$ % i.e. maximum load-difference is $deg(i)/2$

end Scheduling step

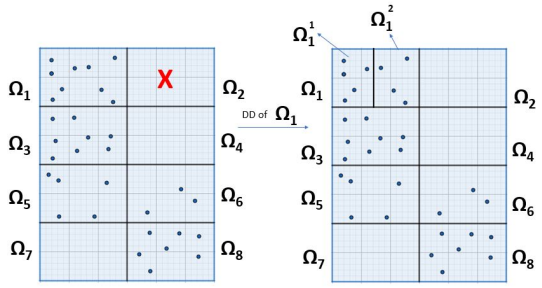
Begin Migration Step

Shift boundaries of two adjacent subdomains in order to achieve a balanced load.

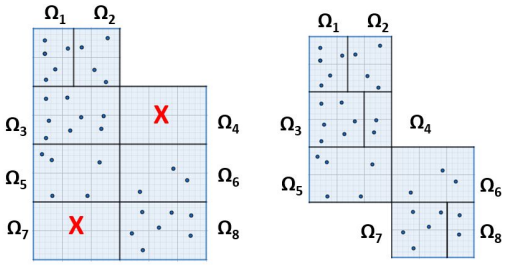
end Migration Step

Update DD of Ω

end Procedure DyDD



(a) Ω_1 is identified as having the maximum load w.r.t. its neighbourhoods.



(b) Ω_4 and Ω_7 are identified as having the maximum load w.r.t. their neighbourhoods.

Figure 4.1: DyDD framework - Step 1. Check of the initial partitioning, identification of subdomains which do not have data or they suffer of any load imbalance and redefinition of subdomains. We observe that the workload of each subdomain after this re-partitioning is now $l_r(1) = 5$, $l_r(2) = 4$, $l_r(3) = 6$, $l_r(4) = 2$, $l_r(5) = 5$, $l_r(6) = 3$, $l_r(7) = 5$ and $l_r(8) = 2$. The average load is then $\overline{l} = 4$.

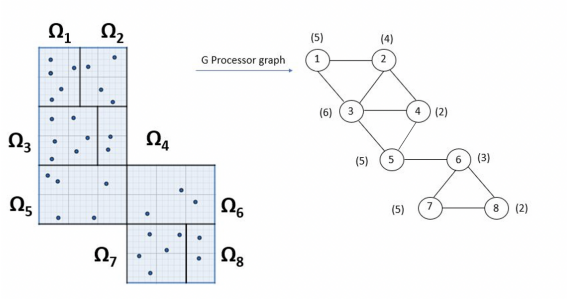


Figure 4.2: DyDD framework - Step 2. Scheduling. On the right, the graph G associated to the DD of Ω. In brackets the number $l_r(i)$ is displayed.

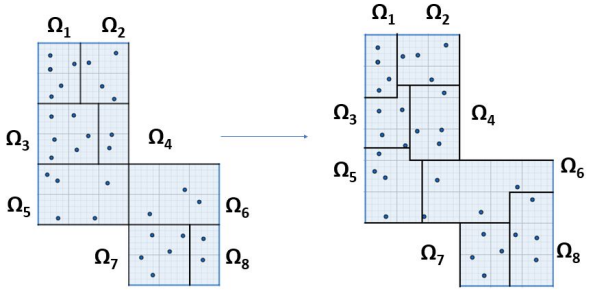


Figure 4.3: DyDD framework - Step 3. Migration. Redefinition of the boundaries of adjacent subdomains.

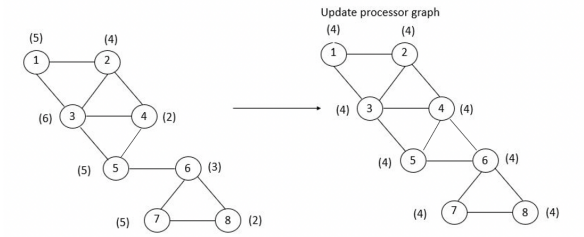


Figure 4.4: DyDD framework - Step 4. Update step. Updating of the processor graph. In brackets, the number of observations $l_{fi}(i)$ after DyDD is displayed. We observe that the workload of each subdomain after DyDD is equal to the average load $\overline{l} = 4$.

4.2 Dynamic Domain Decomposition in Space and Time (DyDDST)

In many problems within the earth and environmental sciences observations are non uniformly distributed and its distribution change during time. DYDDST algorithm is proposed to support real time applications where load measurement is necessary to determine when load imbalance occurs. DyDDST is an extension to time windows of DyDD introduced in Section 4.1. We apply DyDDST on algorithm proposed in [8] in order to ensure a balanced distribution of load between spatial subdomains in each time interval. The load balancing scheme proposed involves, at each time interval, an adaptive and dynamic repartitioning of load among spatial subdomains. Load repartition is performed by shifting boundaries of adjacent subdomains defined by the initial domain partitioning. As shown in Algorithm 4.2, DyDDST framework consists in five steps:

- DD check: starting from the initial DD of $\Omega \times \Delta$ provided by DD–DA framework as in Section 3.1, DyDDST performs a check of the partitioning. If a spatial subdomain is empty, it decomposes in two subdomains the adjacent subdomain which has the maximum load.

- Scheduling step: DyDDST computes the amount of observations needed for achieving, in each subdomain Ω_i, the average load in Δ_k; this is performed by using the connected graph G^k associated to the DD of Ω in Δ_k; i.e. G^k depends on the configuration in Δ_k of spatial subdomains. This is achieved computing the Laplace matrix $L^k = \{L^k_{ij}\}$ as follows [8]:

$$L^k_{i,j} = \begin{cases} -1 & i \neq j \text{ and } (i,j) \in G^k \\ d^k(i) & i = j \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

  and the load imbalance $b^k(i) = \left(l^k(i) - \bar{l^k}\right)$, where $d^k(i)$ is the degree of vertex i in

Δ_k; finally, $l^k(i)$ and $\bar{l^k}$ are the number of observations and the average load in $\Omega_i \times \Delta_k$, respectively. Solution of the Laplacian system:

$$L^k \lambda^k = b^k \tag{4.4}$$

associated to G^k gives the amount of data which should have migrated in Δ_k.

- Migration step: DyDDST shifts the boundaries of spatial subdomains.
- Updating step: DyDDST redefines spatial subdomains in Δ_k such that each one contains the number of observations computed in the scheduling step and redistributes subdomains among processors grids. After, for all subdomains it is necessary to re-evaluate the workload to balance the number of observations in Δ_{k+1}.

Table 4.2: Procedure DyDDST

Procedure DyDDST-Dynamic Load Balancing in Space and Time(in: p,N_t, Ω, out: l_1,...,l_p)
%Procedure DyDDST allows to balance observations between adjacent subdomains in Δ
% Domain $\Omega \times \Delta$ is decomposed in $p \times N_t$ subdomains and some of spatial subdomains may be empty.

Initial DD step
% DD of $\Omega \times \Delta$ in $(\Omega_1, \Omega_2, \ldots, \Omega_p)$ and $(\Delta_1, \Delta_2, \ldots, \Delta_{N_t})$
end of Initial DD step

DD step
Define n_i: the number of adjacent subdomains of Ω_i
Define l_i^k: the amount of observations in $\Omega_i \times \Delta_k$

repeat for any i

% identification of Ω_m, the subdomain adjacent to Ω_i

with the maximum load

Compute m such that $l_m^k = max_{j=1,...,n_i}\ (l_j^k)$: the maximum amount

of observations

Decompose Ω_m in 2 subdomains: $\Omega_m \leftarrow (\Omega_m^1, \Omega_m^2)$

until $(l_i^k \neq 0)$

end of DD Step

Begin Scheduling step

Define G^k: the graph associated with initial partition of $\Omega \times \Delta_k$: vertex i corresponds to Ω_i in Δ_k

Distribute the amount of observations l_i^k in Ω_i

Define $d^k(i) = n_i$, the degree of node i of G^k:

repeat

Compute the average load: $\bar{l}^k = \frac{\sum_{i=1}^{p} l_i^k}{p}$

Compute load imbalance: $b^k = (l_i^k - \bar{l^k})_{i=1,...,p}$

Compute L^k, Laplacian matrix of G^k

Call PCG(in:L^k, b^k, out:λ^k) % Preconditioned Conjugate Gradient algorithm solving the linear system $L^k\lambda^k = b^k$

Compute $\delta_{i,j}^k$, the load increment between the adjacent subdomains Ω_i and Ω_j. $\delta_{i,j}^k$ is the nearest integer of $(\lambda_i^k - \lambda_j^k)$

Define $n_{s_i}^k, n_{r_i}^k$, number of those subdomains whose configuration has to be updated

Update G^k

Update amount of observations of Ω_i^k: $l_i^k = l_i^k - \sum_{j=1}^{n_{s_i}^k} \delta_{i,j}^k + \sum_{j=1}^{n_{r_i}^k} \delta_{j,i}^k, \forall i$

until $(max_i \| l_i^k - \overline{l}^k \| == \frac{d^k(i)}{2})$ % i.e. maximum load-difference is $d^k(i)/2$

end Scheduling step

Begin Migration Step

Shift boundaries of two adjacent subdomains in order to achieve a balanced load in Δ_k.

end Migration Step

Update DD of Ω

Update DD

Update $l_i^k \equiv l_i^{k+1}$ the number of observations of subdomain Ω_i in Δ_{k+1} (not yet balanced)

end Update step

Define $l_i \equiv l_i^k$ on $\Omega_i \times \Delta_k$.

endProcedure DyDDST

Chapter 5

Validation Analysis

Simulations were aimed to validate the proposed approaches by measuring the scalability of DD–4DVAR and DD-KF algorithms. We consider two problems outlined in Appendix A.1 and A.2 Performance evaluation was carried out using Parallel Computing Toolbox of MATLAB. We consider two DD configurations, on two different computing environments.

1. Computing environment no.1: we run DD–4DVAR and DD-KF algorithms using MATLABR2018b on a CPU shared memory, namely a Laptop with 1.6GHz CPU, with 2 physical cores and 4 GB of memory. In this case for testing the algorithm we consider up to $N_{sub} = 8$ subdomains equally distributed among the cores. This is essentially an *intra-node* DD configuration which realizes a fine-grained parallelization strategy on a single node with many-core CPU.

2. Computing environment no.2: we run DD–4DVAR and DD-KF algorithms using MATLABR2013a on the high performance hybrid compunting architecture of the SCoPE (Sistema Cooperativo Per Elaborazioni scientifiche multidiscipliari) data center, located in the University of Naples Federico II. More precisely, the HPC architecture is made of 8 nodes,

consisting of distributed memory DELL M600 blades connected by a 10 Gigabit Ethernet technology. Each blade consists of 2 Intel Xeon@2.33GHz quadcore processors sharing 16 GB RAM memory for a total of 8 cores/blade and of 64 cores, in total. In this case for testing the algorithm we consider up to $N_{sub} = 64$ subdomains equally distributed among the cores. This is an *intra-node* DD configuration implementing a parallelization strategy on multiprocessor systems with many-core CPUs. Finally, we compare time execution of parallel KF algorithm with respect to DD–4DVAR and DD-KF algorithms.

5.1 DD-KF applied to CLS problem

We perform validation analysis of the proposed DD-KF approach by considering the *intra-node* DD configuration on computing environment no.1.

DD-KF set up:

- $H_0 \in \mathbb{R}^{11\times 6}$: random matrix;
- $H_1 \equiv h^T \in \mathbb{R}^{1\times 6}$: random vector;
- $y_0 \in \mathbb{R}^{11}$: random vector;
- $y_1 \in \mathbb{R}$: a random constant;
- $b = \left[\; y_0, y_1 \; \right] \in \mathbb{R}^{12}$ the vector in (A.2);
- $R_0 = 0.5 \cdot I$: weight matrix, with $I \in \mathbb{R}^{11\times 11}$ identity matrix, $R_1 = 0.5$ and $R = diag(R_0, R_1) \in \mathbb{R}^{12\times 12}$ weight matrix.

We calculate:

- $\widehat{x}_0 \in \mathbb{R}^6$: solution of normal equations in (A.1);

- $\widehat{x} \in \mathbb{R}^6$: solution of normal equations in (A.6) obtained by using Conjugate Gradient method;
- $\widehat{x}_1 \in \mathbb{R}^6$ Kalman estimate as in (2.20) at step $k = 1$.

We apply the DD approach to CLS problem in (A.4) by using:

- $nmax = 50$: maximum number of iterations;
- $tol = 10^{-6}$: tolerance;
- $\widehat{x} \in \mathbb{R}^6$ solution of normal equations in (A.6) by Conjugate Gradient method.

Decomposition of $I = \{1, 2, 3, 4, 5, 6\}$ without overlap i.e.

- $I_1 = \{1, 2, 3, 4\}$, $I_2 = \{5, 6\}$;
- $N_p \equiv |I| = 6$,
- $n_1 \equiv |I_1| = 4$,
- $n_2 \equiv |I_2| = 2$;
- $\widehat{x}_{1,0} \equiv \widehat{x}_0|_{I_1} \in \mathbb{R}^{n_1}$,
- $\widehat{x}_{2,0} \equiv \widehat{x}_0|_{I_2} \in \mathbb{R}^{n_2}$: as in (3.128) with $\widehat{x}_0$ solution in least squares sense of (A.1);
- for $i = 1, 2$, $\widehat{x}^0_{i,1} \equiv zeros(n_i) \in \mathbb{R}^{n_i}$, where $zeros(n_i)$ is the null vector;
- for $n = 1, 2, ..., nmax$, $\widehat{x}^{n+1}_{1,1} \in \mathbb{R}^4$, $\widehat{x}^{n+1}_{2,1} \in \mathbb{R}^2$: the Kalman estimates;
- $\|r^{n+1}\| < tol$: stopping criterion, where $r^{n+1} := (A^T RA)x^{n+1} - A^T Rb$ is the residual at iteration $n + 1$ in (A.6);

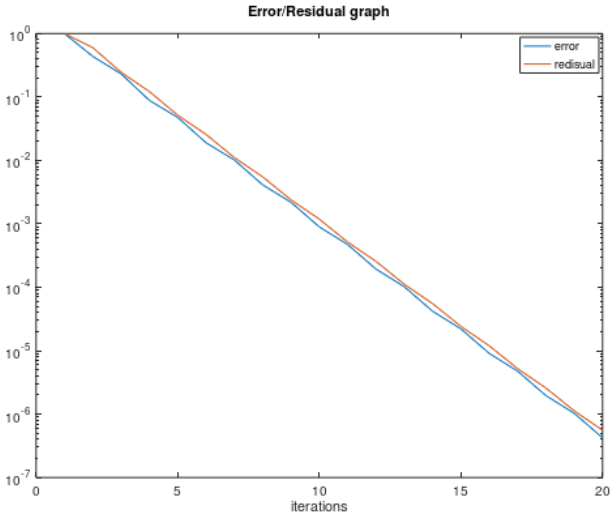


Figure 5.1: The norm residual $\|r^{n+1}\|$ and the $error = \|\widehat{x} - x^{n+1}\|$ graph. $\|r^{n+1}\|$| and $error$ decrease as the number of iterations grows. tolerance $tol = 10^{-6}$ is exceeded at $ns = 20$.

- ns the number of iterations needed to stop of the iterative procedure.

x^{n+1}, i.e. the DD solution, is:

$$x^{n+1} = \begin{cases} \widehat{x}_{1,1}^{n+1} & on\ I_1 \\ \widehat{x}_{2,1}^{n+1} & on\ I_2 \end{cases}. \tag{5.1}$$

In Figure 5.1, we report the $\|r^{n+1}\|$ when n grows. We see that the residual norm exceeds $tol = 10^{-6}$ in correspondence of $ns = 20$. In particular, we note that the order of magnitude of $error = \|\widehat{x} - x^{ns}\| \approx 6.2668 \times 10^{-7}$ is the same of $\|r^{ns}\| \approx 6.6801 \times 10^{-7}$. In Table 5.1, we report values of $error$ and the relative number of iterations (ns).

Decomposition of $I = \{1, 2, 3, 4, 5, 6\}$ in I_1 and I_2 with overlap, for $\delta/2 = 1, 2, 3$:

- $I_1 = \{1, 2, 3, 4\}$, $I_2 = \{4 - \frac{\delta}{2}, ..., n\}$ and $I_{1,2} = \{4 - \frac{\delta}{2}, ..., 4\}$;
- $N_p \equiv |I| = 6$,

Table 5.1: Values of $error = \|\widehat{x} - x^{ns_\delta}\|$ for different values of tol.

tol	ns_δ	$error$
10^{-6}	20	$6.4037e-07$
10^{-9}	29	$4.8394e-10$
10^{-14}	33	$6.7045e-15$

- $n_1 \equiv |I_1| = 4$,
- $n_2 \equiv N_p + \frac{\delta}{2} - n_1 \equiv |I_2| = 2 + \frac{\delta}{2}$;
- for $i = 1, 2$, $\widehat{x}^0_{i,1} \equiv zeros(n_i) \in \mathbb{R}^{n_i}$, where $zeros(n_i)$ is the null vector;
- for $n = 1, 2, ..., nmax$, we compute $\widehat{x}^{n+1}_{1,1} \in \mathbb{R}^4$, $\widehat{x}^{n+1}_{2,1} \in \mathbb{R}^2$: the Kalman estimates as in (3.138).

DD estimate $x^{n+1}_s \in \mathbb{R}^9$ is obtained as follows

$$x^{n+1}_\delta = \begin{cases} \widehat{x}^{n+1}_{1,1}|_{I_1 \setminus I_{1,2}} & \text{on } I_1 \setminus I_{1,2} \\ \frac{\mu}{2}(\widehat{x}^{n+1}_1|_{I_{1,2}} + \widehat{x}^{n+1}_2|_{I_{1,2}}) & \text{on } I_{1,2} \\ \widehat{x}^{n+1}_{2,1}|_{I_2 \setminus I_{1,2}} & \text{on } I_2 \setminus I_{1,2} \end{cases}, \tag{5.2}$$

with $\mu \equiv 1$ regularization parameter; $\|r^{n+1}_\delta\| < tol$ the stopping criterion, where $r^{n+1}_\delta := (A^T R A)x^{n+1}_\delta - A^T R b$ is the residual at iteration $n+1$ of (A.6); ns_δ is the corresponding iteration. As expected and shown in Table 5.1, the size of the overlapping set impacts the convergence behaviour of the algorithm i.e. increasing ns_δ, consistently to tolerance tol, the accuracy improved.

Table 5.2: Values of $error_\delta = \|\widehat{x} - x_\delta^{ns}\|$ for $tol = 10^{-6}$.

ns_s	$error_\delta$	$\delta/2$
17	$7.2526e-07$	1
15	$5.1744e-07$	2
22	$7.2741e-07$	3

5.1.1 Trustworthy analysis

We perform trustworthy analysis of the proposed DD-KF approach by considering the *inter-node* DD configuration on computing environment no.2.

We consider the columnwise domain decomposition of $I = \{1, \ldots, N_p\}$ where $N_p \in \mathbb{N}$, into $N_{sub} < N_p$ subdomain:

$$I = \bigcup_{i=1}^{N_{sub}} I_i$$

where

$$I_i := \left\{ (i-1) \times \frac{N_p}{N_{sub}} + 1, \ldots, i \times \frac{N_p}{N_{sub}} \right\}.$$

This choice of data involves exchanges of data only between two adjacent subdomains so that the surface to volume ratio is minimized.

Communication functions *labSend* and *labReceive* transfer data between workers so that each worker proceeds with the parallel execution of the procedure.

We let:

- $H_0 \in \mathbb{R}^{m_0 \times N_p}$, $H_1 \equiv h^T \in \mathbb{R}^{m_1 \times N_p}$: identity matrices;
- $m := m_0 + m_1$; where $m_1 \equiv 1$, $m_0 \equiv N_p + l$, and $l := 5$;
- $y_0 \in \mathbb{R}^{m_0}$, $y_1 \in \mathbb{R}^{m_1}$: random vectors;

- $b = \left[\; y_0, y_1 \; \right] \in \mathbb{R}^{m_0+m_1}$;
- $R_0 = 0.5 \times I$, with $I \in \mathbb{R}^{m_0 \times m_0}$ identity matrix,
- $R_1 = 0.5$ and $R = diag(R_0, R_1) \in \mathbb{R}^{m_0+m_1 \times m_0+m_1}$ weight matrix.

We underline that, as $m_0 := N_p + l$ where $l, m_1 \in \mathbb{N}$ are fixed, it follows that $r := ((m_0 + m_1) \times N_p) = N_p + l + 1$ in (A.4) only depends on n.

We apply DD-KF to CLS model in (A.4) by using:

- $nmax = 50$: maximum number of iterations;
- $tol = 10^{-6}$: tolerance required to the solution accuracy;
- $\widehat{x} \in \mathbb{R}^{N_p}$: numerical solution of normal equations in (A.6) computed by using Conjugate Gradient method.

Let $n_{loc} := \frac{N_p}{N_{sub}}$ be local problem size. Metrics we use for analysing performance of DD-KF algorithm running on p workers for N_{sub} subdomains are:

- $T^1(m_1, N_p)$: amount of time (in seconds) needed to compute CLS solution using KF;
- $T^{1,p}(m_1, N_p)$: amount of time (in seconds) needed to solve CLS using a parallel KF algorithm running on p cores;
- $T^p\,(m_1, n_{loc})$: amount of time (in seconds) needed to perform DD-KF on p workers;
- $T^p_{oh}\,(m_1, n_{loc})$: amount of overhead time (measured in seconds) due to synchronization, memory accesses and communication time among p workers;
- $\frac{S}{V}^p\,(m_1, n_{loc}) := \frac{T^p_{oh}(m_1, n_{loc})}{T^1(m_1, n_{loc})}$: surface-to-volume ratio;

- $S^p\left(m_1, n_{loc}\right) := \frac{T^1(m_1,n)}{T^p(m_1,n_{loc})}$: speed-up on p workers;
- $E^p\left(m_1, n_{loc}\right) := \frac{S^p(m_1,n_{loc})}{p}$: efficiency on p workers;
- $Sc^f_{N_{sub},1}\left(m_1, n_{loc}\right) = \frac{T^1(m_1,N_p)}{N_{sub}\cdot T^1(m_1,n_{loc})}$: measured value of scale up factor;

Note that measured values of the scale up factor give us an estimate on the real reduction of the serial execution time of KF algorithm we could expect by sequentially solving the N_{sub} sub-problems resulting from DD-KF. This value provides additional and valuable insights into any real world employment of DD-KF algorithm.

In the following tables and figures we report values of scalability, computed in terms of *strong scaling* and *weak scaling*.

1. Intra-node DD configuration.

   In Tables 5.3 and Figure 5.2 we report performance results of the algorithm running on the CPU shared memory. We note that at $N_{sub} = 2$ and $N_{sub} = 4$, as the computing environment is a dual-core CPU shared memory, overhead is so small that super-linear speed up arises. At $N_{sub} = 8$, instead, overheads due to traffic memory and synchronizations, measured by the surface-to-communication value, become so predominant to drastically reduce the speed up.

   Note: Strong scaling results at $N_p = 512$, $n_{loc} := N_p/N_{sub}$ and $r := (m_0 + m_1) \times N_p = (N_p + 6) \cdot N_p$ for $N_{sub} = 2, 4, 8$ subdomains and $p = 2$. We note that as the computing environment is a dual-core CPU shared memory, at $N_{sub} = 2$ and $N_{sub} = 4$ the overhead is negligible so superlinear speed up arises. If $N_{sub} = 8$, instead, the surface-to-communication value becomes predominant so that it drastically turns down speed up.

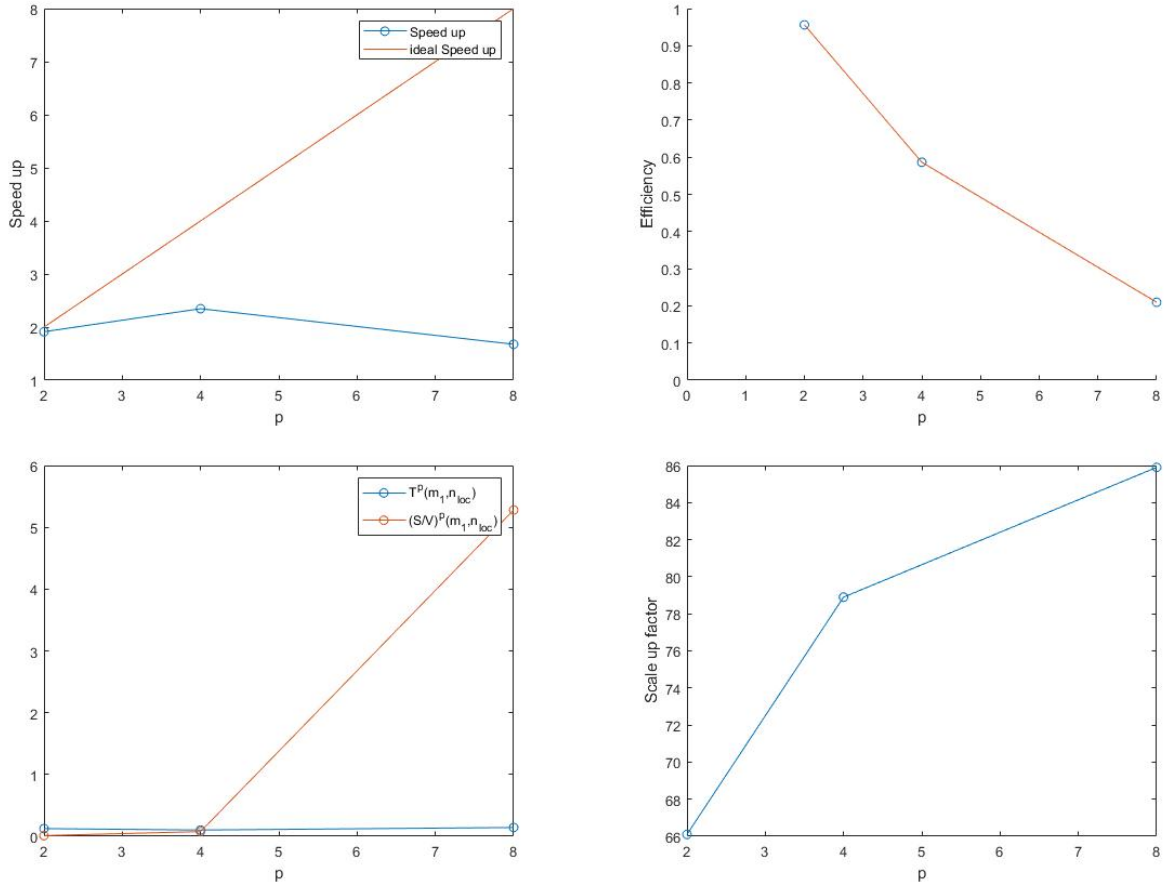


Figure 5.2: Intra-node DD configuration. Performance results at $N_p = 512$, $n_{loc} := N_p/N_{sub}$ and $r := (m_0+m_1) \times N_p = (N_p+6) \cdot N_p$ for $N_{sub} = 2, 4, 8$ subdomains and $p = 2$. We note that as the computing environment is a dual-core CPU shared memory, at $N_{sub} = 2$ and $N_{sub} = 4$ the overhead is negligible so super-linear speed up arises. If $N_{sub} = 8$, instead, the surface-to-communication value becomes predominant so that it drastically turns down speed up.

Table 5.3: Internode DD configuration .

N_{sub}	N_p		$T^1(m_1, N_p)$			
1	512		$2.38 \cdot 10^{-1}$			
N_{sub}	N_p	$\frac{S}{V}^p(m_1, n_{loc})$	$T^p(m_1, n_{loc})$	$S^p(m_1, n_{loc})$	$E^p(m_1, n_{loc})$	$Sc^f_{N_{sub},1}(m_1, n_{loc})$
2	256	1.13×10^{-2}	1.24×10^{-1}	1.92×10^{0}	9.60×10^{-1}	6.61×10^{1}
4	128	7.58×10^{-2}	1.01×10^{-1}	2.36×10^{0}	5.60×10^{-1}	7.89×10^{1}
8	64	5.28×10^{0}	1.42×10^{-1}	1.68×10^{0}	2.10×10^{-1}	8.59×10^{1}

2. Inter-node DD configuration. In these experiments we let

$$p \equiv N_{sub}$$

In Tables 5.4 and 5.5 and Figures 5.3 and 5.5 results related to strong and weak scaling obtained running DD-KF algorithm on the HPC architecture based on many-core distributed processors, are reported. In Figure 5.4 we plot Speed up and Efficiency lines versus p. Moreover, in Table 5.6, we compare execution time of KF algorithm where a straightforward parallelism at fine grained level is introduced but without using DD, and DD-KF. We note that DD-KF gets a performance gain of two orders of magnitude with respect to the parallelism at fine-grained level of KF algorithm.

5.2 DD–KF applied to SWEs problem

We apply DD-KF method to the initial boundary problem of SWEs described in Appendix A.2 by considering the *intra-node* DD configuration on computing environment no.1. The discrete model is obtained by using Lax-Wendroff scheme [81]. Experiments are aimed to prove that DD-KF provide same solutions of KF. Reliability is assessed w.r.t. KF and DA, by

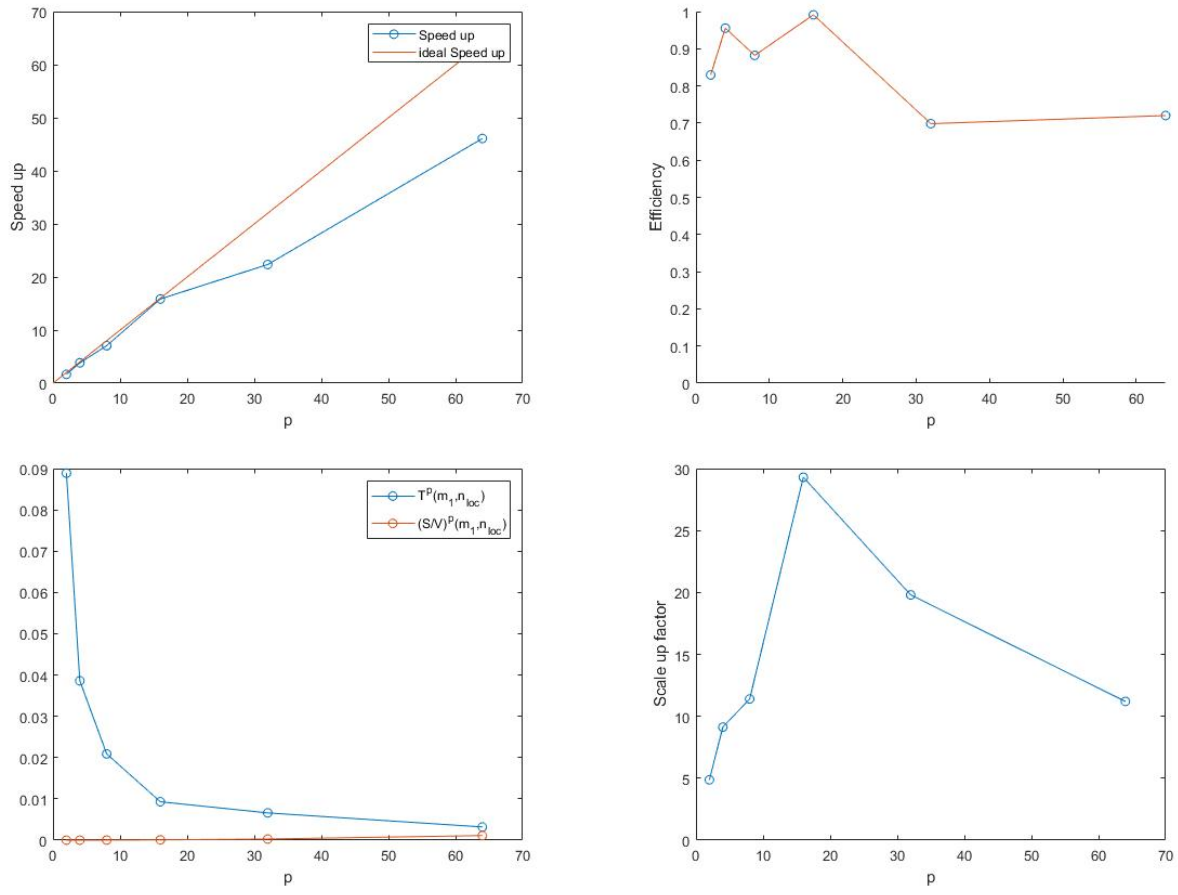


Figure 5.3: Inter-node DD configuration. Performance results at $N_p := 1920$ and $r := (m_0 + m_1) \times N_p = (N_p + 6) \times N_p$ for $N_{sub} \equiv p = 2, 4, 8, 16, 32, 64$ subdomains/workers.

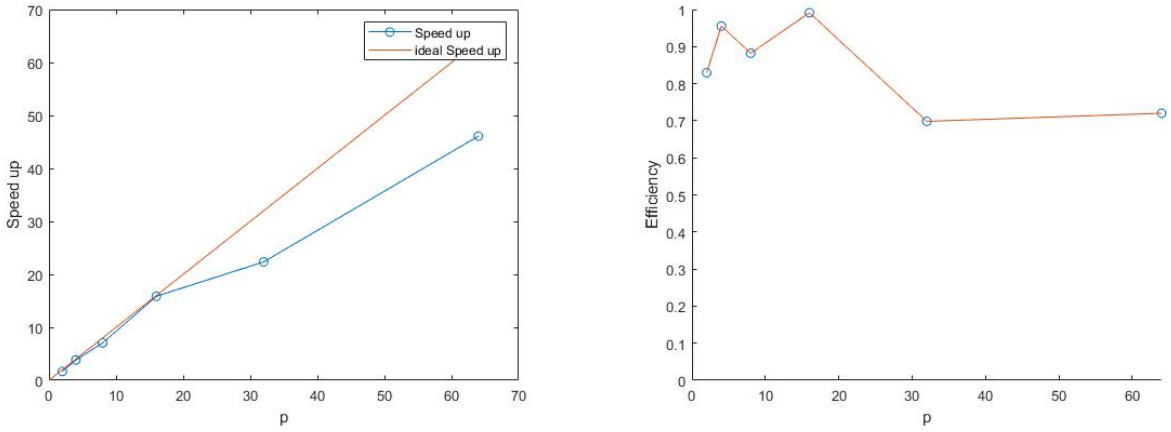


Figure 5.4: Inter-node DD configuration. (Left) Speed-up; (right) Efficenty.

Table 5.4: Inter-node DD configuration.

N_{sub}	N_p		$T^1(m_1, N_p)$			
1	1920		1.46×10^{-1}			
N_{sub}	N_p	$\frac{S}{V}^p(m_1, n_{loc})$	$T^p(m_1, n_{loc})$	$S^p(m_1, n_{loc})$	$E^p(m_1, n_{loc})$	$Sc^f_{N_{sub},1}(m_1, n_{loc})$
2	960	1.09×10^{-6}	8.89×10^{-2}	1.64×10^{0}	8.20×10^{-1}	4.87×10^{0}
4	480	4.34×10^{-6}	3.86×10^{-2}	3.78×10^{0}	9.45×10^{-1}	9.13×10^{0}
8	240	1.74×10^{-5}	2.09×10^{-3}	6.99×10^{0}	8.74×10^{-1}	1.14×10^{1}
16	120	6.94×10^{-5}	9.30×10^{-3}	1.57×10^{1}	9.81×10^{-1}	2.93×10^{1}
32	60	2.78×10^{-4}	6.60×10^{-3}	2.21×10^{1}	6.91×10^{-1}	1.98×10^{1}
64	30	1.11×10^{-3}	3.20×10^{-3}	4.56×10^{1}	7.13×10^{-1}	1.12×10^{1}

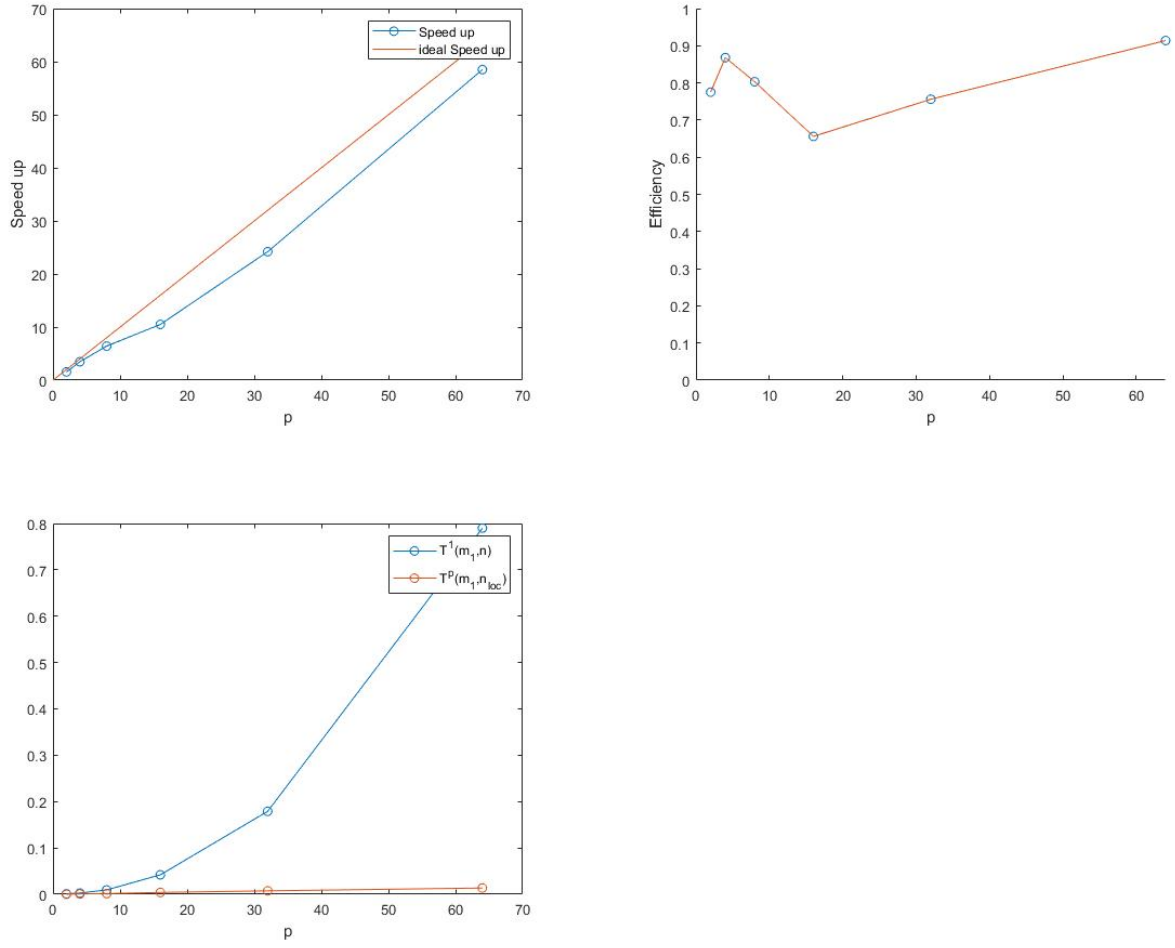


Figure 5.5: Weak Scaling of inter-node configuration. Scalability prediction of DD-KF algorithm computed fixing $n_{loc} = 64$, $N_p = p \times n_{loc}$, and $r := (m_0 + m_1) \times N_p = (N_p + 6) \times N_p$ where $N_{sub} = p = 2, 4, 8, 16, 32, 64$.

