Computers and Geosciences 124 (2019) 103–114

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Parallel implementation of a data assimilation scheme for operational oceanography: The case of the MedBFM model system

A. Teruzzi[a,1], P. Di Cerbo[a,2], G. Cossarini[a,3], E. Pascolo[b,4], S. Salon[a,*,5]

[a] OGS - Istituto Nazionale di Oceanografia e di Geofisica Sperimentale, Via Beirut 2-4, 34151, Trieste, Italy
[b] CINECA, Via Magnanelli 6/3, Casalecchio di Reno, BO, 40033, Italy

A B S T R A C T

The MedBFM model system provides forecasts and reanalysis of the Mediterranean Sea biogeochemistry for the European Copernicus Services. The system integrates model and observations through a 3D variational assimilation scheme, whose performance capitalizes on present HPC systems and ensures compliance with the service requirements. Domain decomposition with message passing paradigm was implemented to parallelize the assimilation code and maximize performance and scalability. In particular, the parallelization of the horizontal recursive filtering algorithm was implemented using a dynamic sliced domain decomposition. Moreover, the efficient parallel solver of the PETSc/TAO library was adopted for optimizing the cost function minimization on which the variational assimilation is based. Considering the complex domain decomposition of the Mediterranean Sea and the high variability of the data input, a modern and scalable software application for variational assimilation schemes that exploits the performance of modern HPC architectures on the whole node was developed.

1. Introduction

Numerical models are powerful tools for the investigation of geophysical processes, providing a full-domain representation of the temporal evolution of three-dimensional fields. However, models are intrinsically affected by errors owing to approximations in their mathematical formulation. On the other hand, observations are often discontinuous in time and space and affected by measurement errors.

Data Assimilation (DA) is a technique widely used in geophysical sciences to optimally integrate information provided by observations into a prediction model. DA algorithms are extensively used, for example, in numerical weather prediction and operational oceanography (Ghil and Malanotte-Rizzoli, 1991; Bertino et al., 2003; Rabier, 2005). Among operational applications, DA in biogeochemical fields is relatively recent (Teruzzi et al., 2014; Ford et al., 2012) and is relevant for the absence of standardized formulation of the governing equations and the lack of extensive observational data availability.

Two main approaches are typically used in DA applications: Kalman filters and variational schemes. Owing to the high dimensionality of biogeochemical oceanography, the application of Kalman filters requires the use of approximate schemes (Cossarini et al., 2009; Ciavatta et al., 2016; Tsiaras et al., 2017). The variational approach is less common to assimilate biogeochemical fields (Teruzzi et al., 2014), however, it is quite frequently adopted in numerical weather prediction (Bannister, 2017) and physical oceanography (Bennett et al., 2008; Dobricic and Pinardi, 2008).

A recent example of a successful implementation of biogeochemical variational DA is the 3DVarBio code (Teruzzi et al., 2014, 2018), that adopts a three-dimensional variational scheme in the MedBFM operational system for the assimilation of surface chlorophyll concentration estimated by satellite. MedBFM integrates 3DVarBio with the coupled physical-biogeochemical model OGSTM-BFM (Lazzari et al., 2010, 2012), based on a medium complexity biogeochemical model (Lazzari et al., 2016; Cossarini et al., 2015) that describes the plankton, bacteria and nutrient dynamics. MedBFM provides the short-term forecast and the reanalysis of the Mediterranean Sea biogeochemistry for the Copernicus Marine Environment Monitoring Services (CMEMS[6]). In this framework, a 10-day forecast of biogeochemical variables (chlorophyll, nitrates, phosphates, phytoplankton biomass, primary production, dissolved oxygen, ocean pH, and pCO2) is released weekly (Bolzon et al.,

* Corresponding author.
E-mail addresses: ateruzzi@inogs.it (A. Teruzzi), pierluigi.dicerbo@gmail.com (P. Di Cerbo), gcossarini@inogs.it (G. Cossarini), eric.pascolo@cineca.it (E. Pascolo), ssalon@inogs.it (S. Salon).

[1] 3DVarBio code developer and manager; code implementation; results analysis and discussion; manuscript drafting.
[2] Development and implementation of parallel version of 3DVarBio; results analysis and discussion; manuscript drafting.
[3] Development of ideas for parallel 3DVarBio; results discussion and presentation.
[4] Implementation of parallel version of 3DVarBio; results discussion.
[5] Results discussion and presentation; manuscript critical revision.
[6] http://marine.copernicus.eu.

https://doi.org/10.1016/j.cageo.2019.01.003
Received 1 March 2018; Received in revised form 6 November 2018; Accepted 7 January 2019
Available online 09 January 2019

2017), and a reanalysis (i.e., a multiyear simulation carried out by the most recently validated operational forecasting system) is performed annually (Teruzzi et al., 2016). Operational systems rely on the computational efficiency of the numerical codes embedded in their workflows to provide timely service to users, and the time-to-solution of 3DVarBio may hamper the performance of MedBFM.

The variational approach relies on the minimization of a cost function (Lorenc, 1986), and crucial issues for its implementation concern the definition and the inversion of the background error covariance matrix, which accounts for the uncertainty of the model variables and their covariance. In geophysical problems, the covariance matrix is proportional to the square of the spatial size of the model domain multiplied by the number of state variables, so it may have large dimensions. Using the approach proposed in Dobricic and Pinardi (2008), the background covariance matrix can be factorised into a series of operators, allowing to define the background error covariance matrix in a modular framework, where each operator describes a different aspect of the covariance. The operators may be matrices, diagnostic models, or even dynamical models that can be developed separately. Using the factorization, the inversion of the background error covariance matrix is no longer required for the evaluation of the cost function. However, the computational cost of this approach is still high for the typical dimensions and constraints of operational oceanography applications and foreseen increases of spatial resolution (Guest et al., 2012). Consequently, the parallelization of 3DVarBio is critical in improving the performance of the MedBFM system within the CMEMS requirements.

