Journal of Scientific Computing (2022) 91:59
https://doi.org/10.1007/s10915-022-01826-7

A Scalable Space-Time Domain Decomposition Approach for Solving Large Scale Nonlinear Regularized Inverse Ill Posed Problems in 4D Variational Data Assimilation

Luisa D'Amore[1] · Emil Constantinescu[2] · Luisa Carracciuolo[3]

Received: 10 March 2020 / Revised: 29 November 2021 / Accepted: 7 March 2022

Abstract
We address the development of innovative algorithms designed to solve the strong-constraint Four Dimensional Variational Data Assimilation (4DVar DA) problems in large scale applications. We present a space-time decomposition approach which employs the whole domain decomposition, i.e. both along the spacial and temporal direction in the overlapping case, and the partitioning of both the solution and the operator. Starting from the global functional defined on the entire domain, we get to a sort of regularized local functionals on the set of sub domains providing the order reduction of both the predictive and the Data Assimilation models. The algorithm convergence is developed. Performance in terms of reduction of time complexity and algorithmic scalability is discussed on the Shallow Water Equations on the sphere. The number of state variables in the model, the number of observations in an assimilation cycle, as well as numerical parameters as the discretization step in time and in space domain are defined on the basis of discretization grid used by data available at repository *Ocean Synthesis/Reanalysis Directory* of Hamburg University.

Keywords Data assimilation · Space and time decomposition · Scalable algorithm · Inverse problems · Nonlinear least squares problems

✉ Luisa D'Amore
luisa.damore@unina.it

Emil Constantinescu
emcosta@mcs.anl.org

Luisa Carracciuolo
luisa.carracciuolo@cnr.it

1 University of Naples Federico II, Naples, Italy

2 Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA

3 Istituto per i Polimeri, Compositi e Biomateriali of the CNR (IPCB-CNR), Naples, Italy

1 Introduction and Motivations

Assimilation of observations into models is a well-established practice in the meteorological community. In view of the numbers (on the order of 10^7 or 10^8) of model variables and (on the order of 10^6) of observations in use for operational models, a variety of approaches have been proposed for reducing the complexity of an assimilation method to a computationally affordable version while retaining its advantages. Ensemble approaches and reduced order models are the most significant approximations. Other approaches take full advantage of existing Partial Differential Equations(PDEs)-based solvers, based on spatial Domain Decomposition (DD) methods, where the DD-solver is suitably modified to also handle the adjoint system. A different approach is the combination of DD-methods in space and DA, where spatial domain-decomposed uncertainty quantification approach performs DA at the local level by using Monte Carlo sampling [1, 2, 27, 54]. The Parallel Data Assimilation Framework [41] implements parallel ensemble-based Kalman Filters algorithms coupled within the PDE-model solver.

The above mentioned methods reduce the spacial dimensionality of the predictive model and the resulting reduced order model is then resolved in time via numerical integration, typically with the same time integrator and time step employed for the high-fidelity model leading to high time synchronization. In last decades, parallel-in-time methods have been investigated for reducing the temporal dimensionality of evolutionary problems. Since Nievergelt, in 1964, which proposed the first time decomposition algorithm for finding the parallel solutions of evolutionary ordinary differential equations, and Hackbusch in 1984, who noted that relaxation operators in multigrid can be employed on multiple time steps simultaneously, the methods of time-parallel time integration have been extensively expanded and several relevant works can be found in the literature. An extensive and updated literature list can be found at the website [47] collecting information about people, methods and software in the field of parallel-in-time integration methods. Among them, we mention the Parallel Full Approximation Scheme in Space and Time (PFASST), introduced in [14]. PFASST is based on a simultaneous approach reducing the optimization overhead by integrating the PDE-based model directly into the optimization process, thus solving the PDE, the adjoint equations and the optimization problem simultaneously. Recently, a non-intrusive framework for integrating existing unsteady partial differential equation (PDE) solvers into a parallel-in-time simultaneous optimization algorithm, using PFASST, is provided in [22]. Finally, we cite the parallel PDE solvers based on Schwarz preconditioner in space-time, proposed in [19, 29, 53].

We propose the design of an innovative mathematical model, and the development and analysis of the related numerical algorithms, based on the simultaneous introduction of space-time decomposition in the overlapping case on the PDEs equations governing the physical model and on the DA model. The core of our approach is that the DA model acts as coarse/predictor operator solving the local PDE model, by providing the background values as initial conditions of the local PDE models. Moreover, in contrast to the other decomposition in time approaches, in our approach local solvers (i.e. both the coarse and the fine solvers) run concurrently from the beginning. As a consequence, the resulting algorithm only requires exchange of boundary conditions between adjacent sub-domains. It is worth to mention that the proposed method belongs to the so-called reduced-space optimization techniques, in contrast to the full-space approaches such as the PFASST method, reducing the runtime of the forward and the backward integration time loops. As a consequence, we could combine the proposed approach with the PFASST algorithm. Indeed, PFASST could be concurrently employed as local solver of each reduced-space PDE-constrained optimization subproblem, exposing even more temporal parallelism.

The article will describe the more general DA problem setup for describing the domain decomposition approach, then it will focus on parallel algorithm solving the reduced order model analysing the impact of the space and time decomposition on the performance of the algorithm and finally we provide the analysis of the algorithm's scalability. Results presented here are should be intended as the starting point for the software development to make decisions about computer architecture, future estimates of the problem size (e.g., the resolution of the model and the number of observations to be assimilated) and the performance and parallel scalability of the algorithms.

In conclusion, specific contributions of this work include:

– a novel decomposition approach in space-time leading to a reduced order model of the coupled PDE-based 4DVar DA problem.
– strategies for computing the 'kernels' of the resulting Regularized Nonlinear Least Square computational problem.
– a priori performance analysis that enables a suitable implementation of the algorithm in advanced computing environments.

The article is organized as follows. Sect. 2 gives a brief introduction to the Data Assimilation framework, where we follow the discretize-then-optimize approach. Main result is the 4DVar functional decomposition, which is given in Sect. 3. In Sect. 4 we review the whole parallel algorithm while its performance analysis is discussed in Sect. 5 on the Shallow Water Equations on the sphere. The number of state variables in the model, the number of observations in an assimilation cycle, as well as numerical parameters as the discretization step in time and in space domain are defined on the basis of discretization grid used by data available, at repository *Ocean Synthesis/Reanalysis Directory* of Hamburg University (see [15]). Scalability prediction of the case study based on the Shallow Water Equations is performed in Sect. 6. Finally, conclusions are provided in Sect. 7.

2 The Data Assimilation framework

We start with the more general DA problem setup and then, for simplicity, for describing the domain decomposition approach, we will consider a more convenient setup.

Let $\mathcal{M}^{\Delta\times\Omega}$ denote a forecast model described by nonlinear Navier-Stokes equations[1] where $\Delta \subset \Re$ is the time interval, and $\Omega \subset \Re^N$ is the spatial domain. If $t \in \Delta$ denotes the time variable and $x \in \Omega$ the spatial variable, let[2]

$$u^b(t,x) : \Delta \times \Omega \mapsto \Re$$

be the function, which we assume belonging to the Hilbert space $\mathcal{K}(\Delta\times\Omega)$ equipped with the standard euclidean norm, representing the solution of $\mathcal{M}^{\Delta\times\Omega}$. Following [8], we assume that $\mathcal{M}^{\Delta\times\Omega}$ is symbolically described as the following initial value problem:

$$\begin{cases} u^b(t,x) = \mathcal{M}^{\Delta\times\Omega}[u^b(t_0,x)], & \forall\,(t,x) \in \Delta\times\Omega, \\ u^b(t_0,x) = u_0^b(x), & t_0 \in \Delta\,,\ x \in \Omega\,. \end{cases} \tag{1}$$

[1] Such as the primitive equations of oceanic circulation models which are based on Boussinesq, hydrostatic momentum, mass balances, material tracer conservation, seawater's equation of state, and parameterized subgrid-scale transports [32–35, 42].

[2] Although typical prognostic variables are temperature, salinity, horizontal velocity, and sea surface displacement, here, for simplicity of notations, we assume that $u^b(t,x) \in \Re$.

Function $u^b(t,x)$ is said background state in $\Delta \times \Omega$. Function $u_0^b(x)$ is the initial condition of $\mathcal{M}^{\Delta\times\Omega}$ and this is the value of the background state in $t_0 \times \Omega$. Let:

$$v(\tau, y) = \mathcal{H}(u(t,x)), \quad (t,x) \in \Delta \times \Omega, \quad (\tau, y) \in \Delta' \times \Omega' \tag{2}$$

where $\Delta' \subset \Delta$ be the observations time interval and $\Omega' \subset \Re^{nobs}$, with $\Omega' \subset \Omega$ be the observation spatial domain. Finally,

$$\mathcal{H} : \mathcal{K}(\Delta \times \Omega) \mapsto \mathcal{K}(\Delta' \times \Omega')$$

denotes the observations mapping, where $\mathcal{H}$ is a nonlinear operator which includes transformations and grid interpolations. According to the practical applications of model-based assimilation of observations, we will use the following definition of a Data Assimilation (DA) problem associated to $\mathcal{M}^{\Delta\times\Omega}$.

Definition 1 (The DA problem set-up) Let [3]

- $\{t_k\}_{k=0,M-1}$, where $t_k = t_0 + k\Delta t$, be a discretization of Δ, such that $\Delta_M := [t_0, t_{M-1}] \subseteq \Delta$.
- $D_K(\Omega) := \{x_j\}_{j=1,K} \in \Re^K$, be a discretization of Ω, such that $D_K(\Omega) \subseteq \Omega$.
- $\Delta_M \times \Omega_K = \{\mathbf{z}_{ji} := (t_j, x_i)\}_{i=1,K; j=1,M}$;
- $\mathbf{u}_0^b := \{u_0^j\}_{j=1,K}^b \equiv \{u(t_0, x_j)^b\}_{j=1,K} \in \Re^K$ be the discretization of initial value in (1);
- $\mathbf{u}_k^b := \{u^b(t_k, x_j)\}_{j=1,K} \in \Re^K$ be the numerical solution of (1) at t_k;
- $\mathbf{u}^b = \{\mathbf{u}_k^b\}_{k=0,M-1}$;
- $nobs << K$;
- $\Delta'_M = [\tau_0, \tau_{M-1}] \subseteq \Delta_M$;
- $D'_{nobs}(\Omega') := \{x_j\}_{j=1,nobs} \in \Re^{nobs}$, be a discretization of Ω', such that
$$D_{nobs}(\Omega') \subseteq \Omega' \quad .$$
- $\mathbf{v}_k := \{v(\tau_k, x_j)\}_{j=1,nobs} \in \Re^{nobs}$ be the values of the observations on x_j at τ_k;
- $\mathbf{v} = \{\mathbf{v}_k\}_{k=0,M-1} \in \Re^K$;
- $\{\mathbf{H}^{(k)}\}_{k=0,M-1}$, the Tangent Linear Model (TLM) of $\mathcal{H}(u(t_k, x))$ at time t_k;
- $\mathbf{M}^{\Delta_M\times\Omega_K}$ be a discretization of $\mathcal{M}^{\Delta\times\Omega}$.
- $\mathbf{M}^{0,M-1}$, is the TLM of $\mathcal{M}^{\Delta\times\Omega}$, i.e. it is the first order linearization [4] of $\mathcal{M}^{\Delta\times\Omega}$ in $\Delta_M \times \Omega_K$ [24];
- $\mathbf{M}^T$is the Adjoint Model (ADM)[5] of $\mathbf{M}^{0,M-1}$ [20][6].

♠

[3] In the following and throughout the paper, for simplicity, we use the notation $j = 1, K$ to indicate $j = 1, \ldots, K$.

[4] For nonlinear Navier Stokes equations which we are considering here, first order linearization of $\mathcal{M}^{\Delta\times\Omega}$ is formed by truncating at the first order the Taylor series expansion of $\mathbf{M}^{\Delta_M\times\Omega_K}$ about $\mathbf{u}^b$ over the interval Δ_M. In some cases this approach performs equally to the approach based on first linearization of the continuous model $\mathcal{M}^{\Delta\times\Omega}$ and then discretization [24].

[5] Let $\mathbf{A} : \mathbf{x} \to \mathbf{y} = \mathbf{A}\mathbf{x}$ be a linear operator on $\Re^N$ equipped with the standard euclidean norm. The operator $\mathbf{A}^T : \mathbf{y} \to \mathbf{x} = \mathbf{A}^T\mathbf{y}$ such that

$$< \mathbf{y}, \mathbf{A}\mathbf{x} > = < \mathbf{A}^T\mathbf{y}, \mathbf{x} >, \quad \forall \mathbf{x}, \forall \mathbf{y} \tag{3}$$

where $< \cdot, \cdot >$ denotes the scalar product in $\Re^N$, is the adjoint of $\mathbf{A}$.

[6] If $\mathbf{M}^{i-1,i}$ is the TLM of $\mathcal{M}^{\Delta\times\Omega}$, in $[t_{i-1}, t_i] \times \Omega_K$, then it holds that:

$$(\mathbf{M}^{0,M-1})^T = (\mathbf{M}^{0,1} \cdot \mathbf{M}^{1,2} \cdots \mathbf{M}^{M-2,M-1})^T = (\mathbf{M}^{M-2,M-1})^T \cdots (\mathbf{M}^{1,2})^T (\mathbf{M}^{0,1})^T. \tag{4}$$

The aim of DA is seeking to produce the optimal combination of the background and observations throughout the assimilation window Δ'_M i.e. to find an optimal tradeoff between the estimate of the system state $\mathbf{u}^b$ and $\mathbf{v}$. The best estimate that optimally fuses all these information is called the analysis, and it is denoted as $\mathbf{u}^{DA}$. It is then used as an initial condition for the next forecast.