Table 5.5: Weak Scaling of inter-node DD configuration. Scalability prediction of DD-KF algorithm computed fixing $n_{loc} = 64$, $N_p = p \times n_{loc}$, and $r := (m_0 + m_1) \times N_p = (N_p + 6) \times N_p$ where $N_{sub} = p = 2, 4, 8, 16, 32, 64$.

p	N_p	r	$T^1(m_1, N_p)$	$T^p(m_1, n_{loc})$	$S^p(m_1, n_{loc})$	$E^p(m_1, n_{loc})$
2	128	$17,152$	9.59×10^{-4}	6.18×10^{-5}	1.55×10^{0}	7.75×10^{-1}
4	256	$67,072$	2.60×10^{-3}	7.50×10^{-4}	3.47×10^{0}	8.68×10^{-1}
8	512	$265,216$	9.40×10^{-3}	1.50×10^{-3}	6.42×10^{0}	8.03×10^{-1}
16	1024	$1,054,720$	4.20×10^{-2}	4.00×10^{-3}	1.05×10^{1}	6.56×10^{-1}
32	2048	$4,206,592$	1.79×10^{-1}	7.40×10^{-3}	2.42×10^{1}	7.56×10^{-1}
64	4096	$16,801,792$	7.90×10^{-1}	1.35×10^{-2}	5.85×10^{1}	9.14×10^{-1}

Table 5.6: Inter-node DD configuration. Comparison between execution time of KF algorithm where a straightforward parallelism at fine grained level is introduced but without using DD, and DD-KF, at $N_p := 1920$ and $N_{sub} \equiv p = 64$ subdomains/workers.

p	N_p	$T^{1,p}(m_1, N_p)$	$T^p(m_1, n_{loc})$
64	1920	1.36×10^{-1}	3.20×10^{-3}

using maximum error and RMSE, respectively.

KF configuration. We consider the following experimental scenario:

- $\Omega = (0, L)$: where $L = 1$;
- $\Delta = [0, T]$: where $T = 1.5$;
- $p_v = 2$: number of physical variables;
- $N_p = 500$ and $b_c = 2$: numbers of nodes of Ω and $\partial\Omega$, respectively;
- $n_x := N_p + b_c = 502$ and $N = 53$: numbers of elements of $\bar{\Omega}$ and Δ, respectively;
- $\Delta x = \frac{1}{500}$ and Δt: step size of Ω and Δ, respectively, where Δt varies to satisfy the stability condition in (A.28) of Lax-Wendroff method;
- $\Omega_I = \{x_i\}_{i=0,\dots,N_p-1}$: discretization of Ω where $x_i = i \cdot \Delta x$;
- $\Delta_K = \{t_k\}_{k=0,\dots,N-1}$: discretization of Δ where $t_k = t_{k-1} + \Delta t$;
- for $i = 1, 2$, $x_l[i]$: KF estimate of height (for $i = 1$) and horizontal velocity (for $i = 2$);
- $n_{obs} = 14$: number of observations at step $l = 0, 1, \dots, N - 1$;
- $v_l := 10^{-2} \cdot \bar{v}_l \in \mathbb{R}^{n_{obs}}$: observations errors, with $\bar{v}_l$ a random vector drawn from the standard normal distribution, for $l = 0, 1, \dots, N - 1$;
- $y_l := x(t_l, x_j) + v_l \in \mathbb{R}^{n_{obs}}$: observations vector for $j = 1, \dots, m$ at step $l = 0, 1, \dots, N - 1$; observations are obtained by adding observation errors to the full solution of the SWE's without using domain decomposition while using exact initial and boundary conditions.
- $H_l \in \mathbb{R}^{n_{obs} \times N_p}$: piecewise linear interpolation operator whose coefficients are computed using the points of the domain nearest to observation values;

- $\sigma_m^2 = 5.0 \times 10^{-1}$, $\sigma_0^2 = 3.50 \times 10^{-1}$: model and observational error variances;
- $B \equiv B_l = \sigma_m^2 C$: covariance matrix of the model error at step $l = 0, 1, \ldots, N-1$, where $C \in \mathbb{R}^{N_p \times N_p}$ denotes Gaussian correlation structure of model errors in (5.3);
- $R \equiv R_k = \sigma_0^2 I_{n_{obs}, n_{obs}} \in \mathbb{R}^{n_{obs} \times n_{obs}}$: covariance matrix of the errors of the observations at step $l = 0, 1, \ldots, N-1$.

DD-KF configuration. We consider the following experimental scenario:

- $\delta = 2, 4, \ldots, 200$: number of inner nodes in spatial overlap;
- $n_1 = 250 + \delta/2$, $I_1 = \{1, \ldots, 250\}$ with $|I_1| = n_1$;
- $n_2 = 250 + \delta/2$, $I_1 = \{n_1 - \delta/2 + 1, \ldots, 500\}$ with $|I_2| = n_2$;
- $s_{1,2} = 2, 3, \ldots, 50$: number of instants of time in temporal overlap;
- $\bar{s}_0 := 0$, $s_{0,1} := 0$, $s_1 = 25 + s_{1,2}/2$, $\bar{s}_1 := s_1 - s_{1,2}$ and $s_2 = 28 + s_{1,2}/2$;
- $C := \{c_{i,j}\}_{i,j=1,\ldots,N_p} \in \mathbb{R}^{N_p \times N_p}$: Gaussian correlation structure of model error where[1]

$$c_{i,j} = \rho^{|i-j|^2}, \quad \rho = exp\left(\frac{-\Delta x^2}{2L^2}\right), \quad |i-j| < N_p/2 \qquad \begin{array}{l} \text{for } i = 1, \ldots, n_1, \\ j = 1, \ldots, n_1 - \delta/2 \text{ and} \\ \text{for } i, j = n_1 - \delta/2 + 1, \ldots, N_p \end{array} . \tag{5.3}$$

1. Decomposition Step.

[1] According to [30] here we assume the model error to be gaussian . As explained in [30], a research collaboration between us and CMCC (Centro Euro Mediterraneo per i Cambiamenti Climatici) give us the opportunity to use the software called OceanVar. OceanVar is used in Italy to combine observational data (Sea level anomaly, sea-surface temperatures, etc.) with backgrounds produced by computational models of ocean currents for the Mediterranean Sea (namely, the NEMO framework [22]). OceanVAR assumes gaussian model errors.

- Decomposition of Ω into two subdomains with overlap region:

$$\begin{aligned}\Omega_1 &= [0, x_{n_1}] \\ \Omega_2 &= [x_{n_1-\delta/2+1}, 1]\end{aligned} \tag{5.4}$$

  where n_1 is the number of inner nodes in Ω_1 and δ is the number of nodes in overlap region.

- Decomposition of Δ into two subsets with overlap region:

$$\begin{aligned}\Delta_1 &= [0, t_{s_1-1}] \\ \Delta_2 &= [t_{s_1-s_{1,2}+1}, 1.5],\end{aligned} \tag{5.5}$$

  where s_1 is the number of instants of time in Δ_1 and $s_{1,2}$ is the number of instants of time in common among 2 adjacent time intervals.

2. Local initial conditions on Δ_1 and Δ_2:

$$\begin{aligned} x[1]_0^{\Delta_1} &= h(0,x) =: x[1]_0^{\Delta_0} & x[1]_{s_1-1}^{\Delta_2} &= x[1]_{s_1-1}^{\Delta_1} \\ x[2]_0^{\Delta_1} &= h(0,x)v(0,x) =: x[2]_0^{\Delta_0} & x[2]_{s_1-1}^{\Delta_2} &= x[2]_{s_1-1}^{\Delta_1} \end{aligned} . \tag{5.6}$$

3. Model reduction in (A.25).
   Decomposition of matrices $M[1]_{l+1} \in \mathbb{R}^{n_x-2}$ at each step $l = 0, 1, \ldots, nt-2$

$$M[1]_{l,l+1}^1 := M[1]_{l,l+1}|_{I_1\times I_1} = \begin{bmatrix} \psi_1^l & \eta_2^l & & & \\ -\eta_1^l & \psi_2^l & \eta_3^l & & \\ & \ddots & \ddots & \ddots & \\ & & -\eta_{n_1-2}^l & \psi_{n_1-1}^l & \eta_{n_1}^l \\ & & & -\eta_{n_1-1}^l & \psi_{n_1}^l \end{bmatrix} \in \mathbb{R}^{n_1\times n_1} \tag{5.7}$$

$$M[1]_{l,l+1}^{1,2} := M[1]_{l,l+1}|_{I_1\times I_2} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & \cdots & 0 \\ \vdots & & & & & & \vdots \\ & & & & & & \\ 0 & \cdots & 0 & \eta_{n_1+1}^k & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{n_1\times n_2} \tag{5.8}$$

$$M[1]^{2,1}_{l,l+1} := M[1]_{l,l+1}|_{I_1\times I_2} = \begin{bmatrix} 0 & \cdots & 0 & 0 & \eta^l_{n_1} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & & & \vdots & \vdots & & \vdots \\ & & & & & & & \\ 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{n_2\times n_1} \tag{5.9}$$

$$M[1]^{2}_{l,l+1} := M[1]_{l,l+1}|_{I_2\times I_2} = \begin{bmatrix} \psi^l_{n_1} & \eta^l_{n_1+1} & & & \\ -\eta^l_{n_1} & \psi^l_{n_1+1} & \eta^l_{n_1+2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\eta^k_{n_x-4} & \psi^k_{n_x-3} & \eta^k_{n_x-2} \\ & & & -\eta^k_{n_1-3} & \psi^k_{n_x-2} \end{bmatrix} \in \mathbb{R}^{n_2\times n_2}. \tag{5.10}$$

We proceed in the same way to get the decomposition of matrix $M[2]_{l,l+1} \in \mathbb{R}^{n_x-2\times n_x-2}$ in (A.25) into $M[2]^1_{l,l+1} \in \mathbb{R}^{n_1\times n_1}$, $M[2]^{1,2}_{l,l+1} \in \mathbb{R}^{n_1\times n_2}$, $M[2]^{2,1}_{l,l+1} \in \mathbb{R}^{n_2\times n_1}$ and $M[2]^2_{l,l+1} \in \mathbb{R}^{n_2\times n_2}$.

4. Decomposition of $H_{l+1} \in \mathbb{R}^{n_{obs}\times(n_x-2)}$ into $H^1_{l+1} \in \mathbb{R}^{n_{obs}\times n_1}$ and $H^2_{l+1} \in \mathbb{R}^{n_{obs}\times n_2}$ as in (3.169).

5.2.1 The algorithm

1. For $k = 1, 2$ and $l = \bar{s}_{k-1} + 1, \ldots, \bar{s}_{k-1} + s_k$, we apply DD-KF method on Δ_1 and Δ_2 by considering the following matrices:

$$\begin{array}{llll} M_1[i] & \equiv M[i]^1_{l,l+1} & M_{1,2}[i] & \equiv M[i]^{1,2}_{l,l+1} \\ M_2[i] & \equiv M[i]^2_{l,l+1} & M_{2,1}[i] & \equiv M[i]^{2,1}_{l,l+1} \\ P_1 & = O_{n_1\times n_1} & P_2 & = O_{n_2\times n_2} \\ P_{1,2} & = O_{n_1\times n_2} & P_{2,1} & = O_{n_2\times n_1} \end{array}. \tag{5.11}$$

In particular, we note:

$$M_{1,2}[i] \equiv \begin{bmatrix} O & M_{1,3}[i] \\ O & M_{2,3}[i] \end{bmatrix} \quad M_{2,1}[i] \equiv \begin{bmatrix} \tilde{M}_{1,2}[i] & O \\ \tilde{M}_{2,1}[i] & O \end{bmatrix}. \tag{5.12}$$

2. Send and receive boundary conditions from adjacent domains and compute the vectors:

$$b_{1,l}[i] = \begin{bmatrix} M_{1,3}[i] \\ M_{2,3}[i] \end{bmatrix} \widehat{x}_{2,l}[i]|_{\Gamma_1} \quad b_{2,l}[i] = \begin{bmatrix} \tilde{M}_{1,2}[i] \\ \tilde{M}_{2,1}[i] \end{bmatrix} \widehat{x}_{1,l}[i]|_{\Gamma_2}. \tag{5.13}$$

3. For $k = 1, 2$ and $l = \bar{s}_{k-1} + 1, \ldots, \bar{s}_{k-1} + s_k$, compute the predicted state estimates $x_{1,l+1}[i] \in \mathbb{R}^{n_1}$ and $x_{2,l+1}[i] \in \mathbb{R}^{n_2}$ for $i = 1, 2$, as follows:

$$\begin{aligned} x_{1,l+1}[i] &= M_1[i]\widehat{x}[i]_l^{\Delta_{k-1}}|_{I_1} + \bar{b}[i]_l|_{I_1} + b_{1,l} \\ x_{2,l+1}[i] &= M_2[i]\widehat{x}[i]_{\bar{s}_k}^{\Delta_{k-1}}|_{I_2} + \bar{b}[i]|_{I_2} + b_{2,l} \end{aligned}, \tag{5.14}$$

with

$$\bar{b}[i]_l = \begin{cases} b[1]_l & \text{in (A.24)} \quad \text{if } i = 1 \\ \tilde{b}[2]_l & \text{in (A.26)} \quad \text{if } i = 2 \end{cases}. \tag{5.15}$$

4. For $k = 1, 2$ and $l = \bar{s}_{k-1}, \ldots, \bar{s}_{k-1} + s_k - 1$, compute DD-KF estimates as in (3.187) on Δ_k i.e. $\widehat{x}_{1,l+1}^{\Delta_k}[i] \equiv \widehat{x}[i]_{1,l+1} \in \mathbb{R}^{n_1}$ and $\widehat{x}_{2,l+1}^{\Delta_k}[i] \equiv \widehat{x}[i]_{2,l+1} \in \mathbb{R}^{n_2}$. In particular, in the time interval Δ_k, by considering the boundary conditions in (A.13), DD-KF estimates on Ω_1 and Ω_2 are

$$\begin{aligned} \widehat{x}_{l+1}^{\Omega_1 \times \Delta_k}[i] &:= \begin{bmatrix} x(t_{l+1}, 0)[i] \\ \widehat{x}_{1,l+1}^{\Delta_k}[i] \\ \widehat{x}_{2,l+1}^{\Delta_k}[i](1) \end{bmatrix} \in \mathbb{R}^{n_1+1} \\ \widehat{x}_{l+1}^{\Omega_2 \times \Delta_k}[i] &:= \begin{bmatrix} \widehat{x}_{1,l+1}^{\Delta_k}[i](n_1 - \delta/2) \\ \widehat{x}_{2,l+1}^{\Delta_k}[i] \\ x(t_{l+1}, x_{n_x-1})[i] \end{bmatrix} \in \mathbb{R}^{n_2+1} \end{aligned} \tag{5.16}$$

where $\widehat{x}_{r,l+1}^{\Delta_k}[i](1)$, $\widehat{x}_{r,k+1}^{\Delta_k}[i](n_1 - \delta/2)$ are the first and the $n_1 - \delta/2$ components of $\widehat{x}_{r,l+1}^{\Delta_k}[i]$, $r = 1, 2$. We refer to

$$\widehat{x}_{l+1}[i]^{\Omega \times \Delta_k} := \begin{bmatrix} \widehat{x}_{l+1}^{\Omega_1 \times \Delta_k}[i] \\ \widehat{x}_{l+1}^{\Omega_{1,2} \times \Delta_k}[i] \\ \widehat{x}_{l+1}^{\Omega_2 \times \Delta_k}[i] \end{bmatrix} \in \mathbb{R}^{n_x} \tag{5.17}$$

as the estimate of the wave height and velocity (if $i = 1, 2$ respectively) obtained by applied the DD-KF method on Ω, where on the spatial overlap $\Omega_{1,2}$, we have considered the arithmetic mean between DD-KF estimates i.e.

$$\widehat{x}_{l+1}^{\Omega_{1,2}\times\Delta_k}[i] := \left[\frac{\widehat{x}_{l+1}^{\Omega_1\times\Delta_k}[i]|_{I_{1,2}} + \widehat{x}_{l+1}^{\Omega_2\times\Delta_k}[i]|_{I_{1,2}}}{2} \right] \tag{5.18}$$

with $I_{1,2}$ the index set defined in (3.154).

Reliability Metrics. For $k = 1, 2$ and $l = \bar{s}_{k-1}+1, \ldots, \bar{s}_{k-1}+s_k$ and fixed the size of temporal overlap $s_{1,2} = 1$, we compute $\widehat{x}_l[1]$, i.e. KF estimate of the wave height h on Ω. In order to quantify the difference between KF estimate and DD-KF estimates on Δ w.r.t. parameter δ, we introduce the following measure:

$$error_\delta^{\Omega\times\Delta} := \max(error_\delta^{\Omega\times\Delta_1}, error_\delta^{\Omega\times\Delta_2})$$

where

$$error_\delta^{\Omega\times\Delta_k} = \max_{l=\bar{s}_{k-1}+1,\ldots,\bar{s}_{k-1}+s_k} (||\widehat{x}_l[1] - \widehat{x}_l[1]^{\Omega\times\Delta_k}||)$$

where $s \in \mathbb{N}$ is the size of the spatial overlap.

In the same way, we fix δ and compute the error between KF estimate and DD-KF estimates on Δ w.r.t. parameter $s_{1,2}$:

$$error_{s_{1,2}}^{\Omega\times\Delta_k} := \max_{l=\bar{s}_{k-1}+1,\ldots,\bar{s}_{k-1}+s_k} (||\widehat{x}_l[1] - \widehat{x}_l[1]^{\Omega\times\Delta_k}||)$$

with $k = 1, 2$, while $s_{1,2} \in \mathbb{N}$ is the size of the overlap in time domain.

Reliability of DD-KF estimates w.r.t. DA is measured in terms of the Root Mean Square Error (RMSE) for $l = 0, 1, \ldots, nt-1$, which is computed as

$$\begin{aligned} RMSE_l^{\Omega_1\cup\Omega_2} &= \sqrt{\frac{\sum_{i=1}^{N_p}(x_l[1](i) - \widehat{x}_l^{\Omega\times\Delta}[1](i))^2}{N_p}} \\ RMSE_l^{\Omega} &= \sqrt{\sum_{i=1}^{N_p} \frac{(x_l[1](i) - \widehat{x}_l[1](i))^2}{N_p}}. \end{aligned} \tag{5.19}$$

Figure 5.6 shows $error_{\delta}^{\Omega\times\Delta}$ versus δ, obtained within the machine precision (10^{-15}). Also, $error_{s_{1,2}}^{\Omega\times\Delta_j}$ where $s_{1,2} = 200$ is again obtained within the maximum attainable accuracy in double precision (i.e. in our case, 10^{-15}), with $k = 1, 2$ as shown in Figure 5.7. As expected, the accuracy does not depend on the size of the overlap because, DD-KF is a direct method using estimates provided by KF as initial and boundary values of the local dynamic model. In particular, in 5.7 (b) when the size of temporal overlap reaches 50 we observe a relative increment of the error of one unit, very significant with respect to the overall magnitude of the error. This effect is due to the increasing impact of round off errors on the accuracy of the solution. As expected, in a DD method, the extra work performed on the overlapped region with an increasing size can be seen as the effect of a preconditioner on overlapping region [23]. These results prove the reliability of DD-KF method w.r.t. KF.

Figure 5.8 shows that the $RMSE_l^{\Omega_1\cup\Omega_2}$ and $RMSE_l^{\Omega}$ decrease, in particular as we expected, $RMSE_l^{\Omega_1\cup\Omega_2}$ and $RMSE_l^{\Omega}$ coincide during the whole assimilation window.
Finally, a qualitative analysis of DD-KF estimate $\widehat{x}_{l+1}[1]^{\Omega\times\Delta_k}$ at time $t_{25} \in \Delta_1$ and $t_{53} \in \Delta_2$, i.e. for $l + 1 = 25, 53$ and $k = 1, 2$, respectively is shown in Figure 5.9. In Figure 5.9 (a), we note that $\widehat{x}_{25}[1]^{\Omega\times\Delta_1}$ moves from the trajectory of the model state $x[1]_{25}$ to DD-KF estimate position closer to the observation. At the second observation there is a significantly smaller alteration of the trajectory towards the observation. At the fifth observation, it is very close to the observation so we would not expect much effect from the assimilation of this observation. As the model evolves in time it is clear to see that observations have a diminishing effect on the correction of the forecast state estimate, as we can note in Figure 5.9 (b). In particular, for different choices of σ_m^2 and σ_0^2 (model and observation error variances), trajectory of DD-KF estimates $\widehat{x}_{l+1}[1]^{\Omega\times\Delta_k}$ changes, for $k = 1, 2$. Figure 5.10 (a) shows that for $\sigma_m^2 = 0$, DD-KF method gives full confidence to the model, indeed, trajectory of $\widehat{x}_{25}[1]^{\Omega\times\Delta_1}$ coincides with the model state $x[1]_{25}$, otherwise considering $\sigma_0^2 = 10^{-5}$, DD-KF method gives more confidence to

observations, as shown in Figure 5.10 (b).

Finally, in order to point out the capability of DD-KF to deal with the presence of different observation errors, in Figures 5.11 and 5.12 we reports results obtained considering two different values of observations errors in Ω_1 and Ω_2, i.e. $v_l = [v_l^1 v_l^2]$, respectively; in particular, in Figure 5.11 we set $v_l^1 = 10^{-15} \times \bar{v}_l^1$ in Ω_1 and $v_l^2 = 2\bar{v}_l^2$ in Ω_2; in Figure 5.12 we set $v_l^1 = \bar{v}_l^1$ in Ω_1 and $v_l^2 = 10^{-15} \times \bar{v}_l^2$ in Ω_2, where $\bar{v}_l^1$ and $\bar{v}_l^2$ are random vectors drawn from the standard normal distribution and $l = 10$. Results shown in Figure 5.11 confirm the effect of the variance observations $\sigma_{2,0}^2 = 4.28 \times 10^0$ on DD-KF estimate $\widehat{x}[1]^{\Omega \times \Delta}$ in Ω_2 ; similarly, it occurs in Ω_1 when $\sigma_{1,0}^2 = 1.04 \times 10^0$, as results in Figure 5.12 show.

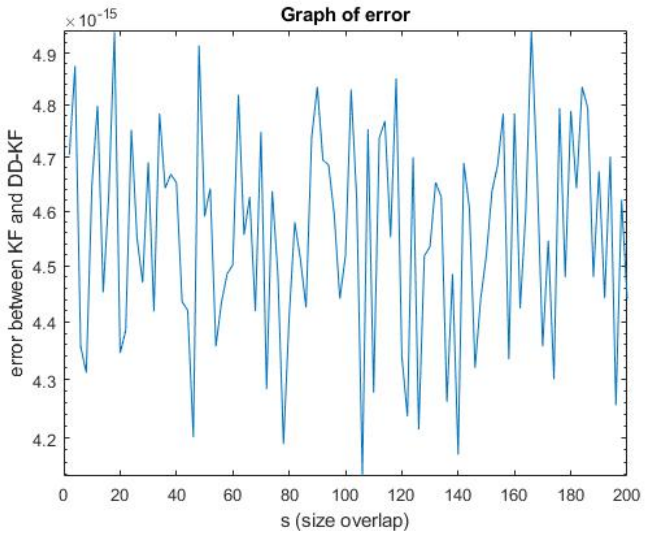
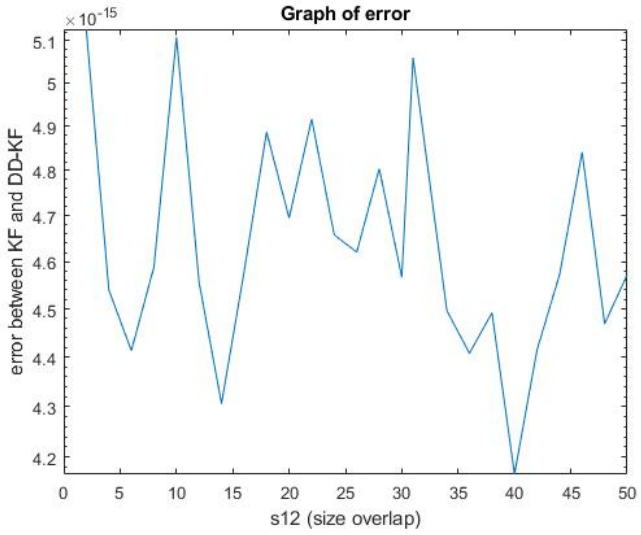
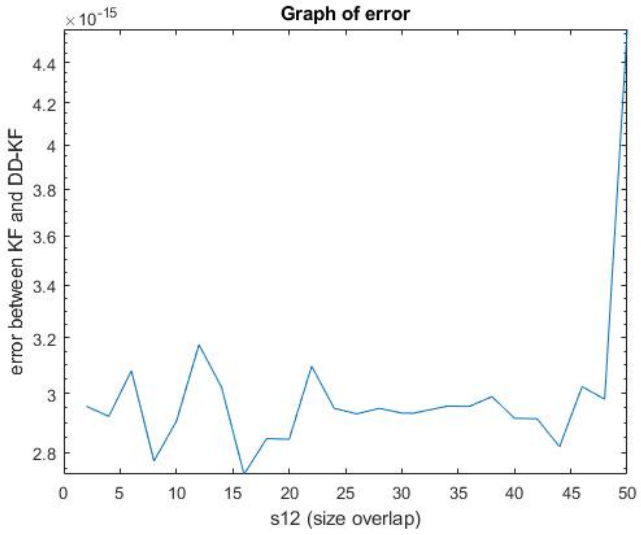


Figure 5.6: Graph of $error_{\delta}^{\Omega \times \Delta}$ versus the spatial overlap δ while temporal overlap is fixed to $s_{1,2} = 1$.



(a) *Behaviour of* $error_{s_{1,2}}^{\Omega\times\Delta_1}$ *versus the size of temporal overlap* $s_{1,2}$.



(b) *Behaviour of* $error_{s_{1,2}}^{\Omega\times\Delta_2}$ *versus the size of temporal overlap* $s_{1,2}$.

Figure 5.7: Behaviour of $error_{s_{1,2}}^{\Omega\times\Delta_1}$ and $error_{s_{1,2}}^{\Omega\times\Delta_2}$ versus the size of temporal overlap $s_{1,2}$ while the size of spatial overlap is $s = 200$.

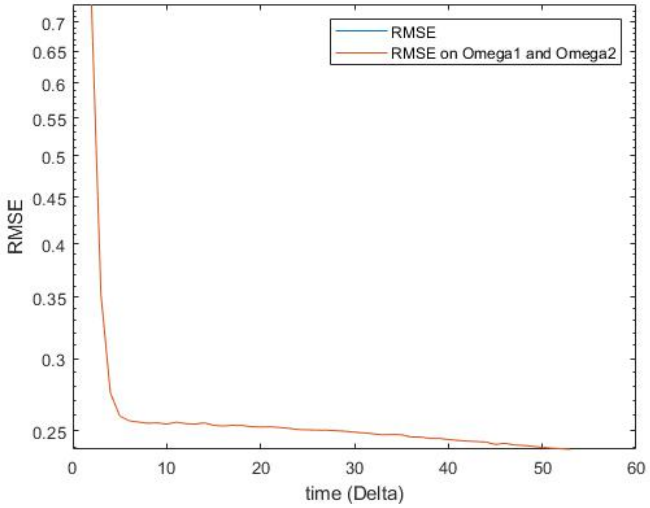


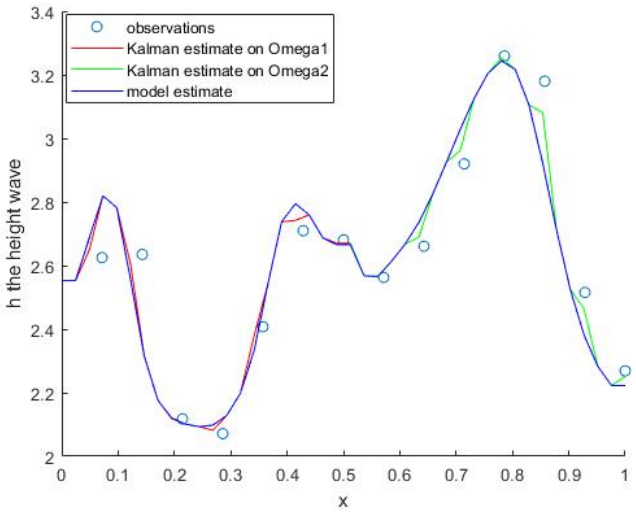
Figure 5.8: Behaviour of $RMSE_l^{\Omega_1 \cup \Omega_2}$ and $RMSE_l^{\Omega}$ versus time.

5.3 DD–4DVAR applied to SWEs problem

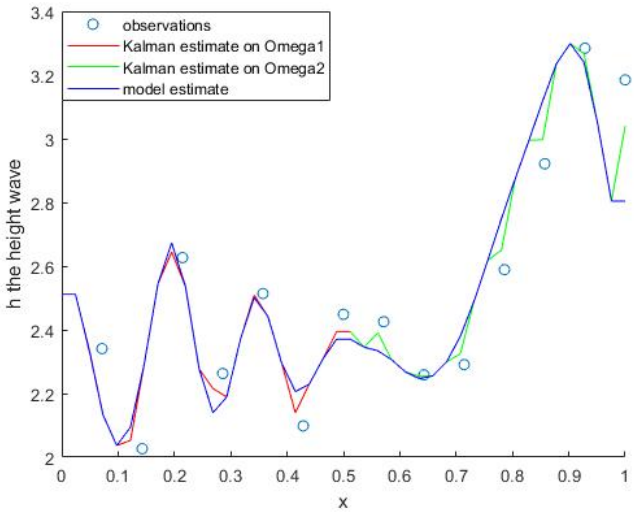
We perform validation analysis of the proposed approach considering the initial boundary problem of the SWEs defined in Section A.2 by considering the *inter-node* DD configuration on computing environment no.1. The discrete model is obtained using Lax–Wendroff scheme [81]. We underline that results we present are concerned not so much with parallel efficiency as with the trustworthy and usability of the proposed approach to solve this problem.

4DVAR DA set up. We consider the following experimental set up:

- $\Omega = (0, 1) \subset \mathbb{R}$: spatial domain;
- $\Delta_0 = [0, 1.5] \subset \mathbb{R}$: time interval;
- $p_v = 2$: number of physical variables;
- $N_p = 200$ and $b_c = 2$: numbers of elements of Ω and $\partial\Omega$, respectively;
- $n_x := N_p + b_c = 202$ and $N = 22$: numbers of elements of $\bar{\Omega}$ and Δ, respectively;



(a) *SWEs solution* $x[1]_{25} \in \mathbb{R}^{n_x}$ *and the DD-KF estimate* $\widehat{x}_{25}^{\Omega\times\Delta_1}[1] \in \mathbb{R}^{n_x}$ *at* t_{25} *on* Ω.



(b) *SWEs solution* $x[1]_{53} \in \mathbb{R}^{n_x}$ *(model estimate) and DD-KF estimate* $\widehat{x}_{53}^{\Omega\times\Delta_2}[1] \in \mathbb{R}^{n_x}$ *at* t_{53} *on* Ω.

Figure 5.9: SWEs solution and its DD-KF estimate.

- $\Delta x = \frac{1}{200}$ and Δt: step size of Ω and Δ, respectively, where Δt varies to satisfy the stability condition in (A.27) of the Lax-Wendroff method;
- $\Omega_I = \{x_i\}_{i=0,\dots,N_p-1}$: discretization of Ω where $x_i = i \cdot \Delta x$;
- $\Delta_K = \{t_k\}_{k=0,\dots,N-1}$: discretization of Δ where $t_k = t_{k-1} + \Delta t$;
- $n_{obs} = 2$: number of observations considered at each step $l = 0, 1, \dots, N$;
- $y \in \mathbb{R}^{N \cdot n_{obs}}$: observations vector at each step $l = 0, 1, \dots, N$;
- $H_l \in \mathbb{R}^{n_{obs} \times N_p}$: piecewise linear interpolation operator whose coefficients are computed using the nodes of model domain nearest to the observation values;
- $G \in \mathbb{R}^{N \cdot n_{obs} \times N_p}$: obtained as in (2.8) from the matrix H_l, $l = 0, 1, \dots, N$;
- $\sigma_m^2 = 0.5$, $\sigma_0^2 = 0.5$: model and the observational error variances;
- $\mathbf{B} \equiv B_l = \sigma_m^2 \cdot C$: covariance matrix of the error of the model at each step $l = 0, 1, \dots, N$, where $C \in \mathbb{R}^{N_p \times N_p}$ denotes the Gaussian correlation structure of the model errors in (5.3);
- $\mathbf{R}_l = \sigma_0^2 \cdot I_{n_{obs}, n_{obs}} \in \mathbb{R}^{n_{obs} \times n_{obs}}$: covariance matrix of the errors of the observations at each step $l = 0, 1, \dots, N-1$.
- $\mathbf{R} := diag(R_0, \dots, R_{N-1}) \in \mathbb{R}^{N \cdot n_{obs} \times N \cdot n_{obs}}$: a diagonal matrix obtained from the matrices R_l, $l = 0, 1, \dots, N-1$.

DD–4DVAR set up: we consider the following set up:

- $N_t = 2$: number of instants of time subdomains;
- $ad_1 = ad_2 = 1$: number of subdomains adjacent to Ω_1 and Ω_2, respectively;

- $\delta = 2$: the number of inner nodes in overlap regions;
- for $k = 1, \ldots, N-1$, $s_{k-1,k} = 1$: number of instants of time in temporal overlap;
- $N_{loc} = N_p/2 + \delta/2$: number of nodes in the interior of subdomains;
- $n_{obs_1} = n_{obs_2} = 1$;
- $N_1 = 11$ and $N_2 = 12$: number of instants of time t_l in time subdomains;
- $C := \{c_{i,j}\}_{i,j=1,\ldots,N_p} \in \mathbb{R}^{N_p \times N_p}$: the Gaussian correlation structure of the model error where

$$c_{i,j} = \rho^{|i-j|^2}, \quad \rho = exp\left(\frac{-\Delta x^2}{2}\right), \quad |i-j| < N_p/2 \quad \text{for } i,j = 1, \ldots, N_p \ . \tag{5.20}$$

1. Space-Time Decomposition Step.
   - Decomposition of Ω into two subdomains with overlap region:

$$\begin{aligned} \Omega_1 &= [0, x_{n_1+1}] \\ \Omega_2 &= [x_{n_1}, 1] \end{aligned} \tag{5.21}$$

     where n_1 is the numebr of inner nodes in Ω_1.
   - Decomposition of Δ into two subsets with overlap region:

$$\begin{aligned} \Delta_1 &= [0, t_{N_1-1}] \\ \Delta_2 &= [t_{N_1-1}, 1.5]. \end{aligned} \tag{5.22}$$

     where N_1 is the number of instants of time in Δ_1.
2. Model reduction of SWEs.
   Decomposition of matrices $M[1]_{l+1} \in \mathbb{R}^{N_p \times N_p}$, for $l = 0, 1, \ldots, N-2$ defined in (A.20)

$$M[1]^1_{l,l+1} := M[1]_{l,l+1}/\Omega_1 = \begin{bmatrix} \psi^l_1 & \eta^l_2 & & & \\ -\eta^l_1 & \psi^l_2 & \eta^l_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -\eta^l_{N_{p_1}-2} & \psi^l_{N_{p_1}-1} & \eta^l_{N_{p_1}} \\ & & & -\eta^l_{N_{p_1}-1} & \psi^l_{N_{p_1}} \end{bmatrix} \in \mathbb{R}^{N_{p_1} \times N_{p_1}} \tag{5.23}$$

$$M[1]^{1,2}_{l,l+1} := M[1]_{l,l+1}/\Omega_1 \times \Omega_2 = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & & & \vdots \\ \\ \eta^l_{N_{p_1}+1} & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{N_{p_1} \times N_{p_2}} \tag{5.24}$$

$$M[1]^{2,3}_{l,l+1} := M[1]_{l,l+1}/\Omega_2 \times \Omega_1 = \begin{bmatrix} 0 & \cdots & 0 & 0 & \eta^l_{n_1} \\ 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & & & \vdots \\ \\ 0 & \cdots & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{N_{p_2} \times N_{p_1}} \tag{5.25}$$

$$M[1]^{2}_{l,l+1} := M[1]_{l,l+1}/\Omega_2 \times \Omega_2 = \begin{bmatrix} \psi^l_{N_{p_1}+1} & \eta^l_{N_{p_1}+2} & & & \\ -\eta^l_{N_{p_1}+1} & \psi^l_{N_{p_1}+2} & \eta^l_{N_{p_1}+2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\eta^l_{N_p-2} & \psi^l_{N_p-1} & \eta^l_{N_p} \\ & & & -\eta^l_{N_p-1} & \psi^l_{N_p} \end{bmatrix} \in \mathbb{R}^{N_{p_2} \times N_{p_2}}. \tag{5.26}$$

We proceed in the same way to get the decomposition of $M[2]_{l,l+1} \in \mathbb{R}^{N_p \times N_p}$ in (A.25) into $M[2]^1_{l,l+1} \in \mathbb{R}^{N_{p_1} \times N_{p_1}}$, $M[2]^{1,2}_{l,l+1} \in \mathbb{R}^{N_{p_1} \times N_{p_2}}$, $M[2]^{2,3}_{l,l+1} \in \mathbb{R}^{N_{p_2} \times N_{p_1}}$ and $M[2]^2_{l,l+1} \in \mathbb{R}^{N_{p_2} \times N_{p_2}}$.