Parallelization of variational assimilation systems for realistic ocean modelling applications can be performed using the domain decomposition approach, as for example in Farina et al. (2015) and Moore et al. (2011). Dedicated studies demonstrated the scalability of the domain decomposition approach and its mathematical consistency when applied in variational assimilation (Arcucci et al., 2017a; D'Amore et al., 2018, 2014, 2012). On the other hand, it has been highlighted that scalability of variational approaches may be limited by the use of 2D recursive filters to model the horizontal correlation (Weaver et al., 2016).

This study describes the parallelization and the optimization of the DA scheme implemented in the 3DVarBio code, in which the error covariance matrix is factorised following Dobricic and Pinardi (2008) with the horizontal error covariance implemented as a 2D recursive filter. Domain decomposition with MPI was used, and the TAO module of the PETSc library (Balay et al., 2016a, b, 1997) was adopted for the cost function minimization, while, differently from previous works, the parallelization of the recursive filters was implemented using a dynamic sliced domain decomposition.

Hereafter, the original serial version of 3DVarBio code will be referred in the text as *original code*. The paper is organized as follows: Section 2 describes the minimization algorithm and the implementation of the solver provided by the TAO module. Section 3 presents the issues related to the parallelization and the adopted approach. In Section 4, the data sets and the computational facilities are first described; subsequently, the results are presented. In Section 5, a discussion on the results is provided, and the conclusions are drawn.

2. Minimization algorithm

2.1. Mathematical problem

In the 3DVarBio code the assimilated field (i.e., analysis) is obtained through the minimization of a cost function that is defined on the basis of Bayes' theorem (Lorenc, 1986). The cost function relies on the misfit between the model results and the observations, weighted according to their accuracy estimations, represented by the error covariance matrices $\mathbf{B}$ and $\mathbf{R}$, respectively (the complete formulation of the cost function is provided in Appendix A). In the present application, the size of the state vector $\mathbf{x}$ (i.e., the number of the model variables times the number of grid points) is nearly equal to 10^6, while the size of the observation vector $\mathbf{y}$ (surface chlorophyll map) is approximately equal to 10^3. Hence, the dimensions of $\mathbf{B}$ and $\mathbf{R}$ are $10^6 \times 10^6$ and $10^3 \times 10^3$ respectively. The cost function formulation (Eq. (A.1)) requires the inversion of both the error covariance matrices. Assuming that the observation errors are uncorrelated, the observation error matrix $\mathbf{R}$ is an easily invertible diagonal matrix, whilst the inversion of the covariance matrix $\mathbf{B}$ is computationally not affordable.

According to the scheme implemented in Dobricic and Pinardi (2008), the background error covariance $\mathbf{B}$ can be approximated by $\mathbf{B} = \mathbf{V}\mathbf{V}^{\intercal}$. Therefore, the cost function can be rewritten as

$$J(\mathbf{v}) = \frac{1}{2}\mathbf{v}^{\intercal}\mathbf{v} + \frac{1}{2}(\mathbf{d} - H\mathbf{V}\mathbf{v})^{\intercal}\mathbf{R}^{-1}(\mathbf{d} - H\mathbf{V}\mathbf{v}), \tag{1}$$

where

$$\delta\mathbf{x} = \mathbf{V}\mathbf{v}. \tag{2}$$

In this formulation, the inversion of $\mathbf{B}$ is no longer required, and the solution of the minimization is a vector $\mathbf{v}$ of size n depending on the definition of $\mathbf{V}$ (in our case order 10^5).

The solution $\mathbf{v}$ within the original 3DVarBio code is iteratively computed using a library that implements a quasi-Newton L-BFGS minimizer requiring the computation of the cost function gradient, namely,

$$\mathbf{J}'(\mathbf{v}) = \mathbf{v} - \mathbf{V}^{\intercal}H^{\intercal}\mathbf{R}^{-1}(\mathbf{d} - H\mathbf{V}\mathbf{v}). \tag{3}$$

Once the solution $\mathbf{v}$ is found, the final increment for the biogeochemical variables $\delta\mathbf{x}$ is evaluated by Eq. (2).

For the application of the 3D variational approach in the framework of biogeochemical DA a factorization of $\mathbf{V}$ based on the approach proposed by Dobricic and Pinardi (2008) has been applied in Teruzzi et al. (2014, 2018). In particular, in order to account of the main aspects of the background covariance matrix in biogeochemical applications, $\mathbf{V}$ is factorised as

$$\mathbf{V} = \mathbf{V}_b\mathbf{V}_h\mathbf{V}_v. \tag{4}$$

These three operators describe the vertical error covariance of the chlorophyll fields ($\mathbf{V}_v$), the horizontal error covariance ($\mathbf{V}_h$), and error covariance of the biogeochemical variables ($\mathbf{V}_b$). As it will be illustrated later, the $\mathbf{V}_h$ operator has a strong data dependency and is the most computationally expensive (details on its formulation are provided in Appendix B). The operators $\mathbf{V}_v$ and $\mathbf{V}_b$ are based, respectively, on synthetic vertical profiles of chlorophyll obtained by an empirical orthogonal function (EOF) decomposition applied to a multiyear run, and on a functional variance operator that accounts for phytoplankton growth status along with phytoplankton internal quotas and phytoplankton species relative quotas (Teruzzi et al., 2014, 2018). In other examples of variational DA EOFs were applied to the whole covariance matrix (Arcucci et al., 2017b).