Definition 2 (The 4DVar DA problem: a regularized nonlinear least square problem (RNL-LS)) Given the DA problem set-up, the 4DVar DA problem consists in computing the vector $\mathbf{u}^{DA} \in \Re^K$ such that

$$\mathbf{u}^{DA} = \arg\min_{\mathbf{u}\in\Re^K} J(\mathbf{u}) \tag{5}$$

with

$$J(\mathbf{u}) = \|\mathbf{u} - \mathbf{u}_0^b\|^2_{\mathbf{B}^{-1}} + \lambda \sum_{k=0}^{M-1} \|\mathbf{H}^{(k)}(\mathbf{M}^{\Delta_M \times \Omega_K}(\mathbf{u})) - \mathbf{v}_k\|^2_{\mathbf{R}_k^{-1}} \tag{6}$$

where $\lambda > 0$ is the regularization parameter, $\mathbf{B}$ and $\mathbf{R}_k$ ($\forall k = 0, M-1$) are the covariance matrices of the errors on the background and the observations respectively, while $\|\cdot\|_{\mathbf{B}^{-1}}$ and $\|\cdot\|_{\mathbf{R}_k^{-1}}$ denote the weighted euclidean norm.

♠

The first term of the (6) quantifies the departure of the solution $\mathbf{u}^{DA}$ from the background state $\mathbf{u}^b$. The second term measures the sum of the mismatches between the new trajectory and observations $\mathbf{v}_k$, for each time t_k in the assimilation window. The weighting matrices $\mathbf{B}$ and $\mathbf{R}_k$ need to be predefined, and their quality influences the accuracy of the resulting analysis [3].

As K exceeds 10^6 in general this problem can be considered as a Large Scale Nonlinear Lest Square problem. We provide a mathematical formulation of a domain decomposition approach, which starts from decomposition of the whole domain $\Delta \times \Omega$, namely both the spatial and temporal domain; it uses a partitioning of the solution and a modified functional describing the RNL-LS problem on the subdomain of the decomposition. Solution continuity equations across interval boundaries are added as constraints of the assimilation functional. We will first introduce domain decomposition of $\Delta \times \Omega$ then, restriction and extension operators will be defined on functions given on $\Delta \times \Omega$. These definitions will subsequently generalized to $\Delta_M \times \Omega_K$.

3 The Space-Time Decomposition of the Continuous 4DVar DA Model

In this section we give a precise mathematical setting for space and function decomposition then we state some notations used later. In particular, we first introduce the function and domain decomposition, then by using restriction and extension operators, we associate to the domain decomposition a functional decomposition. So, we prove the following result: the minimum of the global functional, defined on the entire domain, can be obtained by collecting the minimum of each local functional.

For simplicity we assume that the spacial and temporal domains of the observations are the same of the background state, i.e. $\Delta' = \Delta$ and $\Omega' = \Omega$; furthermore we assume that $t_k = \tau_k$.

Definition 3 (Domain Decomposition) Let $P \in \mathbf{N}$ and $Q \in \mathbf{N}$ be fixed. The set of bounded Lipschitz domains Ω_i, overlapping sub-domains of Ω:

$$DD(\Omega) = \{\Omega_i\}_{i=1,P} \tag{7}$$

is called a decomposition of Ω, if

$$\bigcup_{i=1}^{P} \Omega_i = \Omega \tag{8}$$

with

$$\Omega_{jh} := \Omega_j \cap \Omega_h \neq \emptyset$$

when two subdomains are adjacent. Similarly, the set of overlapping sub-domains of Δ:

$$DD(\Delta) = \left\{\Delta_j\right\}_{j=1,Q} \tag{9}$$

is a decomposition of Δ, if

$$\bigcup_{j=1}^{Q} \Delta_j = \Delta \tag{10}$$

with

$$\Delta_{ik} := \Delta_i \cap \Delta_k =\neq \emptyset$$

when two subdomains are adjacent. We call domain decomposition of $\Delta \times \Omega$ and we denote it as $DD(\Delta \times \Omega)$, the set of $P \times Q$ overlapping subdomains of $\Delta \times \Omega$:

$$DD(\Delta \times \Omega) = \left\{\Delta_j \times \Omega_i\right\}_{j=1,Q;\ i=1,P}\ . \tag{11}$$

♠

From (11) it follows that

$$\Delta \times \Omega = \cup \Delta_j \times \cup \Omega_i = \cup(\Delta_j \times \Omega_i).$$

Associated to the decomposition (11) we define the Restriction Operator of functions belonging to $\mathcal{K}(\Delta \times \Omega)$:

Definition 4 (Restriction of a function) Let

$$RO_{ji} : f \in \mathcal{K}(\Delta \times \Omega) \mapsto RO_{ji}[f] \in \mathcal{K}(\Delta_j \times \Omega_i)$$

be the Restriction Operator (RO) of f in $DD(\Delta \times \Omega)$ as in (11) be such that:

$$RO_{ji}[f(t,x)] \equiv \begin{cases} f(t,x), & \forall\ (t,x) \in \Delta_j \times \Omega_i \\ \frac{1}{2} f(t,x), \forall\, (t,x)\, s.t.\, x \in \Omega_i, & \exists\, \bar{k} \neq j : t \in \Delta_j \cap \Delta_{\bar{k}}, \\ \frac{1}{2} f(t,x), \forall\, (t,x)\, t \in \Delta_j, & \exists\, \bar{h} \neq i : x \in \Omega_i \cap \Omega_{\bar{h}}, \\ \frac{1}{4} f(t,x), \exists\, (\bar{h},\bar{k}) \neq (j,i) : (t,x) \in (\Delta_j \cup \Delta_{\bar{h}}) \times (\Omega_i \cup \Omega_{\bar{k}}), \end{cases}$$

We pose:

$$f_{ji}^{RO}(t,x) \equiv RO_{ji}[f(t,x)].$$

♠

For simplicity, if $i \equiv j$, we denote $RO_{ii} = RO_i$.

In line with this, given a set of $Q \times P$ functions g_{ji}, $j = 1, Q, i = 1, P$ each belonging to $\mathcal{K}(\Delta_j \times \Omega_i)$, we define the Extension Operator of g_{ji}:

Definition 5 (*Extension of a function) Let*

$$EO : g_{ji} \in \mathcal{K}(\Delta_j \times \Omega_i) \mapsto EO[g_{ji}] \in \mathcal{K}(\Delta \times \Omega)$$

be the Extension Operator (EO) of g_{ji} in $DD(\Delta \times \Omega)$ as in (11)be such that:

$$EO[(g_{ji}(t,x)] = \begin{cases} g_{ji}(t,x) \ \forall \ (t,x) \in \Delta_j \times \Omega_i \\ 0 \qquad\qquad elsewhere \end{cases}$$

We pose:

$$g_{ji}^{EO}(t,x) \equiv EO[g_{ji}(t,x)].$$

♠

For any function $u \in \mathcal{K}(\Delta \times \Omega)$, associated to the decomposition (8), it holds that

$$u(t,x) = \sum_{i=1,P;j=1,Q} EO\left[u_{ji}^{RO}(t,x)\right]. \tag{12}$$

Given $P \times Q$ functions $u_{ji}(t,x) \in \mathcal{K}(\Delta_i \times \Omega_j)$, the summation

$$\sum_{i=1,P;j=1,Q} u_{ji}^{EO}(t,x) \tag{13}$$

defines a function $u \in \mathcal{K}(\Delta \times \Omega)$ such that:

$$RO_{ji}[u(t,x)] = RO_{ji}\left[\sum_{i=1,P;j=1,Q} u_{ji}^{EO}(t,x)\right] = u_{ji}(t,x). \tag{14}$$

Main outcome of this framework is the definition of the operator RO_{ji} for the 4DVar functional defined in (6). This definition originates from the definition of the restriction operator of $\mathcal{M}^{\Delta\times\Omega}$ in (1), given as follows.

Definition 6 (*Reduction of* $\mathcal{M}^{\Delta\times\Omega}$) If $\mathcal{M}^{\Delta\times\Omega}$ is defined in (1), we introduce the model $\mathcal{M}^{\Delta_j\times\Omega_i}$ to be the Reduction of $\mathcal{M}^{\Delta\times\Omega}$:

$$RO_{ji} : \mathcal{M}^{\Delta\times\Omega}(t,x)[u(t_0,x)] \mapsto RO_{ji}[\mathcal{M}^{\Delta\times\Omega}[u(t_0,x)]]$$

defined in $\Delta_j \times \Omega_i$, such that:

$$\begin{cases} u^R(t,x) = \mathcal{M}^{\Delta_j\times\Omega_i}[u^b(t_j,x)] \ \forall \ (t,x) \in \Delta_j \times \Omega_i \\ u^b(t_j,x) = u_j^b(x) \qquad\qquad t_j \in \Delta_j\,, \quad x \in \Omega_i \end{cases} \tag{15}$$

♠

It is worth noting that initial condition $u_j^b(x)$ is the value in t_j of the solution of $\mathcal{M}^{\Delta\times\Omega}[u(t_0,x)]$ defined in (1).

3.1 Space-Time Decomposition of the Discrete Model

Let assume that $\Delta_M \times \Omega_K$ can be decomposed into a sequence of $P \times Q$ overlapping subdomains $\Delta_j \times \Omega_i$ such that

$$\Delta_M \times \Omega_K = \bigcup_{i=1,P;\; j=1,Q} \Delta_j \times \Omega_i$$

where $\Omega_i \subset \Re^{r_i}$ with $r_i \leq K$ and $\Delta_j \subset \Re^{s_j}$ with $s_j \leq M$. Finally, let us assume that

$$\Delta_j := [t_j, t_{j+s_j}] \quad .$$

Hence

$$\mathbf{u}_{ji} := RO_{ji}(\mathbf{u}) \equiv \mathbf{u}^{\mathbf{RO_{ji}}} \equiv (u(z_{ji}))_{z_{ji} \in \Delta_j \times \Omega_i}, \quad \mathbf{u}_{ji} \in \Re^{s_j \times r_i}.$$

In this respect, we define the Extension Operator (EO) also. If $\mathbf{u} = (u(z_{ji}))_{z_{ji} \in \Delta_j \times \Omega_i}$, it is

$$EO(\mathbf{u}) = \begin{cases} u(z_{kh}) \; z_{kh} \in \Delta_k \times \Omega_h \\ 0 \qquad elsewhere \end{cases}$$

and $EO(\mathbf{u}) \equiv \mathbf{u}^{\mathbf{EO}} \in \Re^{M \times K}$.

Definition 7 (*Restriction of the Covariance Matrix*) Let $\mathbf{C}(\mathbf{w}) \in \Re^{K \times K}$ be the covariance matrix of a random vector $\mathbf{w} = (w_1, w_2, \ldots, w_K) \in \Re^K$, that is coefficient $c_{i,j}$ of $\mathbf{C}$ is $c_{i,j} = \sigma_{ij} \equiv Cov(w_i, w_j)$. Let $s < K$, we define the Restriction Operator RO_{st} onto $\mathbf{C}(\mathbf{w})$ as follows:

$$RO_{st} : \mathbf{C}(\mathbf{w}) \in \Re^{K \times K} \mapsto RO_{st}[\mathbf{C}(\mathbf{w})] \overbrace{=}^{def} \mathbf{C}(\mathbf{w}^{\mathbf{RO_{st}}}) \in \Re^{s \times s}$$

i.e., it is the covariance matrix defined on $\mathbf{w}^{\mathbf{RO_{st}}}$.

♠

Hereafter, we refer to $\mathbf{C}(\mathbf{w}^{\mathbf{RO_s}})$ using the notation $\mathbf{C_{st}}$.

Definition 8 (*Restriction of the operator* $\mathbf{H}^{(k)}$) We define the Restriction Operator RO_{ji} of $\mathbf{H}^{(k)}$ in $DD(\Delta \times \Omega)$ as in (11) as the TLM at time t_k of the restriction of $\mathcal{H}$ on $\Delta_j \times \Omega_i$.

♠

Definition 9 (*Restriction of* $\mathbf{M}^{\Delta_M \times \Omega_K}$) We let $\mathbf{M}^{\Delta_j \times \Omega_i}$ be the Restriction Operator RO_{ji} of $\mathbf{M}^{\Delta_M \times \Omega_K}$ in $\Delta_j \times \Omega_i$ where:

$$RO_{ji} : \mathbf{M}^{\Delta_M \times \Omega_K}(\mathbf{u}_0^b) \mapsto \mathbf{M}^{\Delta_j \times \Omega_i}(\mathbf{u}_0^b) = \mathbf{u}_{ji}^b$$

defined in $\Delta_j \times \Omega_i$.

♠

Definition 10 (*Restriction of the operator* $\mathbf{M}^{0,M-1}$) We define $\mathbf{M}_i^{j,j+1}$ to be the Restriction Operator RO_{ji} of $\mathbf{M}^{0,M-1}$ in $DD(\Delta \times \Omega)$, as in (11). It is the TLM of the Restriction of $\mathbf{M}^{\Delta_M \times \Omega_K}$ on $\Delta_j \times \Omega_i$.

♠

Finally, we are now able to give the following definition.