3. Decomposition of matrices:

   - $G \in \mathbb{R}^{(N \cdot n_{obs}) \times (N_p \cdot N)}$ into $G^k_i \in \mathbb{R}^{\bar{s}_k \cdot n_{obs_i} \times N_{p_i}}$, $i = 1, 2$;
   - $\mathbf{B} = \mathbf{V}\mathbf{V}^T \in \mathbb{R}^{N_p \times N_p}$ into $\mathbf{B}_i = \mathbf{V}_i\mathbf{V}_i^T, \mathbf{B}_{i,j} = \mathbf{B}/\Gamma_{i,j} \in \mathbb{R}^{N_{p_i} \times N_{p_i}}$, $i, j = 1, 2$;
   - $\mathbf{R} \in \mathbb{R}^{N \cdot n_{obs} \times N \cdot n_{obs}}$ into $\mathbf{R}_{i,k} \in \mathbb{R}^{\bar{s}_k \cdot n_{obs_i} \times \bar{s}_k \cdot n_{obs_i}}$, $i = 1, 2$.

DD procedure for solving 4DVAR DA problem for SWEs :

let $\mathbf{u}^n_{i,0} \equiv \mathbf{u}_0/\Omega_i$, i.e. $u^n_{i,0}[j] \equiv u_0[j]/\Omega_i$, $j = 1, 2$; computation proceeds on each subdomain in

space and time, i.e. on $\Omega_i \times \Delta_k$, along the assimilation window, i.e. for $k = 1, \ldots, N_t$, in the following way:

1. defines the reduced models

$$M_{i,k}[\hat{j}] := M[\hat{j}]^{i}_{\bar{s}_{k-1},\bar{s}_k} \quad M^k_i[\hat{j}]^{i,i+1} := M[\hat{j}]^{i,i+1}_{\bar{s}_{k-1},\bar{s}_k}.$$

2. sends and receives boundary conditions from adjacent domains and computes the vectors

$$b[\hat{j}]^n_{i,k} := M_{i,k}[\hat{j}]^{i,i+1} u^n_{i_j,k-1} + b[\hat{j}]/\Omega_i \quad i_j = 1, 2 : -1,\ j = 1$$

3. computes reduced models

$$u^{M_{i,k},n+1}_{i,k}[\hat{j}] = M_{i,k}[\hat{j}] u^{n+1}_{i,k-1}[\hat{j}] + b[\hat{j}]^n_{i,k}.$$

4. computes DA solution $\mathbf{w}^n_{i,k}[\hat{j}]$ by solving systems in (3.39).

   (4.1) iterative update of r.h.s of (3.41) for $r = 0, 1, \ldots, \bar{r}$ by *sending/receiving* $\mathbf{w}^{r,n}_{i_j,k}[\hat{j}]$ and computation of $\mathbf{w}^{r+1,n}_{i,k}[\hat{j}]$ by solving the system in (3.41) by CG method;

5. updates the solution

$$u^{n+1}_{i,k+1}[\hat{j}] = u^{M_{i,k+1},n+1}_{i,k+1}[\hat{j}] + \mathbf{V_i} \mathbf{w}^n_{i,k}[\hat{j}].$$

The algorithm is made of two nested loops: the outer loop over index n, defines approximations of local model, while the inner loop over index r solve each local 4DVAR DA problems. Step 5 updates the local solution according to the PinT methodology. Final solution is obtained by gathering local solutions

$$u[\hat{j}] := \{u_k[\hat{j}]\}_{k=1,\ldots,N_t}, \tag{5.27}$$

where

$$u_{k+1}[\hat{j}] := [u^{\bar{n}}_{i,k+1}[\hat{j}]]' \tag{5.28}$$

and $\bar{n}$ is the iterations number needed to stop DD algorithm.

Validation results. Figures 5.13-5.14 show $u[1]$, numerical value of h, the SWEs solution on $\Omega_i \times \Delta_k$, with $i, k = 1, 2$.

- Metrics.

  In order to show the convergence behaviour of the algorithm we quantify the error between two successive approximations using the following metrics:

$$E^{r,\Omega_i\times\Delta_k} = \|\mathbf{w}[1]_{i,k}^{r+1} - \mathbf{w}[1]_{i,k}^{r}\|,$$

$$E^{n,\Omega_i\times\Delta_k} = \|u[1]_{i,k}^{n+1} - u[1]_{i,k}^{n}\|,$$

  estimating the accuracy on the computed solution with respect to r-iterations and with respect to n-iterations. Then, we define the following quantity

$$E^{r,\Omega_i\times\Delta} = \{E^{r,\Omega_i\times\Delta_k}\}_{k=1,\dots,N_t}$$

  accounting for the behavior of $E^{r+1,\Omega_i\times\Delta_k}$ on the whole time interval Δ, for $k = 1, \dots, N_t$. Moreover, as ASM convergence in space demonstrated for 3D problems in [32] also holds for DD in space of 4DVAR DA problems, in order to show the reliability of the DD in time, we compute

$$\widetilde{u}_k^{DD-DA}[1] := \{u_l^{ASM}[1]\}_{t_l\in\Delta_k} = \{u_{i,l}^{ASM}[1]\}_{i=1,\dots,N_{sub},l=\bar{s}_{k-1},\dots,\bar{s}_k+N_k}$$

  i.e. the restriction to Δ_k of the solution of the 4DVAR DA problem in (A.18) obtained applying ASM. Then, we show the behaviour of

$$E_k^n = \|\widetilde{u}_k^{DD-DA}[1] - u_k^n[1]\|, \quad k = 1, .., N_t,$$

  where $u_k^n[1] := [u_{i,k}^n[1]]'$.

- Validation analysis.

  Figure 5.15 shows convergence behaviour of the inner loop, i.e. the behaviour of $E^{r+1,\Omega_1\times\Delta}$ and $E^{r+1,\Omega_2\times\Delta}$ versus iteration number. Note that in about four steps ASM reaches the input required tolerance ($tol = 10^{-4}$).
  In Figures 5.16- 5.17 we show convergence history of the outer loop, i.e. the DD in space. We see that $E^{n,\Omega_i\times\Delta_k}$ rapidly decreases, and in about five steps it settle on 10^{-15}, which is the maximum attainable accuracy in double precision. Considering that the values of $u[1]$ are in the range [-1,1]. Concerning the trustworthy of the DD in time, finally, in Figure 5.18 the behaviour of E_1^n and E_2^n versus iteration number is shown. Note that after about four iterations errors reach the order of 10^{-13} (resp. 10^{-14}) and the maximum attainable accuracy in double precision after 13 (resp. 21) iterations.
  Time (measured in seconds) needed to compute $u[1]$, which is the solution on $\Omega \times \Delta$ in (5.27) when $N_p = 200$, $N = 101$ is

$$T(N_p, N) = 4.22 \quad secs$$

  while $T(n_{loc}, N_1) = T(n_{loc}, N_1) = 0.71 \quad secs$

Results validate the trustworthiness of the proposed algorithm.

5.3.1 Performance analysis and scalability prediction

In this section we discuss the impact of the space and time decomposition approach described in Section 6 on the performance of the algorithm. Performance metrics are the time complexity and scalability. As we intend to mainly focus on the benefits arising from using domain decomposition approaches in data assimilation algorithms we consider the so called scale-up factor introduced firstly in [30]. Our aim is to highlight the benefits arising from using the space and

time decomposition approach instead of solving the problem on the whole domain. As we shall discuss later, the performance gain that we get from using the space and time decomposition approach is two fold:

1. Instead of solving one larger problem we can solve several smaller problems which are better conditioned than the former problem. This result leads to a reduction of each local time complexity.

2. Subproblems reproduce the whole problem at smaller dimensions and they are solved in parallel. This result leads to a reduction of software execution time.

In our discussion we assume the following set–up

Definition 16. *Let*

$$n_{proc} := q \times p$$

be an uniform bi–directional decomposition of domain $\Omega \times \Delta$*. Then if*

$$size(\Omega \times \Delta) = N_p \times N,$$

denotes the size of the whole domain, then

$$size(\Omega_i \times \Delta_k) = D_t \times D_s, \quad j = 1, \ldots, q; \quad i = 1, \ldots, p$$

where $D_s = \frac{N_p}{q} \geq 1$*, and* $D_t = \frac{N_t}{p} \geq 1$.

We let

$$N_{tot} := N_p \times N; \quad N_{loc} := D_s \times D_t \quad .$$

Furthermore, we let m_{ji} and l_{ji} be the number of steps of the outer/inner-loop, of local algorithm, respectively. Then, let

$$P(N_{loc}) = a_d N_{loc}^d + a_{d-1} N_{loc}^{d-1} + \ldots + a_0, \quad a_d \neq 0$$

be the polynomial expressing the complexity of most time–consuming operation of local algorithm.

Let

$$m_{max} := \max_{ji} m_{ji}; \quad l_{max} := \max_{ji} l_{ji}.$$

Observe that m_{max} and l_{max} actually are the number of steps of the outer and inner loops of the whole parallel algorithm. Let m_G and l_G denote the number of iterations of inner and outer loop of serial algorithm, we let

$$\rho^G := m_G \times l_G \quad ; \quad \rho^{ji} := m_{ji} \times l_{ji} \quad ; \quad \rho^{DD} := m_{max} \times l_{max}$$

Following result straightforwardly derives from the definition of the scale-up factor and gives an asymptotic estimate of the algorithm's scalability [30, 39]:

Theorem 9 (Scale up factor)**.** *if $T[\mathcal{A}^G]$ denote the computational time of serial algorithm and $T[\mathcal{A}^{loc}]$ the computational time of the local algorithm, we define*

$$Sc_{n_{proc}} := \frac{1}{n_{proc}} \times \frac{T(\mathcal{A}^G)}{T(\mathcal{A}^{loc})} \ ,$$

be the (relative) scale-up factor measuring the performance gain with respect to the serial algorithm. It is:

$$Sc_{n_{proc}} \geq \frac{\rho^G}{\rho^{DD}} \alpha(N_{loc}, n_{proc}) \, (n_{proc})^{d-1} \tag{5.29}$$

where

$$\alpha(N_{loc}, n_{proc}) = \frac{a_d + a_{d-1}\frac{1}{N_{loc}} + \ldots + \frac{a_0}{N_{loc}^d}}{a_d + a_{d-1}\frac{n_{proc}}{N_{loc}} + \ldots + \frac{a_0 (n_{proc})^d}{N_{loc}^d}} \quad ,$$

and, if $N_{loc} = N_{proc}$

$$\lim_{n_{proc} \to N_{loc}} \alpha(N_{loc}, n_{proc}) = \beta \in]0, 1]$$

.

Furthermore, it holds that

$$\lim_{n_{proc}\to N_{loc}} \alpha(N_{loc}, n_{proc}) = 1$$

Finally, if n_{proc} is fixed

$$\lim_{N_{loc}\to\infty} \alpha(N_{loc}, n_{proc}) = 1.$$

♠

If N_{loc} is fixed, it is

$$\lim_{n_{proc}\to N_{loc}} Sc_{1,n_{proc}} = \beta \cdot N_{loc}^{d-1} \quad ;$$

while, if n_{proc} is fixed

$$\lim_{N_{loc}\to\infty} Sc_{1,n_{proc}} = const \neq 0 \quad .$$

From (5.29) it results that, the growth of the scale up factor of DD algorithm is essentially one order less than the time complexity of the serial algorithm.

In particular, in our set–up, we get that performance gain of DD algorithm is

$$Sc_{1,n_{proc}} = \frac{1}{4} \times \frac{4.22}{0.71} = 1.5$$

Besides the time complexity, scalability is also affected by the communication overhead of the parallel algorithm. The surface-to-volume ratio is a measure of the amount of data exchange (proportional to surface area of domain) per unit operation (proportional to volume of domain). This means that, in order provide the best mapping on the given parallel architecture, it needs to find the appropriate value of the number of subdomains giving the right trade off between the scale up and the overhead of the algorithm. This analysis will be provided on the forthcoming parallel software.

5.3.2 The role of the overlapping region

We analyse the role of the overlapping region in DD–4DVAR method in chapter (3.3). We perform DD–4DVAR algorithm on computing environment no.2 by using Parallel Computing Toolbox of MATLABR2013a. MATLAB does not support multi–level parallelism with the Parallel Computing Toolbox, consequently, we only run in parallel local problems defined on $\{\Omega_1 \times \Delta_k\}_{k=1,\dots,Nt}, \dots, \{\Omega_{nsub} \times \Delta_k\}_{k=1,\dots,Nt}$.
We consider the following experimental set up.

4DVAR DA set up.

- $\Omega = (0,1) \subset \mathbb{R}$: spatial domain;
- $\Delta = [0, 1.5] \subset \mathbb{R}$: time interval;
- N_p: numbers of inner nodes of Ω defined in (2.4);
- $N = 9$: numbers of time in Δ;
- $n_{obs} = 64$: number of observations considered at each step $l = 0, 1, \dots, N$;
- $y \in \mathbb{R}^{N \cdot n_{obs}}$: observations vector at each step $l = 0, 1, \dots, N$;
- $H_l \in \mathbb{R}^{n_{obs} \times N_p}$: piecewise linear interpolation operator whose coefficients are computed using the nodes of model domain nearest to the observation values;
- $G \in \mathbb{R}^{N \cdot n_{obs} \times N_p}$: obtained as in (2.8) from the matrix H_l, $l = 0, 1, \dots, N$;
- $\sigma_m^2 = 0.5$, $\sigma_0^2 = 0.5$: model and the observational error variances;
- $\mathbf{B} \equiv B_l = \sigma_m^2 \cdot C$: covariance matrix of the error of the model at each step $l = 0, 1, \dots, N$, where $C \in \mathbb{R}^{N_p \times N_p}$ denotes the Gaussian correlation structure of the model errors in (5.20);

- $\mathbf{R}_l = \sigma_0^2 \cdot I_{n_{obs},n_{obs}} \in \mathbb{R}^{n_{obs}\times n_{obs}}$: covariance matrix of the errors of the observations at each step $l = 0, 1, \ldots, N-1$.

- $\mathbf{R} \in \mathbb{R}^{N\cdot n_{obs}\times N\cdot n_{obs}}$: a diagonal matrix obtained from the matrices R_l, $l = 0, 1, \ldots, N-1$.

DD–4DVAR set up: we consider the following set up:

- p: number of cores;

- $N_{sub} \equiv p$: number of spatial subdomains;

- $N_t = 2$: number of instants of time intervals;

- $n_1 = n_{nsub} = 1$ and $n_2 = n_3 = \ldots = n_{nsub-1} = 2$: number of subdomains adjacent to $\Omega_1,\ \Omega_{nsub}$ and $\Omega_2,\ \Omega_3, \ldots, \Omega_{nsub-1}$, respectively;

- δ: number of inner nodes of overlap regions defined in (3.8);

- N_{loc}: inner nodes of subdomains defined in (3.10);

- $C := \{c_{i,j}\}_{i,j=1,\ldots,N_p} \in \mathbb{R}^{N_p\times N_p}$: the Gaussian correlation structure of the model error where

$$c_{i,j} = \rho^{|i-j|^2}, \quad \rho = exp\left(\frac{-\Delta x^2}{2}\right), \quad |i-j| < N_p/2 \qquad \text{for } i,j = 1,\ldots,N_p\ .$$

- $T^1(N_p, N)$: denoting sequential time (in seconds) to perform DD–4DVAR on $N_{sub} = 1$ domain;

- $T^p(N_{loc}, N)$: denoting time (in seconds) needed to perform in parallel DD–4DVAR on N_{sub} subdomain;

- $S^p(N_{loc}, N) := \frac{T^1(N_p,N)}{T^p(N_{loc},N)}$: is the speed-up of DD–4DVAR parallel algorithm;

- $E^p(N_{loc}, N) := \frac{S^p(N_{loc},N)}{p}$: is the efficiency of DD–4DVAR parallel algorithm.

We introduce

$$e^p = e^p(\delta, \Delta x, \Delta t, N_{sub}, N_t) := \|\mathbf{u}^{\mathbf{DA}} - \tilde{\mathbf{u}}^{\mathbf{DD-DA}}\|_2, \tag{5.30}$$

where $\mathbf{u}^{\mathbf{DA}}$ denotes the minimum of the 4DVAR (global) functional $\mathbf{J}$ in (2.10) while $\tilde{\mathbf{u}}^{\mathbf{DA}}$ is obtained by gathering minimum of the local 4DVAR functionals $\mathbf{J}_{i,k}$ in (3.36) by considering different values of inner nodes of overlap regions δ defined in (3.8) and $p \equiv N_{sub}$. $\mathbf{u}^{\mathbf{DA}} \in \mathbb{R}^{N_p \times N}$ is computed by running DD–4DVar algorithm for $N_{sub} = 1$, while $\tilde{\mathbf{u}}^{\mathbf{DA}} \in \mathbb{R}^{N_p \times N}$ is computed by gathering local solutions obtained by running DD–4DVAR algorithm for different values of $N_{sub} > 1$ with $\delta \geq 0$, as shown in Figure (5.19).
In the following experimental results we analyse three aspects of DD–4DVAR algorithm:

1. The impact of δ defined in (3.8) to DD–4DVAR accuracy.

   In Table 5.7, fixed $\Delta x = 1.56 \times 10^{-3}$ and $\Delta t = 2.19 \times 10^{-2}$, we report values of error e^p defined in (5.30) for different values of δ. For $N_{sub} = 2,\ 4,\ 8,\ 16,\ 32,\ 64$, the order of magnitude in case $\delta = 2 \times 10^0$ is less than or equal to order of magnitude in cases $\delta = 1 \times 10^0, 2 \times 10^1, 1 \times 10^2, 2 \times 10^2, 4 \times 10^2, 6.4 \times 10^2$. In particular, as δ increases, N_{sub} decreases consequently, DD–4DVAR accuracy improved. The experimental results indicate that $\delta = 2$ is the optimal number of inner nodes in overlap regions.

2. Strong scaling of DD–4DVAR method.

   In Tables 5.8, 5.9, 5.10 and 5.11 we fixed the value of $\delta = 2$ and analyze the efficiency of DD–4DVAR algorithm. In Tables 5.8, 5.9 and 5.10 we note that efficiency strongly decreases for $N_{sub} = 16$, $N_{sub} = 32$ and $N_{sub} = 64$, respectively (see Figures 5.20, 5.21 and 5.22). For this reason we increase the value N_p defined in (2.4) and report the values in Tables 5.9, 5.10 and 5.11, respectively.

3. Isoefficiency of DD–4DVAR method.

In Table 5.12, we report the values of efficiency such that it is constant (isoefficiency) by increasing the value of N_p for each value of N_{sub}. For different values of N_{sub}, the number of inner nodes N_p in Ω has to increases at different rates in order to maintain a constant efficiency as number of subdomains increases. Figure 5.23 shows curve best fit data (N_{sub}, N_p) reported in Table 5.12. The equation of curve is

$$N_p = a \cdot p^2 + b \cdot p + c \tag{5.31}$$

where

$$a = -9.68 \times 10^{-2}, \quad b = 1.24 \times 10^{1}, \quad c = 6.25 \times 10^{2}$$

The function in (5.31) dictates how N_p must grow to maintain a fixed efficiency as N_{sub} increases. From the equations in (5.31), we can infer that the the number of inner nodes N_p of Ω needs to increase with the number of subdomains N_{sub}, i.e. the number of processes $p \equiv N_{sub}$, at an overall rate of $O(p^2)$ to maintain a fixed efficiency.

Table 5.7: Fixed $\Delta x = 1.56 \times 10^{-3}$ and $\Delta t = 2.19 \times 10^{-2}$ spatial and temporal step sizes of $M_{i,k}$ defined in (3.32), we report the values of N_{sub}, which is the number of subdomains, N_p number of inner nodes of Ω, N_{loc} number of inner nodes in each subdomains, e^p error defined in (5.30) and δ number of inner nodes in overlap regions, respectively.

$N_{sub} = 1$	$N_p = 640$	$e^1 = 0$	
N_{sub}	N_{loc}	δ	e^p
2	320	0	6.16×10^{-4}
2	321	2	4.69×10^{-4}
2	325	10	4.50×10^{-4}

2	330	20	4.71×10^{-4}
2	370	100	3.72×10^{-4}
2	420	200	3.60×10^{-4}
2	520	400	2.91×10^{-4}
1	640	640	0
4	160	0	1.14×10^{-3}
4	161	2	5.40×10^{-4}
4	165	10	5.68×10^{-2}
4	170	20	4.59×10^{-3}
4	210	100	4.30×10^{-3}
4	260	200	8.43×10^{-3}
2	360	400	3.93×10^{-4}
2	480	640	3.11×10^{-4}
8	80	0	3.05×10^{-3}
8	81	2	7.14×10^{-4}
8	85	10	8.79×10^{-2}
8	90	20	1.32×10^{-2}
8	130	100	1.55×10^{-2}
8	180	200	4.85×10^{-3}
4	280	400	8.07×10^{-3}
2	400	640	3.82×10^{-4}
16	40	0	4.90×10^{-3}
16	41	2	8.53×10^{-4}
16	45	10	8.98×10^{-2}
16	50	20	2.18×10^{-2}

8	90	100	1.34×10^{-2}
8	140	200	4.18×10^{-2}
4	240	400	9.19×10^{-3}
2	360	640	3.85×10^{-4}
32	20	0	1.13×10^{-2}
32	21	2	4.40×10^{-3}
32	25	10	5.07×10^{-2}
32	30	20	2.36×10^{-2}
16	70	100	3.59×10^{-2}
8	120	200	1.53×10^{-2}
4	220	400	4.05×10^{-3}
2	340	640	3.94×10^{-4}
64	10	0	1.11×10^{-2}
64	11	2	9.41×10^{-3}
64	15	10	2.42×10^{-2}
32	20	20	2.36×10^{-2}
16	60	100	2.18×10^{-2}
8	110	200	1.43×10^{-2}
4	210	400	4.23×10^{-3}
2	330	640	4.69×10^{-4}

Remark 10. *The majority of parallelization techniques first decompose domain in nonoverlapping subdomains then extend each subdomain with halo regions to enable stencil operations in a parallel context* [2]*. Instead, DD–4DVAR algorithm is based on overlapping DD allowing us*

[2]Parallelization technique applied to Regional Ocean Modeling System (ROMS) [62] and Nucleus for European

Table 5.8: DD–4DVAR performance results at $N_p = 640$ and $\delta = 2$. We report values of N_{sub}, which is the number of subdomains, N_p number of inner nodes of Ω, N_{loc} number of inner nodes in each subdomains, $T^1(N_p, N)$ sequential time (in seconds) to perform DD–4DVAR on $N_{sub} = 1$ domain, $T^p(N_{loc}, N)$ time (in seconds) needed to perform in parallel DD–4DVAR on N_{sub} subdomain, $S^p(N_{loc}, N)$ and $E^p(N_{loc}, N)$ the speed-up and efficiency of DD-KF parallel algorithm, respectively.

$nsub = 1$	$N_p = 640$	$T^1(N_p, N) = 6.04 \times 10^0$		
$nsub$	N_{loc}	$T^p(N_{loc}, N)$	$S^p(N_{loc}, N)$	$E^p(N_{loc}, N)$
2	321	4.14×10^0	1.47×10^0	7.29×10^{-1}
4	161	1.55×10^0	3.91×10^0	9.76×10^{-1}
8	81	1.44×10^0	4.20×10^0	5.25×10^{-1}
16	41	1.52×10^0	3.90×10^0	2.44×10^{-1}

Table 5.9: DD–4DVAR performance results at $N_p = 832$ and $\delta = 2$. We report values of N_{sub}, which is the number of subdomains, N_p number of inner nodes of Ω, N_{loc} number of inner nodes in each subdomains, $T^1(N_p, N)$ sequential time (in seconds) to perform DD–4DVAR on $p \equiv N_{sub} = 1$ domain, $T^p(N_{loc}, N)$ time (in seconds) needed to perform in parallel DD–4DVAR on $p \equiv N_{sub}$ subdomain, $S^p(N_{loc}, N)$ and $E^p(N_{loc}, N)$ the speed-up and efficiency of DD-KF parallel algorithm, respectively.

$nsub = 1$	$N_p = 832$	$T^1(N_p, N) = 2.97 \times 10^1$		
$nsub$	N_{loc}	$T^p(N_{loc}, N)$	$S^p(N_{loc}, N)$	$E^p(N_{loc}, N)$
16	53	1.95×10^0	1.52×10^1	9.47×10^{-1}
32	27	1.93×10^0	1.58×10^1	4.93×10^{-1}
64	14	2.75×10^0	1.12×10^1	1.74×10^{-1}

Table 5.10: DD–4DVAR performance results at $N_p = 896$ and $\delta = 2$. We report values of N_{sub}, which is the number of subdomains, N_p number of inner nodes of Ω, N_{loc} number of inner nodes in each subdomains, $T^1(N_p, N)$ sequential time (in seconds) to perform DD–4DVAR on $N_{sub} = 1$ domain, $T^p(N_{loc}, N)$ time (in seconds) needed to perform in parallel DD–4DVAR on $p \equiv N_{sub}$ subdomain, $S^p(N_{loc}, N)$ and $E^p(N_{loc}, N)$ the speed-up and efficiency of DD-KF parallel algorithm, respectively.

$nsub = 1$	$N_p = 896$		$T^1(N_p, N) = 5.90 \times 10^1$	
$nsub$	N_{loc}	$T^p(N_{loc}, N)$	$S^p(N_{loc}, N)$	$E^p(N_{loc}, N)$
32	29	1.89×10^0	3.13×10^1	9.73×10^{-1}
64	15	3.11×10^0	1.89×10^1	2.95×10^{-1}

Table 5.11: DD–4DVAR performance results at $N_p = 1024$ and $\delta = 2$. We report values of N_{sub}, which is the number of subdomains, N_p number of inner nodes of Ω, N_{loc} number of inner nodes in each subdomains, $T^1(N_p, N)$ sequential time (in seconds) to perform DD–4DVAR on $N_{sub} = 1$ domain, $T^p(N_{loc}, N)$ time (in seconds) needed to perform in parallel DD–4DVAR on $p \equiv N_{sub}$ subdomain, $S^p(N_{loc}, N)$ and $E^p(N_{loc}, N)$ the speed-up and efficiency of DD-KF parallel algorithm, respectively.

$nsub = 1$	$N_p = 1024$		$T^1(N_p, N) = 1.59 \times 10^2$	
$nsub$	N_{loc}	$T^p(N_{loc}, N)$	$S^p(N_{loc}, N)$	$E^p(N_{loc}, N)$
64	17	2.70×10^0	5.89×10^1	9.21×10^{-1}

Table 5.12: DD–4DVAR isoefficiency: Fixed $\delta = 2$, values of efficiency for different values of number of inner nodes N_p in Ω and number of subdomains N_{sub}.

N_{sub}	N_p	$E^p(N_{loc}, N)$
2	640	7.29×10^{-1}
4	660	9.44×10^{-1}
8	728	6.03×10^{-1}
16	832	9.47×10^{-1}
32	896	9.73×10^{-1}
64	1024	9.21×10^{-1}

to minimize the communications among subdomains adjacent to only nodes in interfaces Γ_{ij} defined in (3.4) $\forall i = 1, \dots N_{sub}$ and $j \in J_i$ where J_i is set of indices of adjacent subdomains to Ω_i. In particular, if width of halo regions are equal to one grids nodes wide, the halo nodes are actually the nodes of Γ_{ij}, $\forall i = 1, \dots N_{sub}$ and $j \in J_i$ (see Figure 5.24). Otherwise halo regions contain more nodes than interfaces nodes and this means more communications among adjacent subdomains. Hence, DD–4DVAR algorithm reduces communications times and increases memory overhead due to increase of number of inner nodes N_{loc} in (3.10) which depends on number of inner nodes in overlap regions δ in (3.8). We analyse the accuracy and efficiency of DD–4DVAR algorithm for different choices of δ and we prove that optimal value of number of inner nodes is $\delta = 2$, in order that it is the number of nodes required to discretization scheme (i.e. Lax- Wendroff scheme [81]).

Modelling of the Ocean (NEMO) [22, 52] needed respectively a halo of width two (or three for advection scheme) and one grid nodes.

5.3.3 Sensitivity Analysis: consistency and stability

We refer to 4DVAR and DD–4DVAR set in section 5.3.2. In the following, we present experimental results of the consistency and stability of DD–4DVAR method considering the initial boundary problem of the SWEs 1D. The discrete model is obtained using Lax–Wendroff scheme [81] on $\Omega \times \Delta$ where the orders of convergence in space and time are $p = q = 2$.

- Consistency of DD–4DVAR method.

  From Table 1.1, we could assume that[3]

  $$e^p\left(\frac{\Delta x}{d}, \frac{\Delta t}{d}\right) \approx \frac{e^p(\Delta x, \Delta t)}{d^2} \quad d = 1, 2, 4, 6, 8, 10. \tag{5.32}$$

  As shown in Table 5.13 and Figure 5.25, *the experimental order of consistency corresponds to the theoretical one obtained in Theorem 3.*

- Stability of DD–4DVAR method.

  In Table 5.13 and Figure 5.26, we report values of $\bar{E}_k$ for different values of perturbation $\bar{e}_k$ on initial condition of $P_{i,k}^{M_{i,k}}$ defined in (3.31). From the values in Table 5.13, we experimentally estimate C_k in (3.110), in particular

  $$C_k \approx 2.00 \times 10^1 \quad \forall k = 1, \ldots, N_t.$$

  Consequently, *DD–4DVAR with the initial boundary problem of SWEs 1D is well-conditioned.*

[3] By fixing the values of $N_{sub} = N_t = 4$ and $\delta = 2$, we get $e^p(2, \Delta x, \Delta t, 4, 4) = e^p(\Delta x, \Delta t)$ i.e. e^p depends only on Δx and Δt.

d	$\frac{\Delta x}{d}$	$\frac{\Delta t}{d}$	$e^p\left(\frac{\Delta x}{d}, \frac{\Delta t}{d}\right)$	$\frac{e^p(\Delta x, \Delta t)}{d^2}$
1	$7.87 \times \times 10^{-3}$	1.09×10^{-1}	1.53×10^{-2}	1.53×10^{-2}
2	$3.92 \times \times 10^{-3}$	5.47×10^{-2}	9.01×10^{-4}	3.83×10^{-3}
4	$1.96 \times \times 10^{-3}$	2.74×10^{-2}	6.45×10^{-4}	9.56×10^{-4}
6	$1.30 \times \times 10^{-3}$	1.83×10^{-2}	3.65×10^{-4}	2.39×10^{-4}
8	$9.78 \times \times 10^{-4}$	1.37×10^{-2}	3.99×10^{-4}	4.25×10^{-4}
10	$7.81 \times \times 10^{-4}$	1.10×10^{-2}	3.77×10^{-4}	1.53×10^{-4}

Table 5.13: Fixed $N_p = 640$ number of inner nodes in Ω, $N = 9$ number of instants of time in Δ, $N_{sub} = 4$ number of spatial subdomain and $N_t = 4$ time intervals. We report the values of e^p defined in (5.30) for different values of Δx and Δt spatial and temporal step sizes of $M_{i,k}$ defined in (3.32).

$\bar{e}_k$	$\bar{E}_k$
3.03×10^{-6}	6.06×10^{-5}
3.03×10^{-5}	6.06×10^{-4}
3.03×10^{-4}	6.06×10^{-3}
3.03×10^{-3}	6.06×10^{-2}

Table 5.14: Fixed $N_p = 640$ number of inner nodes in Ω, $N = 20$ number of instants of time in Δ, $N_{sub} = 4$ number of spatial subdomains and $N_t = 4$ time intervals. For $k = 1, 2, 3, 4$, we report the values of $\bar{E}_k$ defined in (3.98) for different values of perturbation $\bar{e}_k$ to initial condition of $P_{i,k}^{M_{i,k}}$ defined in (3.31).

5.4 DyDD: Performance analysis

Simulations were aimed to validate the proposed approach by measuring performance of DyDD algorithm. Performance evaluation was carried out by considering a *intra-node* DD configuration on computing environment no.2.

DyDD set up. We will refer to the following quantities:

- $\Omega \subset \mathbb{R}^2$: spatial domain;
- $N_p = 2048$: mesh size;
- n_{obs} : number of observations;
- p: number of subdomains and processing units;
- i: identification number of processing unit, which is the same of the associated subdomain;
- for $i = 1, \ldots, p$, $deg(i)$: degree of i, i.e. number of subdomains adjacent to Ω_i;
- $i_{ad}(i) \in \mathbb{N}$: identification of subdomains adjacent to Ω_i;
- $l_{in}(i) \in \mathbb{N}$: number of observations in Ω_i before the dynamic load balancing;
- $l_r(i) \in \mathbb{N}$: number of observations in Ω_i after DD step of DyDD procedure;
- $l_{fi}(i) \in \mathbb{N}$: number of observations in Ω_i after the dynamic load balancing;
- $T^p_{DyDD}(n_{obs})$: time (in seconds) needed to perform DyDD on p processing units;
- $T_r(n_{obs})$: time (in seconds) needed to perform re-partitioning of Ω;
- $Oh_{DyDD}(n_{obs}) = \frac{T_r(n_{obs})}{T^p_{DyDD}(n_{obs})}$ overhead time to the dynamic load balancing.

As measure of the load balance introduced by DyDD algorithm, we use:

$$\mathcal{E} = \frac{\min_i(l_{fi}(i))}{\max_i(l_{fi}(i))}$$

i.e. we compute the ratio of the minimum to the maximum of the number of observations of subdomains $\Omega_1, \ldots, \Omega_p$ after DyDD, respectively. As a consequence, $\mathcal{E}$ = 1 indicates a perfectly

balanced system.

Regarding DD–DA, we let $n_{loc} := \frac{N_p}{p}$ be local problem size and we consider as performance metrics, the following quantities:

- $T^1(n_{obs}, N_p)$ denoting sequential time (in seconds) to perform KF solving CLS problem;
- $T^p_{DD-DA}(n_{obs}, n_{loc})$ denoting time (in seconds) needed to perform in parallel DD-KF solving CLS problem after DyDD;
- $T^p_{oh}(n_{obs}, n_{loc})$ being the overhead time (measured in seconds) due to synchronization, memory accesses and communication time among p cores;
- $\widehat{x}_{KF} \in \mathbb{R}^{N_p}$ denoting KF estimate obtained by applying the KF procedure on CLS problem after DyDD;
- $\widehat{x}_{DD-DA} \in \mathbb{R}^{N_p}$ denoting DD estimate obtained by applying DD-KF on CLS problem after DyDD;
- $error_{DD-DA} := \|\widehat{x}_{KF} - \widehat{x}_{DD-DA}\|$ denoting the error introduced by the DD framework;
- $S^p(n_{obs}, n_{loc}) := \frac{T^1(n_{obs},n)}{T^p_{DD-DA}(n_{obs},n_{loc})}$, which refers to the speed-up of DD-KF parallel algorithm;
- $E^p(n_{obs}, n_{loc}) := \frac{S^p(n_{obs},n_{loc})}{p}$ which denotes the efficiency of DD-KF parallel algorithm.

In the following tables we report results obtained by employing three scenarios, which are defined such that each one is gradually more articulated than the previous one. It means that the number of subdomains which are adjacent to each subdomain increases, or the number of observations and the number of subdomains increase. In this way the workload re distribution increases. Example 1: First configuration: $p = 2$ subdomains and $m = 1500$ observations. In

Table 5.15: Example 1. DyDD parameters in Case 1. Both subdomains have data but they are unbalanced. We report values of p, which is the number of subdomains, i the identification number of processing unit, $deg(i)$ degree of i, i.e. number of subdomains adjacent to Ω_i, $l_{in}(i)$ which is number of observations in Ω_i before dynamic load balancing, $l_{fi}(i)$ number of observations in Ω_i after dynamic load balancing, i_{ad} identification of subdomains adjacent to Ω_i.

p	i	$deg(i)$	l_{in}	l_{fin}	i_{ad}
2	1	1	1000	750	2
	2	1	500	750	1

Case1, both Ω_1 and Ω_2 have data i.e. observations, but they are unbalanced. In Case2, Ω_1 has observations and Ω_2 is empty. In Table 5.15 and Table 5.16, respectively, we report values of the parameters after applying DyDD algorithm. This is the simplest configuration we consider just to validate DyDD framework. In both cases, $l_{fi}(1)$ and $l_{fi}(2)$, i.e. number of observations of Ω_1 and Ω_2, are equal to the average load $\bar{l} = 750$ and $\mathcal{E}$ = 1. As the workload re distribution of Case 1 and Case 2 is the same, DD-KF performance results of Case 1 and Case 2 are the same, and they are reported in Table 5.23, for $p = 2$ only. In Table 5.17 we report performance results of DyDD algorithm.