With the described factorization of $\mathbf{V}$, the size n of the solution of the minimization of Eq. (1) is equal to the number of EOFs used in $\mathbf{V}_v$ multiplied by the number of grid points on which DA is applied.

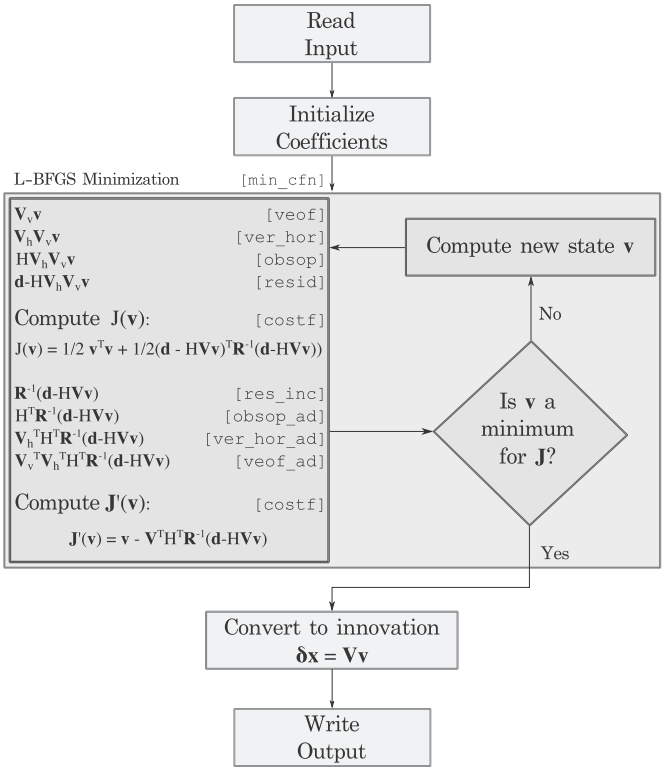


Fig. 1. Workflow scheme of the original version of the 3DVarBio code. Names of subroutines are in typewriter font within square brackets.

2.2. 3DVarBio code structure

The 3DVarBio code is written in Fortran 90 and consists of 55 subroutines. The workflow of the subroutine-call sequence is shown in Fig. 1.

The first group of subroutines (not shown in Fig. 1) initializes the components required for the computation. Namely, they load the observation data, read the grid structure, initialize the coefficients required for the computation, and perform other minor operations. Then, the minimum of the cost function is computed by calling the corresponding library (min_cfn subroutine). The solver is iteratively called until an exit condition is satisfied and requires at each iteration the value of the cost function (Eq. (1)) and its gradient (Eq. (3)). The subroutine costf splits the calculation of the cost function (Eq. (1)) into different subroutines:

- veof: Applies the operator $\mathbf{V}_v$ to the state vector $\mathbf{v}$.
- ver_hor: Applies the operator $\mathbf{V}_h$ to the output of the veof subroutine, yielding the product $\mathbf{V}\mathbf{v}$ in Eq. (1).
- obsop: Applies the observational operator to obtain the product $H\mathbf{V}\mathbf{v}$.
- resid: Computes the difference $\mathbf{d} - H\mathbf{V}\mathbf{v}$.

The gradient (Eq. (3)) is computed through another series of subroutines (adjoint subroutines), that apply the adjoint operators:

- res_inc: Initializes quantities for adjoint calculations; the initial vector is $\mathbf{R}^{-1}(\mathbf{d} - H\mathbf{V}\mathbf{v})$.
- obsop_ad: Applies the adjoint observational operator to obtain $H^T\mathbf{R}^{-1}(\mathbf{d} - H\mathbf{V}\mathbf{v})$.
- ver_hor_ad: Applies the $\mathbf{V}_h^T$ operator to obtain $\mathbf{V}_h^TH^T\mathbf{R}^{-1}(\mathbf{d} - H\mathbf{V}\mathbf{v})$.
- veof_ad: Applies the $\mathbf{V}_v^T$ operator to obtain $\mathbf{V}^TH^T\mathbf{R}^{-1}(\mathbf{d} - H\mathbf{V}\mathbf{v})$.

Among the 3DVarBio subroutines, the most time-consuming ones are ver_hor and ver_hor_ad (see Code Box 1), which require almost 90% of the total computation time (profiling not shown). These subroutines solve the horizontal covariance operator $\mathbf{V}_h$, which is composed of a series of recursive Gaussian filters (Dobricic and Pinardi, 2008) applied in forward and backward direction in the x (longitude) and y (latitude) direction (the details of the recursive filters formulation are resumed in Appendix B).

2.3. Characteristics of the solver

The original version of the 3DVarBio code uses the L-BFGS-B library version 2.1 (Zhu et al., 1997), implementing a limited memory quasi-Newton algorithm for solving large nonlinear optimization problems with simple bounds on the variables. The L-BFGS-B library features the BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm, which is widely used for approximating the Hessian matrix and constraining the solution (Malouf, 2002). The quadratic form of the cost function ensures that using the BFGS method the solution converges.

The minimization solver of the TAO module within the PETSc library (Balay et al., 2016a, b, 1997) was adopted to parallelize the BFGS algorithm in the new code, as it was natively coded for parallel execution. In particular, the Limited Memory Variable Metric solver (Munson et al., 2015) which implements a quasi-Newton optimization based on the BFGS formula was chosen, ensuring the reproducibility of the results with respect to the original version.