Definition 11 (*Restriction of 4DVar DA*) Let

$$RO_{ji}[J] : \mathbf{u}_{ji} \mapsto RO_{ji}[J](\mathbf{u}_{ji})$$

denotes the Restriction operator of the 4DVar DA functional defined in (6). It is defined as

$$\begin{aligned} RO_{ji}[J](\mathbf{u}_{ji}) = \| & \underbrace{RO_{ji}(\mathbf{u})}_{\mathbf{u}_{ji}} - \underbrace{RO_{ji}[\mathbf{M}^{\Delta_M \times \Omega_K}(\mathbf{u}_0^b)]}_{\mathbf{u}_{ji}^b} \|_{(\mathbf{B}^{-1})_{ji}} \\ & + \lambda \textstyle\sum_{k:t_k \in \Delta_j} \| \underbrace{RO_{ji}[\mathbf{H}^{(k)}] RO_{ji}[\mathbf{M}^{\Delta_M \times \Omega_K}(\mathbf{u})}_{(\mathbf{H}^{(k)})_{ji} RO_{ji}[(\mathbf{M}^{\Delta_M \times \Omega_K})(\mathbf{u}_{ji})]} - \underbrace{RO_{ji}[\mathbf{v_k}]}_{\mathbf{v}_{ji}} \|^2_{(\mathbf{R}^{-1})_{ji}} . \end{aligned} \tag{16}$$

♠

Local 4DVar DA functional $J_{ji}(\mathbf{u}_{ji})$ in (16) becomes:

$$J_{ji}(\mathbf{u}_{ji}) = \underbrace{\|\mathbf{u}_{ji} - \mathbf{u}_{ji}^b\|_{(\mathbf{B}^{-1})_{ji}}}_{local\ state\ trajectory} + \tag{17a}$$

$$\lambda \sum_{k:t_k \in \Delta_j} \underbrace{\|(\mathbf{H}^{(k)})_{ji}[\mathbf{M}_i^{k,k+1}(\mathbf{u}_{ji})] - \mathbf{v}_{ji}\|_{(\mathbf{R}^{-1})_{ji}}}_{local\ observations} . \tag{17b}$$

This means that the approach we are following is to firstly decompose the 4DVar functional J then to locally linearize and solve each local functional J_{ji}. For simplicity of notations we let

$$RO_{ji}[J] \equiv J_{\Delta_j \times \Omega_i} .$$

We observe that $RO_{ji}[J](\mathbf{u}_{ji})$ is made of a first term which quantifies the departure of the state $\mathbf{u}_{ji}$ from the background state $\mathbf{u}_{ji}^b$ at time t_j and space x_i. The second term measures the mismatch between the state $\mathbf{u}_{ji}$ and the observation $\mathbf{v}_{ji}$.

Definition 12 (*Extension of 4DVar DA*) Given $DD(\Delta \times \Omega)$ as in (11) let

$$EO[J] : J_{\Delta_j \times \Omega_i} \mapsto J^{EO}_{\Delta_j \times \Omega_i} ,$$

be the Extension Operator (EO) of the 4DVar functional defined in (6), where

$$EO[J](J_{\Delta_j \times \Omega_i}) = \begin{cases} J_{\Delta_j \times \Omega_i} & (t, x) \in \Delta_j \times \Omega_i \\ 0 & elsewhere \end{cases} \tag{18}$$

♠

From (19), it follows the decomposition of J as follows.

$$J \equiv \sum_{i=1,P;j=1,Q} J^{EO}_{\Delta_j \times \Omega_i} . \tag{19}$$

Main outcome of (19) is the capability of defining local 4D Var problems which contribute all together to the 4DVar problem as detailed in the following.

3.2 Local 4DVar DA Problem: The Local RNL-LS Problem

Starting from Local 4DVar functional in (17) which is obtained applying the Restriction Operator to the 4DVar functional defined in (6), we add a *local* constraint to such restriction. This is a sort of regularization of the local 4DVar functional introduced in order to enforce the continuity of each solution of the local problem onto the overlap region between adjacent subdomains. Local constraint consists of the overlapping operator $\mathcal{O}_{(jh)(ik)}$ defined as

$$\mathcal{O}_{(jh)(ik)} := \mathcal{O}_{jh} \circ \mathcal{O}_{ik} \tag{20}$$

where the symbol $\circ$ denotes the operators composition. Each operator in the (20) tackles the overlapping of the solution in the spatial dimension and in the temporal dimension, respectively. More precisely, for $j = 1 \dots Q$; $i = 1 \dots P$, operator $\mathcal{O}_{(jh)(ik)}$ represents the overlap of temporal subdomains j and h and spatial subdomains i and k, where h and k are given as in Definition 4 and

$$\mathcal{O}_{ik} : \mathbf{u}_{ji} \in \Delta_j \times \Omega_i \mapsto \mathbf{u}_{(j)(ik)} \in \Delta_j \times (\Omega_i \cap \Omega_k) \tag{21}$$

and

$$\mathcal{O}_{jh} : \mathbf{u}_{(j)(ik)}) \mapsto \mathbf{u}_{(jh)(ik)} \in (\Delta_j \cap \Delta_h) \times (\Omega_i \cap \Omega_k) \tag{22}$$

Remark 1 We observe that, in the overlapping domain $\Delta_{jh} \times \Omega_{ik}$ we get two vectors: $\mathbf{u}_{(jh)(ik)}$, which is obtained as the restriction of $\mathbf{u}_{(ji)} = argmin J_{ji}(\mathbf{u}_{ji})$ to that region, and $\mathbf{u}_{(hj)(ki)}$, which is the restriction of $\mathbf{u}_{(hk)} = argmin J_{hk}(\mathbf{u}_{hk})$ to the same region. The order of the indexes plays a significant rule from the computing perspectives.

From (20), three cases derive

1. decomposition in space, i.e. $j = Q = 1$ and $P > 1$. Here we get $j = Q = 1$, i.e. time interval is not decomposed, and $P > 1$, i.e. the spatial domain Ω is decomposed according to the domain decomposition in (11). The overlapping operator is defined as in (21). In particular we assume that

$$\mathcal{O}_{ik}(\mathbf{u}_{ji}) := \| \underbrace{RO_{ji}(\mathbf{u}_{jk})}_{\mathbf{u}_{j(ki)}} - \underbrace{RO_{jk}(\mathbf{u}_{ji})}_{\mathbf{u}_{(j)(ik)}} \|_{(\mathbf{B}^{-1})_{ik}}$$

2. decomposition in time, i.e. $Q > 1$ and $P = 1$. We get that $i = P = 1$, i.e. the spatial domain is not decomposed, and $Q > 1$, i.e. the time interval is decomposed according to the domain decomposition in (11). The overlapping operator is defined as in (22). In particular we assume that

$$\mathcal{O}_{jh}(\mathbf{u}_{ji}) := \| \underbrace{RO_{ji}(\mathbf{u}_{hi})}_{\mathbf{u}_{(hj)i}} - \underbrace{RO_{hi}(\mathbf{u}_{ji})}_{\mathbf{u}_{(jh)i}} \|_{(\mathbf{B}^{-1})_{jh}}$$

3. decomposition in space-time, i.e. $Q > 1$ and $P > 1$. We assume that $Q > 1$ and $P > 1$ i.e. both the time interval and the spatial domain are decomposed according to the domain decomposition in (11). The overlapping operator is defined as in (20). In particular we assume that

$$\mathcal{O}_{(jh)(ik)}(\mathbf{u}_{ji}) := \|\mathbf{u}_{(hj)(ki)} - \underbrace{RO_{hi}(RO_{jk}(\mathbf{u}_{ji}))}_{\mathbf{u}_{(jh)(ik)}} \|_{(\mathbf{B}^{-1})_{(jh)(ik)}}$$

We now give the new definition of the local 4DVar DA functional

Definition 13 (*Local 4DVar DA*) Given $DD(\Delta \times \Omega)$ as in (11), let:

$$J_{ji}(\mathbf{u}_{ji}) = RO_{ji}[J](\mathbf{u}_{ji}) + \mu_{ji}\ O_{(jh)(ik)}(\mathbf{u}_{ji}) \tag{23}$$

where $RO_{ji}[J](\mathbf{u}_{ji})$ is given in (16), $\mathcal{O}_{(jh)(ik)}$ will be suitably defined on $\Delta_{jh} \times \Omega_{ik}$, be the local 4DVar functional. Parameter μ_{ji} is a regularization parameter. Finally, let

$$\mathbf{u}_{ji}^{DA} = \arg\min_{\mathbf{u}_{ji}} J_{ji}(\mathbf{u}_{ji}) \tag{24}$$

be the global minimum of J_{ji} in $\Delta_j \times \Omega_i$.

♠

More precisely, the local 4DVar DA functional $J_{ji}(\mathbf{u}_{ji})$ in (23) becomes:

$$J_{ji}(\mathbf{u}_{ji}) = \underbrace{\|\mathbf{u}_{ji} - \mathbf{u}_{ji}^b\|_{(\mathbf{B}^{-1})_{ji}}}_{local\ state\ trajectory} + \tag{25a}$$

$$\lambda \sum_{k:t_k \in \Delta_j} \underbrace{\|(\mathbf{H}^{(k)})_{ji}[\mathbf{M}_i^{k,k+1}(\mathbf{u}_{ji})] - \mathbf{v}_{ji}\|_{(\mathbf{R}^{-1})_{ji}}}_{local\ observations} + \tag{25b}$$

$$\mu\, \underbrace{\|\mathbf{u}_{(hj)(ki)} - \mathbf{u}_{(jh)(ik)}\|_{(\mathbf{B}^{-1})_{(jk)(ih)}}}_{overlap} \tag{25c}$$

where the three terms contributing to the definition of the local DA functional clearly come out. We note that in (17) the operator $\mathbf{M}_i^{k,k+1}$ which is defined in (4) replaces $\mathcal{M}^{\Delta_j \times \Omega_i}$.

Finally, we have to guarantee that the global minimum of the operator J, can be searched among the global minima of *local* functionals.

3.3 Local 4DVar DA Minimization

Let

$$\widetilde{\mathbf{u}}_{\mathbf{ji}} := (\mathbf{u}_{ji}^{DA})^{EO} \in \Re^{M \times K}, \quad \forall\, j = 1, Q; i = 1, P \tag{26}$$

where $\mathbf{u}_{ji}^{DA}$ is defined in (24), be (the extension of) the minimum of the (global) minimums of the *local* functionals J_{ji} as in (24). Let

$$\widetilde{\mathbf{u}}^{\mathbf{DA}} := \arg \min_{j=1,Q;i=1,P} \left\{ J\left(\widetilde{\mathbf{u}}_{\mathbf{ji}}\right) \right\} \tag{27}$$

be its minimum.

Theorem 1 *If $DD(\Delta \times \Omega)$ is a decomposition of $\Delta \times \Omega$ as defined in (11). It follows that:*

$$J(EO(\mathbf{u}^{\mathbf{DA}})) \le J(\widetilde{\mathbf{u}}^{\mathbf{DA}}), \tag{28}$$

with $\mathbf{u}^{DA}$ defined in (5). Moreover, the equality in (28) holds if J is convex.

Proof Let $\mathbf{u}_{ji}^{DA}$ be defined in (24), it is

$$\nabla J_{ji}[\mathbf{u}_{ji}^{DA}] = \underline{0} \in \Re^{NP}, \quad \forall (j,i) : \Delta_j \times \Omega_i \in DD(\Delta \times \Omega). \tag{29}$$

From the (29) follows

$$\nabla EO\left[J_{ji}\left(\mathbf{u}_{ji}^{DA}\right)\right] = \underline{0}, \tag{30}$$

which gives from the (19):

$$\nabla J\left[(\mathbf{u}_{ji}^{DA})^{EO}\right]=\underline{0} \tag{31}$$

then $(\mathbf{u}_{ji}^{DA})^{EO}$ is a stationary point for J in $\Re^{M\times K}$. As $\mathbf{u}^{DA}$ in (5) is the global minimum of J in $\Re^{K}$, it follows that:

$$J(EO(\mathbf{u}^{DA}))\le J\left((\mathbf{u}_{ji}^{DA})^{EO}\right),\quad \forall j=1,Q;i=1,P \tag{32}$$

then, from the (27) it follows that

$$J(EO(\mathbf{u}^{DA}))\le J\left(\widetilde{\mathbf{u}}^{\mathbf{DA}}\right),\quad . \tag{33}$$

Now we prove that if J is convex, then

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

by reduction to the absurd. Assume that

$$J(EO(\mathbf{u}^{DA}))<J(\widetilde{\mathbf{u}}^{\mathbf{DA}}). \tag{34}$$

In particular,

$$J(EO(\mathbf{u}^{DA}))<J(RO_{ji}(\widetilde{\mathbf{u}}^{DA}))\quad .$$

This means that

$$RO_{ji}\left[J(EO(\mathbf{u}^{DA}))\right]<RO_{ji}\left[J(\widetilde{\mathbf{u}}^{\mathbf{DA}})\right]. \tag{35}$$

From the (35) and the (27), it is:

$$RO_{ji}\left[J(EO(\mathbf{u}^{DA}))\right]<RO_{ji}\left[min_i(J\left(\mathbf{u}_{ji}^{DA})^{EO}\right)\right]$$

then, from the (14):

$$J_{ji}\left(RO_{ji}[\mathbf{u}^{DA}]^{EO}\right)<J_{ji}\left(RO_{ji}\left[\mathbf{u}_{ji}^{DA}\right]^{EO}\right)=J_{ji}(\mathbf{u}_{ji}^{DA})\qquad . \tag{36}$$

The (36) is an absurd as the values of $\mathbf{u}_{ji}^{DA}$ is the global minimum for J_{ji}. So the (28) is proved. ♣

□

4 The Space-Time RNLLS Parallel Algorithm

We introduce the algorithm solving the RNL-LS problem by using the space-time decomposition, i.e. solving the $QP=q\times p$ local problems in $\Delta_j\times\Omega_i$, where $j=1,Q$ and $i=1,P$ (see Fig. 1 to see an example of domain decomposition where $Q=4$ and $P=2$.).