Example 2: Second configuration. In this experiment we consider $p = 4$ subdomains and $n_{obs} = 1500$ observations, and four cases which are such that the number of subdomains not having observations, increases from 0 up to 3. In particular, in Case 1, all subdomains have observations. See Table 5.18. In Case 2, only one subdomain is empty, namely Ω_2. See Table 5.19. In Case 3, two subdomains are empty, namely Ω_1 and Ω_2 are empty. See Table 5.20. In Case 4, three subdomains are empty, namely Ω_j, for $j = 1, 2, 3$, is empty. See Table 5.21. In all cases, $\mathcal{E}$ reaches the ideal value 1 and $l_{fin}(i) = \bar{l} = 375$, $i = 1, 2, 3, 4$. Then, DD-KF performance results of all cases are the same and they are reported in Table 5.23 for $p = 4$. In

Table 5.16: Example 1. DyDD parameters in Case 2. Ω_2 is empty. We report values of p i.e. number of subdomains, i identification number of processing unit, $deg(i)$ degree of i, i.e. number of subdomains adjacent to Ω_i, $l_{in}(i)$ which is number of observations in Ω_i before dynamic load balancing, $l_r(i)$ number of observations in Ω_i after DD step of DyDD procedure, $l_{fi}(i)$ number of observations in Ω_i after dynamic load balancing, i_{ad} which is identification of subdomains which are adjacent to Ω_i.

p	i	$deg(i)$	l_{in}	l_r	l_{fin}	i_{ad}
2	1	1	1500	1000	750	2
	2	1	0	500	750	1

Table 5.17: Example 1. Execution times: we report values of $T^p_{DyDD}(n_{obs})$, time (in seconds) needed to perform DyDD on p processing units, $T_r(n_{obs})$, time (in seconds) needed to perform re-partitioning of Ω, $Oh_{DyDD}(n_{obs})$ overhead time due to dynamic load balancing and $\mathcal{E}$ measuring load balance.

Case	$T^p_{DyDD}(n_{obs})$	$T_r(n_{obs})$	$Oh_{DyDD}(m)$	$\mathcal{E}$
1	4.11×10^{-2}	0	0	1
2	3.49×10^{-2}	4.00×10^{-6}	1.15×10^{-4}	1

Table 5.22 we report performance results of the four cases.

Example 3. We consider $n_{obs} = 1032$ observations and a number of subdomains p equals to $p = 2, 4, 8, 16, 32$. We assume that all subdomains Ω_i has observations, i.e. for $i = 1, \ldots, p$, $l_{in}(i) \neq 0$; Ω_1 has $p-1$ adjacent subdomains, i.e. $n_{ad} := deg(1) = p-1$; Ω_i has 1 adjacent subdomain i.e. for $i = 2, \ldots, p$, $deg(i) = 1$; finally $i = 1, \ldots, p$, we let the maximum and the minimum number of observations in Ω_i be such that $l_{max} = max_i(l_{fin}(i))$ and $l_{min} = min_i(l_{fin}(i))$. Table 5.24 shows performance results and Figure 5.27 reports the error of DD-KF with respect to KF.

Example 4 We consider $n_{obs} = 2000$ observations and $p = 2, 4, 8, 16, 32$ we assume that Ω_i has observations, i.e. for $i = 1, \ldots, p$, $l_{in}(i) \neq 0$; Ω_1 and Ω_p have 1 adjacent subdomain i.e. $deg(1) = deg(p) = 1$; Ω_i and Ω_p have 2 adjacent subdomains i.e. for $i = 2, \ldots, p-1$, $deg(i) = 2$. In Table 5.26 we report performance results and in Figure 5.27 the error of DD-KF with respect to KF is shown.

Finally, regarding the accuracy of the DD-DA framework with respect to computed solution, in Table 5.25 (Examples 1-2) and in Figure 5.27 (Examples 3-4), we get values of $error_{DD-DA}$. We observe that the order of magnitude is about 10^{-11} consequently, we may say that the accuracy of local solutions of DD-DA and hence of local KF estimates, are not impaired by DD approach.

Table 5.18: Example 2. DyDD parameters in Case 1. All subdomains have data. We report values of p, which is the number of subdomains, i identification number of processing unit, $deg(i)$ degree of i, i.e. number of subdomains adjacent to Ω_i, $l_{in}(i)$ the number of observations in Ω_i before dynamic load balancing, $l_{fi}(i)$ the number of observations in Ω_i after dynamic load balancing, i_{ad} identification of subdomains which are adjacent to Ω_i.

p	i	$deg(i)$	l_{in}	l_{fin}	i_{ad}
4	1	2	150	375	[2 4]
	2	2	300	375	[3 1]
	3	2	450	375	[4 2]
	4	2	600	375	[3 1]

Table 5.19: Example 2. DyDD parameters in Case 2. Ω_2 is empty. We report values of p, which is number of subdomains, i i.e. identification number of processing unit, $deg(i)$ i.e. degree of i, i.e. number of subdomains which are adjacent to Ω_i, $l_{in}(i)$ i.e. number of observations in Ω_i before dynamic load balancing, $l_{fi}(i)$ number of observations in Ω_i after dynamic load balancing, i_{ad} identification of subdomains adjacent to Ω_i.

p	i	$deg(i)$	l_{in}	l_r	l_{fin}	i_{ad}
4	1	2	450	450	375	[2 4]
	2	2	0	225	375	[3 1]
	3	2	450	225	375	[4 2]
	4	2	600	600	375	[3 1]

Table 5.20: Example 2. DyDD parameters in Case 3. Ω_1 and Ω_2 are empty. We report values of p, which is the number of subdomains, i identification number of processing unit, $deg(i)$ i.e. degree of i, i.e. number of subdomains adjacent to Ω_i, $l_{in}(i)$ number of observations in Ω_i before dynamic load balancing, $l_{fi}(i)$ number of observations in Ω_i after dynamic load balancing, i_{ad} identification of subdomains which are adjacent to Ω_i.

p	i	$deg(i)$	l_{in}	l_r	l_{fin}	i_{ad}
4	1	2	0	300	375	[2 4]
	2	2	0	450	375	[3 1]
	3	2	900	450	375	[4 2]
	4	2	300	600	375	[3 1]

Table 5.21: Example 2. DyDD parameters in Case 4. Ω_1, Ω_2 and Ω_3 are empty. We report values of p i.e. number of subdomains, i identification number of processing unit, $deg(i)$ degree of i, i.e. i.e. number of subdomains which are adjacent to Ω_i, $l_{in}(i)$ the number of observations in Ω_i before dynamic load balancing, $l_{fi}(i)$ number of observations in Ω_i after dynamic load balancing and i_{ad} identification of subdomains which are adjacent to Ω_i.

p	i	$deg(i)$	l_{in}	l_r	l_{fin}	i_{ad}
4	1	2	0	500	375	[2 4]
	2	2	0	250	375	[3 1]
	3	2	0	250	375	[4 2]
	4	2	1500	500	375	[3 1]

Table 5.22: Example2. Execution times: we report values of $T^p_{DyDD}(n_{obs})$, i.e. time (in seconds) needed to perform DyDD algorithm on p processing units, $T_r(n_{obs})$ time (in seconds) needed to perform re-partitioning of Ω, $Oh_{DyDD}(n_{obs})$ overhead time to the dynamic load balancing and $\mathcal{E}$ parameter of load balance.

Case	$T^p_{DyDD}(n_{obs})$	$T_r(n_{obs})$	$Oh_{DyDD}(n_{obs})$	$\mathcal{E}$
1	5.40×10^{-2}	0	0	1
2	5.84×10^{-2}	2.35×10^{-4}	$0.4 \cdot 10^{-2}$	1
3	4.98×10^{-2}	3.92×10^{-4}	$0.8 \cdot 10^{-2}$	1
4	4.63×10^{-2}	5.78×10^{-4}	$0.1 \cdot 10^{-1}$	1

From these experiments, we observe that as the number of adjacent subdomains increases, data communications required by the workload re-partitioning among subdomains increases too. Accordingly, the overhead due to the load balancing increases (for instance see Table 5.22). As expected, the impact of such overhead on the performance of the whole algorithm strongly depends on the problem size and the number of available computing elements. Indeed, in Case 1 of Example 1 and of Example 2, when p is small in relation to n_{loc} (see Table 5.23) this aspect is quite evident. In Example 4, instead, as p increases up to 32, and n_{loc} decreases the overhead does not affect performance results (see Table 5.26). In conclusion, we recognize a sort of trade off between the overhead due to the workload re-partitioning and the subsequent parallel computation.

Table 5.23: Example 1-2: DD-KF performance results in Example 1 and Example 2. We report values of p, which is the number of subdomains, N_p mesh size, n_{loc} i.e. local problem size, m number of observations, $T^1\left(n_{obs}, N_p\right)$ sequential time (in seconds) to perform KF solving CLS problem, $T^p_{DD-DA}\left(n_{obs}, n_{loc}\right)$ time (in seconds) needed to perform in parallel DD–KF solving CLS problem with DyDD, $S^p\left(n_{obs}, n_{loc}\right)$ and $E^p\left(n_{obs}, n_{loc}\right)$ the speed-up and efficiency of DD–KF parallel algorithm, respectively. We applied DyDD to all cases of Example 1 and Example 2 and obtained the perfect load balance, as reported in Table 5.17 and Table 5.22, respectively. As the workload distribution is the same, DD–KF performance results are the same in all cases of Example 1, then we show results for $p = 2$, only. In the same way, for all cases of Example 2, we show results for $p = 4$, only.

$p = 1$	$N_p = 2048$	$n_{obs} = 1500$		$T^1(n_{obs}, N_p) = 5.67 \times 10^0$
p	n_{loc}	$T^p_{DD-DA}\left(n_{obs}, n_{loc}\right)$	$S^p\left(n_{obs}, n_{loc}\right)$	$E^p\left(n_{obs}, n_{loc}\right)$
2	1024	4.95×10^0	1.15×10^0	5.73×10^{-1}
4	512	2.48×10^0	2.29×10^0	5.72×10^{-1}

5.5 DyDDST: Performance analysis

Validation of DyDDST algorithm is performed by considering a *intra-node* DD configuration on computing environment no.2. DyDDST set up:

- $\Omega \subset \mathbb{R}^2$: spatial domain;
- $N_p = 2048$: mesh size;
- $N = 64$: number of elements of Δ;
- $N_{tot} = N_p \times N$: problem size;
- p: number of spatial subdomains and processing units;

Table 5.24: Example 3. Execution times: we report values of p, i.e. the number of subdomains, n_{ad}, the number of adjacent subdomains to Ω_1, $T^p_{DyDD}(n_{obs})$, time (in seconds) needed to perform DyDD on p processing units, $\mathcal{E}$ which measures load balance, l_{max} and l_{min} i.e. maximum and minimum number of observations between subdomains after DyDD, respectively. $\mathcal{E}$ depends on $n_{ad}(\equiv deg(1))$, i.e. as $n_{ad}(\equiv deg(1))$ increases (consequently p increases), then $\mathcal{E}$ decreases. For $i = 1, \ldots, p$, subdomain Ω_i has observations, i.e. $l_{in}(i) \neq 0$, consequently we do not need to perform re-partitioning of Ω, then $T_r(m) \equiv 0$.

p	n_{ad}	$T^p_{DyDD}(n_{obs})$	l_{max}	l_{min}	$\mathcal{E}$
2	1	6.20×10^{-3}	516	515	9.98×10^{-1}
4	3	2.60×10^{-2}	258	257	9.96×10^{-1}
8	7	9.29×10^{-2}	129	128	9.92×10^{-1}
16	15	1.11×10^{-1}	71	64	8.88×10^{-1}
32	31	1.36×10^{-1}	39	32	8.21×10^{-1}

Table 5.25: Examples 1-2. We report values of $error_{DD-DA}$, i.e. the error introduced by the DyDD framework, in Example 1 (with $p = 2$) and Example 2 (with $p = 4$).

p	$error_{DD-DA}$
2	8.16×10^{-11}
4	8.82×10^{-11}

Table 5.26: Example 4. Performance results of DyDD framework: we report values of p, whic is the number of subdomains, N_p i.e. the mesh size, n_{loc} i.e. the local problem size, n_{obs} the number of observations, $T^1\left(n_{obs}, N_p\right)$ i.e. sequential time (in seconds) needed to perform KF, $T^p_{DyDD}(n_{obs})$ i.e. time (in seconds) needed to perform DyDD on p processing units, $T^p_{DD-DA}\left(n_{obs}, n_{loc}\right)$ i.e. time (in seconds) needed to perform in parallel DD-DA with DyDD, $S^p\left(n_{obs}, n_{loc}\right)$ and $E^p\left(n_{obs}, n_{loc}\right)$, i.e. speed-up and efficiency of DD-DA parallel algorithm, respectively.

p	$N_p = 2048$	$n_{obs} = 2000$	$T^1(n_{obs}, N_p) = 4.88 \times 10^0$		
p	n_{loc}	$T^p_{DyDD}(m)$	$T^p_{DD-DA}\left(n_{obs}, n_{loc}\right)$	$S^p\left(n_{obs}, n_{loc}\right)$	$E^p\left(n_{obs}, n_{loc}\right)$
2	1024	4.10×10^{-3}	4.71×10^0	1.04×10^0	5.18×10^{-1}
4	512	4.29×10^{-2}	2.61×10^0	1.87×10^0	4.67×10^{-1}
8	256	1.07×10^{-1}	8.43×10^{-1}	5.79×10^0	6.72×10^{-1}
16	128	1.42×10^{-1}	3.46×10^{-1}	1.41×10^1	8.81×10^{-1}
32	64	3.49×10^{-1}	1.66×10^{-1}	2.94×10^1	9.19×10^{-1}

- $N_t = p$: number of instants of time intervals;
- i: identification number of processing unit, which is the same of the associated subdomain;
- for $k = 1, \ldots, N_t,\ m_k$: number of observations in Δ_k;
- $\mathbf{n_{obs}} := [n_{obs}(1), \ldots, n_{obs}(N_t)] \in \mathbb{N}^{N_t}$: vector of number of observations in Δ;
- for $i = 1, \ldots, p$, $k = 1, \ldots, N_t$
  - $d^k\left(i\right)$: degree of i in Δ_k, i.e. number of subdomains adjacent to Ω_i in Δ_k;
  - $l^k_{in}\left(i\right) \in \mathbb{N}$: number of observations in $\Omega_i\ in\ \Delta_k$ before the dynamic load balancing;
  - $l^k_{fi}\left(i\right) \in \mathbb{N}$: number of observations in $\Omega_i\ in\ \Delta_k$ after the dynamic load balancing.

- $Iter = max_{k=1,\ldots,N_t}\left(Iter\left(k\right)\right)$: maximum between the number of iterations needed to solve Laplacian system in (4.4) associated to G^k with Preconditioned Conjugate Gradient (PCG) method in each time interval Δ_k;
- $T^p_{DyDDST}\left(\mathbf{n_{obs}}\right)$: time (in seconds) needed to perform DyDDST on p processing units;
- $T_r\left(\mathbf{n_{obs}}\right)$: time (in seconds) needed to perform re-partitioning of Ω;
- $Oh_{DyDDST}\left(\mathbf{n_{obs}}\right) = \frac{T_r(\mathbf{n_{obs}})}{T^p_{DyDDST}(\mathbf{n_{obs}})}$: overhead time to the dynamic load balancing.

Regarding DD-DA, we let:

- $n_{loc} := \frac{N_p}{p} \times \frac{N}{p}$: be local problem size;
- $T^1\left(\mathbf{n_{obs}}, N_p\right)$: sequential time (in seconds) needed to DA;
- $T^p_{DD-DA}\left(\mathbf{n_{obs}}, n_{loc}\right)$: time (in seconds) needed to perform in parallel DD-DA solving CLS problem after DyDDST procedure;
- $S^p\left(\mathbf{n_{obs}}, n_{loc}\right) := \frac{T^1(\mathbf{n_{obs}},n)}{T^p_{DD-DA}(\mathbf{n_{obs}},n_{loc})}$: algorithm speed-up;
- $E^p\left(\mathbf{n_{obs}}, n_{loc}\right) := \frac{S^p(\mathbf{n_{obs}},n_{loc})}{p}$: algorithm efficiency.

As measure of the load balance of DyDDST algorithm, for $k = 1, \ldots, N_t$, we use [80]:

$$\mathcal{E}^k = \frac{\min_i\left(l^k_{fi}\left(i\right)\right)}{\max_i\left(l^k_{fi}\left(i\right)\right)} \tag{5.33}$$

i.e. we compute the ratio of the minimum to the maximum of the number of observations of subdomains $\Omega_1, \ldots, \Omega_p$ in Δ_k after applying DyDDST algorithm, respectively. Further, $\mathcal{E}^k = 1$ indicates a perfectly balanced system in Δ_k. In the following tables we report results obtained by employing three configurations. More precisely, fixed $p = 4$, for $k = 1, \ldots, 4$ configuration considered in Example 1 changes in Δ_k i.e. some subdomains are such that its number of adjacent subdomains changes in Δ_k, and the degree $d^k\left(i\right)$ of the vertex i of processor graph changes.

In Examples 2 and 3, for $p = 2, 4, 8, 32, 64$ and $k = 1, \ldots, 64$ configurations are the same in Δ_k while it changes the number of subdomains adjacent to each subdomain. Namely, in Example 2, subdomains Ω_1 and Ω_p have 1 adjacent subdomain and for $i = 2, \ldots, p-1$, Ω_i has 2 adjacent subdomains while in Example 3 Ω_1 has $p-1$ adjacent subdomains and for $i = 2, \ldots, p$, Ω_i has 1 adjacent subdomain. From these experiments, we observe that the number of subdomains adjacent to each subdomain increases parameter $\mathcal{E}^k$. On the contrary, as the number of adjacent subdomains increases, communications required by the workload repartitioning among subdomains increases, accordingly, the number of operations needed to compute the amount of observations required to obtain load balance increases, increasing $T^p_{DyDDST}(\mathbf{n_{obs}})$.

Example 1. *(Tables 5.27-5.29) First configuration: p=4 spatial subdomains and time intervals such that:*

1. *k=1:*
   - *for* $i = 1, \ldots, p : \Omega_i \times \Delta_k$*: have data i.e. observations;*
   - $d^k(1) = d^k(4) = 1, d^k(2) = d^k(3) = 2$.
2. *k=2:*
   - $\Omega_1 \times \Delta_k$*: is empty;*
   - *for* $j = 2, 3, 4$*:* $\Omega_j \times \Delta_k$ *have data;*
   - $d^k(1) = d^k(2) = 2, d^k(3) = 1, d^k(4) = 3$.
3. *k=3:*
   - *for* $i = 1, 2 : \Omega_i \times \Delta_k$*: is empty;*
   - *for* $j = 3, 4 : \Omega_i \times \Delta_k$*: have data;*

- $d^k(1) = d^k(4) = 1, d^k(2) = d^k(3) = 2.$

4. k=4:

- *for* $i = 1, 2, 3$*:* $\Omega_i \times \Delta_k$*: is empty;*
- $\Omega_4 \times \Delta_k$*: have data;*
- $d^k(1) = d^k(4) = 1, d^k(2) = d^k(3) = 2.$

Table 5.27: Example 1. For $k = 1$ all subdomains have data, consequently, it is not necessary to perform re-partitioning of Ω and $T_r(n_{obs}(k)) \equiv 0$.

p	k	$n_{obs}(k)$	$T^p_{DyDDST}(n_{obs}(k))$	$T_r(n_{obs}(k))$	$Oh_{DyDDST}(n_{obs}(k))$
4	1	2217	2.58×10^{-1}	0	0
	2	2933	8.11×10^{-2}	8.00×10^{-4}	9.90×10^{-3}
	3	1925	7.05×10^{-2}	8.00×10^{-4}	1.13×10^{-2}
	4	1678	5.82×10^{-2}	1.20×10^{-4}	1.37×10^{-2}

Table 5.28: Example 1. We report the values of $\mathcal{E}^k$ and $Iter(k)$. For $k = 1$ all subdomains have data, consequently, it is not necessary to perform re-partitioning of Ω and $T_r(n_{obs}(k)) \equiv 0$.

p	k	$n_{obs}(k)$	$\mathcal{E}^k$	$Iter(k)$
4	1	2217	9.98×10^{-1}	1
	2	2933	9.99×10^{-1}	2
	3	1925	9.98×10^{-1}	2
	4	1678	9.98×10^{-1}	2

Table 5.29: Example 1: We report values obtained by applying DD-DA after DyDDST in Example 1 (Tables 5.27- 5.28) where $\mathbf{n_{obs}} = [1617\ 2894\ 1098\ 2445]$.

$p=1$	$N_p = 131072$	$T^1(\mathbf{n_{obs}}, N_p) = 11.92 \times 10^0$		
p	n_{loc}	$T^p_{DD-DA}(\mathbf{n_{loc}}, n_{loc})$	$S^p(\mathbf{n_{obs}}, n_{loc})$	$E^p(\mathbf{n_{obs}}, n_{loc})$
4	8192	3.45×10^0	3.46×10^0	8.65×10^{-1}

Example 2. *(Table 5.30 and Figure 5.28): We consider* $p = 2, 4, 8, 16, 32, 64$ *such that: for* $k = 1, \ldots, 64$

- $d^k(1) = d^k(p) = 1$*: Ω_1 and Ω_p have 1 adjacent subdomain in Δ_k;*
- *for* $i = 2, \ldots, p-1$, $d^k(i) = 2$*: Ω_i has 2 adjacent subdomains in Δ_k;*
- $n_{obs}(k)$*: number of observations available in Δ_k as defined in Table 5.32.*

Example 3. *(Table 5.31 and Figure 5.29) Second configuration: we consider p=2, 4, 8, 16, 32, 64 such that: for* $k = 1, \ldots, 64$

- $d^k(1) = p - 1$*: Ω_1 has $p-1$ adjacent subdomains in Δ_k;*
- *for* $i = 2, \ldots, p$, $d^k(i) = 1$*: Ω_i has 1 adjacent subdomain in Δ_k;*
- $n_{obs}(k)$*: number of observations available in Δ_k defined in Table 5.32.*

We note that in Example 2 the number of subdomains which are adjacent to Ω_1 increases along the time window, while in Example 3 it equals to 2. Consequently, in Example 2, iterations needed to solve linear system in (4.4) (see Tables 5.30-5.31) are less than in Example 3.

Table 5.30: Example 2. Performance results where $\mathbf{n_{obs}}$ are defined in Table 5.32 .

p	$T^1(\mathbf{n_{obs}}, N_p)$

1		6.41×10^2				
p	n_{loc}	$T^p_{DyDDST}(\mathbf{n_{obs}})$	$T^p_{DD-DA}(\mathbf{n_{obs}}, n_{loc})$	$S^p(\mathbf{n_{obs}}, n_{loc})$	$E^p(\mathbf{n_{obs}}, n_{loc})$	$Iter$
2	32768	9.34×10^{-1}	3.62×10^2	1.78×10^0	8.89×10^{-1}	1
4	8192	3.13×10^0	1.65×10^2	3.88×10^0	9.69×10^{-1}	3
8	2048	8.12×10^0	9.95×10^1	6.65×10^0	8.07×10^{-1}	7
16	512	1.65×10^1	5.70×10^1	1.11×10^1	6.93×10^{-1}	15
32	128	3.08×10^1	3.26×10^1	1.98×10^1	6.15×10^{-1}	20
64	32	5.77×10^1	3.18×10^1	2.02×10^1	3.56×10^{-1}	20

Table 5.31: Example 3. Performance results where $\mathbf{n_{obs}}$ are defined in Table 5.32.

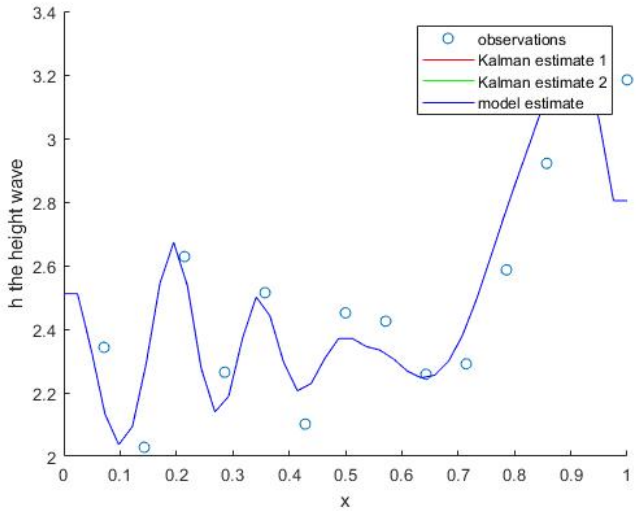
$p = 1$	N_p	$T^1(\mathbf{n_{obs}}, N_p)$				
1	131072	6.41×10^2				
p	n_{loc}	$T^p_{DyDDST}(\mathbf{n_{obs}})$	$T^p_{DD-DA}(\mathbf{n_{obs}}, n_{loc})$	$S^p(\mathbf{n_{obs}}, n_{loc})$	$E^p(\mathbf{n_{obs}}, n_{loc})$	$Iter$
2	32768	9.34×10^{-1}	3.62×10^2	1.78×10^0	8.89×10^{-1}	1
4	8192	2.44×10^0	1.65×10^2	3.90×10^0	9.74×10^{-1}	2
8	2048	6.72×10^0	1.02×10^2	6.31×10^0	7.89×10^{-1}	2
16	512	1.21×10^1	5.53×10^1	1.16×10^1	7.26×10^{-1}	2
32	128	2.14×10^1	3.37×10^1	1.91×10^1	5.96×10^{-1}	2
64	32	3.78×10^1	2.76×10^1	2.33×10^1	3.64×10^{-1}	2

Table 5.32: Example 2-3. For $k = 1, \ldots, 64$, values of $n_{obs}(k) \in \Delta_k$.

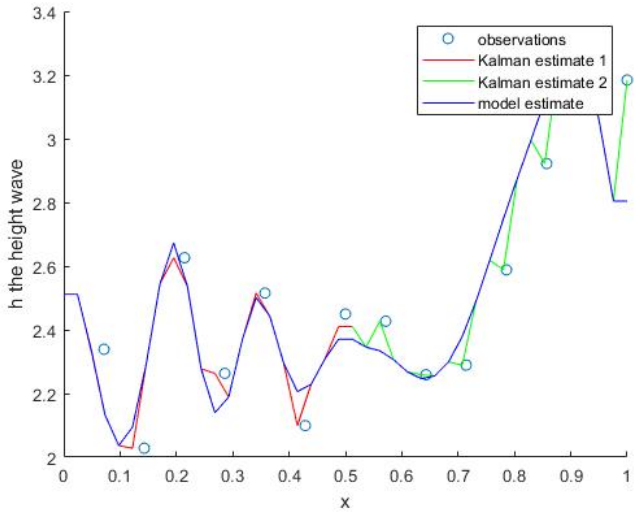
k	$n_{obs}(k)$	k	$n_{obs}(k)$	k	$n_{obs}(k)$	k	$n_{obs}(k)$	k	$n_{obs}(k)$	k	$n_{obs}(k)$
1	1618	9	2327	17	1579	25	2651	33	1209	41	2256
2	2419	10	1678	18	2744	26	2571	34	2626	42	1343

3	2523	11	1739	19	1493	27	2174	35	1683	43	2048
4	2869	12	1968	20	1912	28	2555	36	2332	44	2667
5	2260	13	2078	21	28906	29	1603	37	1691	45	2687
6	1874	14	2512	22	2439	30	963	38	2146	46	2184
7	2036	15	2613	23	2214	31	2270	39	233	47	2763
8	2536	16	2584	24	2476	32	2611	40	1772	48	2392

k	$n_{obs}(k)$	k	$n_{obs}(k)$
49	1552	57	2148
50	2397	58	2375
51	1547	59	1852
52	2987	60	2338
53	1656	61	2356
54	2581	62	2869
55	2332	63	1524
56	2477	64	1462



(a) *SWEs solution* $x[1]_{53} \in \mathbb{R}^{n_x}$ *(model estimate) and DD-KF estimates* $\widehat{x}_{53}^{\Omega\times\Delta_1}[1] \in \mathbb{R}^{n_x}$ *at* t_{53} *on* Ω *(Kalman estimate 1 and Kalman estimate 2) by considering* $\sigma_m^2 = 0$*. As expected, Kalman estimates and model estimate overlap.*



(b) *SWEs solution* $x[1]_{99} \in \mathbb{R}^{n_x}$ *(model estimate) and DD-KF estimates* $\widehat{x}_{25}^{\Omega\times\Delta_1}[1] \in \mathbb{R}^{n_x}$ *at* t_{53} *on* Ω_1 *and* Ω_2 *(Kalman estimate 1 and Kalman estimate 2) by considering the error variance* $\sigma_0^2 = 10^{-5}$*.*

Figure 5.10: SWEs solution and DD-KF estimates for different choices of σ_m^2 and σ_0^2.

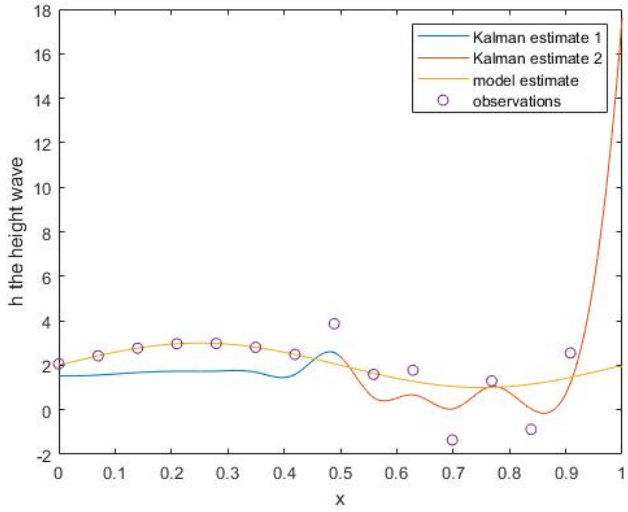


Figure 5.11: *SWEs solution* $x[1]_{10} \in \mathbb{R}^{n_x}$ *(model estimate) and DD-KF estimate* $\widehat{x}_{10}^{\Omega\times\Delta}[1] \in \mathbb{R}^{n_x}$ *at* t_{10} *on* Ω *(Kalman estimate 1 and Kalman estimare 2) by considering different errors and variances observations; i.e.* $v_{10}^1 = 10^{-15} \cdot \bar{v}_{10}^1$ *and* $\sigma_{1,0}^2 = 6.67 \times 10^{-1}$ *in* Ω_1 *and* $v_{10}^2 = 2 \cdot \bar{v}_{10}^2$ *and* $\sigma_{2,0}^2 = 4.28 \times 10^0$ *in* Ω_2. $\bar{v}_{10}^1$ *and* $\bar{v}_{10}^2$ *are random vectors drawn from the standard normal distribution.*

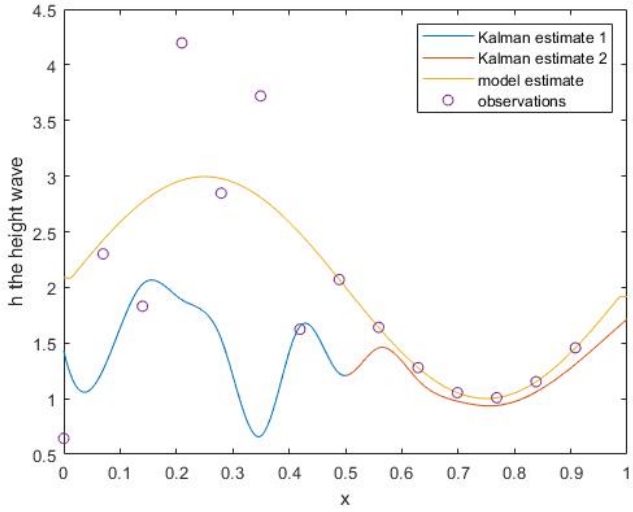
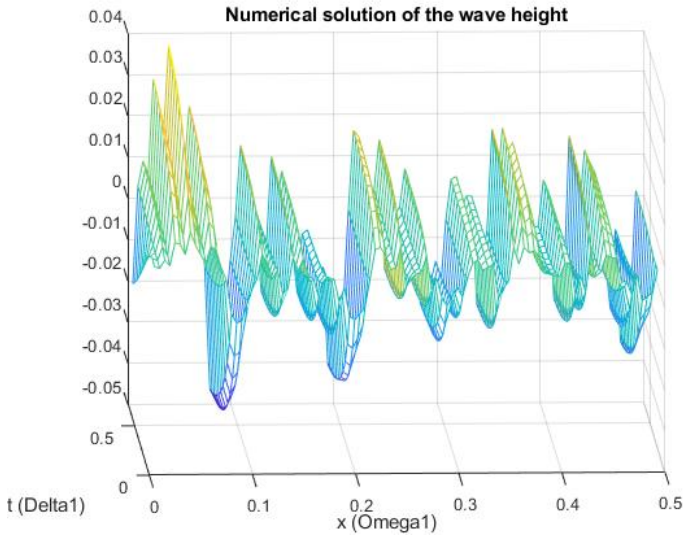
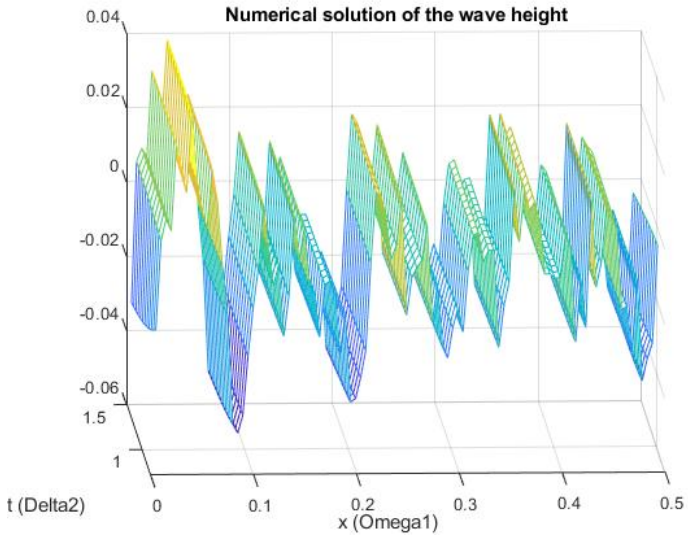


Figure 5.12: *SWEs solution* $x[1]_{10} \in \mathbb{R}^{n_x}$ *(model estimate) and DD-KF estimates* $\widehat{x}_{10}^{\Omega\times\Delta}[1] \in \mathbb{R}^{n_x}$ *(Kalman estimate 1 and Kalman estimate 2) at* t_{10} *on* Ω *by considering different errors and variances observations; i.e.* $v_{10}^1 = 1 \cdot \bar{v}_{10}^1$ *and* $\sigma_{1,0}^2 = 1.04 \times 10^0$ *in* Ω_1 *and* $v_{10}^2 = 10^{-15} \cdot \bar{v}_{10}^2$ *and* $\sigma_{2,0}^2 = 2.22 \times 10^{-1}$ *in* Ω_2. $\bar{v}_{10}^1$ *and* $\bar{v}_{10}^2$ *are random vectors drawn from standard normal distribution.*

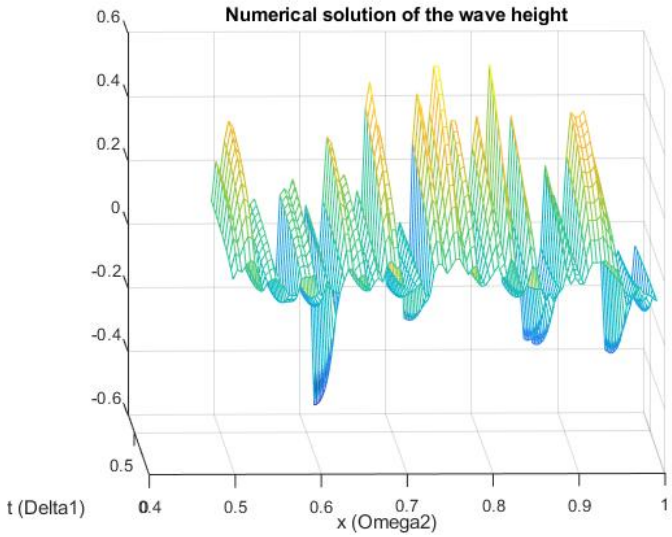


(a) Numerical solution $u[1]$ of the SWEs solution h on $\Omega_1 \times \Delta_1$.