3. Computational methods and implementation

3.1. Dynamic domain decomposition for filtering

Domain decomposition exploits the computing power of a parallel computer and relies on a partition of the computational domain into sub-domains, which are assigned to processes that store only a portion of the entire data structures. Ideally, each process performs the operations required for updating its part of the domain, and the output is the composition of all local results. This programming paradigm was also chosen for the parallelization of the OGSTM code, using MPI (Lazzari et al., 2010). Domain decomposition in the 3DVarBio code cannot be easily implemented, as the filter has a strong data dependency. Indeed, the application of the filter in a specific direction, x or y, requires information of the entire row or column (intra-row dependency; Eq. (B.2)), while, the result of filtering on a specific row does not depend on the other rows (inter-row independency). This data-dependency structure motivates the application of a sliced domain decomposition, which must dynamically change to allow the filter application on the other direction after the application of the filter in the x (or y) direction.

3.2. Implementation

Figure 2a shows a domain decomposition of the Mediterranean Sea using four processes. The axes are defined with the origin set in the bottom left corner, the x axis right-oriented (eastwards) and the y axis upwards (northwards). The sub-domains are assigned starting from the top (the northernmost area belongs to process zero). Initially, each process has a portion of rows (Fig. 2a), on which the filter is separately

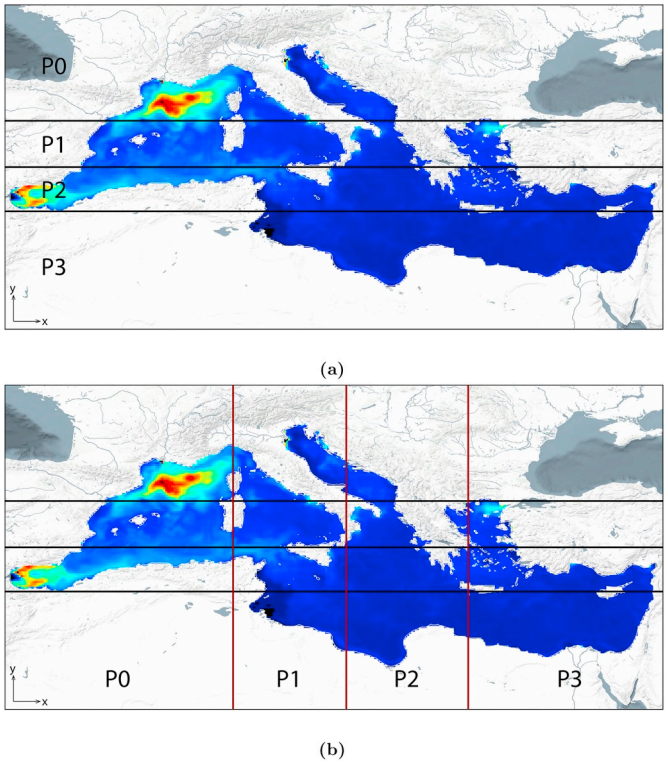


Fig. 2. Dynamic domain decomposition. a) Example of a generic *x*-domain decomposition with four processes: this decomposition allows filtering along the *x* direction. b) Overlap among the *x*-domain decomposition (black lines) and the *y*-domain decomposition (red lines). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

applied along the *x* direction (*x*-decomposition). Then, when the filter is applied along the *y* direction, the domain decomposition must change as shown in Fig. 2b (red lines) with each process having a portion of the columns of the domain. Therefore, to obtain the new decomposition (and return to the previous one), each process must communicate with the others, exchanging parts of sub-domains. All the processes are involved in the communication sending a part of their sub-domain to the other processes and receiving the remaining parts.

The MPI_Alltoallv function is used to execute the dynamical switch among decompositions. This function implements *collective* communication, acting on a group of processes and involving all the processes of the group, and allows to dynamically reconstruct the domain for filter application (Schulz, 2008). Indeed, the MPI_Alltoallv function takes as input an array, divides it into parts, and sends the first part to process zero, the second to process one, etc., with an analogous procedure occurring in data gathering.

The dynamic domain decomposition was implemented by suitably executing the MPI_Alltoallv function before the filtering operations. The implementation of the filtering is provided in Code Box 1, which shows the structure of the ver_hor before parallelization. The calling sequence shown in Code Box 1 consists of subroutines that implement the *recursive filters*: the direct application of the filters $((\mathbf{W}_y\mathbf{G}_y\mathbf{W}_x\mathbf{G}_x)_{dir}$ in Eq. (B.1)) is implemented by the execution of the subroutines rcfl_x_dir and rcfl_y_dir, resulting in the grd%chl_dir field. The second term of Eq. (B.1), i.e., $(\mathbf{W}_x\mathbf{G}_x\mathbf{W}_y\mathbf{G}_y)_{rev}$, is implemented by rcfl_y_rev and rcfl_x_rev, resulting in the grd%chl_rev field. Following the direct and reverse application of the filters, the 3D fields grd%chl_dir and grd%chl_rev are estimated. Each rcfl subroutine implements a recursive filter that acts in forward and backward directions (see Eq. (B.2)), where the first three entries are the global grid dimensions (grd%i, grd%j, and grd%k), while the others are coefficients required for recursive filtering (grd%aex and grd%aey). For instance, grd%aex provides the value of $\alpha(x, y, z)$ of Eq. (B.2) for the application of filter $\mathbf{G_x}$.