Definition 14 (DD-RNLLS Algorithm) Let $\mathcal{A}_{RNLLS}^{loc}(\Delta_j\times\Omega_i)$ denote the algorithm solving the local 4DVar DA problem defined in $\Delta_j\times\Omega_i$. The space-time DD-RNLLS parallel algorithm solving the RNL-LS problem in $DD(\Delta\times\Omega)$, is symbolically denoted as

$$\mathcal{A}_{RNNLS}^{DD}(\Delta_M\times\Omega_K),$$

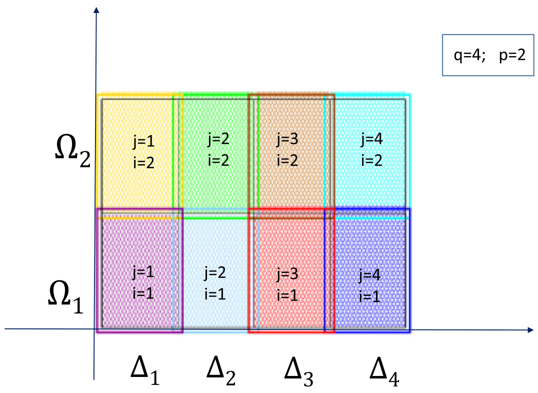

Fig. 1 Configurations of the decomposition $DD(\Delta_M \times \Omega_K)$, if $\Omega \subset \Re$, and $Q = 4$, $P = 2$

and it is defined as the merging of the $QP = Q \times P$ local algorithms $\mathcal{A}^{loc}_{RNLLS}(\Delta_j \times \Omega_i)$, i.e.:

$$\mathcal{A}^{DD}_{RNLLS}(\Delta_M \times \Omega_{NP}) := \bigcup_{j=1,Q;i=1,P} \mathcal{A}^{loc}_{RNLLS}(\Delta_j \times \Omega_i). \tag{37}$$

♠

The DD-RNLLS algorithm can be sketched as described by **Algorithm 1**. Similarly, the Local RNLLS algorithm $\mathcal{A}^{loc}_{RNLLS}$ is described by **Algorithm 2**.

```
Algorithm 1; $\mathcal{A}^{DD}_{RNLLS}$: solves the RNL-LS problem on $\Delta_M \times \Omega_{NP}$
1: % **Domain Decomposition** Step
2: Compute $\mathbf{M}^{\Delta_M \times \Omega_K}$ from $\mathcal{M}^{\Delta \times \Omega}$
3: **procedure** DD- 4DVAR($in : \mathbf{M}^{\Delta_M \times \Omega_K}, \mathbf{u}_0^b, \mathbf{R}, \mathbf{B}, \mathbf{H}, \mathbf{v}, \Delta_M, \Omega_K; out : \mathbf{u}^{\mathbf{DA}}$)
4:     % Run $\mathbf{M}^{\Delta_M \times \Omega_K}$ in (1) with initial condition $\mathbf{u}_0^b$
5:     $\mathbf{u}^b = \mathbf{M}^{\Delta_M \times \Omega_K}(\mathbf{u}_0^b)$
6:     % **Local Model Linearization Step**
7:     **for** $j = 1, q;\ i = 1, p$ **do**
8:     $l := 0, \mathbf{u}_{ji}^0 = \mathbf{u}_{ji}^b$
9:     **repeat**
10:        $l := l + 1$
11:        **Call** Loc-4DVar ($in : \mathbf{M}^{\mathbf{0},\mathbf{M-1}}, \mathbf{R}, \mathbf{B}, \mathbf{H}, \mathbf{v}, \mathbf{u}^b, \Delta_j, \Omega_i; out : \mathbf{u}_{ji}^l$)
12:        **Exchange** $\mathbf{u}_{ji}^k$ between adjacent subdomains
13:     **until** $\|\mathbf{u}_{ji}^l - \mathbf{u}_{ji}^{l-1}\| < eps$
14:     **endfor**
15:     % **End the Domain Decomposition** Step
16:     **Gather** of $\mathbf{u}_{ji}^l : \mathbf{u}^{\mathbf{DA}} = \sum_{ji} \mathbf{u}_{ji}^l$
```

```
Algorithm 2; $\mathcal{A}^{loc}_{RNLLS}$: solves the Local RNL-LS problem on $\Delta_j \times \Omega_i$
1: **procedure** LOC- RNLLS($in : \mathbf{M}^{\mathbf{0},\mathbf{M-1}}, \mathbf{R}, \mathbf{B}, \mathbf{H}, \mathbf{v}, \mathbf{u}^b, \Delta_j, \Omega_i; out : \mathbf{u}_{ji}^l$)
2:     **Compute** $\mathbf{J}_{ji}$ on $\Delta_j \times \Omega_i$
3:     **Compute** $\mathbf{u}_{ji} = argmin\, \mathbf{J}_{ji}$
```

Remark 2 We observe that $\mathcal{A}^{DD}_{RNNLS}(\Delta_M \times \Omega_K)$ algorithm is based on two main steps, i.e. the domain decomposition step (see line 1) and the model linearization step (see line 6). This means that this algorithm uses a convex approximation of the objective DA functional so that Theorem 1 holds.

The common approach for solving RNL-LS problems consists in defining a sequence of local approximations of $\mathbf{J}_{ij}$ where each member of the sequence is minimized by employing Newton's method or one its variants (such as Gauss-Newton, L-BFGS, Levenberg-Marquardt). Approximations of $\mathbf{J}_{ij}$ are obtained by expanding $\mathbf{J}_{ij}$ in a truncated Taylor's series, while the minimum is obtained by using second-order sufficient conditions [13, 44]. Let's consider **Algorithm 3** solving the RNL-LS problem on $\Delta_j \times \Omega_i$.

```
Algorithm 3; $\mathcal{A}^{loc}_{RNLLS}$: solves a RNL-LS problem on $\Delta_j \times \Omega_i$
1: procedure Loc- RNLLS($in$ : $\mathbf{M^{0,M-1}}$, $\mathbf{R}$, $\mathbf{B}$, $\mathbf{H}$, $\mathbf{v}$, $\mathbf{u}^b$, $\Delta_j$, $\Omega_i$; $out$ : $\mathbf{u}^l_{ji}$)
2:     Initialize $\mathbf{u}^0_{ij} := \mathbf{u}^b_{ij}$;
3:     Initialize $l := 0$;
4:     repeat % at each step $l$, a local approximation of $\tilde{\boldsymbol{J}}_{ij}$ is minimized
5:             Compute $\delta\mathbf{u}^l_{ij} = argmin\ \tilde{\boldsymbol{J}}_{ji}$
6:             Update $\mathbf{u}^l_{ji} = \mathbf{u}^l_{ji} + \delta\mathbf{u}^l_{ji}$
7:             Update $l = l + 1$
8:     until (convergence is reached)
```

Main computational task occurs at step 5 of **Algorithm 3** concerning the minimization of $\tilde{\mathbf{J}}_{ji}$, which is the local approximation of $\mathbf{J}_{ij}$. Two approaches could be employed in **Algorithm 3**:

(a) by truncating Taylor's series expansion of $\mathbf{J}_{ij}$ at the second order, we get

$$\mathbf{J}^{QD}_{ij}(\mathbf{u}^{l+1}_{ji}) = \mathbf{J}_{ij}(\mathbf{u}^l_{ji}) + \nabla\mathbf{J}_{ij}(\mathbf{u}^l_{ji})^T\delta\mathbf{u}^l_{ji} + \left(\delta\mathbf{u}^l_{ji}\right)^T \nabla^2\mathbf{J}_{ij}(\mathbf{u}^l_{ji})\delta\mathbf{u}^l_{ji} \tag{38}$$

giving a quadratic approximation of $\mathbf{J}_{ji}$ at $\mathbf{u}^l_{ji}$. Newton'methods (including LBFGS and Levenberg-Marquardt) use $\tilde{\boldsymbol{J}}_{ji} = \mathbf{J}^{QD}_{ij}$.

(b) by truncating Taylor's series expansion of $\mathbf{J}_{ij}$ at the first order we get the following linear approximation of $\mathbf{J}_{ij}$ at $\mathbf{u}^k_{ji}$:

$$\mathbf{J}^{TL}_{ij}(\mathbf{u}^{l+1}_{ji}) = \mathbf{J}_{ij}(\mathbf{u}^l_{ji}) + \nabla\mathbf{J}_{ij}(\mathbf{u}^l_{ji})^T\delta\mathbf{u}^l_{ji} = \frac{1}{2}\|\nabla\mathbf{F}_{ji}(\mathbf{u}^l_{ji})\delta\mathbf{u}^l_{ji} + \mathbf{F}_{ji}(\mathbf{u}^l_{ji})\|_2^2 \tag{39}$$

where we let[7] $\mathbf{J}_{ij} := \|\mathbf{F}_{ji}\|_2^2$ which gives a linear approximation of $\mathbf{J}_{ji}$ at $\mathbf{u}^l_{ji}$. Gauss-Newton's methods (including Truncated or Approximated Gauss-Newton [21]) use $\mathbf{J}^{TL}_{ji} = \tilde{\mathbf{J}}_{ji}$.

[7] If $\mathbf{C}_{ji} = diag((\mathbf{B}^{-1})_{ji}, (\mathbf{R}^{-1})_{ji})$, and $\tilde{\boldsymbol{d}}^l_{ji} = (\mathbf{u}^l_{ji} - \mathbf{u}^b_0, \mathbf{H}^0_{ji}(\mathbf{u}^l_{ji}) - \mathbf{v}^k_{ji}, \ldots, (\mathbf{H}^{M-1})_{ji}[(\mathbf{M}^k_{M-2,M-1})_{ji}(\mathbf{u}^k_{ji})] - \mathbf{v}^l_{ji})$ then $\mathbf{J}_{ij} := \frac{1}{2}((\mathbf{C}^{-1/2})_{ji}\tilde{\boldsymbol{d}}^l_{ji})^T((\mathbf{C}^{-1/2})_{ji}\tilde{\boldsymbol{d}}^l_{ji}) == \|\mathbf{F}_{ji}\|_2^2$ where $\mathbf{F}_{ji} = (\mathbf{C}^{-1/2})_{ji}\tilde{\boldsymbol{d}}^l_{ji}$.

Observe that from (38) it follows

$$\mathbf{J}_{ij}^{QD}(\mathbf{u}_{ji}^{l+1}) = \mathbf{J}_{ij}^{TL}(\mathbf{u}_{ji}^{l}) + \frac{1}{2}\left(\delta\mathbf{u}_{ji}^{l}\right)^{T} \nabla^2\mathbf{J}_{ij}(\mathbf{u}_{ji}^{l})\delta\mathbf{u}_{ji}^{l}. \tag{40}$$

Algorithm 3 becomes **Algorithm 4**, described below.

Algorithm 4; $\mathcal{A}_{RNLLS}^{loc}$: solves a RNL-LS problem on $\Delta_j \times \Omega_i$

```
1: procedure Loc- RNLLS($in$ : $\mathbf{M^{0,M-1}}, \mathbf{R}, \mathbf{B}, \mathbf{H}, \mathbf{v}, \mathbf{u}^b, \Delta_M, \Omega_K$; $out$ : $\mathbf{u}_{ji}^l$)
2:     Initialize $\mathbf{u}_{ij}^0 := \mathbf{u}_{ij}^b$;
3:     Initialize $l := 0$;
4:     repeat
5:         % Compute $\delta\mathbf{u}_{ij}^l = argmin\,\mathbf{J}_{ji}$ by using $\mathcal{A}_{QN}^{loc}$ or $\mathcal{A}_{LLS}^{loc}$
6:         If (QN) then
7:             Call Loc-QN ($in$ : $\mathbf{M^{0,M-1}}, \mathbf{R}, \mathbf{B}, \mathbf{H}, \mathbf{v}, \mathbf{u}^b, \Delta_M, \Omega_K$; $out$ : $\mathbf{u}_{ji}^l$ )
8:         ElseIf (LLS) then
9:             Call Loc-LLS ($in$ : $\mathbf{M^{0,M-1}}, \mathbf{R}, \mathbf{B}, \mathbf{H}, \mathbf{v}, \mathbf{u}^b, \Delta_M, \Omega_K$; $out$ : $\mathbf{u}_{ji}^l$)
10:        EndIf
11:        Update $\mathbf{u}_{ji}^l = \mathbf{u}_{ji}^l + \delta\mathbf{u}_{ji}^l$
12:        Update $l = l + 1$
13:    until (convergence is reached)
```

(a) $\mathcal{A}_{QN}^{loc}$: computes a local minimum of $\mathbf{J}_{ji}^{QN}$ following the Newton's descend direction. The minimum is computed solving the linear system involving the Hessian matrix $\nabla^2\mathbf{J}_{ij}$, and the negative gradient $-\nabla\mathbf{J}_{ij}$ at $\mathbf{u}_{ji}^l$, for each value of l (see **Algorithm 5** described below);

(b) $\mathcal{A}_{LLS}^{loc}$: computes a local minimum of $\mathbf{J}_{ji}^{TL}$ following the steepest descend direction. The minimum is computed solving the normal equations arising from the local Linear Least Squares (LLS) problem (see **Algorithm 6** described below).

Algorithm 5; $\mathcal{A}_{QLS}^{loc}$: solves a Q-LS problem on $\Delta_j \times \Omega_i$

```
1: procedure Loc- QN($\mathbf{M^{0,M-1}}, \mathbf{R}, \mathbf{B}, \mathbf{H}, \mathbf{v}, \mathbf{u}^b, \Delta_M, \Omega_K$; $out$ : $\mathbf{u}_{ji}^l$)
2:     Initialize $\mathbf{u}_{ji}^0 := \mathbf{u}_{ji}^b$;
3:     Initialize $l := 0$;
4:     repeat
5:         %Compute $\delta\mathbf{u}_{ij}^l = argmin\,\mathbf{J}_{ji}^{QD}$, by Newton's method
6:         1.1 Compute $\nabla\mathbf{J}_{ji}(\mathbf{u}_{ij}^l) = \nabla\mathbf{F}_{ji}^T(\mathbf{u}_{ji}^l)\nabla\mathbf{F}_{ji}(\mathbf{u}_{ji}^l)$
7:         1.2 Compute $\nabla^2\mathbf{J}_{ji}(\mathbf{u}_{ij}^l) = \nabla\mathbf{F}_{ji}^T(\mathbf{u}_{ji}^l)\nabla\mathbf{F}_{ji}(\mathbf{u}_{ji}^l) + \mathbf{Q}((\mathbf{u}_{ij}^l))$
8:         1.3 Solve $\nabla^2\mathbf{J}_{ji}(\mathbf{u}_{ij}^l)\delta\mathbf{u}_{ij}^l = -\nabla\mathbf{J}_{ji}(\mathbf{u}_{ij}^l)$
9:         Update $\mathbf{u}_{ji}^l = \mathbf{u}_{ji}^l + \delta\mathbf{u}_{ji}^l$
10:        Update $l = l + 1$
11:    until (convergence is reached)
```

```
Algorithm 6; $\mathcal{A}_{LLS}^{loc}$: solves LLS problems in $\Delta_j \times \Omega_i$
1: procedure LOC- LLS($\mathbf{M^{0,M-1}}, \mathbf{R}, \mathbf{B}, \mathbf{H}, \mathbf{v}, \mathbf{u}^b, \Delta_M, \Omega_K; out : \mathbf{u}_{ji}^l$)
2:     Initialize $\mathbf{u}_{ij}^0 := \mathbf{u}_{ij}^b$;
3:     Initialize $l := 0$;
4:     repeat
5:         Compute $\nabla \mathbf{J}_{ji} = \nabla \mathbf{F}_{ji}^T(\mathbf{u}_{ji}^l)\nabla \mathbf{F}_{ji}(\mathbf{u}_{ji}^l)$
6:         %Compute $\delta \mathbf{u}_{ij}^l = argmin\, \mathbf{J}_{ji}^{TL}$ by solving the normal equations system:
7:         Solve $\nabla \mathbf{F}_{ji}^T(\mathbf{u}_{ji}^l)\nabla \mathbf{F}_{ji}(\mathbf{u}_{ji})\delta \mathbf{u}_{ji}^l = -\nabla \mathbf{F}_{ji}^T(\mathbf{u}_{ji}^l)\mathbf{F}_{ji}(\mathbf{u}_{ji}^l)$
8:         Update $\mathbf{u}_{ji}^l = \mathbf{u}_{ji}^l + \delta \mathbf{u}_{ji}^l$
9:         Update $l = l + 1$
10:    until (convergence is reached)
```

Remark 3 : We observe that if, in $\mathcal{A}_{QN}^{loc}$ algorithm, matrix $\mathbf{Q}(\mathbf{u}_{ij}^l)$ (see line 6 of Algorithm 5) is neglected we get the Gauss-Newton method described by $\mathcal{A}_{LLS}^{loc}$ algorithm. More generally, term $\mathbf{Q}(\mathbf{u}_{ij}^l)$

1. in case of Gauss Newton, $\mathbf{Q}(\mathbf{u}_{ij}^l)$ is neglected;
2. in case of Levenberg-Marquardt, $\mathbf{Q}(\mathbf{u}_{ij}^l)$ equals to λI, where the damping term, $\lambda > 0$, is updated at each iteration and I is the identity matrix [26, 30];
3. in case of the L-BFGS, the Hessian matrix is Rank-1 updated at every iteration [45].