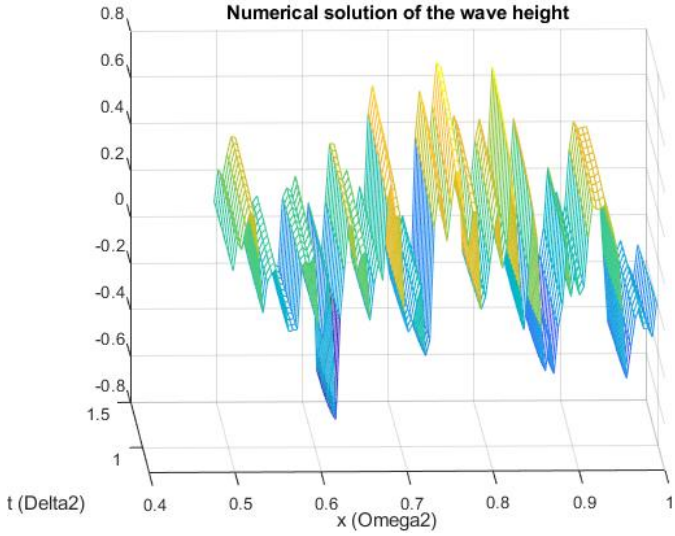


(b) Numerical solution $u[1]$ of the SWEs solution h on $\Omega_1 \times \Delta_2$.

Figure 5.13: For $i, j = 1, 2$, numerical solution $u[1]$ on $\Omega_i \times \Delta_j$.

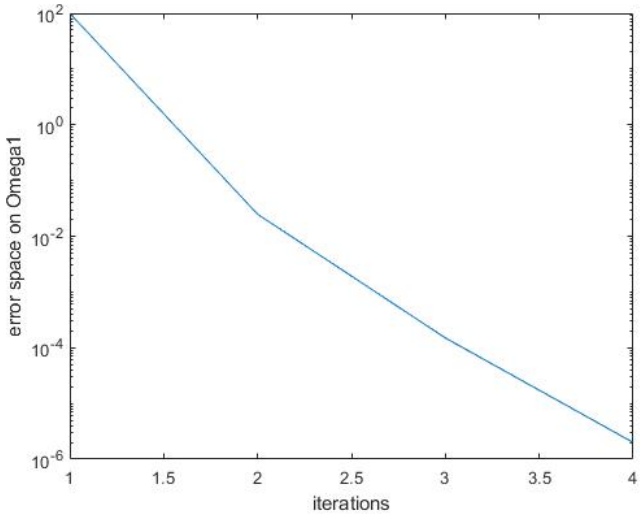


(a) Numerical solution $u[1]$ of the SWEs solution h on $\Omega_1 \times \Delta_1$.

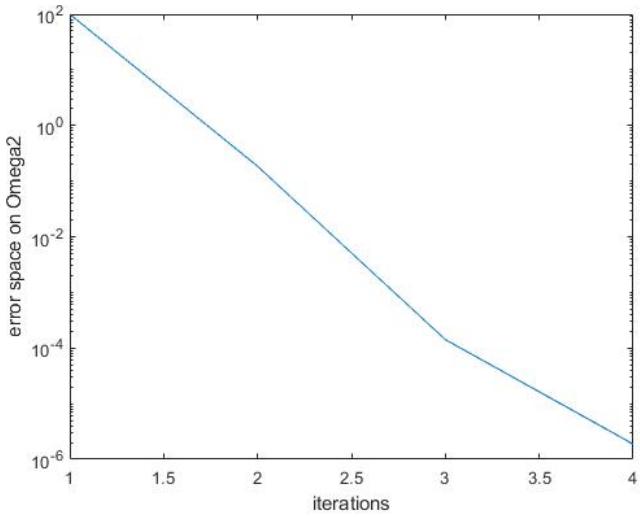


(b) Numerical solution $u[1]$ of the SWEs solution h on $\Omega_1 \times \Delta_2$.

Figure 5.14: For $i, j = 1, 2$, numerical solution $u[1]$ on $\Omega_i \times \Delta_j$.

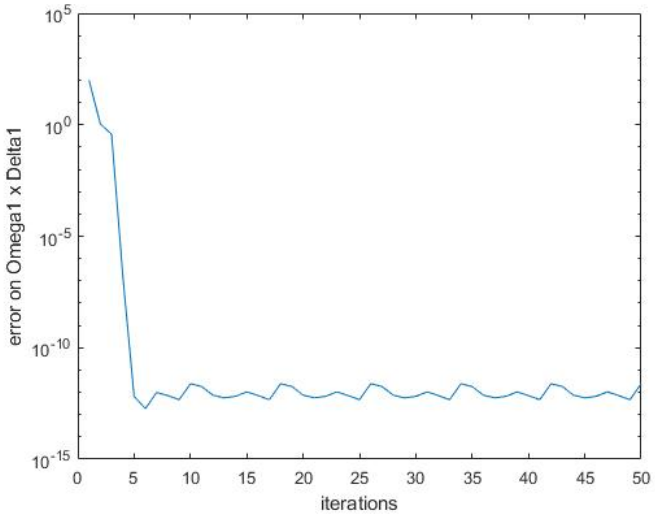


(a) Behaviour of $E^{\Omega_1 \times \Delta}$ versus iterations.

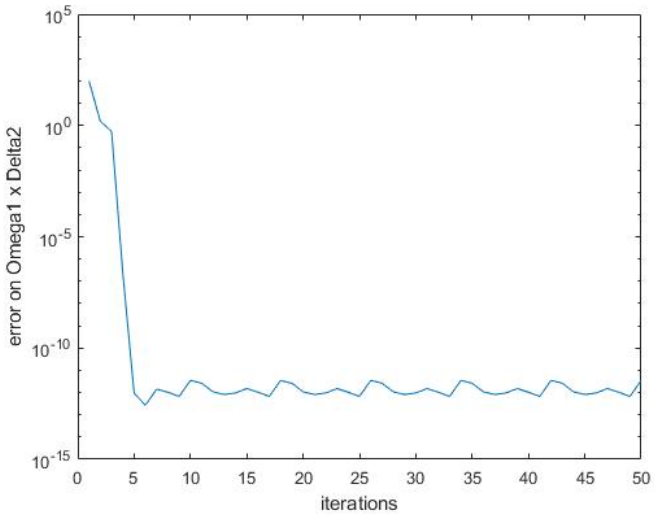


(b) Behaviour of $error^{\Omega_2 \times \Delta}$ versus iterations.

Figure 5.15: Behaviour of $E^{\Omega_1 \times \Delta}$ and $E^{\Omega_2 \times \Delta}$ versus iterations.

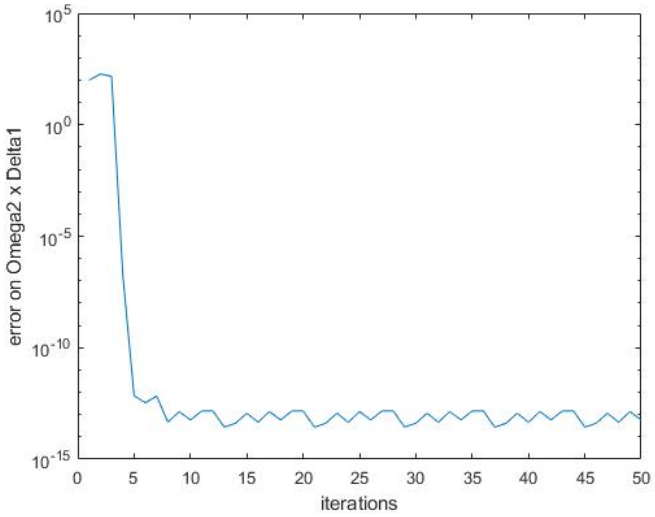


(a) Behaviour of $E^{\Omega_1 \times \Delta_1}$ versus iterations.

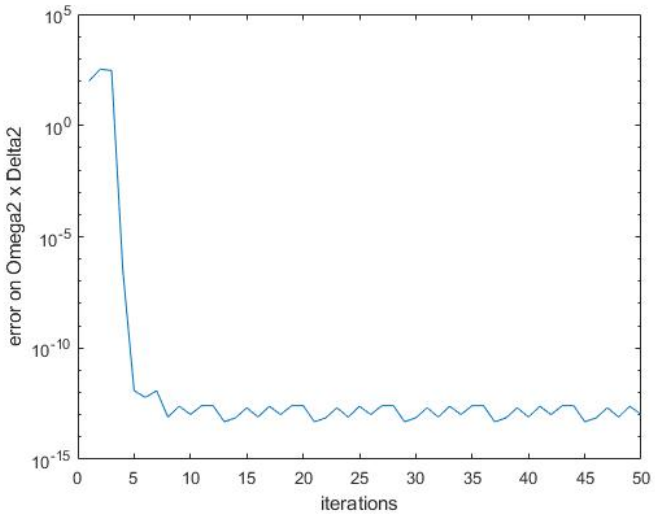


(b) Behaviour of $E^{\Omega_1 \times \Delta_2}$ versus iterations.

Figure 5.16: Behaviour of $E^{\Omega_1 \times \Delta_1}$ and $E^{\Omega_1 \times \Delta_2}$ versus iterations.

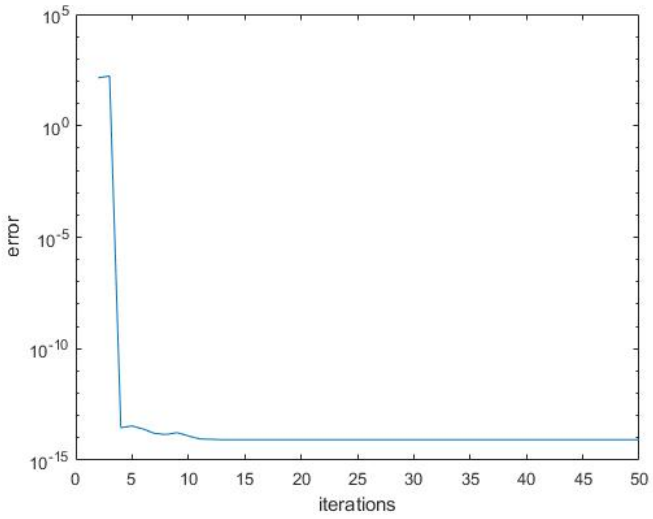


(a) Behaviour of $E^{\Omega_2 \times \Delta_1}$ versus iterations.

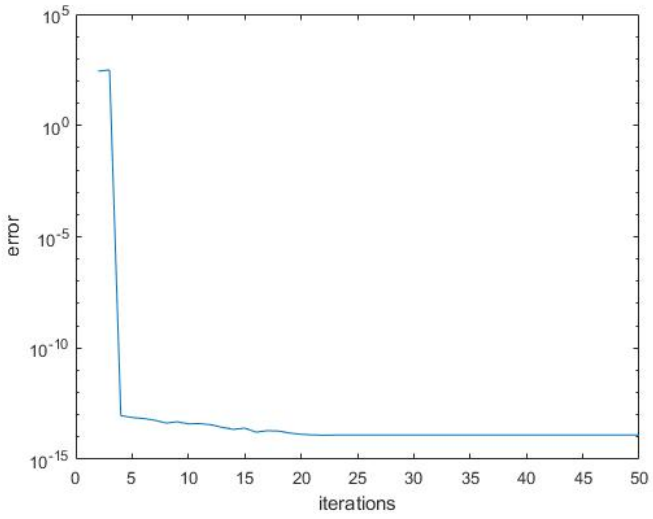


Behaviour of $E^{\Omega_2 \times \Delta_2}$ versus iterations.

Figure 5.17: (b) Behaviour of $E^{\Omega_2 \times \Delta_1}$ and $E^{\Omega_2 \times \Delta_2}$ versus iterations.



(a) Behaviour of E_1^n versus iterations.



(b) Behaviour of E_2^n versus iterations.

Figure 5.18: Behaviour of E_1^n and E_2^n versus iterations.

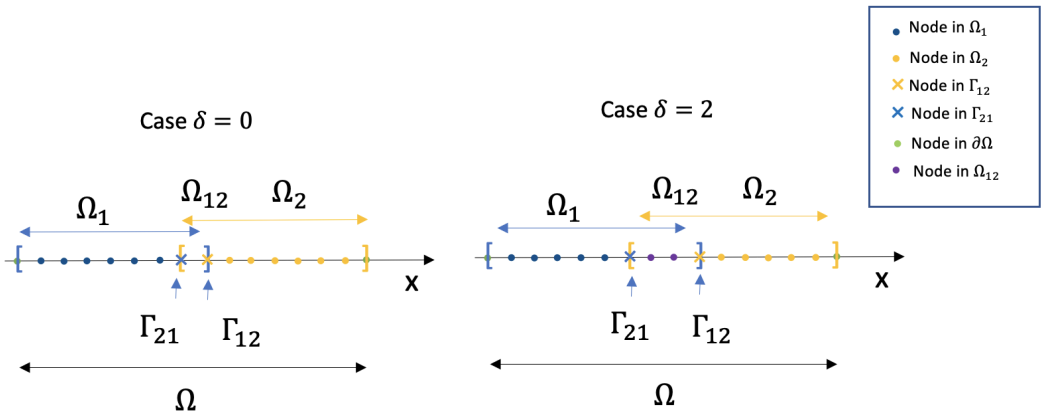


Figure 5.19: Decomposition of spatial domain $\Omega \subset \mathbb{R}$ in two subdomains $\{\Omega_i\}_{i=1,2}$ by identifying overlap region Ω_{12} defined in (3.3) and interfaces Γ_{12} and Γ_{21} defined in (3.4). On the left case $\delta = 0$ i.e. no inner nodes in Ω_{12}, on the right case $\delta = 2$ i.e. two inner nodes in overlap region Ω_{12}.

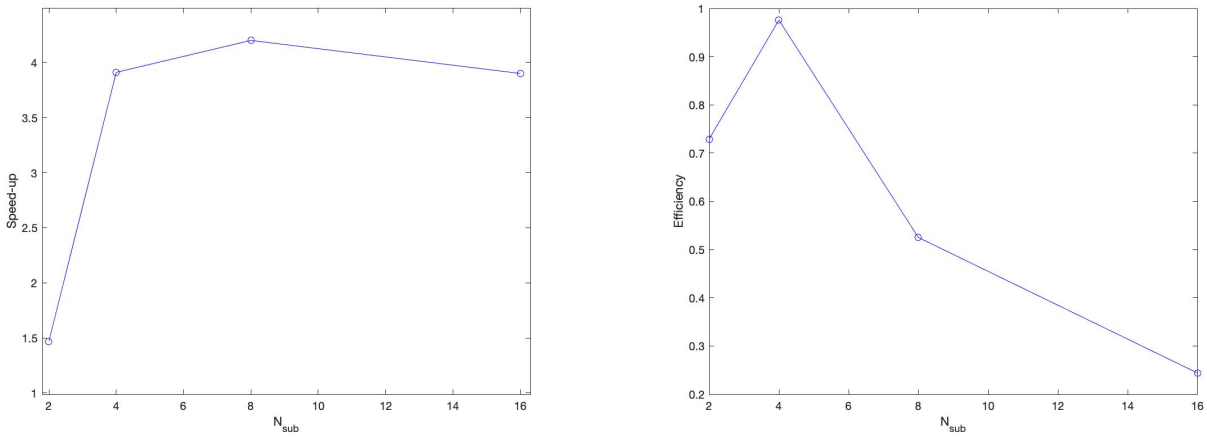


Figure 5.20: Performance results at $N_p = 640$ for $N_{sub} = 2, 4, 8, 16$. The values of speed up (left) and efficiency (right) are reported in Table 5.8. For $N_{sub} = 16$, speed up and efficiency strongly decrease.

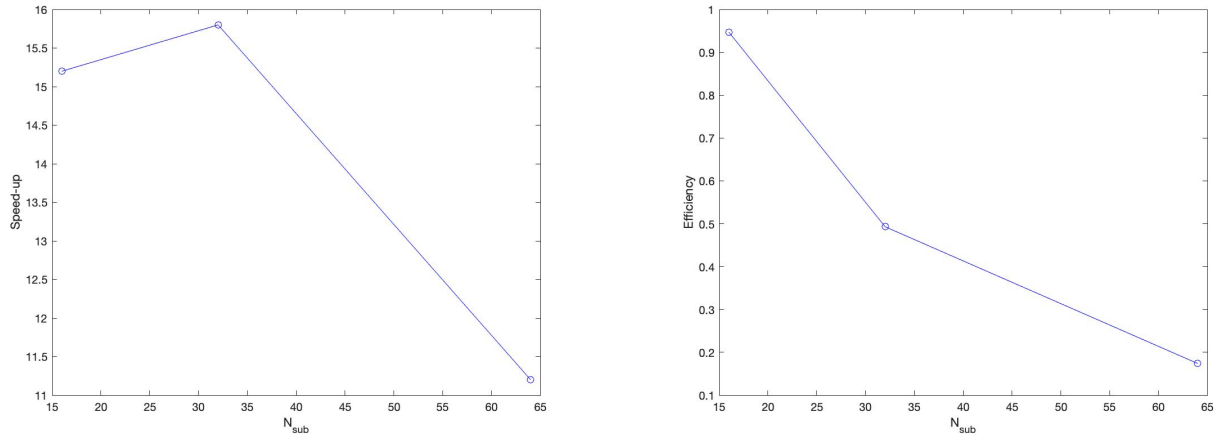


Figure 5.21: Performance results at $N_p = 832$ for $N_{sub} = 16, 32, 64$. The values of speed up (left) and efficiency (right) are reported in Table 5.9. For $N_{sub} = 64$, speed up and efficiency strongly decrease.

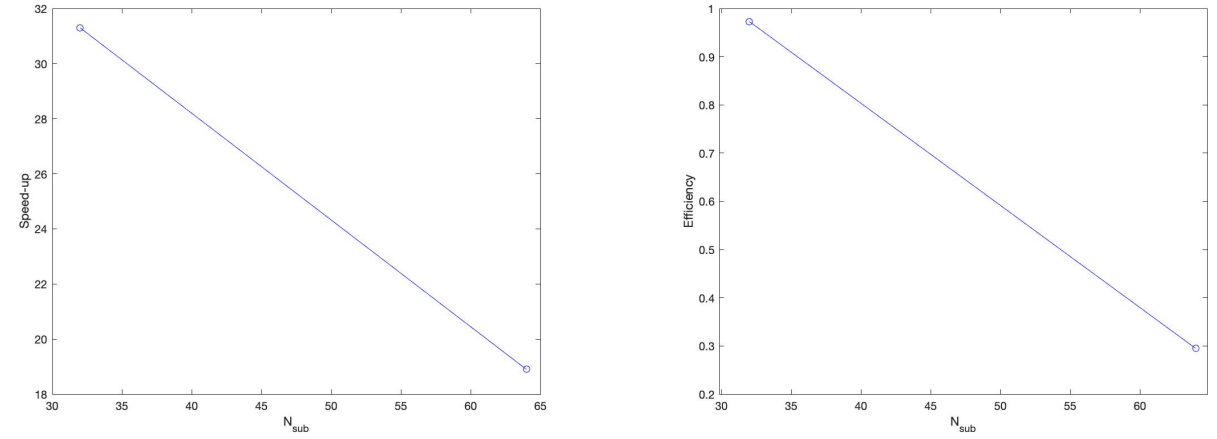


Figure 5.22: Performance results at $N_p = 896$ for $N_{sub} = 32, 64$. The values of speed up (left) and efficiency (right) are reported in Table 5.10. For $N_{sub} = 64$, speed up and efficiency strongly decrease.

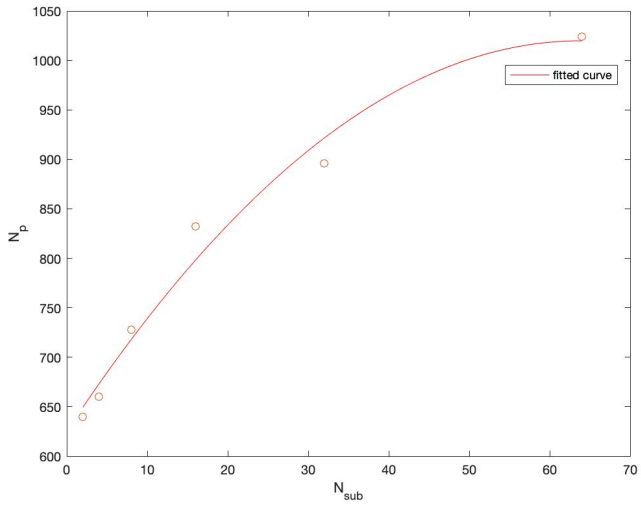


Figure 5.23: DD–4DVAR isoefficiency: curve best fit data (N_{sub}, N_p) reported in Table 5.12.

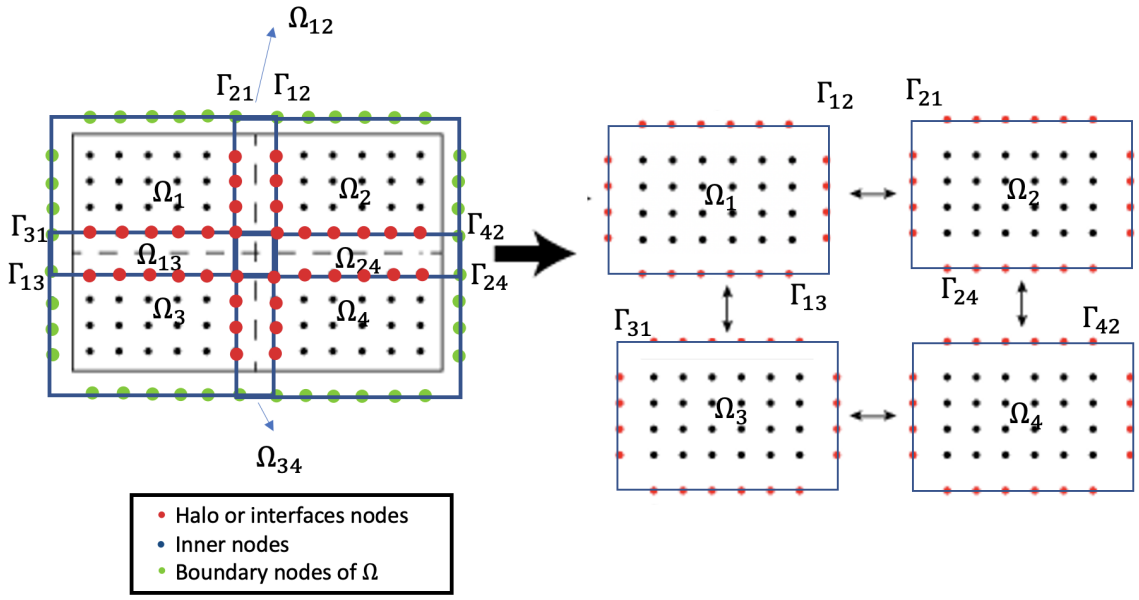


Figure 5.24: Decomposition of a spatial domain $\Omega \subset \mathbb{R}^2$ in 4 subdomains $\{\Omega_i\}_{i=1,2,3,4}$ by identifying for $i = 1, 2, 3, 4$ overlap regions Ω_{ij} defined in (3.3) and interfaces Γ_{ij} defined in (3.4) $\forall j \in J_i := \{s \in \{1, \ldots, N_{sub}\} \ : \ s \neq i\}$

.

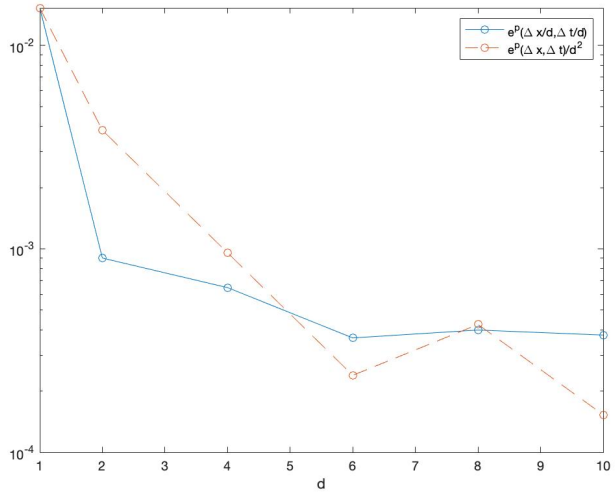


Figure 5.25: Plot of values of $e^p\left(\frac{\Delta x}{d}, \frac{\Delta t}{d}\right)$ (orange dashed line) and $\frac{e^p(\Delta x,\Delta t)}{d^2}$ (blue full line) for $d = 1, 2, 4, 6, 8, 10$ reported in Table 1.1.

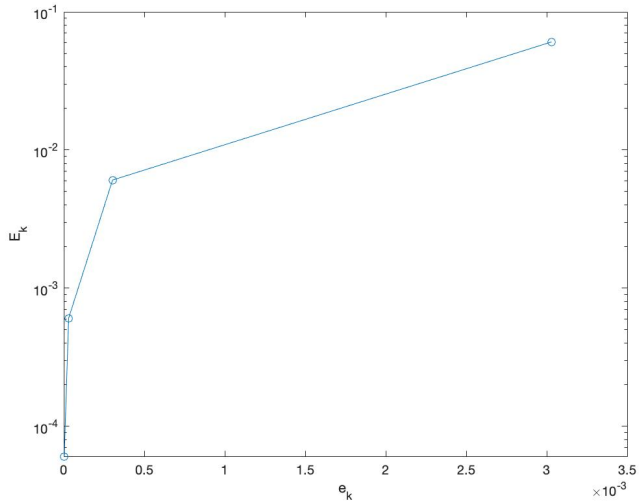


Figure 5.26: Plot of values $(\bar{e}_k, \bar{E}_k)$ reported in Table 1.2.

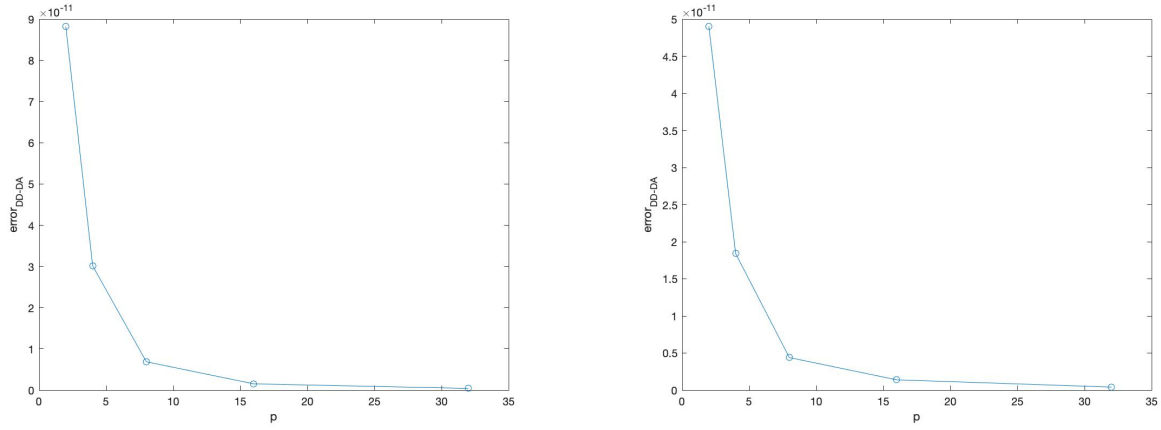


Figure 5.27: Examples 3 (left)- 4 (right). We report values of $error_{DD-DA}$ versus p.

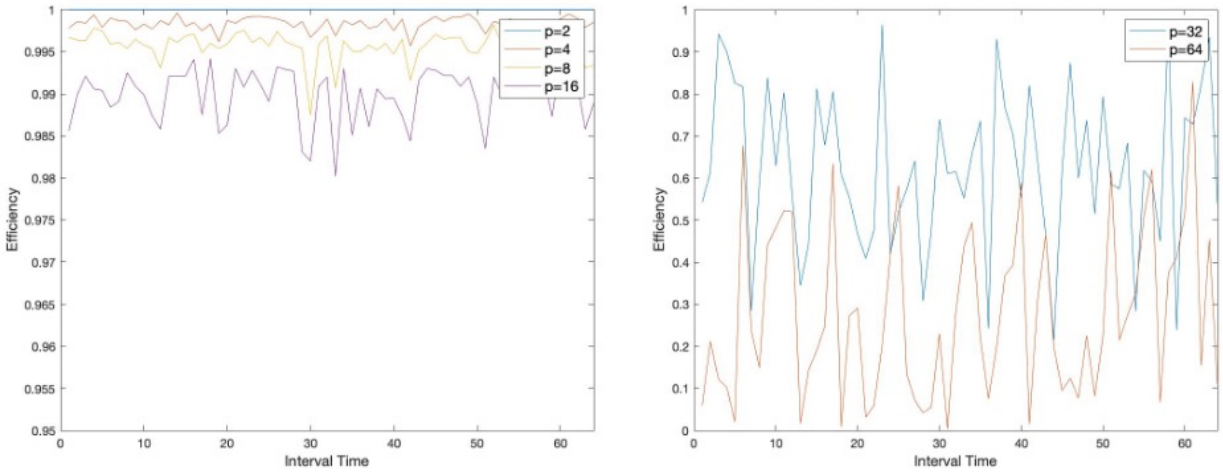


Figure 5.28: Example 2. For $k = 1, \ldots, 64$, value of parameter $\mathcal{E}^k \in \Delta_k$.

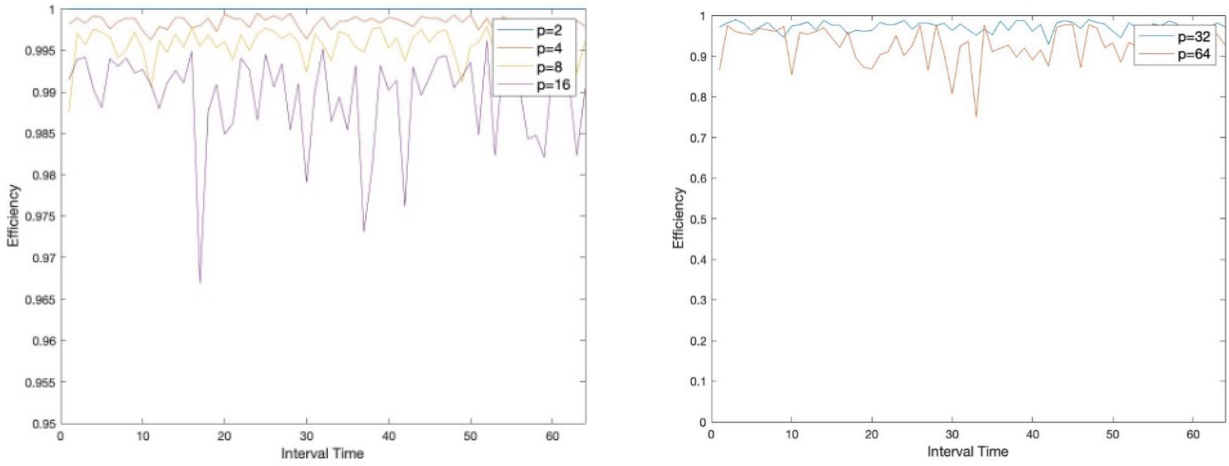


Figure 5.29: Example 3. For $k = 1, \ldots, 64$, value of parameter $\mathcal{E}^k \in \Delta_k$.

Chapter 6

Conclusions

The present work is placed in the context of a research activity devoted to the development of scalable algorithms for using Data Assimilation in large scale applications [9, 10, 32, 30]. Main focus is the mathematical framework for using a DD-based approach for KF, 3DVAR and 4DVAR methods that are computationally efficient. DD approach is based on decomposition along both spatial and temporal directions, a space-time partitioning of the PDE–based model and of the DA functional. DA acts as predictor propagator for the local PDE–based model, providing the approximations needed to the PinT method for solving the initial value problem on each subinterval concurrently. Leveraging Schwarz and PinT methods consistency constraints for PDEs-based models, the framework iteratively adjust local solutions by adding the contribution of adjacent subdomains to the local filter, along overlapping regions. Furthermore, in terms of time complexity reduction it results that applying Schwarz method the performance gain of DD–DA algorithm scales as the number of subdomains squared. As a consequence this approach increases the accuracy of local solutions and it allows to apply in parallel both the fine and coarse solvers, increasing the efficiency of the resulting algorithm. The key point of the present work is to prove the necessary results that underpin this framework by considering, let us say, a first-level decomposition. Nevertheless, such configuration should be considered as

a part of a multilevel DD scheme designed according to the features of the application and of the computing environment. An interesting advantage of this approach is that it can be potentially applied in a moderately non-intrusive manner to existing codes in navigation, computer graphics, robotics, arising from the so-called *discretize-then-optimize* approach. As DD configuration may depend both on the particular state estimation application (in terms of different data distribution) and on the mapping on the available parallel computing environment (in terms of different computing power), we employed a dynamic and adaptive DD configuration which could be used in concrete scenarios. In particular, we focused on the introduction of a dynamic redefining of initial DD in order to deal with problems where the observations are non uniformly distributed and general sparse. We called them DyDD and DyDDST. Results confirm that the accuracy of local solutions of the forecast model and hence of local KF and 4DVAR estimates, are not impaired by DD approach. As a consequence this approach increases the accuracy of local solutions and it allows to apply in parallel both the fine and coarse solvers, increasing the efficiency of the resulting algorithm. We derived and discussed main features of DD–DA framework using shallow water equations which are commonly used for monitoring and forecasting the water flow in rivers and open channels and constrained least square model as a reference state estimation problem. In particular, DD–DA methods coupled with DyDD or DyDDST can be properly modified to be adapted for solving real-world problems in oceanography. In discretization phase, DD-DA methods have to deal with the fact that the planet Earth has emerged land areas, this means that some discretization nodes are located over the land. Hence, it could be applied a pre-processing phase to identify land and sea nodes and remove only-land subdomains. Then, applying a DyDD or DyDDST related to sea nodes, we could get a balanced DD. Nevertheless, this framework has application on the plentiful literature of PDE-based state estimation real-world problems. This work makes possible numerous extensions. Among them, it makes it possible to apply deep-learning techniques to develop consistency constraints which will ensure that the solutions are physically meaningful even at the boundary

of the small domains in the output of the local models [59, 59, 88, 21].

Appendix A

Appendix

A.1 Constrained Least Squares (CLS) Problem

Let

$$H_0 x_0 = y_0, \quad H_0 \in \mathbb{R}^{m_0 \times N_p}, \quad y_0 \in \mathbb{R}^{m_0}, \quad x_0 \in \mathbb{R}^{N_p} \tag{A.1}$$

be an overdetermined linear system (the state), where $rank(H_0) = N_p > 0$, $m_0 > N_p$.
Given $H_1 \in \mathbb{R}^{m_1 \times N_p}$, $y_1 \in \mathbb{R}^{m_1}$ (the observations), $x_1 \in \mathbb{R}^{N_p}$, $x \in \mathbb{R}^{N_p}$, we consider the system which couples state and observations equations

$$S: \quad Ax = b \tag{A.2}$$

where

$$A = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} \in \mathbb{R}^{(m_0+m_1) \times N_p}, \quad b = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} \in \mathbb{R}^{m_0+m_1}, \tag{A.3}$$

and $m_1 > 0$. Let $R_0 \in \mathbb{R}^{m_0 \times m_0}$, $R_1 \in \mathbb{R}^{m_1 \times m_1}$ be weight matrices and $R = diag(R_0, R_1) \in \mathbb{R}^{(m_0+m_1)\times(m_0+m_1)}$.

CLS problem consists in the computation of $\widehat{x}$ such that:

$$CLS: \quad \widehat{x} = argmin_{x \in \mathbb{R}^{N_p}} J(x) \tag{A.4}$$

with

$$J(x) = ||Ax - b||_R^2 = ||H_0 x - y_0||_{R_0}^2 + ||H_1 x - y_1||_{R_1}^2, \tag{A.5}$$

where $\widehat{x}$ is hence given by

$$(A^T R A)\widehat{x} = A^T R b \Rightarrow \widehat{x} = (A^T R A)^{-1} A^T R b \tag{A.6}$$

or,

$$\widehat{x} = (H_0^T R_0 H_0 + H_1^T R_1 H_1)^{-1}(H_0^T R_0 y_0 + H_1^T R_1 y_1). \tag{A.7}$$

We refer to $\widehat{x}$ as the least squares solution of system in (A.2).

In particular,

$$\widehat{x}_0 = (H_0^T R_0 H_0)^{-1} H_0^T R_0 y_0, \tag{A.8}$$

is the least squares solution of system in (A.1)

A.2 Shallow Water Equations (SWEs) set up

Neglecting the Coriolis force and frictional forces and assuming unit width, SWEs are:

$$\begin{cases} \frac{\partial h}{\partial t} + \frac{\partial vh}{\partial x} = 0 \\ \frac{\partial vh}{\partial t} + \frac{\partial (v^2 h + \frac{1}{2} g h^2)}{\partial x} = 0 \end{cases} \tag{A.9}$$

where the independent variables x and t make up the spatial dimension and time. The dependent variables are h, which is the height with respect to the surface and v, i.e. the horizontal velocity, while g is the gravitational acceleration.

In order to write SWEs in a compact form, we introduce the vectors

$$u := \begin{bmatrix} h \\ vh \end{bmatrix} \quad f(u) := \begin{bmatrix} vh \\ v^2 h + \frac{1}{2} g h^2 \end{bmatrix}, \tag{A.10}$$

the SWEs can be rewritten as follows

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0. \tag{A.11}$$

The initial boundary problem for the SWEs is

$$\begin{cases} u(x,t+\Delta t) = \mathcal{M}_{t,t+\Delta t}(u(x,t)) \quad \forall t, t+\Delta t \in [0,1.5] \\ u(x,0) := \begin{bmatrix} h(x,0) \\ h(x,0)v(x,0) \end{bmatrix} = \begin{bmatrix} 2+\sin(2\pi x) \\ 0 \end{bmatrix} \quad \forall x \in \Omega \end{cases} \tag{A.12}$$

with the following reflective boundary conditions on v and free boundary conditions on h

$$u(0,t) := \begin{bmatrix} h(x_1,t) \\ -h(x_1,t)v(x_1,t) \end{bmatrix} \quad u(x_{n_x-1},t) := \begin{bmatrix} h(x_{n_x-2},t) \\ -h(x_{n_x-2},t)v(x_{n_x-2},t) \end{bmatrix} \quad \forall t \in \Delta \tag{A.13}$$

where

$$\mathcal{M}_{t,t+\Delta t}(u(x,t)) := u(x,t) - \int_t^{t+\Delta t} \frac{\partial f(u)}{\partial x} ds. \tag{A.14}$$

We note that $\mathcal{M}_{t,t+\Delta t}(u(x,t))$ should include Coriolis forces and also frictional forces, if they are present in the SWEs; in particular, as these quantities show off as the right hand side of (A.11), they will be also included in the right hand side of (A.14) as additive terms under the integral.