Code Box 1. Pseudo-code for the ver_hor subroutine of the original code. Further details are in the text.

```
!
!... Initialize some quantities ....
!

! apply direct recursive filter along the x direction
call rcfl_x_dir(grd%i, grd%j, grd%k, grd%aex, grd%chl_dir)

!
! ... Perform some operations ...
!

! apply direct recursive filter along the y direction
call rcfl_y_dir(grd%i, grd%j, grd%k, grd%aey, grd%chl_dir)

!
! ... Perform some operations ...
!

! apply reverse recursive filter along y direction
call rcfl_y_rev(grd%i, grd%j, grd%k, grd%aey, grd%chl_rev)

!
! ... Perform some operations ...
!

! apply reverse recursive filter along x direction
call rcfl_x_rev(grd%i, grd%j, grd%k, grd%aex, grd%chl_rev)

! compose the results
do k=1,grd%k
   grd%chl(:,:,k) = 0.5 * (grd%chl_dir(:,:,k) + grd%chl_rev(:,:,k) )
enddo

!
!... Exiting from ver_hor ...
!
```

All the communications required to carry out the parallel filtering occur before the call to rcfl subroutines (rcfl_x_dir, rcfl_y_dir, rcfl_x_rev and rcfl_y_rev in Code Box 1). Therefore, these subroutines do not need to be modified to implement the 3DVarBio parallel version, and are executed in local sub-domains. As rcfl_y_dir and rcfl_y_rev are applied in sequence on two different arrays (grd%chl_dir and grd%chl_rev), a new MPI data type composed of the pairs (grd%chl_dir(i,j,k), grd%chl_rev (i,j,k)) is defined to minimize the number of MPI_Alltoallv executions. This allows the exchange of both arrays by a single MPI_Alltoallv call. Thus, starting with a domain decomposition that allows filtering along the *x* direction (*x*-decomposition), only two calls to MPI_Alltoallv function must be performed to execute the ver_hor subroutine. The first call shifts from the *x*-decomposition to the *y*-decomposition; the second returns to the *x*-decomposition and applies the rcfl_x_rev subroutine.

Code Box 2. Pseudo-code for the parallel version of ver hor subroutine. Further details are in the text.

```
! apply direct recursive filter along the x direction
call rcfl_x_dir(grd%i, grd%j, grd%k, grd%aex, grd%chl_dir)

!
! ... construct the array to send ...
!
call MPI_Alltoallv(MPI_Array)
!
! ... unpack the MPI_Alltoall output
!

! apply direct recursive filter along the y direction
call rcfl_y_dir(grd%i, grd%j, grd%k, grd%aey, grd%chl_dir)

! apply reverse recursive filter along y direction
call rcfl_y_rev(grd%i, grd%j, grd%k, grd%aey, grd%chl_rev)

!
! ... constuct the array to send ...
!
call MPI_Alltoallv(MPI_Array)
!
! ... unpack the MPI_Alltoall output
!

! apply reverse recursive filter along x direction
call rcfl_x_rev(grd%i, grd%j, grd%k, grd%aex, grd%chl_rev)

! compose the results
do k=1,grd%k
   grd%chl(:,:,k) = 0.5 * (grd%chl_dir(:,:,k) + grd%chl_rev(:,:,k) )
enddo

!
!... Exiting from ver_hor ...
!
```

The pseudo-code for the parallel version of the ver_hor subroutine is shown in Code Box 2. As the subroutine ver_hor_ad is similar to ver_hor, and the computation of the cost function requires the execution of both ver_hor and ver_hor_ad, four MPI_Alltoallv calls are performed for each call to costf.

3.3. Extended grids

Extended grids are structures that support the application of filtering operations (Dobricic and Pinardi, 2008). Indeed, the filter must be applied only on seawater cells, whereas land cells do not contribute to the computation. Therefore, the extended grids are composed by all the seawater cells and by a fixed number of additional cells for each coast point (defined as a land cell followed by a water cell or vice versa) preventing that filtering operation occurs between water cells separated by land cells.

In Fig. 3, a sketch of the extended grid construction for filtering in *x* and *y* direction is shown.

The number of cells in a given row of the extended grid is equal to the sum of the number of water cells in the row and of the number of additional cells (red cells), which depends on the number of coast points in the row.

4. Results

4.1. Computational setup

Implementation, testing, and performance analysis were carried out on a compute node of the *PICO* cluster installed at *CINECA*, the main Italian supercomputing centre. PICO is a Linux Infiniband cluster composed of 74 nodes with two Intel Xeon ten-core processors (E5-2670 v2) at 2.5 GHz capable of eight FPO per cycle. Intel hyper-threading technology is disabled. Each computing node is equipped with 128 GB of RAM with a maximum theoretical bandwidth of 59.7 GB/s, and all the nodes are interconnected through a Mellanox Infiniband FDR 56 Gb/s network, allowing for a low latency/high bandwidth interconnection.

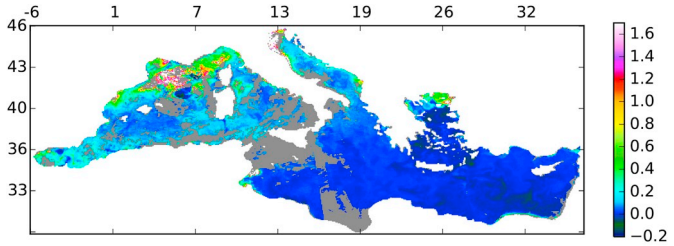
3DVarBio as well as the PETSc library were built using the Intel Fortran compiler from Intel Composer XE-2016 (version 16.0.0). The performance analysis and the communication profiling (presented in Section 5) were obtained using *Intel VTune Amplifier* and *Intel Trace Analyzer*, respectively.