According to the most common implementation of the 4DVar DA [15, 50], we focus the attention on Gauss-Newton(G-N) method described in $\mathcal{A}_{LLS}^{loc}$ in **Algorithm 6**.

For each l, let $\mathbf{G}_{ji}^l = RO_{ji}[\mathbf{G}^l]$, where $\mathbf{G}^l \in \Re^{(M\times nobs)\times(NP\times M)}$, be the block diagonal matrix such that

$$\mathbf{G}^l = \begin{cases} diag\,[\mathbf{H}_0, \mathbf{H}_1\mathbf{M}_{0,1}^l, \ldots, \mathbf{H}_{M-1}\mathbf{M}_{M-2,M-1}^l] & M > 1; \\ \mathbf{H}_0 & M = 1. \end{cases} \tag{41}$$

while $(\mathbf{G}_{ji}^T)^l = RO_{ji}[(\mathbf{G}^T)^k]$ is the restriction of the transpose of $\mathbf{G}^l$, and

$$\mathbf{M}_{0,1}^l, \ldots, \mathbf{M}_{M-2,M-1}^l$$

are the TLMs of $\mathbf{M}_{k,k+1}$, for $s = 0, M-1$, around $\mathbf{u}_{ji}^l$, respectively. Finally, let

$$\mathbf{d}_{ji}^l = \mathbf{v}_{ji} - \mathbf{H}_{ji}\mathbf{u}_{ji}^l$$

be the restriction of the misfit vector. In line 7 of **Algorithm 6**, it is

$$\nabla \mathbf{F}_{ji}^T(\mathbf{u}_{ji}^l)\nabla \mathbf{F}_{ji}(\mathbf{u}_{ji}^l) = \mathbf{B}_{ji}^{-1} + (\mathbf{G}_{ji}^T)^l\mathbf{R}_{ji}\mathbf{G}_{ji}^l, \tag{42}$$

and,

$$-\nabla \mathbf{F}_{ji}^T(\mathbf{u}_{ji}^l)\mathbf{F}_{ji}(\mathbf{u}_{ji}^l) = (\mathbf{G}_{ji}^T)^l\mathbf{R}_{ji}^{-1}\mathbf{d}_{ji}. \tag{43}$$

Most popular 4DVar DA software implement the so called **B**-preconditioned Krylov subspace iterative method [21, 23, 50] arising by using the background error covariance matrix as preconditioner of a Krylov subspace iterative method.

Let $\mathbf{B}_{ji} = \mathbf{V}_{ji}\mathbf{V}_{ji}^T$ be expressed in terms of the deviance matrix $\mathbf{V}_{ji}$, and $\mathbf{w}_i$ such that

$$\mathbf{w}_{ji}^l = \mathbf{V}_{ji}^+(\mathbf{u}_{ji}^l - \mathbf{u}_{ji}^b) \tag{44}$$

with V_i^+ generalised inverse of $\mathbf{V}_i$, the (42) becomes

$$\mathbf{B}_{ji}^{-1} + (\mathbf{G}_{ji}^T)^l \mathbf{R}_{ji} \mathbf{G}_{ji}^l = \mathbf{I}_{ji} + (\mathbf{G}_{ji}^l \mathbf{V}_{ji})^T (\mathbf{R}^{-1})_{ji} \mathbf{G}_{ji}^l \mathbf{V}_{ji}, \tag{45}$$

while the (43) becomes

$$(\mathbf{G}_{ji}^T)^l (\mathbf{R}^{-1})_{ji} \mathbf{d}_{ji} = (\mathbf{G}_{ji} \mathbf{V}_{ji})^T)^k (\mathbf{R}^{-1})_{ji} \mathbf{d}_{ji}. \tag{46}$$

The normal equation system (see line 7 of $\mathcal{A}_{LLS}^{loc}$), i.e. the linear system

$$((\mathbf{B}^{-1})_{ji} + (\mathbf{G}_{ji}^T)^l \mathbf{R}_{ji} \mathbf{G}_{ji}^l) \delta \mathbf{u}_{ji}^l = (\mathbf{G}_{ji}^T)^l (\mathbf{R}^{-1})_{ji} \mathbf{d}_{ji}$$

becomes

$$(\mathbf{I}_{ji} + (\mathbf{G}_{ji}^l \mathbf{V}_{ji})^T (\mathbf{R}^{-1})_{ji} \mathbf{G}_{ji}^l \mathbf{V}_{ji}) \delta \mathbf{u}_{ji}^l = (\mathbf{G}_{ji}^l \mathbf{V}_{ji})^T (\mathbf{R}^{-1})_{ji} \mathbf{d}_{ji} \quad .$$

Definition 15 [DD-4DVar Algorithm] Let $\mathcal{A}_{4DVar}^{loc}(\Delta_j \times \Omega_i)$ denote the algorithm solving the local 4DVar DA problem defined in $\Delta_j \times \Omega_i$. The space-time 4DVar DA parallel algorithm solving the 4DVar DA problem in $DD(\Delta_M \times \Omega_K)$, is symbolically denoted as $\mathcal{A}_{4DVar}^{DD}(\Delta_M \times \Omega_K)$, and it is defined as the union of the $QP = q \times p$ local algorithms $\mathcal{A}_{4DVar}^{loc}(\Delta_j \times \Omega_i)$, i.e.:

$$\mathcal{A}_{4DVar}^{DD}(\Delta_M \times \Omega_K) := \bigcup_{j=1,q;i=1,p} \mathcal{A}_{4DVar}^{loc}(\Delta_j \times \Omega_i). \tag{47}$$

♠

Algorithm $\mathcal{A}_{4DVar}^{loc}$ is algorithm $\mathcal{A}_{LLS}^{loc}$ (see **Algorithm 6**) specialized for the 4D Var DA problem and it is described by **Algorithm 7** and **Algorithm 8** described below [23].

Algorithm 7; $\mathcal{A}_{4DVar}^{loc}$: solves Local 4DVAR DA problem in $\Delta_j \times \Omega_i$

```
1: procedure Loc- 4DVar($\mathbf{M}^{\Delta_M \times \Omega_{NP}}$, $\mathbf{R}, \mathbf{B}, \mathbf{H}, \mathbf{v}, \mathbf{u}^b, \Delta_M, \Omega_K; out : \mathbf{u}_{ji}^l$)
2:     Initialize $\mathbf{u}_{ji}^0 := \mathbf{u}_{ji}^b$;
3:     Initialize $l := 0$;
4:     repeat
5:         Compute $\mathbf{d}_{ji}^l = \mathbf{v}_{ji} - \mathbf{H}_{ji}(\mathbf{u}_{ji}^l)$
6:         Call TLM($in : \mathcal{M}^{\Delta \times \Omega}, \mathbf{u}_{ji}^l; out : \mathbf{M}_{0,M-1}^l$)
7:         Call ADJ($in : \mathbf{M}_{0,M-1}^k; out : (\mathbf{M}_{0,M-1}^T)^l$)
8:         Compute $\mathbf{G}_{ji}, \mathbf{V}_{ji}$
9:         Call $\mathcal{A}_{BLanczos}^{loc}$ ($\mathbf{G}_{ji}^k, \mathbf{V}_{ji}, \mathbf{R}_{ji}, \mathbf{B}_{ji}, \mathbf{d}_{ji}, \mathbf{u}_{ji}^b, \Delta_j, \Omega_i; out : \delta\mathbf{u}_{ji}^k$)
10:        Update $\mathbf{u}_{ji}^l = \mathbf{u}_{ji}^l + \delta\mathbf{u}_{ji}^l$
11:        Update $l = l + 1$
12:    until (convergence is reached)
13: endprocedure
14: procedure TLM($in : \mathcal{M}^{\Delta \times \Omega}, \mathbf{u}_{ji}^l; out : \mathbf{M}_{0,M-1}^l$)
15:        %Linearize $\mathbf{M}^{\Delta_M \times \Omega_{NP}}$ about $\mathbf{u}_{ji}^l$
16: endprocedure
17: procedure ADJ($in : \mathbf{M}_{0,M-1}^k; out : (\mathbf{M}_{0,M-1}^T)^l$)
18:        %Compute the adjoint of $\mathbf{M}_{0,M-1}$
19: endprocedure
```

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

```
Algorithm 8; $\mathcal{A}^{loc}_{BLanczos}$: BLanczos for 4DVAR DA problem in $\Delta_j \times \Omega_i$
1: procedure BLANCZOS- 4DVAR($\mathbf{G}_{ji}, \mathbf{V}_{ji}, \mathbf{R}_{ji}, \mathbf{B}_{ji}, \mathbf{d}_{ji}, \mathbf{u}^b_{ji}, \Delta_j, \Omega_i; out : \delta\mathbf{u}^l_{ji}$)
2:     % Solve $(\mathbf{I}_{ji} + (\mathbf{G}_{ji}\mathbf{V}_{ji})^T(\mathbf{R}^{-1})_{ji}\mathbf{G}_{ji}\mathbf{V}_{ji})\delta\mathbf{u}^l_{ji} = (\mathbf{G}_{ji}\mathbf{V}_{ji})^T(\mathbf{R}^{-1})_{ji}\mathbf{d}_{ji}$
3:     % by using BLanczos algorithm (see [23])
```

5 Convergence Analysis

In the following we pose $\|\cdot\| = \|\cdot\|_\infty$.

Proposition 1 *Let $u^{ASM,r}_{j,i}$ be the approximation of the increment $\delta\mathbf{u}_{ji}$ to the solution $\mathbf{u}_{ji}$ obtained at step r of ASM-based inner loop on $\Omega_j \times \Delta_i$. Let $u^n_{j,i}$ be the approximation of $\mathbf{u}_{j,i}$ obtained at step n of the outer loop i.e. the space-time decomposition approach on $\Omega_j \times \Delta_i$. Let us assume that the numerical scheme discretizing the model $\mathbf{M}^{j,j+1}_i$ is convergent. Then, given i and j fixed, it holds that:*

$$\forall \epsilon > 0 \ \exists M(\epsilon) > 0 \ : \ n > M(\epsilon) \ \Rightarrow \ E^n_{j,i} := \|\mathbf{u}_{j,i} - u^n_{j,i}\| \le \epsilon. \tag{48}$$

Proof let $u^{\mathbf{M}^{j,j+1}_i, n+1}_{j,i}$ be the numerical solution of $\mathbf{M}^{j,j+1}_i$ at step n; taking into account that, according to the incremental update of the solution of the 4D Var DA functional (for instance, see line 10 of the Algorithm 7), the approximation $\mathbf{u}^n_{j,i}$ is computed as

$$\mathbf{u}^n_{j,i} = u^{\mathbf{M}^{j,j+1}_i, n+1}_{j,i} + [u^{ASM,r}_{j,i} - u^{\mathbf{M}^{j,j+1}_i, n}_{j,i}]$$

then, it is

$$\begin{aligned} E^n_{j,i} := \|\mathbf{u}_{j,i} - u^n_{j,i}\| &= \|\mathbf{u}_{j,i} - u^{\mathbf{M}^{j,j+1}_i, n+1}_{j,i} - [u^{ASM,r}_{j,i} - u^{\mathbf{M}^{j,j+1}_i, n}_{j,i}]\| \\ &\le \|\mathbf{u}_{j,i} - u^{ASM,r}_{j,i}\| + \|u^{\mathbf{M}^{j,j+1}_i, n}_{j,i} - u^{\mathbf{M}^{j,j+1}_i, n+1}_{j,i}\| \end{aligned} \tag{49}$$

from the hypothesis, we have

$$\begin{aligned} &\forall \epsilon^{\mathbf{M}^{j,j+1}_i} > 0 \, \exists M^1(\epsilon^{\mathbf{M}^{j,j+1}_i}) > 0 : n > M^1(\epsilon^{\mathbf{M}^{j,j+1}_i}) \\ &\Rightarrow E^n_{j,i} := \|u^{\mathbf{M}^{j,j+1}_i, n+1}_{j,i} - u^{\mathcal{M}^j, n}_{j,i}\| \le \epsilon^{\mathbf{M}^{j,j+1}_i} \end{aligned} \tag{50}$$

and (49) can be rewritten as follows

$$\|\mathbf{u}_{j,i} - u^n_{j,i}\| \le \|\mathbf{u}_{j,i} - u^{ASM,r}_{j,i}\| + \epsilon^{\mathbf{M}^{j,j+1}_i}. \tag{51}$$

Convergence of ASM is proved in [5], similarly, applying ASM to 4D Var DA problem, it holds that

$$\forall \epsilon^{ASM} > 0 \ \exists M^2(\epsilon^{ASM}) > 0 \ : \ n > M^2(\epsilon^{ASM}) \ \Rightarrow \ \|u_{j,i} - u^{ASM,r}_{j,i}\| \le \epsilon^{ASM}, \tag{52}$$

and for $n > M^2(\epsilon^{ASM})$, we get

$$\|u_{j,i} - u^{ASM,n}_{j,i}\| \le \epsilon^{ASM} + \epsilon^{\mathbf{M}^{j,j+1}_i}. \tag{53}$$

Hence, by using $\epsilon := \epsilon^{ASM} + \epsilon^{\mathbf{M}^{j,j+1}_i}$ and $M(\epsilon) := \max\{M^1(\epsilon^{ASM}), M^2(\epsilon^{\mathcal{M}^{j,j+1}_i})\}$, we get the thesis in (52). □

6 Performance Analysis

Performance metrics we will employ are time complexity and scalability. Our aim is to highlight the benefits arising from using the 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 algorithm's 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.