The state of the system at each time $t_{l+1} \in \Delta$, $l = 0, 1, \ldots, N-2$ is:

$$u(t_{l+1}) \equiv u_{l+1} := \begin{bmatrix} x[1]_{l+1} \\ x[2]_{l+1} \end{bmatrix} \in \mathbb{R}^{2(n_x-2)} \tag{A.15}$$

where

$$\begin{aligned} u[1]_{l+1} &= \{u[1](x_i,t_{l+1})\}_{i=1,\ldots,n_x-1} := \{h(x_i,t_{l+1})\}_{i=1,\ldots,n_x-1} \in \mathbb{R}^{(n_x-2)} \\ u[2]_{l+1} &:= \{u[2](x_i,t_{l+1})\}_{i=1,\ldots,n_x-1} := \{v(x_i,t_{l+1})h(x_i,t_{l+1})\}_{i=1,\ldots,n_x-1} \in \mathbb{R}^{(n_x-2)} \end{aligned}. \tag{A.16}$$

From Lax-Wendroff scheme [81], we obtain the following discrete formulation of the SWEs

$$u_{l+1} = M_{l,l+1}u_l + b_l + w_l \tag{A.17}$$

and 4DVAR DA problem for the SWEs is

$$\mathbf{u^{DA}} = argmin_{u \in \mathbb{R}^{N_p \times N}} ||u - u^M||^2_{\mathbf{B}^{-1}} + ||Gu - y||^2_{\mathbf{R}^{-1}} \tag{A.18}$$

where $u^M := \{u^M_{l+1}\}_{l=0,1,\dots,N-1}$ is the background, $M \equiv M_{0,N-1}$ and

$$M_{l,l+1} = \begin{bmatrix} M[1]_{l,l+1} & O_{n_x-2} \\ M[2,1]_{l,l+1} & M[2]_{l,l+1} \end{bmatrix} \in \mathbb{R}^{2(n_x-2)\times 2(n_x-2)} \tag{A.19}$$

with $O_{n_x-2} \in \mathbb{R}^{n_x-2\times n_x-2}$ the null matrix and $M[1]_{l,l+1} \in \mathbb{R}^{(n_x-2)\times(n_x-2)}$, $M[2,1]_{l,l+1} \in \mathbb{R}^{(n_x-2)\times(n_x-2)}$, $M[2]_{l,l+1} \in \mathbb{R}^{(n_x-2)\times(n_x-2)}$ the following tridiagonal matrices

$$M[1]_{l,l+1} = \begin{bmatrix} \psi^l_1 & \eta^l_2 & & & \\ -\eta^l_1 & \psi^l_2 & \eta^l_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -\eta^l_{n_x-4} & \psi^l_{n_x-3} & \eta^l_{n_x-2} \\ & & & -\eta^l_{n_x-3} & \psi^l_{n_x-2} \end{bmatrix} \tag{A.20}$$

$$M[2,1]_{l,l+1} = \begin{bmatrix} 0 & \chi^l_2 & & & \\ -\chi^l_1 & 0 & \chi^l_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -\chi^k_{n_x-4} & 0 & \chi^l_{n_x-2} \\ & & & -\chi^l_{n_x-3} & 0 \end{bmatrix} \tag{A.21}$$

$$M[2]_{l,l+1} = \begin{bmatrix} \phi^l_1 & \xi^l_2 & & & \\ -\xi^l_1 & \phi^l_2 & \xi^l_3 & & \\ & \ddots & \ddots & \ddots & \\ & & -\xi^l_{n_x-4} & \phi^l_{n_x-3} & \xi^l_{n_x-2} \\ & & & -\xi^l_{n_x-3} & \phi^l_{n_x-2} \end{bmatrix} \tag{A.22}$$

and the vector $b \in \mathbb{R}^{2(n_x-2)}$

$$b_l := \begin{bmatrix} b[1]_l \\ b[2]_l \end{bmatrix} \in \mathbb{R}^{2(n_x-2)} \tag{A.23}$$

with

$$b[1]_l := \begin{bmatrix} \eta^l_0 h^l_0 \\ 0 \\ \vdots \\ 0 \\ \eta^l_{n_x-1} h^l_{n_x-1} \end{bmatrix} \in \mathbb{R}^{n_x-2} \quad b[2]_l := \begin{bmatrix} \xi^l_0 v^l_0 h^l_0 - \chi^l_0 h^l_0 \\ 0 \\ \vdots \\ 0 \\ \xi^l_{n_x-1} h^l_{n_x-1} + \xi^l_{n_x-1} v^l_{n_x-1} h^l_{n_x-1} \end{bmatrix} \in \mathbb{R}^{n_x-2} \tag{A.24}$$

where

$$\begin{aligned} \eta_i^l &:= \tfrac{\alpha}{2}[v_i^l + \alpha((v_i^l)^2 + \tfrac{1}{2}gh_i^l)] & \psi_i^l &:= 1 - 4\alpha^2((v_i^l)^2 + \tfrac{1}{2}gh_i^l) \\ \xi_i^l &:= \tfrac{\alpha}{2}[\alpha + v_i^l] & \phi_i^l &:= (1 + 4\alpha^2) \end{aligned}$$

$$\chi_i^l := \frac{1}{2}gh_i^l$$

with $\alpha := \frac{\Delta t}{2\Delta x}$ and $h_i^l := h(t_l, x_i)$, $v_i^l := v(t_l, x_i)$.

While, $G \in \mathbb{R}^{N \cdot n_{obs} \times n_x}$ is a matrix interpolating the background fields to the observation locations and transforming the model variables to observed quantities, and $y \in \mathbb{R}^{N \times n_{obs}}$ is observations in $\Omega \times \Delta$.

We note that the discrete model in (A.17) can be rewritten as follows

$$\begin{aligned} x[1]_{l+1} &= M[1]_{l,l+1}x[1]_l + b[1]_l \\ x[2]_{l+1} &= M[2]_{l,l+1}x[2]_l + \tilde{b}[2]_l \end{aligned} \tag{A.25}$$

with

$$\tilde{b}[2]_l := b[2] - M[2,1]_{l,l+1}x[1]_l. \tag{A.26}$$

In particular, if we define

$$S := \max\left(|v - \sqrt{gh}|, |v + \sqrt{gh}|\right), \tag{A.27}$$

the stability condition of Lax-Wendroff is

$$S \cdot \frac{\Delta t}{\Delta x} \leq 1, \tag{A.28}$$

for the condition (A.28) to be satisfied, at each iteration $k = 0, 1, \ldots, nt$ we choose: $\Delta t = 0.8 \cdot \frac{\Delta x}{S}$.

A.3 Regional Ocean Modeling System (ROMS)

Regional Ocean Modeling System (ROMS) is an open-source, mature numerical framework used by both the scientific and operational communities to study ocean dynamics over 3D spatial domain and time interval.

ROMS supports different 4DVAR DA methodologies IS4D-Var and RBL4DVAR search best circulation estimate in space spanned by control vector and observations, respectively. IS4D-Var and RBL4DVAR algorithm consist of two nested loop, the outer-loop involves the module (1), namely nonlinear ROMS (NLROM) solving ROMS equations, the inner-loop involves modules (2)-(3), namely tangent linear approximation of ROMS (TLROMS) and adjoint model of ROMS (ADROMS); TLROMS and ADROMS are used for minimizing 4DVAR functional [98] (see Figures A.1-A.2).

NLROMS is a three-dimensional, free-surface, terrain-following ocean model that solves the Reynolds-averaged Navier-Stokes equations using the hydrostatic vertical momentum balance and Boussinesq approximation.

NLROMS computes

$$x^{ROMS}(t_l) = M_{l-1,l}(x(t_{l-1}), f(t_l), b(t_l)) \tag{A.29}$$

with the state-vector $x^{ROMS}(t_l) = (T, S, \varsigma, u, v)^T$, temperature T, salinity S, (x, y) components of vector velocity u, v, sea surface displacement ς. $M_{l-1,l}$ represents nonlinear ROMS acting on $x^{ROMS}(t_{l-1})$, and subject to forcing $f(t_l)$, and boundary conditions $b(t_l)$ during the time interval $[t_{l-1}, t_l]$.

Minimization of the 4DVAR functional:

$$\mathbf{J}^{ROMS}(\delta z) = \frac{1}{2}\delta z \mathbf{B}^{-1}\delta z + \frac{1}{2}(G\delta z - \mathbf{d})^T \mathbf{R}^{-1}(G\delta z - \mathbf{d}) \tag{A.30}$$

where δz are the control variable increments, $\mathbf{d}$ is vector of innovations, $G = (..., H_l^T, ...)^T$, where H_l is the observation matrix; $\mathbf{R}$ is observation error covariance matrix and $\mathbf{B}$ is covariance matrix of model error, is computed in the inner-loop in Figure A.1.

Analysis increment, δz^a, that minimizes 4DVAR function in (A.30) corresponds to the solution of the equation $\partial \mathbf{J}^{ROMS}/\partial \delta z = 0$, and is given by:

$$\delta z^a = (\mathbf{B}^{-1} + G^T \mathbf{R}^{-1} G)^{-1} G^T \mathbf{R}^{-1} \mathbf{d} \tag{A.31}$$

or, equivalently

$$\delta z^a = \mathbf{B}G^T(G\mathbf{B}G^T + \mathbf{R})^{-1}\mathbf{d}. \tag{A.32}$$

Equation (A.31) is referred to as the dual form (RBL4DVAR), while (A.32) is referred to as the primal form (IS4DVAR). In particular, we define

$$\mathbf{K} = \mathbf{B}G^T(G\mathbf{B}G^T + \mathbf{R})^{-1} \tag{A.33}$$

as Kalman gain matrix.

TLROMS computes

$$\delta x^{ROMS}(t_l) \simeq M_{l-1,l}u(t_{l-1}) \tag{A.34}$$

where $\delta x^{ROMS}(t_l) = x^{ROMS}(t_l) - x^b(t_l)$, $\delta f(t_l) = f(t_l) - f^b(t_l)$, $\delta b(t_l) = b(t_l) - b^b(t_l)$, and $x^b(t_l)$, $f^b(t_)$, $b^b(t_l)$ are the background of the circulation, surface forcing and open boundary conditions respectively, and

$$u(t_{l-1}) = ((\delta x^{ROMS})^T(t_{l-1}), \delta f^T(t_l), \delta b^T(t_l))^T.$$

Equation (A.34) is obtained from first-order Taylor expansion of NLROMS in (A.29).

ADROMS computes

$$u^*(t_{l-1}) = M^T_{l-1,l}p(t_l) \tag{A.35}$$

where $u^*(t_{l-1}) = (p^T(t_{l-1}), \delta f^T(t_l), \delta b^{*T}(t_l))^T$ where p is the adjoint state-vector, δf^T and δb^{*T} are the adjoint of the surface forcing and the open boundary condition increments.

A.3.1 DD–4DVAR DA in ROMS model

The DD method proposed in Section 3.3 is made up of decomposition of the domain of computation $\Omega \times \Delta$ into subdomains where Ω is the 3D spatial domain and Δ is the time interval,

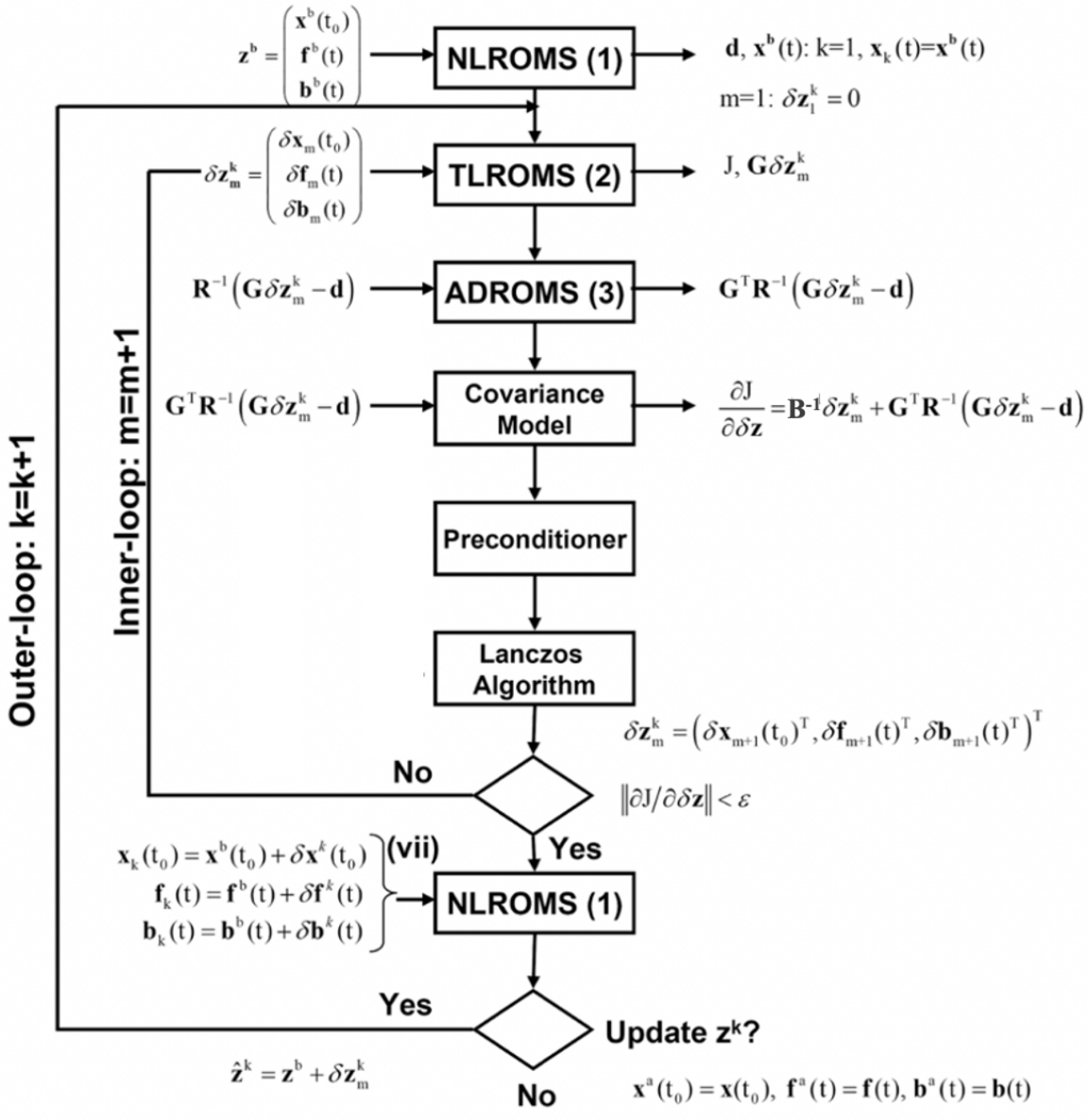


Figure A.1: A flow chart illustrating IS4D-Var algorithm.

solution of reduced forecast model and minimization of local 4DVAR functionals. Relying on the existing software implementation, in the next we describe the main components of DD–4DVAR method, highlighting the topics that we will address both on the mathematical problem

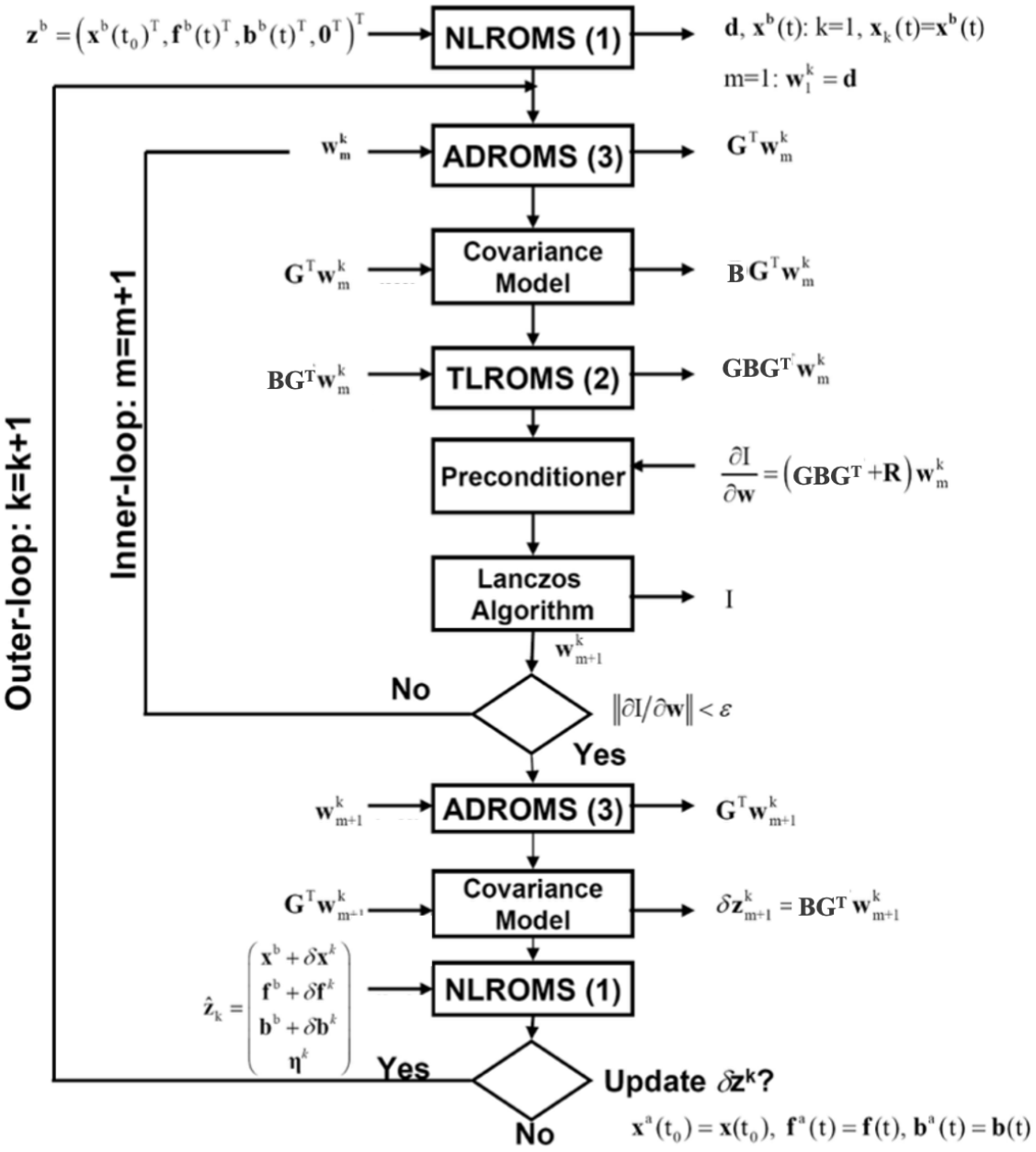


Figure A.2: A flow chart illustrating RBL4DVAR algorithm.

underlying ROMS and the code implementation.

We focus on IS4DVAR formulation described in Section A.3.

In the following, we introduce the decomposition in space and time of the ocean model.

- Decomposition of spatial domain Ω.

  We will consider a 2D decomposition of $\Omega \subseteq \mathbb{R}^3$ in x- and y-direction and denote Ω_{xy} the spatial domain to decompose.

  ROMS uses a parallelization[1] approach that partitions domain Ω_{xy} into tiles (see Figure A.3).

$$\Omega_{xy} = \bigcup_{i=0}^{N_{sub}-1} tile_i \tag{A.36}$$

  where $N_{sub} = NtileI \times NtileJ$; $NtileI$ and $NtileJ$ are the number of tiles set in the input file in x- and y-direction, respectively.

  We denote by HI and HJ the overlapping tiles regions (i.e ghost or halo area in ROMS[2]) in x- and y-direction, respectively.

  Step 0: DD of Ω in ROMS.

  In our study we will assume the decomposition available in ROMS, as given in (A.36).

- Decomposition of time interval Δ.

  ROMS does not implement a decomposition in time direction.

[1] https://www.myroms.org/wiki/Parallelization

[2] In ROMS, the halo area would be two grids points wide unless the MPDATA advection scheme is used, in which case it needs three.

Step 1: DD of Δ in ROMS.

In our study we will introduce a decomposition of time interval Δ into N_t intervals:

$$\Delta = \bigcup_{k=1}^{N_t} \Delta_k := \bigcup_{k=1}^{N_t} [t_{\bar{s}_{k-1}}, t_{\bar{s}_{k-1}+N_k}], \tag{A.37}$$

where $N_k = |D(\Delta_k)|$ are respectively the number of subdomains of $[0, T]$ and of time $t_l \in \Delta_k$ such that $\sum_{k=1}^{N_t} N_k - (N_t - 1) = N$, $\bar{s}_{k-1} := \sum_{j=1}^{k-1} N_j - (k-1)$ and $\bar{s}_0 := 0$.

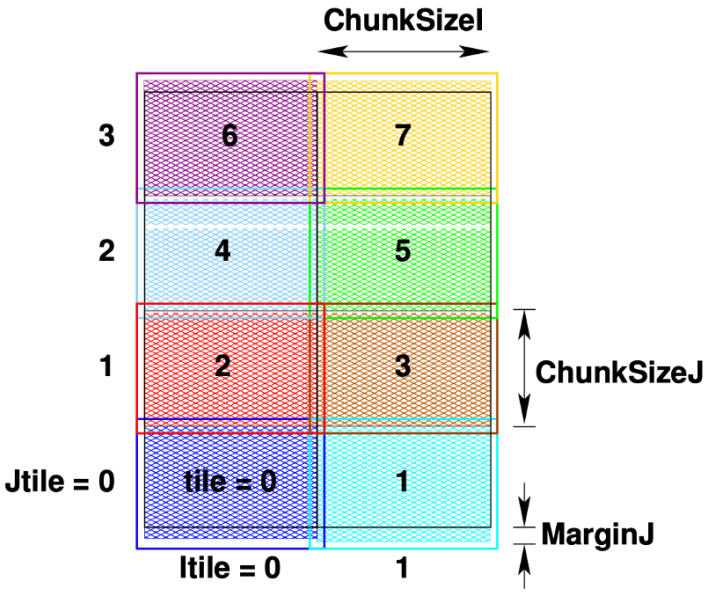


Figure A.3: A tiled grid in xy-plane with some internal ROMS parameters.

- Ocean model reduction.

  ROMS allows each tile (or subdomain, see Figure A.3) to compute local solutions of TL-ROMS and ADROMS.

For $i = 0, 1, \ldots, N_{sub} - 1$ and $k = 1, \ldots, N_t$ local[3] TLROMS on local domain $tile_i$ computes

$$\delta x_i^{ROMS}(t_l) \simeq M_{i,(l-1,l)} u_i(t_{l-1}) \tag{A.38}$$

and local ADROMS on local domain $tile_i$ computes

$$u_i^*(t_{l-1}) = M_{i,(l-1,l)}^T p_i(t_l), \tag{A.39}$$

where x_i, u_i, u_i^*, M_i, p_i are the restriction on $tile_i$ of variables x, u, u^*, M and p and $M_{i,(l-1,l)}$ is discrete model from t_{l-1} to t_l.

DD–4DVAR method introduces the model reduction by using the background x^b as local initial values. For $n = 0, 1, \ldots, \bar{n}$ (outer loop of DD–4DVAR method [33]) do: for $k = 1, \ldots, N_t$, posed $x_{i,k}^0 \equiv x_{i,k}^b \; \forall i = 0, 1, \ldots, N_{sub} - 1$, we let $x_{i,k}^{M_{i,k}}$ be the solution of the local model

$$(P_{i,k}^{M_{i,k},n})_{i=0,1\ldots,N_{sub}-1,r=1,\ldots,N_t} \; : \; \begin{cases} x_{i,k}^{\mathcal{M}_{i,k},n} = M_{i,k} x_{i,k-1}^n + b_{i,k}, \\ x_{i,k-1}^n = x_{i,k}^{\mathcal{M}_{i,k},n}, \\ x_{i,k-1}^n / HI = x_{i_I,k-1}^n / HI, \;\; (A.40.1) \\ x_{i,k-1}^n / HJ = x_{i_J,k-1}^n / HJ, \;\; (A.40.2) \end{cases} \tag{A.40}$$

where $i_I = 0, \ldots, n_I - 1$, $i_J = 0, \ldots, n_J - 1$, n_I and n_J are respectively numbers of adjacent tiles in x- and y-direction, $b_{i,r}^k$ and M_i^r are respectively the background on $tile_i \times \Delta_r$, the vector accounting boundary conditions of $tile_i$ and the restriction in $tile_i$ of the matrix in (A.29) that is

$$M_k \equiv M_{\bar{s}_{r-1},\bar{s}_r} := M_{\bar{s}_{r-1},\bar{s}_{r-1}+1} \cdots M_{\bar{s}_r-1,\bar{s}_r}.$$

[3]Let $x \in \mathbb{R}^{N_p}$ and $y \in \mathbb{R}^{N_p \times N}$ be vectors, for simplicity of notations, we refer to x_i as a restriction of x to Ω_i, i.e. $x_i \equiv x/\Omega_i$ and $x_{i,k} \equiv x/(\Omega_i \times \Delta_k)$, similarly for matrix $A \in \mathbb{R}^{N_p \times N_p}$, i.e. $A_i \equiv A/\Omega_i$ and $A_{i,k} \equiv A/(\Omega_i \times \Delta_k)$, according the description in [8].

In the following, we neglect the dependency on outer loop iteration n of DD–4DVAR method. We underline that local TLROMS and ADROMS in (A.34) and (A.35) are obtained with MPI exchange for boundary conditions, regardless of local solution on overlap area, namely they not consider overlapping tiles conditions in (A.40.1) and (A.40.2). Consequently, we need to modify local TLROMS and ADROMS in (A.38) and (A.39) taking into account of (A.40.1) and (A.40.1) conditions. More precisely,

Step 2: TLROMS.

for $k = 1, \ldots, N_t$, $\forall i = 0, 1, \ldots, N_{sub} - 1$, local TLROMS on $tile_i \times \Delta_k$ will be modified such that

$$\delta x_{i,k} \simeq M_{i,k} u_{i,k-1} + \theta_I(u_{i,k-1}) + \theta_J(u_{i,k-1}) \tag{A.41}$$

where

$$\begin{aligned} \theta_J(u_{i,k-1}) := \ & \textstyle\sum_{i_J=1}^{n_J} \gamma_{i_J} (M_{i,k}/HJ \cdot u_{i,k-1}/HJ - \\ & M_k/HJ \cdot u_{i,k-1}/HJ) \end{aligned} \tag{A.42}$$

and

$$\begin{aligned} \theta_I(u_{i,r-1}) := \ & \textstyle\sum_{i_I=1}^{n_I} \gamma_{i_I} (M_i^r/HI \cdot u_{i,r-1}/HI - \\ & M_i^r/HI \cdot u_{i,r-1}/HI) \end{aligned} \tag{A.43}$$

are overlapping vectors in x- and y-direction, respectively; where parameters γ_{i_I}, γ_{i_J} denote weights.

Similarly, we need to modify local ADROMS in (A.39). More precisely,

Step 3: ADROMS.

for $k = 1, \ldots, N_t$, $\forall i = 0, 1, \ldots, N_{sub} - 1$, local ADROMS in (A.39) on local domain $tile_i \times \Delta_k$ will be modified such that

$$u_{i,k-1} = (M_{i,r})^T \cdot p_{i,k} + \theta_J(p_{i,k-1}) + \theta_I(p_{i,k-1}) \tag{A.44}$$

where θ_J and θ_I are defined in (A.42) and (A.43).

- 4DVAR DA Operator Reduction. Operator reduction involves the inner loop in Figure A.1.

  ROMS computes and minimizes the operator

$$\mathbf{J}^{ROMS}_{tile_i} := \mathbf{J}^{ROMS}/tile_i \tag{A.45}$$

where $\mathbf{J}^{ROMS}$ is defined in (A.30) and $\mathbf{J}^{ROMS}/(tile_i)$ is the IS4D-Var functional in each tile $tile_i$ with MPI exchange of boundary conditions, $\forall i = 0, 1, ..., N_{sub} - 1$.

Local 4D–VAR DA functional in [33] is defined as follows

$$\mathbf{J}^{DD-4DVAR}_{tile_i \times \Delta_r} := \mathbf{J}_{i,k}(x_{i,k}) = \mathbf{J}(x_{i,k})/(tile_i \times \Delta_k) + \mathcal{O}_{IJ}(x_{i,k}) \tag{A.46}$$

where

$$\mathbf{J}(x) = \alpha \|x - x^b\|^2_{\mathbf{B}^{-1}} + \|Gx - y\|^2_{\mathbf{R}^{-1}}, \tag{A.47}$$

is the DD–4DVAR functional in [33],

$$\mathcal{O}_{IJ}(x_{i,k}) = \sum_{i_I=1}^{n_I} \beta_{i_I,k} \cdot \|x_{i,k}/HI - x_{i_I,k}/HI\|^2_{\mathbf{B}^{-1}_{i_I}} + \sum_{i_J=1}^{n_J} \beta_{i_J,k} \cdot \|x_{i,k}/HJ - x_{i_J,k}/HJ\|^2_{\mathbf{B}^{-1}_{i_j}} \tag{A.48}$$

is the overlapping operator on overlapping tiles region HI and HJ, and

$$\mathbf{J}_{i,k}(x_{i,k})/(tile_i \times \Delta_k) = \alpha_{i,k} \cdot \|x_{i,k} - x^{M_{i,k}}_{i,k}\|_{\mathbf{B}^{-1}_i} + \|G_{i,r}x_{i,k} - y_{i,k}\|^2_{\mathbf{R}^{-1}_i} \tag{A.49}$$

is the restriction of $\mathbf{J}$ on $tile_i \times \Delta_k$, where x^b is background, y is observations vector in Δ, $y_{i,k}$ is observations vector in Δ_k; $\mathbf{B}_i$, $\mathbf{B}_{i_I}$, $\mathbf{B}_{i_J}$ are respectively the restrictions of covariance matrix $\mathbf{B}$ to $tile_i$, HI and HJ; $G_{i,r}$, $\mathbf{R}_i$ are the restriction of matrices G and $\mathbf{R}$ to $tile_i$ in Δ_k. Parameters $\alpha_{i,r}$, β_{i_I} and β_{i_J} in (A.48) denotes the regularization parameters. We let $\alpha_{i,k} = \beta_{i_I,k} = \beta_{i_J,k} = 1, \forall i_I = 1, \ldots, n_I$ and $\forall i_J = 1, \ldots, n_J$.
Incremental formulation of local 4D–VAR DA functional in (A.46) is

$$J_{i,k}(\delta z_{i,k}) = J(\delta z_k)/(tile_i \times \Delta_k) + \mathcal{O}_{IJ}(\delta z_{i,k})$$

where $\delta z_{i,k}$ are the control variable increments in $tile_i \times \Delta_k$.

Step 4: IS4D-Var.

From (A.47) local IS4D-Var function in (A.45) can be written

$$\mathbf{J}_{loc}^{ROMS} = \mathbf{J}_{tile_i \times \Delta}^{DD-4DVAR} - \mathcal{O}_{IJ} \tag{A.50}$$

Consequently, we need to add overlapping operator in order to enforce the matching of local solutions on the overlapping tiles in each time interval.

A.3.2 DD–4DVAR DA in ROMS code

In Figure A.5 the ROMS directory structure is shown. We focus on ROMS folder, in particular, on its folders: Tangent and Adjoint.

1. ROMS.

   We need to modify routines in ROMS folder implementing decomposition as in

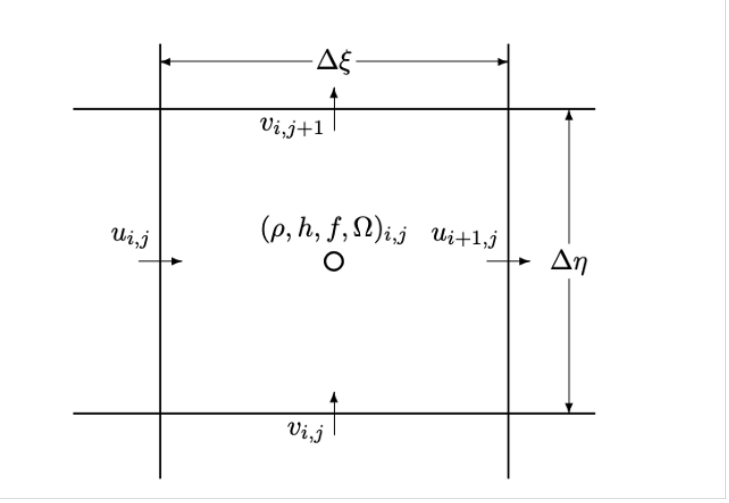


Figure A.4: Placement of variables on an Arakawa C grid.

(A.37) (see step 1).

Decomposition of time interval involves modification of initial conditions of non-linear model, in TLROMS and ADROMS.

Routines involving initialization are (see Figure A.5):

(a) initial routine in Nonlinear folder: initializes all model variables (it is called in main3d).

> **Note 1.1 (see step 1)**
> Initial routine initializes all model variable before calling main3d routine (in Nonlinear folder). Main3d routine is the main driver solving nonlinear ROMS model (background).

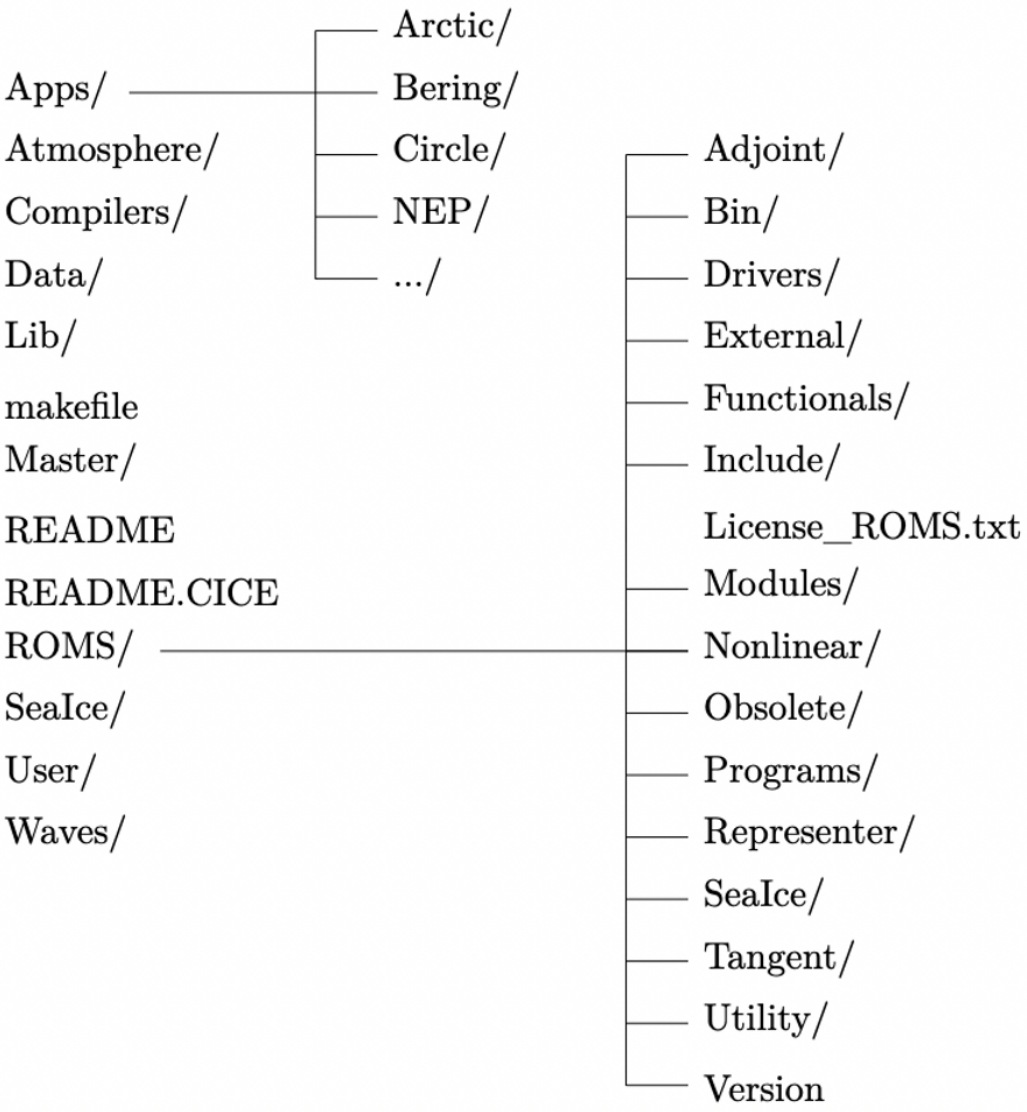


Figure A.5: ROMS directory structure.

DD in x- and y- directions is not applied for computing background, i.e. there are not MPI communications among processes.

This means that background is computed without using DD in space, consequently we not apply DD in time.