4.2. Benchmark datasets

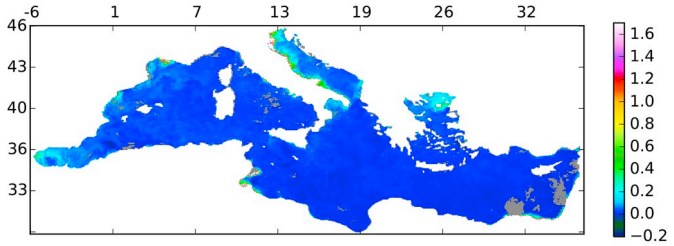
The 3DVarBio code was executed on a grid with dimensions 720 × 256 × 72 (1/16° horizontal resolution). The number of iterations for minimizing the cost function strongly depends on:

- the number of missing values of the satellite chlorophyll input (the cloud cover is higher in winter than in summer);
- the shape of the spatial gradients of the input, which depends on the variability of the chlorophyll patterns (higher in winter than in summer);

Fig. 3. Sketch of the extended grid construction for filtering along the *x* (bottom-right) and *y* (left) directions, with seawater cells (blue squares) and land cells (brown squares). In the extended grid, the red cells are added to prevent diffusion in the presence of coast points. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)



(a) April dataset.



(b) July dataset.

Fig. 4. Maps of misfit ($\mathbf{d}_k$, as defined in Section 2) of the April (top) and July (bottom) datasets. No data points are shown in grey. The axes represent the longitude and the latitude.

Table 1
List of simulation tests conducted for the 3DVarBio code.

Test name	Input data	Precision	Nodes	Processes
Original code	April/July	double	1	1
SingleApril	April	single	1	1, 2, 4–20
SingleJuly	July	single	1	1, 2, 4–20
DoubleApril	April	double	1	1, 2, 4–20
DoubleJuly	July	double	1	1, 2, 4–20
TwoNodes	April	double	2	2, 4–20
FourNodes	April	double	4	2, 4–20

- the heterogeneity of the vertical EOF profiles of the $\mathbf{V}_v$ operator, which vary on a monthly basis.

As it is not straightforward to evaluate the relation of these three factors to the number of iterations, two different misfit conditions (spring case and summer case, Fig. 4) were chosen as input for the simulations (see Table 1).

The spring case (23 April 2013, hereafter referred to as “April”) has a larger number of missing data points (grey regions) and higher spatial pattern heterogeneity compared with the summer case (16 July 2013, hereafter referred to as “July”). By contrast, the $\mathbf{V}_v$ operator for the July case exhibits higher vertical heterogeneity due to the non-uniform vertical distribution of chlorophyll in summer, characterised by the presence of a deep chlorophyll maximum.

4.3. Time-to-solution and scalability

Given the 3-D spatial heterogeneity of the domain, different choices for the domain decomposition were available. The results presented here were obtained using an equally distributed domain decomposition among the processes for x and y direction.

Several tests (Table 1) were performed in the April and July conditions using two precision versions of the code (single and double), and different node distributions (1, 2, or 4 nodes). We computed strong scalability, or speedup (also referred to as *fixed-size scaling*; Keyes, 1998), as the ratio between the single-process and the multiple-processes time-to-solution: for each test, the code was executed 10 times, and the mean value of the time-to-solution was considered (the time-to-solution in the case of a single process refers to the original version of the code).

Figure 5 shows that the time-to-solution generally decreases as the number of processes increases. The April case has a longer time-to-solution compared with the July case, and the single precision version of the code is always significantly faster than the double precision version. The best scaling obtained is 12.7 with 20 processes for the *SingleApril* dataset and 18.6 with 18 processes for the *SingleJuly* dataset. The corresponding values for the version of the code that employs double precision are 7.6 with 20 processes for April and 8.8 with 18 processes for July. Though a significant reduction of the time-to-solution is obtained with the parallelization (Fig. 5, top), code scalability does not result optimal, particularly for the double precision tests (Fig. 5, bottom).

In addition to the single node configuration (all processes equally distributed between the two sockets of the node), the processes were placed on two nodes and four nodes minimizing the number of processes per socket.

Figure 6 shows that for the different node configurations the time-to-solution differs with an increased number of processes. In particular, scalability does not improve in multiple-node configurations (except the case of 8 processes distributed over 2 nodes, bottom panel), with execution times even becoming 25 to 30% slower than the single-node configuration using 16 processes, while, with less than 8 processes, the execution times change little. Moreover, as shown in Fig. 7, a significant portion of the time-to-solution is spent in MPI communication and increases linearly as the number of processes increases (e.g., 30% of the time-to-solution is devoted to communication in the run with 20 processes).

The relevant decrease of time-to-solution in the single precision case along with the similar time-to-solution obtained with less than 8 processes in the multiple- and in the single-node configuration suggests that code performance is limited by a memory bandwidth issue. Indeed, at fixed memory bandwidth, in the single precision case each process doubles the amount of data available for processing with respect to the double precision case, resulting in increased code scalability. Moreover, the total time-to-solution would be expected to increase on multiple nodes considering the communication aspect. However, since the available bandwidth per process increases when decreasing the number of processes per node (even though the bandwidth of the Infiniband network connecting the nodes is one tenth slower than the intranode one), the time-to-solution is almost the same in the single- and multiple node configurations for less than 8 processes. When the number of processes is greater than 8, the time-to-solution is higher in the multiple-node case because the total network overload increases (more communications are executed, Fig. 7), and the narrow Infiniband bandwidth becomes relevant. Additional tests of performance analysis (not shown), carried out using *Intel VTune Profiler*, confirm that the code is significantly memory bound, and the optimal scaling is mainly slowed down by the memory bandwidth saturation.