We give the following

Definition 16 An uniform bi-directional decomposition of the space and time domain $\Delta_M \times \Omega_K$, is such that if we let

$$size(\Delta_M \times \Omega_K) = M \times K,$$

be the size of the whole domain, then each subdomain $\Delta_j \times \Omega_i$ is such that

$$size(\Delta_j \times \Omega_i) = D_t \times D_s, \quad j = 1, \dots, q; \quad i = 1, \dots, p$$

where $D_t = \frac{M}{q} \geq 1$, and $D_s = \frac{K}{p} \geq 1$.

♠

In the following we let

$$N := M \times K; \quad N_{loc} := D_t \times D_s; \quad QP := q \times p \quad .$$

Let $T(\mathcal{A}^{DD}_{4DVar}(\Delta_M \times \Omega_K))$ denote time complexity of $\mathcal{A}^{DD}_{4DVar}(\Delta_M \times \Omega_K)$.

We now provide an estimate of the time complexity of each local algorithm, denoted as $T(\mathcal{A}^{Loc}_{4DVar}(\Delta_j \times \Omega_i))$. This algorithm consists of two loops. The outer-loop, over l-index, for computing local approximations of $\mathbf{J}_{ji}$, and the inner-loop over m-index, for performing Newton's or Lanczos' steps. The major computational tasks to be performed at each step of the outer-loop is the computation of $\mathbf{J}_{ji}$. The major computational tasks to be performed at each step l of the inner-loop, in case of G-N method (see algorithm $\mathcal{A}^{Loc}_{4DVar}$), involving the predictive model is[8]

1. the computation of the tangent linear model $RO_{ji}[\mathbf{M}_{k,k+1}]$ (the time complexity of such operation scales as the problem size squared) ,
2. the computation of the adjoint model $RO_{ji}[\mathbf{M}^T_{k,k+1}]$ (which is at least 4 times more expensive than the computation of $RO_{ji}[\mathbf{M}_{k,k+1}]$,
3. the solution of the normal equations, involving at each iteration, two matrix-vector products with $RO_{ji}[\mathbf{M}^T_{k,k+1}]$ and $RO_{ji}[\mathbf{M}_{k,k+1}]$ (whose time complexity scales as the problem size squared).

As the most time consuming operation involving the predictive model is the computation of the tangent linear model, we prove that

[8] These assumptions hold true for the so-called local discretization schemes, i.e. those schemes where each grid point receives contribution from a neighborhood (for instance, using finite difference and finite volume discretization schemes as in [51]).

Proposition 2 *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 of degree $d = 2$ denoting the time-complexity of the tangent linear model $RO_{ji}[\mathbf{M}_{s,s+1}]$. Let m_{ji} and l_{ji} be the number of steps of the outer/inner-loop, of $\mathcal{A}_{4DVAR}^{Loc}$, respectively. We get

$$T(\mathcal{A}_{4DVAR}^{Loc}(\Delta_j \times \Omega_i))) = O\left(m_{ji} l_{ji} P(N_{loc})\right)$$

Proof It is:

$$\begin{aligned} & T(\mathcal{A}_{4DVAR}^{Loc}(\Delta_j \times \Omega_i)) = \\ l_{ji} \times & \left[T(RO_{ji}[\mathbf{M}_{k,k+1}]) + m_{ji} \times O\left(T(RO_{ji}[\mathbf{M}_{k,k+1}]) + T(RO_{ji}[\mathbf{M}_{k,k+1}^T])\right)\right] = \\ l_{ji} \times & \left[T(RO_{ji}[\mathbf{M}_{k,k+1}]) + m_{ji} \times O\left(T(RO_{ji}[\mathbf{M}_{k,k+1}]) + T(\mathbf{M}_{ji}^T)\right)\right] = \\ & = O\left(m_{ji} l_{ji} P(N_{loc})\right) \end{aligned} \tag{54}$$

♣

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 $\mathcal{A}^{DD}(\Delta_M \times \Omega_K)$, respectively. Let m_G and l_G denote the number of iterations of inner and outer loop of $\mathcal{A}^G(\Delta_M \times \Omega_K)$ algorithm, we give the following:

Definition 17 Let

$$\rho^G := m_G \times l_G \quad ; \quad \rho^{ji} := m_{ji} \times l_{ji} \quad ; \quad \rho^{DD} := m_{max} \times l_{max}$$

denote the total number of iterations of $\mathcal{A}_{4DVAR}^G(\Delta_M \times \Omega_K)$, of $\mathcal{A}_{4DVAR}^{Loc}(\Delta_j \times \Omega_i)$ and of $\mathcal{A}_{4DVAR}^{DD}(\Delta_M \times \Omega_K)$, respectively.

If we denote by $\mu(J)$ the condition number of the DA operator , as it holds that [3]

$$\forall i, j \quad \mu(J_{4DVAR}^{Loc}) < \mu(J_{4DVAR})$$

then it is

$$\rho^{ji} < \rho^G,$$

and

$$\rho^{DD} < \rho^G.$$

This result says that the number of iterations of $\mathcal{A}_{4DVar}^{DD}(\Delta_M \times \Omega_K)$ algorithm is always smaller than the number of iterations of $\mathcal{A}_{4DVar}^G(\Delta_M \times \Omega_K)$ algorithm. This is one of the benefits of using the space and time decomposition.

Algorithm scalability is measured in terms of the *strong scaling* (which is the measure of the algorithm's capability to exploit performance of high performance computing architectures in order to minimise the time to solution for a given problem with a fixed dimension) and of the *weak scaling* (which is the measure of the algorithm's capability to use additional computational resources effectively to solve increasingly larger problems). A variety of metrics have been developed to assist in evaluating the scalability of a parallel algorithm, speedup, model throughput, scale up, efficiency are the most used. Each one highlights specific needs and limits to be answered by the parallel algorithm. In our case, as we intend to mainly focus on the benefits arising from the use of hybrid computing architectures we consider the so called scale-up factor firstly introduced in [7].

First result straightforwardly derives from the definition of the scale-up factor:

Proposition 3 (DD-4DVar Scale up factor) *The (relative) scale-up factor of* $\mathcal{A}^{DD}_{4DVar}(\Delta_M \times \Omega_K)$ *related to* $\mathcal{A}^{loc}_{4DVar}(\Delta_j \times \Omega_i)$*, denoted as* $Sc_{QP}(\mathcal{A}^{DD}_{4DVar}(\Delta_M \times \Omega_K))$ *is:*

$$Sc_{QP}(\mathcal{A}^{DD}(\Delta_M \times \Omega_K)) := \frac{1}{QP} \times \frac{T(\mathcal{A}^{G}_{4DVar}(\Delta_M \times \Omega_K))}{T(\mathcal{A}^{loc}_{4DVar}(\Delta_j \times \Omega_i))} \quad ,$$

where $QP := q \times p$ *is the number of sub domains. It is:*

$$Sc_{QP}(\mathcal{A}^{DD}) \geq \frac{\rho^G}{\rho^{DD}} \alpha(N_{loc}, QP)\, (QP)^{d-1} \tag{55}$$

where

$$\alpha(N_{loc}, QP) = \frac{a_d + a_{d-1}\frac{1}{N} + \ldots + \frac{a_0}{N^d_{loc}}}{a_d + a_{d-1}\frac{QP}{N_{loc}} + \ldots + \frac{a_0 (QP)^d}{N^d_{loc}}} \quad ,$$

and

$$\lim_{QP \to N_{loc}} \alpha(N_{loc}, QP) = \beta \in]0, 1]$$

♠

Corollary 1 *If* $a_i = 0 \quad \forall i \in [0, d-1]$*, then* $\beta = 1$*, i.e.*

$$\lim_{QP \to N_{loc}} \alpha(N_{loc}, QP) = 1$$

Finally

$$\lim_{N_{loc} \to \infty} \alpha(N_{loc}, QP) = 1.$$

♣

Corollary 2 *If* N_{loc} *is fixed, it is*

$$\lim_{QP \to N_{loc}} Sc_{1,QP}(\mathcal{A}^{DD}) = \beta \cdot N^{d-1}_{loc} \quad ;$$

while, if QP *is fixed*

$$\lim_{N_{loc} \to \infty} Sc_{1,QP}(\mathcal{A}^{DD}) = const \neq 0 \quad .$$

♣

From (55) it results that, considering one iteration of the whole parallel algorithm, the growth of the scale up factor essentially is one order less than the time complexity of the reduced model. In other words, the time complexity of the reduced model mostly impacts the scalability of the parallel algorithm. In particular, as parameter d equal to 2, it follows that the asymptotic scaling factor of the parallel algorithm, with respect to QP, is bounded above by two.ECMWF

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). We prove that

Theorem 2 *The surface to volume ratio of a uniform bi-dimensional decomposition of space-time domain $\Delta_M \times \Omega_K$, is*

$$\frac{\mathcal{S}}{\mathcal{V}} = 2\left(\frac{1}{D_t} + \frac{1}{D_s}\right) \quad . \tag{56}$$

Let $\mathcal{S}$ denote the surface of each subdomain, then

$$\mathcal{S} = 2\left(\frac{M}{q} + \frac{K}{p}\right)$$

and $\mathcal{V}$ denote its volume, then

$$\mathcal{V} = \frac{M}{q} \times \frac{K}{p} \quad .$$

It holds that

$$\frac{\mathcal{S}}{\mathcal{V}} = \frac{2\left(\frac{M}{q} + \frac{K}{p}\right)}{\frac{M}{q} \times \frac{K}{p}} = 2\left(\frac{1}{D_t} + \frac{1}{D_s}\right)$$

and the (56) follows.

Definition 18 (*Measured Software Scale-up*) Let

$$Sc_{1,QP}^{meas}(\mathcal{A}^{\mathcal{DD}}) := \frac{T_{flop}(N_{loc})}{QP \cdot (T_{flop}(N_{loc}) + T_{oh}(N_{loc}))}. \tag{57}$$

be the measured software scale-up in going from 1 to QP.

♠

Proposition 4 *If*

$$0 \leq \frac{S}{V}(\mathcal{A}_{4DVar}^{loc}) < 1 - \frac{1}{s_{nproc}^{loc}(\mathcal{A}_{4DVar}^{loc})} \quad ,$$

then, it holds that

$$Sc_{1,QP}^{meas}(\mathcal{A}_{4DVar}^{DD}) = \alpha(N_{loc}, QP) Sc_{1,QP}(\mathcal{A}_{4DVar}^{DD}) \tag{58}$$

with

$$\alpha(N_{loc}, QP)^{meas}(\mathcal{A}_{4DVar}^{DD}) = \frac{T_{flop}(N_{loc})}{\frac{QP\,T_{flop}(N_{loc})}{s_{nproc}^{loc}} + QP\,T_{oh}(N_{loc})} = \frac{s_{nproc}^{loc}\frac{T_{flop}(N_{loc})}{QP\,T_{flop}(N_{loc})}}{1 + \frac{s_{nproc}^{loc}T_{oh}(N_{loc})}{T_{flop}(N_{loc})}}. \tag{59}$$

If

$$\alpha(N_{loc}, QP) := \frac{s_{nproc}^{loc}}{1 + \frac{s_{nproc}^{loc}T_{oh}(N_{loc})}{T_{flop}(N_{loc})}} = \frac{s_{nproc}^{loc}}{1 + s_{nproc}^{loc}\frac{S}{V}}$$

from (59) it comes the thesis in (58).

♣

In the following we will denote the measured scale up as $Sc_{1,QP}^{meas}(\mathcal{A}_{4DVar}^{DD})$ or as $Sc_{1,QP}^{meas}(N)$, respectively.

Finally, last proposition allows us to examine the benefit on the measured scale up arising from the speed up of the local parallel algorithm, mainly in the presence of a multilevel decomposition, where $s_{nproc}^{loc}(\mathcal{A}_{4DVar}^{loc}) > 1$.

Proposition 5 *It holds that*

$$s_{nproc}^{loc}(\mathcal{A}_{4DVar}^{loc}) \in [1, QP] \Rightarrow Sc_{QP}^{meas}(\mathcal{A}_{4DVar}^{DD}) \in]Sc_{1,QP}(\mathcal{A}_{4DVar}^{DD}), QP\,Sc_{1,QP}(\mathcal{A}_{4DVar}^{DD})[.$$

Proof – if $s_{nproc}^{loc}(\mathcal{A}_{4DVar}^{loc}) = 1$ then

$$\alpha(N, QP) < 1 \Leftrightarrow Sc_{1,QP}^{meas}(\mathcal{A}_{4DVar}^{DD}) < Sc_{1,QP}(\mathcal{A}_{4DVar}^{DD})$$

– if $s_{nproc}^{loc}(\mathcal{A}_{4DVar}^{loc}) > 1$ then

$$\alpha(N, QP) > 1 \Leftrightarrow Sc_{1,QP}^{meas}(\mathcal{A}_{4DVar}^{DD}) > Sc_{1,QP}^{f}(\mathcal{A}_{4DVar}^{DD});$$

– if $s_{nproc}^{loc}(\mathcal{A}_{4DVar}^{loc}) = QP$ then

$$1 < \alpha(N, QP) < QP \Rightarrow Sc_{1,QP}^{meas}(\mathcal{A}_{4DVar}^{DD}) < QP \cdot Sc_{1,QP}^{f}(\mathcal{A}_{4DVar}^{DD});$$

♣

□

We may conclude that

1. strong scaling: if QP increases and $M \times K$ is fixed, the scale up factor increases but the surface-to-volume ratio increases.
2. weak scaling: if QP is fixed and $M \times K$ increases, the scale up factor stagnate and the surface-to-volume ration decreases.