Moreover, some values of u, v (components velocity) are not set to zero at the end of time step, because its values on some grid points are necessary for the next time step.

(b) tl_initial routine in Tangent folder: initializes all tangent model variables (it is called in i4dvar).

> **Note 1.2 (see step 1).**
> We consider tl_initial routine. Parts involving initializations are:
>
> – line 123: Initializes time stepping indices and counter.
>
> – line 162: initialization time.

(c) ad_initial routine in Adjoint folder: initializes all adjoint model variables(it is called in i4dvar).

> **Note 1.3 (see step 1).**
> We consider ad_initial routine. Parts involving initializations are:
>
> – line 113: Initializes time stepping indices and counter.
>
> – line 152: initialization time.

Main actions to apply DD in time in ROMS are described below.

Action 1.1 (see step 1).

(a) Modify initials routines by adding MPI communications for initial conditions to tangent and adjoint routines in each time interval.

(b) Decompose time interval (RunInterval variable in routines). One possible way forward:

- We need to use OMP threadprivate directive for replicating variables related to time interval such that each thread has its own copy. (see mod_parallel routine in Module folder at line 51 related to DD in space).
  Allocate_routine in mod_parallel allocates tiles; equally we can allocate local time interval and related variables.
- We can add allocation and OMP threadprivate directive of local time intervals in mod_parallel routine. We can define first time interval (first_time_interval) and last time interval (last_time_interval) and add a *for loop*, after *for loop* involving tiles, started from first time interval up to last time interval adding a time step (dt) defined in driver.
  - tl_main3d: at line 277 stars *for loop* involves tiles.
  - ad_main3d: at line 629 stars *for loop* involves tiles.
- Identify tangent and adjoint variables for MPI communications in time.

(c) Introduce MPI communications in time.

- We need to split ROMS MPI communicator (OCN_COMM_WORLD) for obtaining MPI communicators needed to communications among processes related to same spatial subdomain but different time intervals.

  By splitting ROMS MPI communicator we obtained a new MPI communicator namely one single communicator for the current process.

Action 1.2 (see step 1).

Step 1 involves:

- tl_main3d routine: at line 142 starts *while loop* on time interval (RunInterval) by increasing the step time (my_steptime). Moreover, at lines 278 and 280 it calls tl_set_massflux and tl_rho_eos, i.e. the routines we need to modify (see Action 2.1,2,3,4). Consequently, we probably need to introduce in tl_main3d MPI communications in time after the tl_set_massflux and tl_rho_eos routines.

- ad_main3d routine: at line 177 starts *while loop* on time interval (RunInterval) by increasing the step time (my_steptime). Moreover, at lines 632 and 634 it calls ad_rho_eos and ad_set_massflux, i.e. the routines we need to modify (see Action 3.1 and 3.2). Consequently, we probably need to introduce in ad_main3d MPI communications in time after the ad_set_massflux and ad_rho_eos routines.

2. TLROMS.

   We need to modify TLROMS as in (A.41) (see step 2), namely we need to compute the overlapping vector in (A.42) and (A.43) and add them to tangent variables.

Action 2.1 (see step 2).
We could consider another inner loop (inside the loop over m in Figure A.1) over index n and initial approximation of solution on local tiles at $n = 0$. For each iteration we need local solution on tile adjacent to each tile, this means using MPI exchange of information between adjacent tiles (in two direction see Figure A.4).

We note that primitive equations of motion [99] are written in flux form transformed using orthogonal curvilinear coordinates (ξ, η) (see Figure A.4).
We consider tl_main3d routine. In tangent folder (see Figure A.5) this routine is the main driver of TLROMS configurated as a full 3D baroclinic ocean model.
tl_main3d routine calls the following subrountines.

(a) tl_set_massflux calls tl_set_massflux_tile.
tl_set_massflux_tile: computes "in situ" tangent linear horizontal mass flux.

Action 2.2 (see step 2).
Taking into account (A.42) and (A.43) we need to modify the code starting from line 155.

(b) tl_rho_eos calls tl_rho_eos_tile.
tl_rho_eos_tile: computes "in situ" the density and other quantities (temperature, salinity,...).

Action 2.3 (see step 2).
Parts to be modified:

i. Line 505: computes ”in situ” density anomaly;

ii. Line 591: computes ”in situ” Brunt-Vaisala frequency;

iii. Line 1158: computes ”in situ” Brunt-Vaisala frequency;

(c) tl_omega: computes vertical velocity (no modifications because DD is only horizontal, there are not overlap region along vertical direction).

3. ADROMS.

Similarly to TLROMS, we need to modify ADROMS as in (A.44) (see step 3), namely we need to compute and add the overlapping vector in (A.42) and (A.43) to adjoint variables.

We consider ad_main3d routine. In Adjoint folder (see Figure A.5) this routine is the main driver of ADROMS configurated as a full 3D baroclinic ocean model.

ad_main3d routine calls the following subrountines.

(a) ad_rho_eos: computes “in situ” density and other associated quantities.

Action 3.1 (see step 3).
Parts that should be modified:

i. Lines 797 and 1758: compute ”in situ” adjoint Brut-Vaisala frequency at horizontal points.

ii. Line 1070: computes ”in situ” adjoint density anomaly.

(b) ad_set_mass_flux: compute “in situ” adjoint horizontal mass fluxes.

Action 3.2 (see step 3).
Part that should be modified:

i. Line 201: computes ”in situ” adjoint horizontal mass fluxes

(c) ad_set_avg: accumulates and computes output time-averaged adjoint fields. (probably no modifications are needed).

4. IS4D-Var.

We need to modify routines in ROMS folder as in (A.50) (see step 4).

Action 4.1 (see step 4).
We need to modify IS4DVAR cost function in (A.30) to take in account overlap region i.e. halo region.

Main modification to apply DD in time in ROMS is described by point (c) in Action 1.1, i.e. introduction of MPI communications in time. More precisely, we need to introduce and manage the MPI communication in space and time. The introduction of DD in time involves the following communications among processes:

- Intra communications: by splitting MPI communicator (OCN_COMM_WORLD) to create local communicators (TASK_COMM_WORLD) to allow communications among processes related to same spatial subdomain but different time intervals.
  MPI commands are
  - MPI_Comm_split: partitions the group of MPI processes into disjoint subgroups and

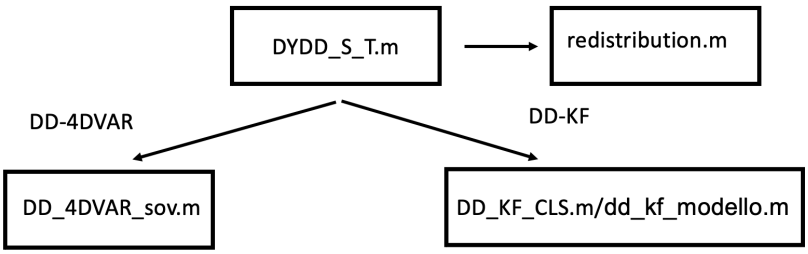


Figure A.6: Scheme of M-File functions related to DyDDST algorithm: DYDD_S_T.m and redistribution.m; DD–4DVAR algorithm: DD_4DVAR_sov.m; DD-KF-CLS algorithm: DD_KF_CLS.m; and DD-KF algorithm: dd_kf_modello.m.

creates a new communicator (TASK_COMM_WORLD) for each subgroup.

- MPI_Isend and MPI_Irecv: sends and receives initial conditions by setting the new communicator obtained from MPI_Comm_split.

- Inter communications: by creating new communicators (OCNi_COMM_WORLD) to allow communications among processes related to different spatial subdomains but same time interval.
  MPI commands are

  - MPI_Intercomm_create: creates an intercommunicator for each subgroup.
  - MPI_Isend and MPI_Irecv: sends and receives boundary conditions by setting the new communicator obtained from MPI_Intercomm_create.

A.4 MATLAB codes

In this section, we report M-files used to obtain the results in section 5.

DD_KF_CLS.m

```
function  [Xk_1,T_tot_i] = DD_KF_CLS(R,A_tot,Xk_prev_tot,Q2,b,n1,m,nsub,a,m1,nt)
%function of DD-KF algorithm to CLS problem

%input
%a_tot,A_tot:matrices to costruct a CLS problem
%m,n:matrices dimensions
%Xk_prev_tot: initial estimate
%X: KF estimate
%R,Q,Q1:covariance matrices
%nsub:number of workers

%output
%Xk_1: DD-KF estimete
%T_tot_i: execution time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%costriction of local matrices


%initializzazion
A=zeros(size(A_tot,1),n1/p,p);
K=zeros(size(A_tot,2),size(A_tot,2),p);
Xk_prev=zeros(n1/p,p);
for i=1:p
A(:,:,i)=A_tot(:,1+n1/p*(i-1):n1/p*i);
K(:,:,i)=A(:,:,i)'*Q2(1:m,1:m)*A(:,:,i);
Xk_prev(:,i)=Xk_prev_tot(1+n1/p*(i-1):n1/p*i);
end
%initializzation local solution
Xk=zeros(n1/p,1);

%covariance matrices
P=zeros(n1,n1,nsub);
for h=1:nsub
P((h-1)*(n1)+1:(h)*(n1),:,h)=0.5*eye(n1,n1);
end
```

```
%paralell section
spmd(p)
h=labindex;
t1=tic;
for j_1=1:nt
    %send and receive informations to compute local Kalman gains
  for h_ad=1:nsub
    if(h_ad~=h)

      labSend(P((h-1)*(n1)+1:(h)*(n1),:,h),h_ad)
      P_adi2((h_ad-1)*(n1)+1:(h_ad)*(n1),:)=labReceive(h_ad);
    end
 end
P_adi2((h-1)*(n1)+1:(h)*(n1),(h-1)*(n1)+1:h*(n1)) =P((h-1)*(n1)+1:h*(n1),(h-1)*(n1)+1:h*(n1),h);
H=a;
Somma=zeros(size(H,1),n1);
for j=1:nsub

    S_d1=Somma+H(:,(j-1)*(n1)+1:j*(n1))*P_adi2((j-1)*(n1)+1:j*(n1),(h-1)*(n1)+1:h*(n1));

    Somma= S_d1;
end
S_d= (S_d1)*H(:,(h-1)*(n1)+1:h*(n1))';
Sd_tot=zeros(size(H,1),size(H,1),nsub);
for h_ad=1:nsub
    if(h_ad~=h)

      labSend(S_d,h_ad)
      Sd_tot(:,:,h_ad)=labReceive(h_ad);
    end
end
somma5=zeros(size(H,1),size(H,1));
Sd_tot(:,:,h)=S_d;
for j=1:nsub
 S_d=somma5+Sd_tot(:,:,j);
 somma5=S_d;
end
S_d=S_d+R;
Somma1=zeros(n1,size(H,1));
 for j=1:nsub-1
```

```
     K=Somma1+P((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:(j)*(n1),h)*H(:,(j-1)*(n1)+1:(j)*(n1))';
 Somma1=K;
 end
K=K*inv(S_d);
%computation con covariances matrices
P_n2=zeros(n,n,nsub);
for h_ad=1:nsub
    if(h~=h_ad)

      labSend(P((h-1)*(n1)+1:(h)*(n1),:,h),h_ad)
      P_adi((h_ad-1)*(n1)+1:(h_ad)*(n1),:)=labReceive(h_ad);
    end
end
for j=1:nsub
    Somma2=zeros(n1,n1);

   for i=1:nsub
       if (i~=h)
           Somma2=-K*H(:,(i-1)*(n1)+1:(i)*(n1))*P_adi((i-1)*(n1)+1:(i)*(n1),(j-1)*(n1)+1:j*(n1))
           +Somma2;
       end
   end
P_n2((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:j*(n1),h)=(eye(n1,n1)-K*H(:,(h-1)*(n1)+1:(h)*(n1)))
*P((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:j*(n1),h)+Somma2;
end

P=P_n2;

%send and receive information to compute local KF solution

if (h==1)
Xk_prev_AD=zeros(n1,nsub);
for h_ad=1:nsub
    if(h~=h_ad)

      labSend(Xk_prev(:,h),h_ad);
     Xk_prev_AD(:,h_ad)=labReceive(h_ad);
    end
end
```

```
somma=zeros(size(H,1),1);
for j=1:nsub
    if(j~=h)
      somma=somma+(H(:,(j-1)*(n1)+1:(j)*(n1))*Xk_prev_AD(:,j));
    end
end

Xk_1 = Xk_prev(:,h)+K*(b((h-1)*m1+1:h*m1,j_t)-((H(:,(h-1)*(n1)+1:(h)*(n1))*Xk_prev(:,h)+somma)));
%send and receive information to compute local KF solutin
labSend(Xk_1,2);

Xk_prev(:,h+1)=labReceive(2);

%local KF solution
Xk_prev(:,h)=Xk_1;
elseif(h==nsub)
Xk_prev_AD=zeros(n1,nsub);
for h_ad=1:nsub
    if(h~=h_ad)

      labSend(Xk_prev(:,h),h_ad)
     Xk_prev_AD(:,h_ad)=labReceive(h_ad);
    end
end
somma=zeros(size(H,1),1);
for j=1:nsub
    if(j~=h)
      somma=somma+(H(:,(j-1)*(n1)+1:(j)*(n1))*Xk_prev_AD(:,j));
    end
end

Xk_1 = Xk_prev(:,h)+K*(y(:,j_t)-((H(:,(h-1)*(n1)+1:(h)*(n1))*Xk_prev(:,h)+somma)));
labSend(Xk_1,h-1);
Xk_prev(:,h-1)=labReceive(h-1);
%local KF solution
 Xk_prev(:,h)=Xk_1;
else
    Xk_prev_AD=zeros(n1,nsub);
for h_ad=1:nsub
    if(h~=h_ad)
```

```
      labSend(Xk_prev(:,h),h_ad)
     Xk_prev_AD(:,h_ad)=labReceive(h_ad);
    end
end
somma=zeros(size(H,1),1);
for j=1:nsub
    if(j~=h)
      somma=somma+(H(:,(j-1)*(n1)+1:(j)*(n1))*Xk_prev_AD(:,j));
    end
end

Xk_1 = Xk_prev(:,h)+K*(b((h-1)*m1+1:h*m1,j_t)-((H(:,(h-1)*(n1)+1:(h)*(n1))*Xk_prev(:,h)+somma)));

labSend(Xk_1,h-1);
Xk_prev(:,h-1)=labReceive(h-1);
labSend(Xk_1,h+1);
Xk_prev(:,h+1)=labReceive(h+1);

 %local KF solution
 Xk_prev(:,h)=Xk_1;
 end
end

Tp=gop(@max,toc(t1));
Xk_1=gcat(Xk,1);
end
toc
%execution time
T_tot_i=Tp{1};
Xk_1=Xk_1{1};
end
```

dd_kf_modello.m

```
function [Xk_1,Xk]=dd_kf_modello(Nx,nt,y,b_t,eta,psi,Xk_prev_t,r_n,r_f,R,nsub)
 %function for DD-KF algorithm to the initial boundary problem of SWEs.


%input
```

```
%Nx: number of discretization points in spatial domain
%s: size of overlap pf spatial subdomains
%nt:number of istants time
%y: observations vector
%b_t: vector concerning boundaries conditions
%eta,psi: vector to costruct model matrix
%Xk_prev_t:background
%r_n: initial istant time of time intrerval
%r_f: final istant time of time intrerval
%R: observation error covariance matrix;
%nsub: number of workers

%output
%Xk_1: DD-KF estimate
%Xk:KF estimate


%define local size
n=Nx;
n1=n/nsub;
%number of observations
k=size(y,1);
%%%Vector of the indices in correspondence of which corresponds a
%%%observation
vett=(1:n);
vett=vett(1:k);
%determine the observations matrix  H
H=zeros(k,n);

for i=1:k
  for j=1:n
    if (j==vett(i))
    H(i,j)=1;
  else
    H(i,j)=0;
    end
  end
end
%costruction of covariances matrices
C=zeros(n,n);
```

```
for i=1:n
  for j=1:n
    if (abs(i-j)<n/2)
C(i,j)=exp(-1/2)^abs(i-j)^2;
   end
  end
end
%costruction of covariance matrices
Q1=rand(n1,n1);
Q=blkdiag(Q1,Q1,Q1,Q1);

% schattering of background among subdomains
Xk_prev=zeros(n,nsub);
for h=1:nsub
Xk_prev(:,h)=Xk_prev_t((h-1)*(n1)+1:h*(n1));
end

%initialization
P_tot=zeros(n,n);

%KF algorithm in sequentially
for j=1:nt
    %model matrix
    M_h_t=diag(psi(1:n,j))+diag(eta(2:n,j),1)-diag(eta(1:n-1,j),-1);
    % Predicted phase.
    Xk_prev_t=M_h_t*Xk_prev_t(1:n)+b_t;
    P_tot=M_h_t*P_tot*M_h_t'+Q;

    %Corrector phase.
    S=H_s*P_tot*H_s'+R;
    %kalman gain
    K_tot = P_tot*H_s'*inv(S);
    %KF estimate
    Xk = Xk_prev_t+K_tot*(y(:,j)-(H_s*Xk_prev_t));
    Xk_prev_t=Xk;
    Xk_array(:,j)=Xk;
end

%initialization
inno=zeros(k,nt);
```

```
P=zeros(n,n,nsub);
for j1=1:nsub
P(:,:,j1)=zeros(n,n);
end
%parallel section
spmd(nsub)
  h=labindex;
for j_t=r_n:r_f
M_h=diag(psi(1:n,j_t))+diag(eta(2:n,j_t),1)-diag(eta(1:n-1,j_t),-1);
% Predicted phase
if(h==1)
    for i=1:nsub
        somma_p=zeros(n1,n1);
        for j=1:nsub
            Tem=somma_p+(M_h((h-1)*(n1)+1:h*(n1),(h-1)*(n1)+1:h*(n1))*P((h-1)*(n1)+1:h*(n1),
            (j-1)*(n1)+1:j*(n1),h)+(M_h((h-1)*(n1)+1:h*(n1),(h)*(n1)+1:(h+1)*(n1))*
            P((h)*(n1)+1:(h+1)*(n1),(j-1)*(n1)+1:j*
            (n1))))*M_h((i-1)*(n1)+1:i*(n1),(j-1)*(n1)+1:(j)*(n1))';
            somma_p=Tem;
        end
            P((h-1)*(n1)+1:h*(n1),(i-1)*(n1)+1:i*(n1),h)=Tem
            +Q((h-1)*(n1)+1:h*(n1),(i-1)*(n1)+1:i*(n1));
    end
    Xk_prev(:,h)=M_h((h-1)*(n1)+1:h*(n1),(h-1)*(n1)+1:h*(n1))
    *Xk_prev(:,h)+M_h((h-1)*(n1)+1:h*(n1),(h)*(n1)+1:(h+1)*(n1))
    *Xk_prev(:,h+1)+b((h-1)*(n1)+1:h*(n1),j_t);
elseif(h==nsub)
     for i=1:nsub
            somma_p=zeros(n1,n1);
        for j=1:nsub
            Tem=somma_p+(M_h((h-1)*(n1)+1:h*(n1),(h-1)*(n1)+1:h*(n1))*
            P((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:j*(n1),h)
            +(M_h((h-1)*(n1)+1:h*(n1),(h-2)*(n1)+1:(h-1)*(n1))
            *P((h-2)*(n1)+1:(h-1)*(n1),(j-1)*(n1)+1:j*(n1),h)))
            *M_h((i-1)*(n1)+1:i*(n1),(j-1)*(n1)+1:(j)*(n1))';
            somma_p=Tem;
        end
       P((h-1)*(n1)+1:h*(n1),(i-1)*(n1)+1:i*(n1),h)=Tem+
       Q((h-1)*(n1)+1:h*(n1),(i-1)*(n1)+1:i*(n1));
     end
```

```
      Xk_prev(:,h)=M_h((h-1)*(n1)+1:h*(n1),(h-1)*(n1)+1:h*(n1))
      *Xk_prev(:,h)+M_h((h-1)*(n1)+1:h*(n1),(h-2)*(n1)+1:(h-1)*(n1))
      *Xk_prev(:,h-1)
      +b((h-1)*(n1)+1:h*(n1),j_t);
else
    for i=1:nsub
            somma_p=zeros(n1,n1);
        for j=1:nsub
            Tem=somma_p+
            (M_h((h-1)*(n1)+1:h*(n1),(h-1)*(n1)+1:h*(n1))
            *P((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:j*(n1),h)+
            (M_h((h-1)*(n1)+1:h*(n1),(h-2)*(n1)+1:(h-1)*(n1))
            *P((h-2)*(n1)+1:(h-1)*(n1),(j-1)*(n1)+1:j*(n1)))
            +(M_h((h-1)*(n1)+1:h*(n1),(h)*(n1)+1:(h+1)*(n1))*
            P((h)*(n1)+1:(h+1)*(n1),(j-1)*(n1)+1:j*(n1))))
            *M_h((i-1)*(n1)+1:i*(n1),(j-1)*(n1)+1:(j)*(n1))';
            somma_p=Tem;
        end
   P((h-1)*(n1)+1:h*(n1),(i-1)*(n1)+1:i*(n1),h)=Tem+Q((h-1)*(n1)+1:h*(n1),(i-1)*(n1)+1:i*(n1));
    end
  Xk_prev(:,h)=M_h((h-1)*(n1)+1:h*(n1),(h-1)*(n1)+1:h*(n1))
  *Xk_prev(:,h)+M_h((h-1)*(n1)+1:h*(n1),(h-2)*(n1)+1:(h-1)*(n1))
  *Xk_prev(:,h-1)+M_h((h-1)*(n1)+1:h*(n1),(h)*(n1)+1:(h+1)*(n1))
  *Xk_prev(:,h+1)+b((h-1)*(n1)+1:h*(n1),j_t);
end
%Corrector phase.
    Somma=zeros(size(H,1),n1);
    if(h==1)
    S_d1=Somma+H(:,(j-1)*(n1)+1:j*(n1))*P((j-1)*(n1)+1:j*(n1),(j-1)*(n1)+1:j*(n1),j);
    S_d= S_d1*H(:,1:1*(n1))'+R;
labSend(S_d,2:nsub);
    else
S_d=labReceive(1);
    end
 Somma1=zeros(n1,size(H,1));
 for j=1:nsub-1
 K=Somma1+P((h-1)*(n1)+1:h*(n1),(j+1-1)*(n1)+1:(j+1)*(n1),h)*H(:,(j+1-1)*(n1)+1:(j+1)*(n1))';
 Somma1=K;
end
```

```
K=(P((h-1)*(n1)+1:h*(n1),(h-1)*(n1)+1:h*(n1),h)*H(:,(h-1)*(n1)+1:(h)*(n1))')*inv(S_d);

Somma2=zeros(n1,n1);

for j=1:nsub-1
P((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:j*(n1),h)=
-K*H(:,(j+1-1)*(n1)+1:(j+1)*(n1))*
P((h-1)*(n1)+1:h*(n1),(j+1-1)*(n1)+1:(j+1)*(n1),h)
+Somma2;
Somma2=P((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:j*(n1),h);
end
for j=1:nsub-1
P((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:j*(n1),h)=P((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:j*(n1),h)
+(eye(n1,n1)-K*H(:,(h-1)*(n1)+1:(h)*(n1)))*P((h-1)*(n1)+1:h*(n1),(j-1)*(n1)+1:j*(n1),h);
end
P((h-1)*(n1)+1:h*(n1),:,h)=(eye(n1,n1)-
K*H(:,(h-1)*(n1)+1:(h)*(n1)))
*P((h-1)*(n1)+1:h*(n1),:,h);
P_tot((h-1)*(n1)+1:h*(n1),:)
%send and receive
if (h==1)
labSend(Xk_prev(:,h),2);
Xk_prev(:,h+1)=labReceive(2);
Xk_1 = Xk_prev(:,h)+K*(y(:,j_t)
 -((H(:,(h-1)*(n1)+1:(h)*(n1))*Xk_prev(:,h)
 +H(:,(h)*(n1)+1:(h+1)*(n1))*Xk_prev(:,h+1))));
Xk_prev(:,h)=Xk_1;
elseif(h==nsub)
labSend(Xk_prev(:,h),h-1);
Xk_prev(:,h-1)=labReceive(h-1);
Xk_1 = Xk_prev(:,h)+K
*(y(:,j_t)-((H(:,(h-1-1)*(n1)+1:(h-1)*(n1))
*Xk_prev(:,h-1)
+H(:,(h-1)*(n1)+1:(h)*(n1))*Xk_prev(:,h))));
Xk_prev(:,h)=Xk_1;
else
labSend(Xk_prev(:,h),h-1);
Xk_prev(:,h-1)=labReceive(h-1);
labSend(Xk_prev(:,h),h+1);
Xk_prev(:,h+1)=labReceive(h+1);
```

```
Xk_1 = Xk_prev(:,h)+
K*(y(:,j_t)
-((H(:,(h-1-1)*(n1)+1:(h-1)*(n1))
*Xk_prev(:,h-1)+H(:,(h-1)*(n1)+1:(h)*(n1))
*Xk_prev(:,h)+H(:,(h)*(n1)+1:(h+1)*(n1))*Xk_prev(:,h+1))));
Xk_prev(:,h)=Xk_1;
end
end
%gathering of local solution
Xk_1=gcat(Xk_1,1);
%gathering background
Xk_prev=gcat(Xk_prev);
end
 Xk_1=Xk_1{1};
Xk_prev=Xk_prev{1};

end

function [X,error_globale_1,Tp,iter] = DD_4DVAR_sov(h_array,n1,s,r,b1_m,y_s,eta,psi,
n,nsub,u1_array_globale,r_n,r_f,beta,eo)

%function for running DD-4DVAR algorithm on nsub computing elements in case
%of size of overlap s>2
n2=n1;
%INPUT:

%h_array:background;
%n1: local size proble
%r: number of istants time
%b1_m: vector for boundary conditions;
%y_s:vector of observations;
%eta,psi: vector to costruct model matrix
%nsub:number of spatial subdomains;
%N:number of discretizations points of global spatial domains;
%b: vector for boundary conditions;
%r_n:initial istant time;
%r_f: final istant time.
%u1_array_globale: solution on global domain.
%beta:overlap parameter
```

```
%OUTPUT:
%X: matrix that contains local solution;
%T1: sequential time.
%e1_p: error between local model solution e DD-4dvar estimate;
%error_global: error between global estimate and estimate obtained by
%gathering local DD-4DVAR estimates.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%numbers of observations
r1=size(y_s,1);
%vector of observations
y=y_s;
%%%Vector of the indices in correspondence of which corresponds a
%%%observation (+ random error (see SWE_DamBreak_JA.m function))
vett=round(linspace(1,n,r1));
%determine the observations matrix  H
H1=zeros(r,n+1);
H1_b=zeros(r1,n+1);
for j=1:r1
    for i=1:n
        if(i==vett(j))
            H1_b(j,i)=1;
        else
            H1_b(j,i)=0;
        end
    end
end
%observation matrix in each istant time
%(I suppose  i t s  always the same matrix for each istant)

for k=1:r
    H1((k-1)*r1+1:r1*k,:)=H1_b;
end
%costruction of covariances matrices
%Matrix construction of the covariances of errors on the model to obtain
% a diagonal block matrix.
C1=zeros(n,n);
```

```
for i=1:n
for j=1:n
if (abs(i-j)<n/2)
C1(i,j)=exp(-1/2)^abs(i-j)^2;
end
end
end
c=0.5;
cr=0.5;
R=cr*eye(r1*r,r1*r);
C2=zeros(n,n);
C2(1:n1-s,1:n1-s)=C1(1:n1-s,1:n1-s);
h=nsub;
C2((h-1)*(n1-s)+1:h*(n1-s),(h-1)*(n1-s)+1:h*(n1-s))=C1(1:n1-s,1:n1-s);
for h=1:nsub
C2((h-1)*(n1-s)+1:h*(n1-s),(h-1)*(n1-s)+1:h*(n1-s))=C1(1:n1-s,1:n1-s);
end
%computation of root means square of covariance matrix B
V=zeros(n1,n1,nsub);
V(:,:,1)=chol(c*C2(1:n1,1:n1));
for h=2:nsub
V(:,:,h)=V(:,:,1);
end
V(n1,n1,nsub)=0;
V(1:s,1:s,nsub)=V(n1-s,n1-s,1);
V(s+1:n1,s+1:n1,nsub)=V(1:n1-s,1:n1-s,1);

B=zeros(n1,n1,nsub);
for h=1:nsub
B(:,:,h)=c*C2(1:n1,1:n1);
end
%costruction of matrices on overlapping region
B_r1=zeros(n1,n1,nsub);
B_r2=zeros(n2,n2,nsub);
for h=1:nsub
B_r1(n1+1-s:n1,n1+1-s:n1,h)=1/nsub*B(n1+1-s:n1,n1+1-s:n1,h);
B_r2(1:s,1:s,h)=1/nsub*B(1:s,1:s,h);
end
%number of observation in each subdomains
m1=r1/nsub;
```

```
for h=1:nsub
condizionamento(h)=cond(B_r1(n1+1-s:n1,n1+1-s:n1,h))
%determine the observations matrix H for each subdomain
end
H=zeros(r*m1,n1,nsub);
for k=1:r
H((k-1)*m1+1:(k)*m1,1:n1,1)=H1((k-1)*r1+1:(k-1)*r1+m1,1:n1);
for h=2:nsub-1
   H((k-1)*m1+1:(k)*m1,:,h)=H1((k-1)*r1+(h-1)*m1+1:(k-1)*r1+(h)*m1,(h-1)*(n1-s)+1:h*(n1-s)+s);
end
H((k-1)*m1+1:(k)*m1,:,nsub)=H1((k-1)*r1+(nsub-1)*m1+1:(k-1)*r1+(nsub)*m1,(nsub-1)*(n1-s)-(s-1):n);
end

% rhs vectors
d1=zeros(r*m1,nsub);
v=zeros(r*m1,nsub);
for k=1:r
for h=1:nsub
    for j=1:m1
    v((k-1)*m1+j,h)=h_array(vett(j+(h-1)*m1),k);
    d1((k-1)*m1+j,h)=y(j+(h-1)*m1,k)'-v((k-1)*m1+j,h);
    end
end
end
% schattering of background among subdomains
u0=h_array;
u_m(:,1:r,1)=h_array(1:n1,1:r);
for h=2:nsub-1
u_m(:,1:r,h)=h_array((h-1)*(n1-s)+1:h*(n1-s)+s,1:r);
end
u_m(:,1:r,nsub)=h_array((nsub-1)*(n1-s)-(s-1):n,1:r);
u01=zeros(n1,r,nsub);
u01(:,1:r,1)=u0(1:n1,1:r);
for h=2:nsub-1
u01(:,1:r,h)=u0((h-1)*(n1-s)+1:h*(n1-s)+s,1:r);
end
u01(:,1:r,nsub)=u0((nsub-1)*(n1-s)-(s-1):n,1:r);
u1_array_pp=zeros(n1,r,nsub);
u_p1_array_g=zeros(n1,size(u_m,2),nsub);
for h=1:nsub
```

```
 u_p1_array_g(:,:,h)=u_m(:,:,h);
end
 %initializzation
maxiter_DD=10;
%iter_DD=1;
tol=10^-3;
e1_b=1000*ones(maxiter_DD,1);
%e2_b(iter_DD,h)=1000;
w1_p=zeros(n1,1);
u1_array=zeros(n1,r);
M1=zeros(n1,n1,nsub);
%parallel section
%tic
spmd(nsub)
%initializzation
u1_p=zeros(n1,r);
w1=zeros(n1,1);
u1=zeros(n1,r);
u1_array_p=zeros(n1,r);
r1=m1;
errore_seq=zeros(r,1);
h=labindex;
%function for execution time
t1=tic;
%%% loop in time for the fist iteration respect to n.
for k=r_n:r_f
   %solving local 4DVAR problem
if h==1
A1_tot=V(:,:,h)'*H(1:k*r1,:,h)'*R(1:k*r1,1:k*r1)*H(1:k*r1,:,h)*V(:,:,h)+eye(n1,n1);
A1_tot=A1_tot+1*B_r1(:,:,h);
c1=(V(:,:,h)'*(H(1:k*r1,:,h)'*(R(1:k*r1,1:k*r1)*d1(1:k*r1,h))));
elseif(h==nsub)
A1_tot=V(:,:,h)'*H(1:k*r1,:,h)'*R(1:k*r1,1:k*r1)*H(1:k*r1,:,h)*V(:,:,h)+eye(n1,n1);
A1_tot=A1_tot+1*B_r2(:,:,h);
c1=(V(:,:,h)'*(H(1:k*r1,:,h)'*(R(1:k*r1,1:k*r1)*d1(1:k*r1,h))));
else
A1_tot=V(:,:,h)'*H(1:k*r1,:,h)'*R(1:k*r1,1:k*r1)*H(1:k*r1,:,h)*V(:,:,h)+eye(n1,n1);
A1_tot=A1_tot+B_r1(:,:,h)+B_r2(:,:,h);
c1=(V(:,:,h)'*(H(1:k*r1,:,h)'*(R(1:k*r1,1:k*r1)*d1(1:k*r1,h))));
end
```

```
%vector on boundaries
w1_r=zeros(n1,1);
w2_r=zeros(n1,1);
%initializzation
iter_DD1=1;
e1_b2=100*ones(maxiter_DD,1);



while (iter_DD1<maxiter_DD && e1_b2(iter_DD1)>tol)

if h==1
b1=c1+beta*B_r1(:,:,h)*w1_r;
elseif(h==nsub)
b1=c1+beta*B_r2(:,:,h)*w2_r;
else
b1=c1+beta*B_r1(:,:,h)*w1_r+beta*B_r2(:,:,h)*w2_r;
end

[w1,~,~,ITER]=pcg(A1_tot,b1,10^-4,size(B,1),B(:,:,h));
%DD error
e1_b2(iter_DD1)=norm(w1-w1_p);
w1_p=w1;
iter_DD1=iter_DD1+1;
%define indices
if h < numlabs
    rcvWkrIdx = labindex + 1;
else
    rcvWkrIdx = [];
end
if labindex > 1
    srcWkrIdx = labindex - 1;
else
    srcWkrIdx = [];
end

%send and receive to adjcent subdomains
w1_pr2 = labSendReceive(rcvWkrIdx,srcWkrIdx,w1_p((n1)-s+1:(n1)));
```

```
%define indices
if h>1
    rcvWkrIdx = labindex - 1;
else
    rcvWkrIdx = [];
end
if h<numlabs
    srcWkrIdx = labindex + 1;
else
    srcWkrIdx = [];
end
%send and receive to adjcent subdomains
w1_pr1 = labSendReceive(rcvWkrIdx,srcWkrIdx,w1_p(1:s));

   w1_r=[ zeros(n1-s,1)' w1_pr1']';
   w2_r=[w1_pr2' zeros(n1-s,1)']';
end

%local solutions
if(h==1)
u1_p(:,k)=u_m(:,k,h)+V(:,:,h)*w1;
elseif(h==nsub)
 u1_p(:,k)=u_m(:,k,h)+V(:,:,h)*w1;
else
 u1_p(:,k)=u_m(:,k,h)+V(:,:,h)*w1;
end
u1_array(:,k)=u1_p(:,k);

end

%initializzation
iter=1;
maxiter=5;
error_iter(1)=100;
error_iter(2:maxiter)=zeros(maxiter-1,1);
while (iter<maxiter  && error_iter(iter)>tol  )
    %initial condition to local models
u1_k=u01;
%initialization
e1_p=zeros(maxiter_DD,r,nsub);
```

```
  for k=r_n+1:r_f
      M=diag(psi(1:n,k))+diag(eta(2:n,k),1)-diag(eta(1:n-1,k),-1);
      if(h==1)
      M1(:,:,h)=M(1:n1,1:n1);
      elseif(h==nsub)
      M1(:,:,h)=M((h-1)*(n1-s)-(s-1):n,(h-1)*(n1-s)-(s-1):n);
      else
      M1(:,:,h)=M((h-1)*(n1-s)+1:(h)*(n1-s)+s,(h-1)*(n1-s)+1:(h)*(n1-s)+s);
      end
 %solving local models
if (h==nsub)
      u1_g=M1(:,:,h)*u1_k(:,k-1,h)+[ M((h-1)*(n1-s)-(s-1):h*(n1-s),(h-2)*(n1-s)+1:(h-1)*(n1-s)-s)
zeros(n1,2*s)]
      *u1_k(:,k-1,h-1)+b1_m(:,k,h);
  elseif(h==1)
        u1_g=M1(:,:,h)*u1_k(:,k-1,h)+
        [zeros(n1,s) M((h-1)*(n1-s)+1:h*(n1-s)+s,(h)*(n1-s)+1:(h+1)*(n1-s))]
        *u1_k(:,k-1,h+1)+b1_m(:,k,h);
elseif(h==2)
    u1_g=M1(:,:,h)*u1_k(:,k-1,h)+
    [zeros(n1,s) M((h-1)*(n1-s)+1:h*(n1-s)+s,(h)*(n1-s)+1+s:(h+1)*(n1-s))  zeros(n1,s) ]
    *u1_k(:,k-1,h+1)+[M((h-1)*(n1-s)+1:h*(n1-s)+(s),(h-2)*(n1-s)+1:(h-1)*(n1-s)) zeros(n1,s)]
    *u1_k(:,k-1,h-1)+b1_m(:,k,h);
elseif(h==nsub-1)
    u1_g=M1(:,:,h)*u1_k(:,k-1,h)+
    [zeros(n1,2*s) M((h-1)*(n1-s):h*(n1-s)+(s-1),(h)*(n1-s)+1:(h+1)*(n1-s)-s) ]
    *u1_k(:,k-1,h+1)+
    [M((h-1)*(n1-s)+1:h*(n1-s)+s,(h-2)*(n1-s)+1:(h-1)*(n1-s)) zeros(n1,s)]
    *u1_k(:,k-1,h-1)+b1_m(:,k,h);
else
    u1_g=M1(:,:,h)*u1_k(:,k-1,h)+
    [zeros(n1,s) M((h-1)*(n1-s):h*(n1-s)+(s-1),(h)*(n1-s)+1:(h+1)*(n1-s)-2*s+1+(s-1)) zeros(n1,s)]
    *u1_k(:,k-1,h+1)+[ M((h-1)*(n1-s)+1:h*(n1-s)+s,(h-2)*(n1-s)+1:(h-1)*(n1-s)) zeros(n1,s)]
    *u1_k(:,k-1,h-1)+b1_m(:,k,h);
end
%DD-4DVar solution on subdomains
u1(:,k)=u1_g+u1_array(:,k)-u_p1_array_g(:,k,h);

u1_array_p(:,k)=u1(:,k);
u_p1_array_g(:,1,h)=u1(:,1);
```

```
u_p1_array_g(:,k,h)=u1_g;
     %error
      if(iter==1)
      e1_p(iter,k,h)=norm(u1(:,k)-u_m(:,k,h));
      end
  end
%initializzation
%iter_DD2=1;
%e1=zeros(maxiter_DD,nsub);
%e1(iter_DD2,h)=100;
for k=r_n:r_f
     %solving local 4DVAR
if h==1
A1_tot=V(:,:,h)'*H(1:k*r1,:,h)'*R(1:k*r1,1:k*r1)*H(1:k*r1,:,h)*V(:,:,h)+eye(n1,n1);
A1_tot=A1_tot+B_r1(:,:,h);
c1=(V(:,:,h)'*(H(1:k*r1,:,h)'*(R(1:k*r1,1:k*r1)*d1(1:k*r1,h))));
elseif(h==nsub)
A1_tot=V(:,:,h)'*H(1:k*r1,:,h)'*R(1:k*r1,1:k*r1)*H(1:k*r1,:,h)*V(:,:,h)+eye(n1,n1);
A1_tot=A1_tot+B_r2(:,:,h);
c1=(V(:,:,h)'*(H(1:k*r1,:,h)'*(R(1:k*r1,1:k*r1)*d1(1:k*r1,h))));
else
A1_tot=V(:,:,h)'*H(1:k*r1,:,h)'*R(1:k*r1,1:k*r1)*H(1:k*r1,:,h)*V(:,:,h)+eye(n1,n1);
A1_tot=A1_tot+B_r1(:,:,h)+B_r2(:,:,h);
c1=(V(:,:,h)'*(H(1:k*r1,:,h)'*(R(1:k*r1,1:k*r1)*d1(1:k*r1,h))));
end
iter_DD2=1;
while (iter_DD2<maxiter_DD && e1_b(iter_DD2)>tol )
if h==1
b1=c1+beta*B_r1(:,:,h)*w1_r;
elseif(h==nsub)
b1=c1+beta*B_r2(:,:,h)*w2_r;
else
b1=c1+beta*B_r1(:,:,h)*w1_r+beta*B_r2(:,:,h)*w2_r;
end
[w1,~,~,ITER2]=pcg(A1_tot,b1,10^-4,size(B,1),B(:,:,h));
iter_DD2=iter_DD2+1;
 e1_b(iter_DD2)=norm(w1-w1_p);
 w1_p=w1;
  iter_DD2=iter_DD2+1;
if h < numlabs
```

```
    rcvWkrIdx = labindex + 1;
else
    rcvWkrIdx = [];
end
if labindex > 1
    srcWkrIdx = labindex - 1;
else
    srcWkrIdx = [];
end
%communication send/receive
w1_pr2 = labSendReceive(rcvWkrIdx,srcWkrIdx,w1_p((n1)-s+1:(n1)));
if h>1
    rcvWkrIdx = labindex - 1;
else
    rcvWkrIdx = [];
end
if h<numlabs
    srcWkrIdx = labindex + 1;
else
    srcWkrIdx = [];
end
%communication send/receive
w1_pr1 = labSendReceive(rcvWkrIdx,srcWkrIdx,w1_p(1:s));

   w1_r=[ zeros(n1-s,1)' w1_pr1']';
   w2_r=[w1_pr2' zeros(n1-s,1)']';
end
%update
u1_array(:,k)=u_p1_array_g(:,k,h)+V(:,:,h)*w1;
%e1_p(iter,h)=norm(u1_array_pp(:,1:r/2,h)-u1_array_p(:,1:r/2,h))
  u1_array_pp(:,k,h)=u1_array_p(:,k);
  u1_array_p(:,k)=u1_array(:,k);

%error between iterations
 e1_p(iter,k,h)=norm(u1_array_pp(:,k,h)-u1_array_p(:,k));

if (h==1)
u1_array(:,1)=h_array(1:n1,1);
elseif(h==nsub)
u1_array(:,1)=h_array((nsub-1)*(n1-s)-(s-1):n,1);
```

```
else
    u1_array(:,1)=h_array((h-1)*(n1-s)-s+1:h*(n1-s),1);
end
end
%error
error_iter(iter)=norm(u1_array(:,k)-u1(:,k)) ;
 iter=iter+1
end


%execution time
Tp=gop(@max,toc(t1));
%gathering
e1_p=gather(e1_p);
e1_b=gather(e1_b);
X=gcat(u1_array);
error_seq_par=gcat(errore_seq);
Iter_max=gcat(ITER);
Iter_max2=gcat(ITER2);
%eo(1)
end
Tp=Tp{1};
e1_p=e1_p{1};
e1_b=e1_b{1};
X=X{1};
iter_h=Iter_max{1};
iter_pcg=max(iter_h);
iter_h2=Iter_max2{1};
iter_pcg2=max(iter_h2);
 iter=max([iter_pcg2 iter_h2]);
u1_array=(u1_array{1});
error_seq_par=error_seq_par{1};
vector_seq=u1_array_globale(1:n1-2*s,r_n:r_f);
vector_dd=X(1:n1-2*s,r_n:r_f);
for h=2:nsub-1
  vector_seq=[vector_seq;u1_array_globale(1+s:n1-s,(h-1)*r_f+1:h*r_f)];
  vector_dd=[vector_dd; X(1+s:n1-s,(h-1)*r_f+1:h*r_f)];
end
vector_seq=[vector_seq;u1_array_globale(1+s:n1,(nsub-1)*r_f+1:nsub*r_f)];
vector_dd=[vector_dd; X(1+s:n1,(nsub-1)*r_f+1:nsub*r_f)];
```

```
error_global=norm(vector_seq(:,r_n+1:r_f)-vector_dd(:,r_n+1:r_f));
end
```