All the results presented above were obtained assigning an equal number of rows (columns) to each process. We also tested an alternative domain decomposition balancing the workload among processes. The optimal load balancing was performed by an algorithm based on the extended grids (where the horizontal filters are applied) that precomputes two optimal thresholds for the x and for the y directions, respectively. As an example, using 10 processes the number of rows (columns) per each process ranged from 6% to 19% of the total number of rows (columns), and the maximum wait time decreased of nearly 50%. However, the time-to-solution with the balanced domain decomposition increases of about 12% with respect to the configuration with equal number of rows (columns) for each process. Indeed, the time per process spent in the MPI_Allreduce function increases significantly using the balanced domain decomposition, because of the application of the whole horizontal covariance operator$\mathbf{V}_h$ (see Eq. (B.1) in Appendix

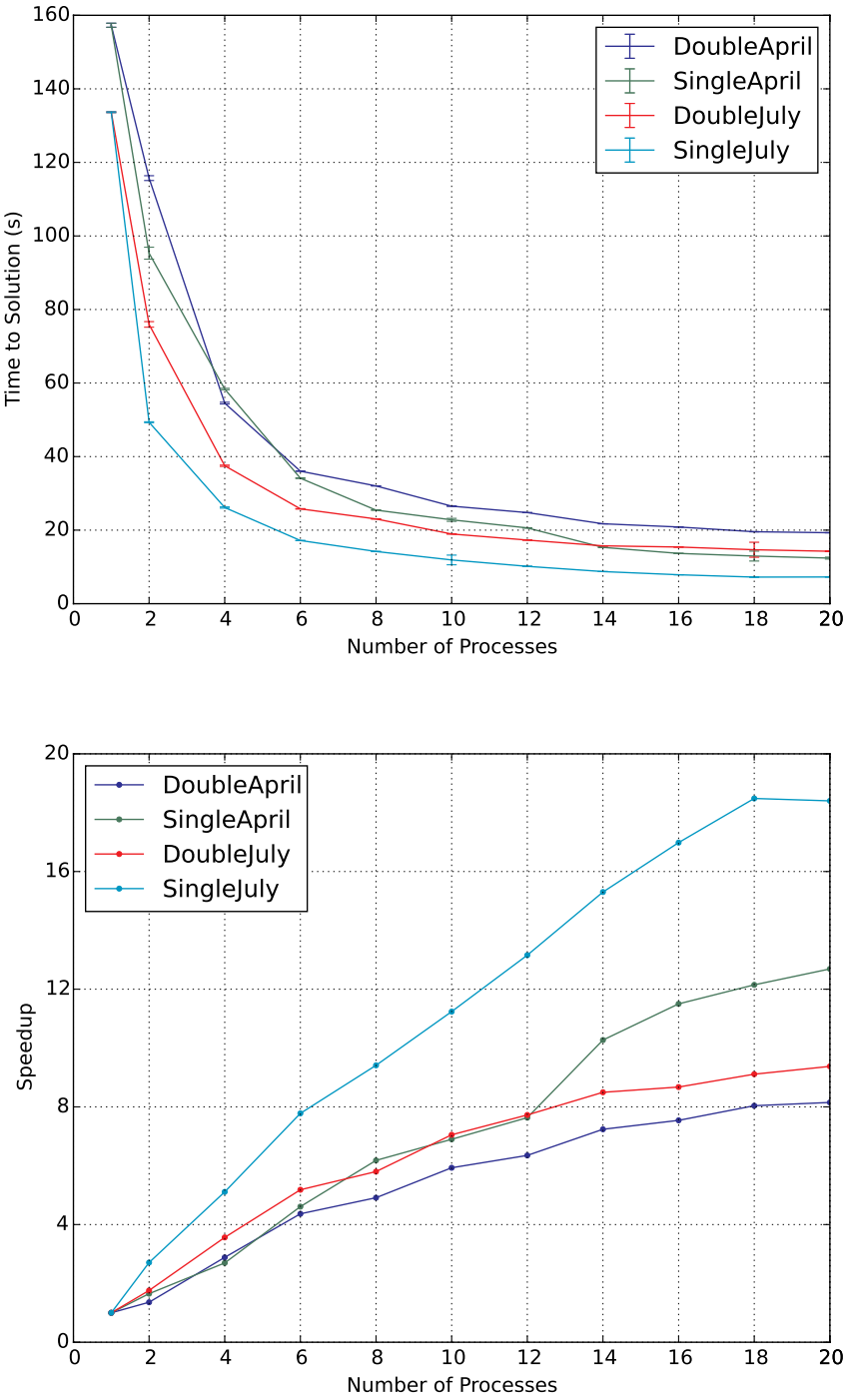


Fig. 5. Time-to-solution (top) and scalability (bottom) obtained using the two input datasets presented in the previous section for the two versions of the code (single and double precision). Only the configurations with an even number of processes are presented, whereas the time-to-solution in the case of a single process was obtained using the original version of the code.

B), that is composed by the two filters ($\mathbf{G}_x$ and $\mathbf{G}_y$) and the two normalizations ($\mathbf{W}_x$ and $\mathbf{W}_y$). While the application of the filters is performed on the balanced domain (with extended grids), normalizations are penalized, since they are performed on the unbalanced domain composed by seawater cells without extended grids.

5. Discussion and conclusion

The results shown in Fig. 5 indicate significant heterogeneity, showing that the best parallel configuration is dependent on the data input. However, the number of processes that minimizes the time-to-solution cannot be predicted prior to scaling analysis. The *SingleJuly* case for less than 20 processes is super-linear with respect to the original code, while in the other cases parallelization saves a large amount of computational time even though not following the optimal scaling. Moreover, we verified that a domain decomposition balanced on extended grids does not improve the code performance in terms of time-to-solution. Thus, for the present operational implementation of 3DVarBio, where execution time is critical, we decomposed the domain without a load balancing based on the extended grid, assigning the same number of rows (columns) to each process.