This means that it needs to find the appropriate value of the number of sub domains, QP, giving the right trade off between the scale up and the overhead of the algorithm.

7 Scalability Results

Results presented here are just a starting point towards the assessment of the software scalability. More precisely, we introduce simplifications and assumptions appropriate for a proof-of-concept study in order to get values of the measured scale up of the one iteration of the parallel algorithm.

As the main outcome of the decomposition is that the parallel algorithm is oriented to better exploit the high performance of new architectures where concurrency is implemented both at the coarsest and finest levels of granularity, such as a distributed memory multiprocessor (MIMD) and a Graphic Processing Units (GPUs) [56], we consider a distributed computing environment located in the University of Naples Federico II campus, connected by local-area network made of

- PE_1 (for the coarsest level of granularity): a *Multiple-Instruction, Multiple-Data* (MIMD) architecture made of 8 nodes which consist 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 the same local 16 GB RAM memory for a total of 8 cores per blade and of 64 total cores.
- PE_2 (for the finest level of granularity): a Kepler architecture of the GK110 GPU [46], which consists of a set of 13 programmable *Single-Instruction, Multiple-Data* (SIMD) *Streaming Multiprocessors* (SMXs), connected to a quad-core Intel i7 CPU running at 3.07GHz, 12 GB of RAM. For host(CPU)-to-device(GPU) memory transfers CUDA enabled graphic cards are connect to a PC motherboard via a PCI-Express (PCIe) BUS [48]. For this architecture the maximum number of active threads per multiprocessor is 2048, which means that the maximum number of active warps per SMX is 64.

Our implementation uses the matrix and vector functions in the Basic Linear Algebra Subroutines (BLAS) for PE_1 and the CUDA Basic Linear Algebra Subroutines (CUBLAS) library for PE_2. The routines used for computing the minimum of J on PE_1 and PE_2 are described in [28] and [10] respectively.

The case study is based on the Shallow Water Equations (SWEs) on the sphere. The SWE have been used extensively as a simple model of the atmosphere or ocean circulation since they contain the essential wave propagation mechanisms found in general circulation models [52].

The SWEs in spherical coordinates are:

$$\frac{\partial u}{\partial t} = -\frac{1}{a\cos\theta}\left(u\frac{\partial u}{\partial\lambda} + v\cos\theta\frac{\partial u}{\partial\theta}\right) + \left(f + \frac{u\tan\theta}{a}\right)v - \frac{g}{a\cos\theta}\frac{\partial h}{\partial\lambda} \tag{60}$$

$$\frac{\partial v}{\partial t} = -\frac{1}{a\cos\theta}\left(u\frac{\partial v}{\partial\lambda} + v\cos\theta\frac{\partial v}{\partial\theta}\right) + \left(f + \frac{u\tan\theta}{a}\right)u - \frac{g}{a}\frac{\partial h}{\partial\theta} \tag{61}$$

$$\frac{\partial h}{\partial t} = -\frac{1}{a\cos\theta}\left(\frac{\partial(hu)}{\partial\lambda} + \frac{\partial(hu\cos\theta)}{\partial\theta}\right) \tag{62}$$

Here f is the Coriolis parameter given by $f = 2\Omega\sin\theta$, where Ω is the angular speed of the rotation of the Earth, h is the height of the homogeneous atmosphere (or of the free ocean surface), u and v are the zonal and meridional wind (or the ocean velocity) components, respectively, θ and λ are the latitudinal and longitudinal directions, respectively, a is the radius of the earth and g is the gravitational constant.

We express the system of equations (60)–(62) using a compact form, i.e.:

$$\frac{\partial \mathbf{Z}}{\partial t} = \mathcal{M}_{t-\Delta t \to t}\left(\mathbf{Z}\right) \tag{63}$$

where

$$\mathbf{Z} = \begin{pmatrix} u \\ v \\ h \end{pmatrix} \tag{64}$$

and

$$\begin{aligned} \mathcal{M}_{t-\Delta t \to t}\left(\mathbf{Z}\right) &= \begin{pmatrix} -\frac{1}{a\cos\theta}\left(u\frac{\partial u}{\partial \lambda} + v\cos\theta\frac{\partial u}{\partial \theta}\right) + \left(f + \frac{u\tan\theta}{a}\right)v - \frac{g}{a\cos\theta}\frac{\partial h}{\partial \lambda} \\ -\frac{1}{a\cos\theta}\left(u\frac{\partial v}{\partial \lambda} + v\cos\theta\frac{\partial v}{\partial \theta}\right) + \left(f + \frac{u\tan\theta}{a}\right)u - \frac{g}{a}\frac{\partial h}{\partial \theta} \\ -\frac{1}{a\cos\theta}\left(\frac{\partial (hu)}{\partial \lambda} + \frac{\partial (hu\cos\theta)}{\partial \theta}\right) \end{pmatrix} \\ &= \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix} \end{aligned} \tag{65}$$

We discretize (63) just in space using an un-staggered Turkel-Zwas scheme [37, 38], and we obtain:

$$\frac{\partial \mathbf{Z}_{disc}}{\partial t} = \mathcal{M}_{disc}^{t-\Delta t \to t}\left(\mathbf{Z}_{disc}\right) \tag{66}$$

where

$$\mathbf{Z}_{disc} = \begin{pmatrix} \left(u_{i,j}\right)_{i=0,\ldots,nlon-1;\, j=0,\ldots,nlat-1} \\ \left(v_{i,j}\right)_{i=0,\ldots,nlon-1;\, j=0,\ldots,nlat-1} \\ \left(h_{i,j}\right)_{i=0,\ldots,nlon-1;\, j=0,\ldots,nlat-1} \end{pmatrix} \tag{67}$$

and

$$\mathcal{M}_{disc}^{t-\Delta t \to t}\left(\mathbf{Z}_{disc}\right) = \begin{pmatrix} \left(U_{i,j}\right)_{i=0,\ldots,nlon-1;\, j=0,\ldots,nlat-1} \\ \left(V_{i,j}\right)_{i=0,\ldots,nlon-1;\, j=0,\ldots,nlat-1} \\ \left(H_{i,j}\right)_{i=0,\ldots,nlon-1;\, j=0,\ldots,nlat-1} \end{pmatrix} \tag{68}$$

so that

$$\begin{aligned} U_{i,j} = & -\sigma_{lon}\frac{u_{i,j}}{\cos\theta_j}\left(u_{i+1,j} - u_{i-1,j}\right) \\ & -\sigma_{lat}\ v_{i,j}\left(u_{i,j+1} - u_{i,j-1}\right) \\ & -\sigma_{lon}\frac{g}{p\cos\theta_j}\left(h_{i+p,j} - h_{i-p,j}\right) \\ & +2\Big[(1-\alpha)\left(2\Omega\sin\theta_j + \frac{u_{i,j}}{a}\tan\theta_j\right)v_{i,j} \\ & + \frac{\alpha}{2}\left(2\Omega\sin\theta_j + \frac{u_{i+p,j}}{a}\tan\theta_j\right)v_{i+p,j} \\ & + \frac{\alpha}{2}\left(2\Omega\sin\theta_j + \frac{u_{i-p,j}}{a}\tan\theta_j\right)v_{i-p,j}\Big] \end{aligned}$$

$$
\begin{aligned}
V_{i,j} = &-\sigma_{lon}\frac{u_{i,j}}{\cos\theta_j}\left(v_{i+1,j}-v_{i-1,j}\right)\\
&-\sigma_{lat}\; v_{i,j}\left(u_{i,j+1}-u_{i,j-1}\right)\\
&-\sigma_{lat}\frac{g}{q}\left(h_{i,j+q}-h_{i,j-q}\right)\\
&-2\Big[(1-\alpha)\left(2\Omega\sin\theta_j+\frac{u_{i,j}}{a}\tan\theta_j\right)u_{i,j}\\
&+\frac{\alpha}{2}\left(2\Omega\sin\theta_{j+q}+\frac{u_{i,j+q}}{a}\tan\theta_{j+q}\right)u_{i,j+q}\\
&+\frac{\alpha}{2}\left(2\Omega\sin\theta_{j-q}+\frac{u_{i,j-q}}{a}\tan\theta_{j-q}\right)u_{i,j-q}\Big]\\
H_{i,j} = &-\alpha\Bigg\{\frac{u_{i,j}}{\cos\theta_j}\left(h_{i+1,j}-h_{i-1,j}\right)\\
&+v_{i,j}\left(h_{i,j+1}-h_{i,j-1}\right)\\
&+\frac{h_{i,j}}{\cos\theta_j}\Big[(1-\alpha)\left(u_{i+p,j}-u_{i-p,j}\right)\\
&+\frac{\alpha}{2}\left(u_{i+p,j+q}-u_{i-p,j+q}+u_{i+p,j-q}-u_{i-p,j-q}\right)\Big]\frac{1}{p}\\
&+\Big[(1-\alpha)\left(v_{i,j+q}\cos\theta_{j+q}-v_{i,j-q}\cos\theta_{j-q}\right)\\
&+\frac{\alpha}{2}\left(v_{i+p,j+q}\cos\theta_{j+q}-v_{i+p,j-q}\cos\theta_{j-q}\right)\\
&+\frac{\alpha}{2}\left(v_{i-p,j+q}\cos\theta_{j+q}-v_{i-p,j-q}\cos\theta_{j-q}\right)\Big]\frac{1}{q}\Bigg\}
\end{aligned}
$$

The numerical model depends on a combination physical parameters, including the number of state variables in the model, the number of observations in an assimilation cycle, as well as numerical parameters as the discretization step in time and in space domain are defined on the basis of discretization grid used by data available, at repository *Ocean Synthesis/Reanalysis Directory* of Hamburg University (see [15]).

To begin our data assimilation, an initial value of the model state is created by choosing snapshots from the run prior to the start of the assimilation experiment and treating it as realization valid at the nominal time. Then, the model state is advanced to the next time using the forecast model, and the observations are combined with the forecasts (i.e., the background) to produce the analysis. This process is iterated. As it proceeds, the process fills gaps in sparsely observed regions, converts observations to improved estimates of model variables, and filters observation noise. All this is done in a manner that is physically consistent with the dynamics of the ocean as represented by the model. In our experiments, the simulated observations are created by sampling the model states and adding random errors to those values. A detailed description of the simulation, together with the results and the software implemented, is presented in [11]. In the following, we are mainly interested to focus on performance results.

The reference domain decomposition strategy uses the following correspondence between QP and $nproc$:

$$QP \leftrightarrow nproc,$$

which means that the number of subdomains coincides with the number of available processors.

According to the characteristics of the physical domain in SWEs, the total number of grid points in space is

$$M = nlon \times nlat \times n_z \quad .$$

Let assume that

$$nlon = nlat = n$$

while $n_z = 3$. Since the unknown vectors are the fluid height or depth, and the two-dimensional fluid velocity fields, the problem size in space is

$$M = n^2 \times 3 \quad .$$

We assume a 2D uniform domain decomposition along the latitude-longitude directions, such that

$$D_s := \frac{M}{p} = nloc_x \times nloc_y \times 3 \tag{69}$$

with

$$nloc_x := \frac{n}{p_1} + 2o_x \;\; , \; nloc_y := \frac{n}{p_2} + 2o_y \;\; , \; n_z := 3 \; , \tag{70}$$

where $p_1 \times p_2 = p$.

Since the GPU (PE_2) can only process the data in its global memory [55], in a generic parallel algorithm execution, the host acquires these input data and sends them to the device memory, which concurrently calculates the minimization of the 4DVar functional. To avoid continuous relatively slow data transfer from the host to the device and in order to reduce the overhead, it was decided to store the device with the entire work data prior to any processing. Namely, the maximum value of D_s in (69) is chosen such that the amount of data related each subdomain (we denote it with $Data_{mem}(Mbyte)$) can be completely stored in the memory.

If we assume that $nloc_x = nloc_y$ and we let $n_{loc} = nloc_x = nloc_y$, as the global GPU memory is of 5Gbyte, we have the values of usable n_{loc} described in Table 1, Values of the speed-up s^{loc}_{nproc} in terms of gain obtained by using the GPU versus the CPU are reported in Table 2. We note that CUBLAS routines allow to reduce in average by 18 times the execution time necessary to a single CPU for the minimization part (Table 3).