DYDD_S_T.m

```
function[l_p1,l_bar,T,ITER,T_totale,T_tot,Tempo_L]=DYDD_S_T(p,l,ntem)
%FUNCTION for parallel dynamic distribution in space and time of
%observations.

%input
%p number of workers
%l initial distribution
%ntem number of time intervals

%output
%l_p1: finale balanced observations
%l_bar: average load
%T: executuion time of parallel section
%ITER: numer of iterations
%T_totale: time execution for each time iterval
%T_tot: total execution time
%Tempo_L: execution time to costruct Laplacian matrix
%initialization
r=zeros(p,p);
%define decomposition:
%%%% DECOMPOSITION WITH P-1  ADJACENT SUBDOMINES TO OMEGA 1
for i=1:p

r(1,i)=i;
r(i,1)=1;
r(i,i)=i;
end
for i=2:p
for j=2:p
if (i~=1)
r(i,j)=i;
end
if (j~=1)
r(i,j)=i;
end
```

```
end
end
ad_dom=zeros(p,p);

for i=1:p
ad_dom(1,i)=1;
end
for i=2:p
ad_dom(i,i)=1;
ad_dom(i,1)=1;
end
%define degree of graph's nodes
deg=zeros(1,p);
deg(1)=p-1;
for i=2:p
deg(i)=1;
end
%%%%%%% HORIZONTAL DECOMPOSITION (3 adjacent subdomains and 2 for omega 1,2, p-1, p)
%%%_________
%%% 1 |  2  |
%%%_________|
%%% 3 |  4  |
%%%_________|
%%% 5 |  6  |
%%%_________|
%%% 7 |  8  |
%%%_________|
r_in=zeros(p,p);
for i=1:p
r_in(i,:)=i;
end
for i=3:p-2
  if (floor(i/2)*2 == i)
  r_in(i,i-2)=i-2;
  r_in(i,i-1)=i-1;
  r_in(i,i+2)=i+2;

 else
   r_in(i,i-2)=i-2;
   r_in(i,i+1)=i+1;
```

```
   r_in(i,i+2)=i+2;
  end
end
r_in(1,2)=2;
r_in(1,3)=3;

r_in(2,1)=1;
r_in(2,4)=4;

r_in(p-1,p-3)=p-3;
r_in(p-1,p)=p;

r_in(p,p-1)=p-1;
r_in(p,p-2)=p-2;
ad_dom_in=zeros(p,p);
for i=1:p
    for j=1:p
        if (r_in(i,j)~=i)
            ad_dom_in(i,j)=1;
        end
    end
    ad_dom_in(i,i)=1;
end
deg_in=zeros(1,p);
for i=1:p
deg_in(i)=sum(ad_dom_in(i,:))-1;
end
r=r_in;
ad_dom=ad_dom_in;
deg=deg_in;
%DD-CHECK
for j=1:ntem
      r=r_in;
      ad_dom=ad_dom_in;
      deg=deg_in;
    tic;
    for i=1:p
    if(l(j,i)==0)
        %call routine to redistribute the observations
     [l(j,:),r,ad_dom,deg]=redistribution(l(j,:),r,ad_dom,p);
```

```
    end
    end
 t_r=toc;
tic;
 %computation of average load
l_bar=sum(l(j,:))/p;
%computation of misfit
b=l(j,:)'-l_bar*ones(p,1);
%execution time
T1(j)=toc;
tic
%costruction of Laplacian matrix
L1=zeros(p,p);
for i=1:p
for j_1=1:p
if(ad_dom(i,j_1)==1)
L1(i,j_1)=-1;
else
L1(i,j_1)=0;
end
L1(j_1,i)=L1(i,j_1);
end
L1(i,i)=deg(i);
end
Tempo_L(j)=toc;
tic
%solving laplaciansistem
[lambda,FLAG,RELRES,ITER(j)]=pcg(L1,b);
%execution time
Tempo_pcg(j)=toc;
%compute the adjacent subdomains for each spatial subdomains
tic
n_ad=zeros(p,1);
for i=1:p
    for j1=1:p
        if (r(i,j1)~=i)
            n_ad(i)=n_ad(i)+1;
        end
    end
end
```

```
T2(j)=toc;
spmd(p)
%codistribution of misfit
b=codistributed(b,codistributor('1d',1));
end
%paralle section
 spmd(p)
t1=tic;
l_in=l(j,labindex);
%parallel dynamic distribution
for i=1:p
    if(r(labindex,i)~=labindex)
     c=lambda(r(labindex,i));
     lambda(labindex);
     delta1=lambda(labindex)-c;
     l_p=l_in-round((delta1));
     l_in=l_p;
    end
    i=i+1;
end
%gathering of solutions
l_p=gcat(l_p);
%execution time in parallel
Tp=gop(@max,toc(t1));
 end
l_p=l_p{1};
T(j)=Tp{1};
l_p1(j,:)=l_p;
delta1=delta1{1};
delta=(delta1);
tic
T3(j)=toc;
% execution time
T_tot(j)=T3(j)+T2(j)+T(j)+Tempo_pcg(j)+T1(j)+Tempo_L(1);
end
%total execution time
T_totale=sum(T_tot);
end
```

redistribution.m

```
function [l,r,ad_dom,deg]=redistribution(l,r,ad_dom,p,deg)
%Function to redistribute the observations after during dd-check some
%subdomains are empty (no observation)


%input
%l: vector initial distribution
%r,ad_dom:vector contains informations about adjacent subdomains
%p: number of workers
%deg: degree of nodes's processor graph

%output
%l: vector of observation distribution after redistribution
%r,ad_dom: update vector contains informations about adjacent subdomains
%deg: update degree of nodes's processor graph


for i=1:p
    for j=1:p
        if (l(i)==0)
        if (l(r(i,j))==max(l(r(i,:))))
         j_1=r(i,j);

l(i)=round(l(j_1)/2);
l(j_1)=l(j_1)-l(i);

if (p-2>=i>=3 )
  if (floor(i/2)*2 == i)
  r(i,i-3)=i-3;
  r(i,i-1)=i-1;
  r(i,i+1)=i+1;

 else
   r(i,i+3)=i+3;
  r(i,i-1)=i-1;
  r(i,i+1)=i+1;
  end
end
```

```
if (i==1)
r(1,2)=2;
r(1,4)=4;
elseif(i==2)
r(2,1)=1;
r(2,4)=3;
elseif(i==p-1)
r(p-1,p-2)=p-2;
r(p-1,p)=p;
elseif(i==p)
r(p,p-1)=p-1;
r(p,p-3)=p-3;
end

for j1=1:p
if (r(i,j1)~=j1)
    r(i,j1)=i;
end
end

r(i,i)=i;

        end
        end
    end
end
for i=1:p
for j1=1:p
if (r(i,j1)==i)
    r(j1,i)=j1;
end
end
end
for i=1:p
    for j1=1:p

    if (r(i,j1)~=i)
     ad_dom(i,j1)=1;
    else
```

```
        ad_dom(i,j1)=0;
    end
    end
    ad_dom(i,i)=1;
end
for i=1:p
deg(i)=sum(ad_dom(i,:))-1;

end
end
```

Bibliography

[1] S. Amaral, D. Allaire, and K. Willcox. A decomposition-based approach to uncertainty analysis of feed-forward multicomponent systems. *Internat. J. Numer. Methods Engrg.*, 100(13):982–1005, 2014.

[2] G. M. Amdahl. 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, 1967.

[3] G. M. Amdahl. 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, 1967.

[4] J. L. Anderson. An ensemble adjustment kalman filter for data assimilation. *Monthly weather review*, 129(12):2884–2903, 2001.

[5] H. Antil, M. Heinkenschloss, R. H. W. Hoppe, and D. C. Sorensen. Domain decomposition and model reduction for the numerical solution of PDE constrained optimization problems with localized optimization variables. *Comput. Vis. Sci.*, 13(6):249–264, 2010.

[6] L. Antonelli, L. Carracciuolo, M. Ceccarelli, L. D'Amore, and A. Murli. Total variation regularization for edge preserving 3d spect imaging in high performance computing

environments. In *International Conference on Computational Science*, pages 171–180. Springer, 2002.

[7] R. Arcucci, L. D'Amore, and L. Carracciuolo. On the problem-decomposition of scalable 4d-var data assimilation models. In *2015 International Conference on High Performance Computing & Simulation (HPCS)*, pages 589–594. IEEE, 2015.

[8] R. Arcucci, L. D'Amore, J. Pistoia, R. Toumi, and A. Murli. On the variational data assimilation problem solving and sensitivity analysis. *J. Comput. Phys.*, 335:311–326, 2017.

[9] R. Arcucci, L. D'Amore, L. Carracciuolo, G. Scotti, and G. Laccetti. A decomposition of the tikhonov regularization functional oriented to exploit hybrid multilevel parallelism. *International Journal of Parallel Programming*, 45(5):1214–1235, 2017.

[10] R. Arcucci, L. D'Amore, S. Celestino, G. Laccetti, and A. Murli. A scalable numerical algorithm for solving tikhonov regularization problems. In *Parallel Processing and Applied Mathematics*, pages 45–54. Springer, 2016.

[11] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah. Parallel-in-time molecular-dynamics simulations. *Physical Review E*, 66(5):057701, 2002.

[12] J. K. Baksalary and O. M. Baksalary. Particular formulae for the moore–penrose inverse of a columnwise partitioned matrix. *Linear algebra and its applications*, 421(1):16–23, 2007.

[13] G. Bal. 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, 2005.

[14] G. Bal and Y. Maday. 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 (Zürich, 2001)*, volume 23 of *Lect. Notes Comput. Sci. Eng.*, pages 189–202. Springer, Berlin, 2002.

[15] A. T. Barker and M. Stoll. Domain decomposition in time for PDE-constrained optimization. *Comput. Phys. Commun.*, 197:136–143, 2015.

[16] G. Battistelli and L. Chisci. Stability of consensus extended kalman filter for distributed state estimation. *Automatica*, 68:169–178, 2016.

[17] E. Benhamou. Kalman filter demystified: from intuition to probabilistic graphical model to real case in financial markets. *Available at SSRN 3292762*, 2018.

[18] C. H. Bischof. Automatic differentiation, tangent linear models, and (pseudo) adjoints. In *High Performance Computing in the Geosciences*, pages 59–80. Springer, 1995.

[19] J. Blum, F. Le Dimet, and I. Navon. Data assimilation for geophysical fluids, volume xiv of handbook of numerical analysis, chapter 9, 2005.

[20] J. E. Boillat. Load balancing and poisson equation in a graph. *Concurrency: practice and experience*, 2(4):289–313, 1990.

[21] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[22] D. Bruciaferri, M. Tonani, H. W. Lewis, J. R. Siddorn, A. Saulter, J. M. Castillo, N. G. Valiente, D. Conley, P. Sykes, I. Ascione, and et al. Nemo community ocean model for multifarious space and time scales, Jan 1970.

[23] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. *Acta numerica*, 3:61–143, 1994.

[24] S. Clerc. Etude de schémas décentrés implicites pour le calcul numérique en mécanique des fluides. *Résolution par décomposition de domaines-These de l'université Paris VI-1997 (non publiée)*, 1997.

[25] S. E. Cohn. An introduction to estimation theory (gtspecial issueltdata assimilation in meteology and oceanography: theory and practice). *Journal of the Meteorological Society of Japan. Ser. II*, 75(1B):257–288, 1997.

[26] S. Cuomo, A. Galletti, G. Giunta, and L. Marcellino. Numerical effects of the gaussian recursive filters in solving linear systems in the 3dvar case study. *Numerical Mathematics: Theory, Methods and Applications*, 10(3):520–540, 2017.

[27] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of parallel and distributed computing*, 7(2):279–301, 1989.

[28] D. N. Daescu and I. M. Navon. Part b: Sensitivity analysis in non-linear variational data assimilation: Theoretical aspects and applications. *Advanced numerical methods for complex environmental models: needs and availability, edited by: Farago, I., Havasi, A., and Zlatev, Z., Bentham Science Publishers*, pages 276–300, 2013.

[29] G. Dahlquist. Convergence and stability in the numerical integration of ordinary differential equations. *Mathematica Scandinavica*, pages 33–53, 1956.

[30] L. D'Amore, R. Arcucci, L. Carracciuolo, and A. Murli. A scalable approach for variational data assimilation. *J. Sci. Comput.*, 61(2):239–257, 2014.

[31] L. D'Amore, R. Arcucci, L. Marcellino, and A. Murli. HPC computation issues of the incremental 3D variational data assimilation scheme in OceanVar software. *JNAIAM. J. Numer. Anal. Ind. Appl. Math.*, 7(3-4):91–105, 2012.

[32] L. D'Amore and R. Cacciapuoti. A note on domain decomposition approaches for solving 3D variational data assimilation models. *Ric. Mat.*, 68(2):679–691, 2019.

[33] L. D'Amore and R. Cacciapuoti. Model reduction in space and time for ab initio decomposition of 4d variational data assimilation problems. *Applied Numerical Mathematics*, 160:242–264, 2021.

[34] L. D'Amore and R. Cacciapuoti. Parallel framework for dynamic domain decomposition of data assimilation problems: a case study on kalman filter algorithm. *Computational and Mathematical Methods*, page e1145, 2021.

[35] L. D'Amore, R. Cacciapuoti, and V. Mele. A scalable kalman filter algorithm: Trustworthy analysis on constrained least square model. *Concurrency and Computation: Practice and Experience*, 33(4):e6022, 2021.

[36] L. D'Amore, R. Campagna, A. Galletti, L. Marcellino, and A. Murli. A smoothing spline that approximates Laplace transform functions only known on measurements on the real axis. *Inverse Problems*, 28(2):025007, 37, 2012.

[37] L. D'amore, R. Campagna, V. Mele, and A. Murli. Algorithm 946: Reliadiff—a c++ software package for real laplace transform inversion based on algorithmic differentiation. *ACM Transactions on Mathematical Software (TOMS)*, 40(4):1–20, 2014.

[38] L. D'Amore, G. Laccetti, D. Romano, G. Scotti, and A. Murli. Towards a parallel component in a gpu–cuda environment: a case study with the l-bfgs harwell routine. *International Journal of Computer Mathematics*, 92(1):59–76, 2015.

[39] L. D'Amore, V. Mele, D. Romano, and G. Laccetti. Multilevel algebraic approach for performance analysis of parallel algorithms. *Computing and Informatics*, 38(4):817–850, 2019.

[40] L. D'Amore and A. Murli. Regularization of a fourier series method for the laplace transform inversion with real data. *Inverse Problems*, 18(4):1185, 2002.

[41] M. D'Elia and A. Veneziani. A bayesian approach to data assimilation for the incompressible navier-stokes equations with applications to hemodynamics. *online open, https://www. linkedin. com/in/marta-delia-04551a44*, 2012.

[42] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996. Corrected reprint of the 1983 original.

[43] P. Diniz, S. Plimpton, B. Hendrickson, and R. Leland. Parallel algorithms for dynamically partitioning unstructured grids. Technical report, Sandia National Labs., Albuquerque, NM (United States), 1994.

[44] M. Dryja and O. B. Widlund. Some domain decomposition algorithms for elliptic problems. In *Iterative methods for large linear systems (Austin, TX, 1988)*, pages 273–291. Academic Press, Boston, MA, 1990.

[45] L. D'Amore, R. Arcucci, L. Carracciuolo, and A. Murli. Dd-oceanvar: A domain decomposition fully parallel data assimilation software for the mediterranean forecasting system. *Procedia Computer Science*, 18:1235–1244, 2013.

[46] L. D'Amore, R. Arcucci, L. Marcellino, and A. Murli. A parallel three-dimensional variational data assimilation scheme. In *AIP Conference Proceedings*, volume 1389, pages 1829–1831. American Institute of Physics, 2011.

[47] L. D'Amore and R. Cacciapuoti. Parallel dynamic domain decomposition in space-time for data assimilation problems. In *Journal of Physics: Conference Series*, volume 1828, page 012131. IOP Publishing, 2021.

[48] L. D'Amore, R. Cacciapuoti, and V. Mele. Ab-initio functional decomposition of kalman filter: A feasibility analysis on constrained least squares problems. In *International Conference on Parallel Processing and Applied Mathematics*, pages 75–92. Springer, 2019.

[49] L. D'Amore, R. Campagna, V. Mele, and A. Murli. Relative. an ansi c90 software package for the real laplace transform inversion. *Numerical Algorithms*, 63(1):187–211, 2013.

[50] L. D'Amore, V. Mele, G. Laccetti, and A. Murli. Mathematical approach to the performance evaluation of matrix multiply algorithm. In *Parallel Processing and Applied Mathematics*, pages 25–34. Springer, 2016.

[51] M. D'Elia, M. Perego, and A. Veneziani. A variational data assimilation procedure for the incompressible navier-stokes equations in hemodynamics. *Journal of Scientific Computing*, 52(2):340–359, 2012.

[52] I. Epicoco, S. Mocavero, F. Macchia, M. Vichi, T. Lovato, S. Masina, and G. Aloisio. Performance and results of the high-resolution biogeochemical model pelagos025 v1. 0 within nemo v3. 4. *Geoscientific Model Development*, 9(6):2115–2128, 2016.

[53] G. Evensen. The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

[54] M. FUJIMOTO and M. KAWAHARA. 34. domain decomposition for kalman filter method and its application to tidal flow at onjuku coast. In *Proceedings of 12th International Conference on Domain Decomposition Methods*. Citeseer, 2001.

[55] M. J. Gander and S. Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM Journal on Scientific Computing*, 29(2):556–578, 2007.

[56] W. Gander. Least squares with a quadratic constraint. *Numerische Mathematik*, 36(3):291–307, 1980.

[57] M. Ghil and P. Malanotte-Rizzoli. Data assimilation in meteorology and oceanography. *Advances in geophysics*, 33:141–266, 1991.

[58] S. Gürol, A. T. Weaver, A. M. Moore, A. Piacentini, H. G. Arango, and S. Gratton. B-preconditioned minimization algorithms for variational data assimilation with the dual formulation. *Quarterly Journal of the Royal Meteorological Society*, 140(679):539–556, 2014.

[59] P. Hähnel, J. Mareček, J. Monteil, and F. O'Donncha. Using deep learning to extend the range of air pollution monitoring and forecasting. *Journal of Computational Physics*, 408:109278, 2020.

[60] A. Hannachi, I. T. Jolliffe, and D. B. Stephenson. Empirical orthogonal functions and related techniques in atmospheric science: A review. *International Journal of Climatology: A Journal of the Royal Meteorological Society*, 27(9):1119–1152, 2007.

[61] Q. He, D. Barajas-Solano, G. Tartakovsky, and A. M. Tartakovsky. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. *Advances in Water Resources*, 141:103610, 2020.

[62] K. Hedstrom. Technical manual for a coupled sea-ice/ocean circulation model (version 5). u.s. dept. of the interior, bureau of ocean energy management, alaska ocs region. ocs study boem 2018-007 182 pp. 2018.

[63] L. Heidari, V. Gervais, M. Le Ravalec, and H. Wackernagel. History matching of reservoir models by ensemble kalman filtering: The state of the art and a sensitivity study. 2011.

[64] C. Homescu, L. R. Petzold, and R. Serban. Error estimation for reduced-order models of dynamical systems. *SIAM Journal on Numerical Analysis*, 43(4):1693–1714, 2005.

[65] G. Horton. A multi-level diffusion method for dynamic load balancing. *Parallel Computing*, 19(2):209–218, 1993.

[66] P. L. Houtekamer and H. L. Mitchell. A sequential ensemble kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 129(1):123–137, 2001.

[67] Y. Hu, R. J. Blake, and D. R. Emerson. An optimal migration algorithm for dynamic load balancing. *Concurrency: practice and experience*, 10(6):467–483, 1998.

[68] Y. Hu, R. J. Blake, and D. R. Emerson. An optimal migration algorithm for dynamic load balancing. *Concurrency: practice and experience*, 10(6):467–483, 1998.

[69] B. R. Hunt, E. J. Kostelich, and I. Szunyogh. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform kalman filter. *Physica D: Nonlinear Phenomena*, 230(1-2):112–126, 2007.

[70] T. Janjić, L. Nerger, A. Albertella, J. Schröter, and S. Skachko. On domain localization in ensemble-based kalman filter algorithms. *Monthly Weather Review*, 139(7):2046–2060, 2011.

[71] R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 82:35–45, 1960.

[72] E. Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge university press, 2003.

[73] E. Kalnay, B. Hunt, E. Ott, and I. Szunyogh. Ensemble forecasting and data assimilation: two problems with the same solution. *Predictability of weather and climate*, 157:180, 2006.

[74] V. Karavasilis, C. Nikou, and A. Likas. Visual tracking by adaptive kalman filtering and mean shift. In *Hellenic Conference on Artificial Intelligence*, pages 153–162. Springer, 2010.

[75] C. L. Keppenne. Data assimilation into a primitive-equation model with a parallel ensemble kalman filter. *Monthly Weather Review*, 128(6):1971–1981, 2000.

[76] C. L. Keppenne and M. M. Rienecker. Initial testing of a massively parallel ensemble kalman filter with the poseidon isopycnal ocean general circulation model. *Monthly weather review*, 130(12):2951–2965, 2002.

[77] U. A. Khan and J. M. Moura. Distributing the kalman filter for large-scale systems. *IEEE Transactions on Signal Processing*, 56(10):4919–4935, 2008.

[78] U. A. Khan and J. M. Moura. Distributing the kalman filter for large-scale systems. *IEEE Transactions on Signal Processing*, 56(10):4919–4935, 2008.

[79] Y. Kim and H. Bang. Introduction to kalman filter and its applications. *Introduction and Implementations of the Kalman Filter, F. Govaers, Ed. IntechOpen*, 2019.

[80] G. A. Kohring. Dynamic load balancing for parallelized particle simulations on mimd computers. *Parallel computing*, 21(4):683–693, 1995.

[81] R. J. LeVeque and R. J. Leveque. *Numerical methods for conservation laws*, volume 132. Springer, 1992.

[82] Q. Liao and K. Willcox. A domain decomposition approach for uncertainty analysis. *SIAM J. Sci. Comput.*, 37(1):A103–A133, 2015.

[83] J. Lions, Y. Maday, and G. Turinici. A "parareal" in time discretization of pde's. comptes rendus de l'acadmie des sciences-series i-mathematics 332, 661–668, 2001.

[84] J.-L. Lions, Y. Maday, and G. Turinici. Résolution d'EDP par un schéma en temps "pararéel". *C. R. Acad. Sci. Paris Sér. I Math.*, 332(7):661–668, 2001.

[85] P.-L. Lions. On the schwarz alternating method. iii: a variant for nonoverlapping subdomains. In *Third international symposium on domain decomposition methods for partial differential equations*, volume 6, pages 202–223. SIAM Philadelphia, PA, 1990.

[86] P.-L. Lions et al. On the schwarz alternating method. i. In *First international symposium on domain decomposition methods for partial differential equations*, volume 1, page 42. Paris, France, 1988.

[87] J. Liu and Z. Wang. Efficient time domain decomposition algorithms for parabolic PDE-constrained optimization problems. *Comput. Math. Appl.*, 75(6):2115–2133, 2018.

[88] Z. Long, Y. Lu, and B. Dong. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019.

[89] F. Lu and H. Zeng. Application of kalman filter model in the landslide deformation forecast. *Scientific reports*, 10(1):1–12, 2020.

[90] P. Lyster, J. Guo, T. Clune, and J. Larson. The computational complexity and parallel scalability of atmospheric data assimilation algorithms. *Journal of Atmospheric and Oceanic Technology*, 21(11):1689–1700, 2004.

[91] G. Meurant. Domain decomposition methods for partial differential equations on parallel computers. *The International Journal of Supercomputing Applications*, 2(4):5–12, 1988.

[92] T. Miyoshi. Computational challenges in big data assimilation with extreme-scale simulations, talk at bdec workshop. *Charleston, SC*, 2013.

[93] R. Montella, D. Di Luccio, P. Troiano, A. Riccio, A. Brizius, and I. Foster. Wacomm: A parallel water quality community model for pollutant transport and dispersion operational predictions. In *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pages 717–724. IEEE, 2016.

[94] C. Montzka, V. Pauwels, H.-J. H. Franssen, X. Han, and H. Vereecken. Multivariate and multiscale data assimilation in terrestrial systems: A review. *Sensors*, 12(12):16291–16333, 2012.

[95] A. M. Moore, H. G. Arango, G. Broquet, C. Edwards, M. Veneziani, B. Powell, D. Foley, J. D. Doyle, D. Costa, and P. Robinson. The regional ocean modeling system (roms) 4-dimensional variational data assimilation systems: part ii–performance and application to the california current system. *Progress in Oceanography*, 91(1):50–73, 2011.

[96] A. M. Moore, H. G. Arango, G. Broquet, C. Edwards, M. Veneziani, B. Powell, D. Foley, J. D. Doyle, D. Costa, and P. Robinson. The regional ocean modeling system (roms) 4-dimensional variational data assimilation systems: Part iii–observation impact and observation sensitivity in the california current system. *Progress in Oceanography*, 91(1):74–94, 2011.

[97] A. M. Moore, H. G. Arango, G. Broquet, B. S. Powell, A. T. Weaver, and J. Zavala-Garay. The regional ocean modeling system (roms) 4-dimensional variational data assimilation systems: Part i–system overview and formulation. *Progress in Oceanography*, 91(1):34–49, 2011.

[98] A. M. Moore, H. G. Arango, G. Broquet, B. S. Powell, A. T. Weaver, and J. Zavala-Garay. The regional ocean modeling system (roms) 4-dimensional variational data assimilation systems: Part i–system overview and formulation. *Progress in Oceanography*, 91(1):34–49, 2011.

[99] A. M. Moore, H. G. Arango, E. Di Lorenzo, B. D. Cornuelle, A. J. Miller, and D. J. Neilson. A comprehensive ocean prediction and analysis system based on the tangent linear and adjoint of a regional ocean model. *Ocean Modelling*, 7(1-2):227–258, 2004.

[100] A. Murli. *Matematica numerica: metodi, algoritmi e software*, volume 2. Liguori Editore Srl, 2013.

[101] A. Murli, L. D'Amore, G. Laccetti, F. Gregoretti, and G. Oliva. A multi-grained distributed implementation of the parallel block conjugate gradient algorithm. *Concurrency and Computation: Practice and Experience*, 22(15):2053–2072, 2010.

[102] I. M. Navon. Data assimilation for numerical weather prediction: a review. *Data assimilation for atmospheric, oceanic and hydrologic applications*, pages 21–65, 2009.

[103] L. Nerger. *Parallel filter algorithms for data assimilation in oceanography*. PhD thesis, Universität Bremen, 2004.

[104] L. Nerger, S. Danilov, W. Hiller, and J. Schröter. Using sea-level data to constrain a finite-element primitive-equation ocean model with a local seik filter. *Ocean Dynamics*, 56(5):634–649, 2006.

[105] L. Nerger, W. Hiller, and J. Schröter. The parallel data assimilation framework: experiences with kalman filtering. In *Use of High Performance in Meteorology, Proceedings of the 11th ECMWF Workshop, World Scientific, Singapore*, pages 63–86, 2005.

[106] N. K. Nichols. Mathematical concepts of data assimilation. In *Data assimilation*, pages 13–39. Springer, 2010.

[107] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[108] A. Quarteroni, R. Sacco, and F. Saleri. *Matematica numerica*. Springer Science & Business Media, 2010.

[109] F. Rabier and Z. Liu. Variational data assimilation: theory and overview. In *Proc. ECMWF Seminar on Recent Developments in Data Assimilation for Atmosphere and Ocean, Reading, UK, September 8–12*, pages 29–43, 2003.

[110] M. Rafiee, A. Tinka, J. Thai, and A. M. Bayen. Combined state-parameter estimation for shallow water equations. In *Proceedings of the 2011 American Control Conference*, pages 1333–1339. IEEE, 2011.

[111] D. Rozier, F. Birol, E. Cosme, P. Brasseur, J.-M. Brankart, and J. Verron. A reduced-order kalman filter for data assimilation in physical oceanography. *SIAM review*, 49(3):449–465, 2007.

[112] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[113] I. Schur. *Gesammelte Abhandlungen*. New York: Springer, 1973.

[114] H. A. Schwarz. Ueber einige abbildungsaufgaben. *J. reine angew. Math.*, 70:105–120, 1869.

[115] J. Sherman. Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix. *Annals of mathematical statistics*, 20(4):621, 1949.

[116] H. W. Sorenson. Least-squares estimation: from gauss to kalman. *IEEE spectrum*, 7(7):63–68, 1970.

[117] M. Stoll and A. Wathen. All-at-once solution of time-dependent stokes control. *Journal of Computational Physics*, 232(1):498–515, 2013.

[118] X. Tang, G. Falco, E. Falletti, and L. L. Presti. Complexity reduction of the kalman filter-based tracking loops in gnss receivers. *Gps Solutions*, 21(2):685–699, 2017.

[119] A. Teruzzi, P. Di Cerbo, G. Cossarini, E. Pascolo, and S. Salon. Parallel implementation of a data assimilation scheme for operational oceanography: The case of the medbfm model system. *Computers & Geosciences*, 124:103–114, 2019.

[120] S. Tirupathi, T. T. Tchrakian, S. Zhuk, and S. McKenna. Shock capturing data assimilation algorithm for 1d shallow water equations. *Advances in Water Resources*, 88:198–210, 2016.

[121] O.-P. Tossavainen, J. Percelay, A. Tinka, Q. Wu, and A. M. Bayen. Ensemble kalman filter based state estimation in 2d shallow water equations using lagrangian sensing and state augmentation. In *2008 47th IEEE Conference on Decision and Control*, pages 1783–1790. IEEE, 2008.

[122] S. Ulbrich. Generalized sqp methods with "parareal" time-domain decomposition for time-dependent pde-constrained optimization. In *Real-time PDE-constrained optimization*, pages 145–168. SIAM, 2007.

[123] C. K. Wikle and N. Cressie. A dimension-reduced approach to space-time kalman filtering. *Biometrika*, 86(4):815–829, 1999.

[124] Y. Wu, Y. Sui, and G. Wang. Vision-based real-time aerial object localization and tracking for uav sensing system. *IEEE Access*, 5:23969–23978, 2017.

[125] C.-Z. Xu and F. C. Lau. The generalized dimension exchange method for load balancing in k-ary n-cubes and variants. *Journal of parallel and distributed computing*, 24(1):72–85, 1995.

[126] C.-Z. Xu and F. C. M. Lau. Analysis of the generalized dimension exchange method for dynamic load balancing. *Journal of Parallel and Distributed Computing*, 16(4):385–393, 1992.

[127] L. Yong. A feasible interior point algorithm for a class of nonnegative least squares problems. In *2009 ETP International Conference on Future Computer and Communication*, pages 157–159. IEEE, 2009.

[128] Z. Zhang and J. C. Moore. *Mathematical and physical fundamentals of climate change*. Elsevier, 2014.

[129] G. Zhao, B. A. Bryan, and X. Song. Sensitivity and uncertainty analysis of the apsim-wheat model: Interactions between cultivar, environmental, and management parameters. *Ecological Modelling*, 279:1–11, 2014.