The outcome we get from these experiments is that the algorithm scales up according to the performance analysis (see Fig. 2). Indeed, as expected, as QP increases, the scale

Table 1 The amount of memory required to store data related to each subdomain on PE_2 expressed in Mbyte

n_{loc}	32	40	48	56	64	72	80	88
$Data_{mem}(Mbyte)$	177	286	485	812	1313	2041	3057	4427

Table 2 Values of the speed-up s^{loc}_{nproc} in terms of gain obtained by using the GPU versus the CPU

n_{loc}	32	40	48	56	64	72	80	88
$\frac{T_{blas}}{T_{cublas}}$	15.3	17.5	18.08	19.0	19.8	20.2	22.5	20.54

The CUBLAS routines allow to reduce in average by 18 times the execution time necessary to a single CPU for the minimization part

Table 3 Weak scalability of one iteration of the parallel algorithm $\mathcal{A}^{DD}_{4DVar}$ with $n_{loc} = 32$ computed using the measured software Scale-up $Sc^{meas}_{1,QP}$ defined in (57)

QP	2	4	8	16	32	64
Problem size	$6.1 \cdot 10^3$	1.2×10^4	$2.4 \cdot 10^4$	$4.9 \cdot 10^4$	$9.8 \cdot 10^4$	1.9×10^5
$Sc^{meas}_{1,QP}$	$3.3 \cdot 10^0$	$1.54 \cdot 10^1$	$5.41 \cdot 10^1$	$1.23 \cdot 10^2$	$2.30 \cdot 10^2$	3.2×10^2

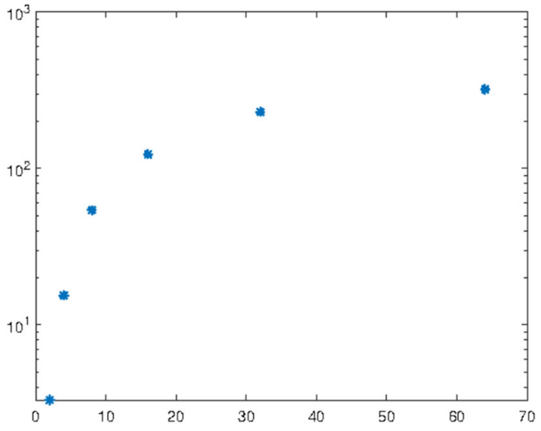

Fig. 2 Weak scalability of one iteration of the parallel algorithm $\mathcal{A}^{DD}_{4DVar}$ with $n_{loc} = 32$ computed using the measured software Scale-up $Sc^{meas}_{1,QP}$ defined in (57)

up factor increases and the surface-to-volume ratio increases, too, so that performance gain tends to become stationary. This the inherent tradeoff between speed up and efficiency of any software architecture.

8 Conclusions

We provided the whole computational framework of a space-time decomposition approach, including the mathematical framework, the numerical algorithm and, finally, its performance validation. We measure performance of the algorithm using a simulation case study based on the SWEs on the sphere. Results presented here are just a starting point towards the assessment of the software scalability. More precisely, we introduce simplifications and assumptions appropriate for a proof-of-concept study in order to get values of the measured scale up of the one iteration of the parallel algorithm. Anyway, the overall insight we get from these experiments is that the algorithm scales up according to the performance analysis. We are currently working on the development of a flexible framework ensuring efficiency and code readability, exploiting future technologies and equipped with a quantitative assessment of scalability [57]. In this regard, we could combine the proposed approach with the PFASST algorithm. Indeed, PFASST could be concurrently employed as local solver of each reduced-space PDE-constrained optimization subproblem, exposingeven more temporal parallelism.

This framework will allow designing, planning and running simulations to identify and overcome the limits of this approach.

Acknowledgements This work was developed within the research activity of the H2020-MSCA-RISE-2016 554 NASDAC Project N. 691184. This work has been realized thanks to the use of the S.Co.P.E. computing infrastructure at the University of Naples.

Funding The authors have not disclosed any funding.

Data Availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest The authors have not disclosed any competing interests.

References

1. Antil, H., Heinkenschloss, M., Hoppe, R.H., Sorensen, D.C.: 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)
2. Amaral, S., Allaire, D., Willcox, K.: A decomposition-based approach to uncertainty analysis of feed-forward multicomponent systems. Int. J. Numer. Methods Eng. **100**(13), 982–1005 (2014)
3. Arcucci, R., D'Amore, L., Pistoia, J., Toumi, R., Murli, A.: On the variational data assimilation problem solving and sensitivity analysis. J. Comput. Phys. **335**, 311–326 (2017)
4. Arcucci, R., D'Amore, L., Carracciuolo, L., Scotti, G., Laccetti, G.: A decomposition of the tikhonov regularization functional oriented to exploit hybrid multilevel parallelism. J. Parallel Program. **45**(5), 1214–1235 (2017)
5. Clerc, S.: Etude de schemas decentres implicites pour le calcul numerique en mecanique des fluides, resolution par decomposition de domaine. Ph.D. thesis, Univesity Paris VI (1997)
6. Constantinescu, E., D'Amore L.: A mathematical framework for domain decomposition approaches in 4D VAR DA problems. H2020-MSCA-RISE-2015-NASDAC project, Report 12-2016, https://doi.org/10.13140/RG.2.2.34627.20002
7. D'Amore, L., Arcucci, R., Carracciuolo, L., Murli, A.: A scalable approach to three dimensional variational data assimilation. J. Sci. Comput. (2014). https://doi.org/10.1007/s10915-014-9824-2
8. Daget, N., Weaver, A.T., Balmaseda, M.A.: Ensemble estimation of background-error variances in a three-dimensional variational data assimilation system for the global ocean. Q. J. R. Meteorol. Soc. **135**, 1071–1094 (2009)
9. D'Amore, L., Arcucci, R., Carracciuolo, L., Murli, A.: A scalable variational data assimilation. J. Sci. Comput. **61**, 239–257 (2014)
10. D'Amore, L., Laccetti, G., Romano, D., Scotti, G.: Towards a parallel component in a GPU–CUDA environment: a case study with the L-BFGS Harwell routine. J. Comput. Math. **93**(1), 59–76 (2015)
11. D'Amore, L., Carracciuolo, L., Constantinescu, E.: Validation of a PETSc based software implementing a 4DVAR data assimilation algorithm: a case study related with an oceanic model based on shallow water equation. Oct. 2018 arXiv:1810.01361v2
12. Dennis, J.E., Jr., Moré, J.J.: Quasi-Newton methods, motivation and theory. SIAM Rev. **19**(1), 46–89 (1977)
13. Dennis, J.E., Jr., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equation. SIAM, Philadelphia (1996)
14. Emmett, M., Minion, M.L.: Toward an efficient parallel in time method for partial differential equations. Commun. Appl. Math. Comput. Sci. **7**, 105–132 (2012)

15. ECMWF Ocean ReAnalysis ORA-S3. Avalaible to: http://icdc.cen.uni-hamburg.de/projekte/easy-init/easy-init-ocean.html
16. Fischer, M., Gurol, S.: Parallelization in the time dimension of the four dimensional variational data assimilation. https://doi.org/10.1002/qj:2996
17. Flatt, H.P., Kennedy, K.: Performance of parallel processors. Parallel Comput. **12**, 1–20 (1989)
18. Gander, M.J.: 50 years of time parallel time integration. In: Carraro, T., Geiger, M., Körkel, S., Rannacher, R. (eds.) Multiple Shooting and Time Domain Decomposition Methods: MuS-TDD, pp. 69–113. Springer International Publishing, Heidelberg (2015)
19. Gander, M.J., Kwok, F.: Schwarz methods for the time-parallel solution of parabolic control problems. Lect. Notes Comput. Sci. Eng. **104**, 207–216 (2016)
20. Giering, R., Kaminski, T.: Recipes for adjoint code construction. ACM Trans. Math. Softw. **24**(4), 437–474 (1998)
21. Gratton, S., Lawless, A.S., Nichols, N.K.: Approximate Gauss–Newton methods for nonlinear least square problems. SIAM J. Optim. **18**(1), 106–132 (2007)
22. Gunther, S., Gauger, N.R., Schroder, J.B.: A non-intrusive parallel-in-time approach for simultaneous optimization with unsteady PDEs. arXiv:1801.06356v2 [math.OC] 28 Feb (2018)
23. Gurol, S., Weaver, A.T., Moore, A.M., Piacentini, A., Arango, H.G., Gratton, S.: B-preconditioned minimization algorithms for variational data assimilation with the dual formulation. Q. J. R. Metereol. Soc. **140**, 539–556 (2014)
24. Lawless, A.S., Gratton, S., Nichols, N.K.: On the convergence of incremental 4D-Var using non tangent linear models. Q. J. R. Meteorol. Soc. **131**, 459–476 (2005)
25. Le Dimet, F.X., Talagrand, O.: Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. Tellus **38A**, 97–110 (1986)
26. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Q. Appl. Math. **2**(2), 164–168 (1944)
27. Liao, Q., Willcox, K.: A domain decomposition approach for uncertainty analysis. SIAM J. Sci. Comput. **37**(1), A103–A133 (2015)
28. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Program. **45**, 503–528 (1989)
29. Liu, J., Wang, Z.: Efficient time domain decomposition algorithms for parabolic PDE-constrained optimization problems. Comput. Math. Appl. **75**(6), 2115–2133 (2018)
30. Marquardt, D.W.: An algorithm for the least-squares estimation of nonlinear parameters. SIAM J. Appl. Math. **11**(2), 431–441 (1963)
31. Miyoshi, T.: Computational challenges in big data assimilation with extreme-scale simulations, talk at BDEC workshop. Charleston, SC (2013)
32. Moore, A.M., Arango, H.G., Broquet, G., Powell, B.S., Weaver, A.T., Zavala-Garay, J.: The regional ocean modeling system (ROMS) 4-dimensional variational data assimilation systems: I-system overview and formulation. Prog. Oceanogr. **91**, 34–49 (2011)
33. Moore, A.M., Arango, H.G., Broquet, G., Edwards, C.A., Veneziani, M., Powell, B.S., Foley, D., Doyle, J.D., Costa, D., Robinson, P.: The regional ocean modeling system (ROMS) 4-dimensional variational data assimilation systems: II performance and application to the California current system. Prog. Oceanogr. **91**, 50–73 (2011)
34. Moore, A.M., Arango, H.G., Broquet, G., Edwards, C.A., Veneziani, M., Powell, B.S., Foley, D., Doyle, J.D., Costa, D., Robinson, P.: The regional ocean modeling system (ROMS) 4-dimensional variational data assimilation systems: III observation impact and observation sensitivity in the California current system. Prog. Oceanogr. **91**, 74–94 (2011)
35. Moore, A.M., Arango, H.G., Di Lorenzo, E., Cornuelle, B.D., Miller, A.J., Neilson, D.J.: A comprehensive ocean prediction and analysis system based on the tangent linear and adjoint of a regional ocean model. Ocean Model. **7**, 227–258 (2004)
36. Murli, A., D'Amore, L., Laccetti, G., Gregoretti, F., Oliva, G.: A multi-grained distributed implementation of the parallel block conjugate gradient algorithm. Concurr. Comput. Pract. Exp. **22**(15), 2053–2072 (2010)
37. Navon, I.M., De Villiers, R.: The application of the Turkel-Zwas explicit large time-step scheme to a hemispheric barotropic model with constraint restoration. Mon. Weather Rev. **115**(5), 1036–1052 (1987)
38. Navon, I.M., Yu, J.: Exshall: a Turkel-Zwas explicit large time-step FORTRAN program for solving the shallow-water equations in spherical coordinates. Comput. Geosci. **17**(9), 1311–1343 (1991)
39. Nerger, L., Hiller, W.: Software for ensemble-based data assimilation systems: implementation strategies and scalability. Comput. Geosci. **55**, 110–118 (2013)

40. Neta, B., Giraldo, F.X., Navon, I.M.: Analysis of the Turkel-Zwas scheme for the two-dimensional shallow water equations in spherical coordinates. J. Comput. Phys. **133**(1), 102–112 (1997). https://doi.org/10.1006/jcph.1997.5657
41. PDAF https://pdaf.awi.de
42. NEMO Web page www.nemo-ocean.eu
43. Nichols, N.K.: Mathematical concepts of data assimilation. In: Lahoz, W., Khattatov, B., Menard, R. (eds.) Data Assimilation: Making Sense of Observations, pp. 13–40. Springer, Cham (2010)
44. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer-Verlag, Cham (1999)
45. Nocedal, J., Byrd, R.H., Lu, P., Zhu, C.: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Trans. Math. Softw. **23**(4), 550–560 (1997)
46. Nvidia.: TESLA K20 GPU active accelerator. Board spec. (2012) Available http://www.nvidia.in/content/PDF/kepler/Tesla-K20-Active-BD-06499-001-v02.pdf
47. https://parallel-in-time.org/
48. PCIsig, tecnology specifications at https://pcisig.com/specifications/pciexpress/
49. Rao, V., Sandu, A.: A time-parallel approach to strong constraint four dimensional variational data assimilation. J. Comput. Phys. **313**, 583–593 (2016)
50. ROMS Web page www.myroms.org
51. Shchepetkin, A.F., McWilliams, J.C.: The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. Ocean Model. **9**, 347–404 (2005)
52. St-Cyr, A., Jablonowski, C., Dennis, J.M., Tufo, H.M., Thomas, S.J.: A comparison of two shallow water models with nonconforming adaptive grids. Mon. Weather Rev. **136**, 1898–1922 (2008)
53. Ulriq, S.: Generalized SQP methods with "Parareal" time-domain decomposition for time-dependent PDE-constrained optimization. In: Biegler, L.T., Ghattas, O., Heinkenschloss, M., Keyes, D., van Bloemen Waanders, B. (eds.) Real-Time PDE-Constrained Optimization. SIAM, Philadelphia (2017)
54. Arcucci R., Carracciuolo L., D'Amore L.: On the problem-decomposition of scalable 4D-Var Data Assimilation models, Proceedings of the 2015 International Conference on High Performance Computing and Simulation, HPCS 2015 pp. 589–594, 2 September 2015 Article number 7237097 13th International Conference on High Performance Computing and Simulation, HPCS 2015Amsterdam20 July 2015 through 24 July 2015
55. D'Amore L., Marcellino L., Mele V., Romano D.: Deconvolution of 3D fluorescence microscopy images using graphics processing units, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) Volume 7203 LNCS, Issue PART 1, pp. 690–699, 2012 9th International Conference on Parallel Processing and Applied Mathematics, PPAM 201111 September 2011 through 14 September 2011
56. D'Amore L., Casaburi D., Galletti A., Marcellino., Murli A.: Integration of emerging computer technologies for an efficient image sequences analysis. Integ. Comput. Aided Eng. **18**(4), 365–378, https://doi.org/10.3233/ICA-2011-0382 (2011)
57. Murli, A., Boccia, V., Carracciuolo, L., D'Amore, L., Laccetti, G., Lapegna, M.: Monitoring and migration of a PETSc-based parallel application for medical imaging in a grid computing PSE. IFIP International Federation for Information Processing, vol. 239, pp. 421–432 (2007)