On the other hand, memory bandwidth has been identified as an

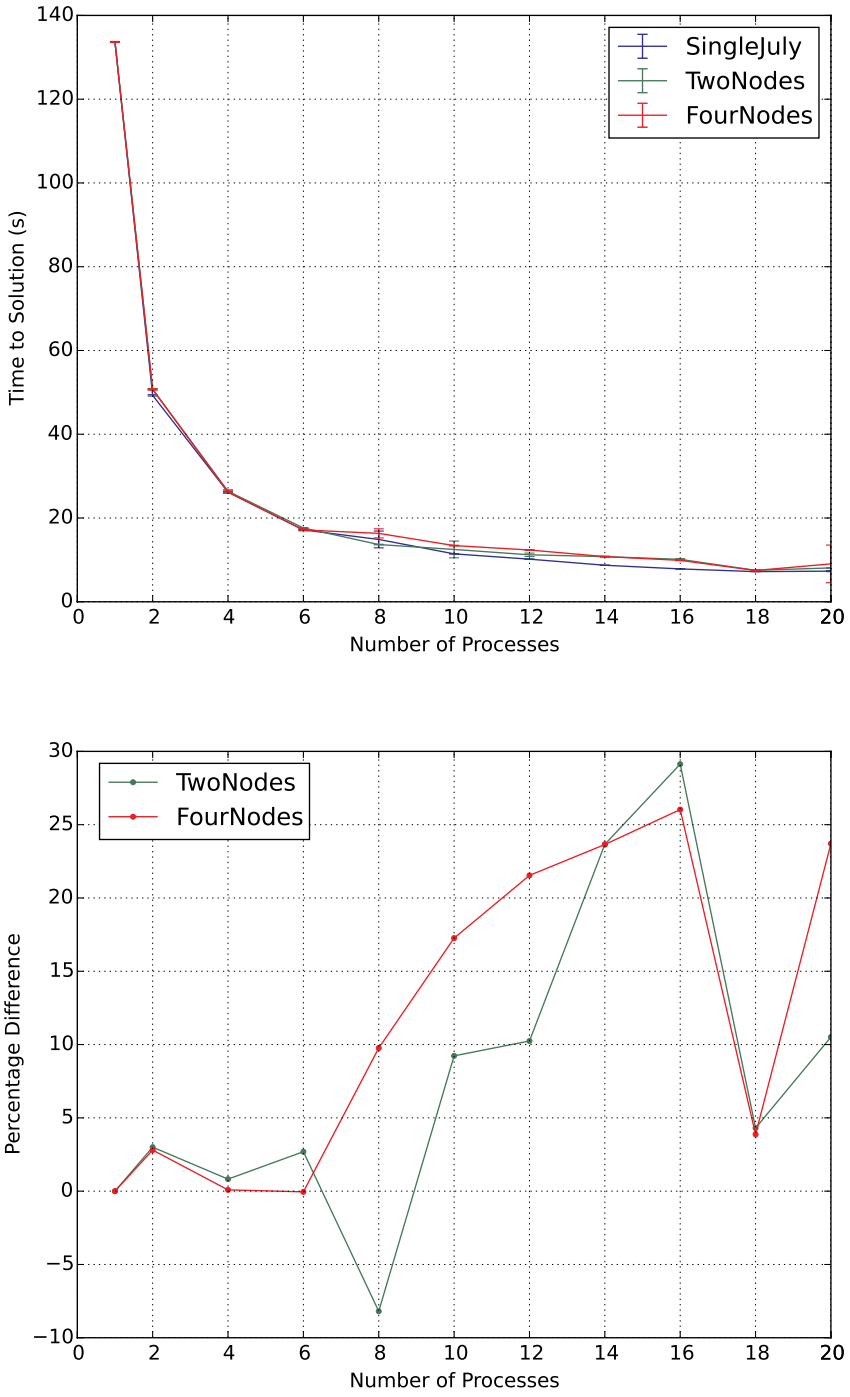


Fig. 6. Time-to-solution (top) obtained using different process configurations on multiple nodes. Bottom figure shows the percentage difference between these configurations and the *SingleJuly* code. The time-to-solution in the case of a single process is obtained using the original version of the code.

issue that can affect the code performance. Geophysical models based on finite differences are usually affected by memory bandwidth issues (e.g., Xu et al., 2014; Pascolo et al., 2016). Indeed, finite differences adopt stencil computations that update the array elements according to a fixed access pattern and are typically memory bound (Krishnamoorthy et al., 2007; Fuhrer et al., 2013). In our case the memory bandwidth issue was confirmed by the significant improvement of the code scalability when using single precision numbers (Fig. 5) and by comparison of single- and multiple-node configurations (Fig. 6). The maximum bandwidth achievable depends on the hardware components of the machine: the higher the number of processes, the more the processes compete for the bandwidth. Thus, our results show that bandwidth saturation limits the benefit of the reduction of the computational workload related to the implemented parallelization.

Literature provides examples of performance optimization of intra-node MPI collective communications in combination with shared memory approach as OpenMP, kernel assisted copy approach (e.g., the KNEM module[7]), or cache-oblivious implementations (Li et al., 2018) especially in the frame of many-core systems. Application of similar techniques is beyond the objectives of the present work, but they might represent promising future developments of 3DVarBio. Presently, our choice to adopt MPI was motivated by the parallelization scheme

[7] http://knem.gforge.inria.fr/.

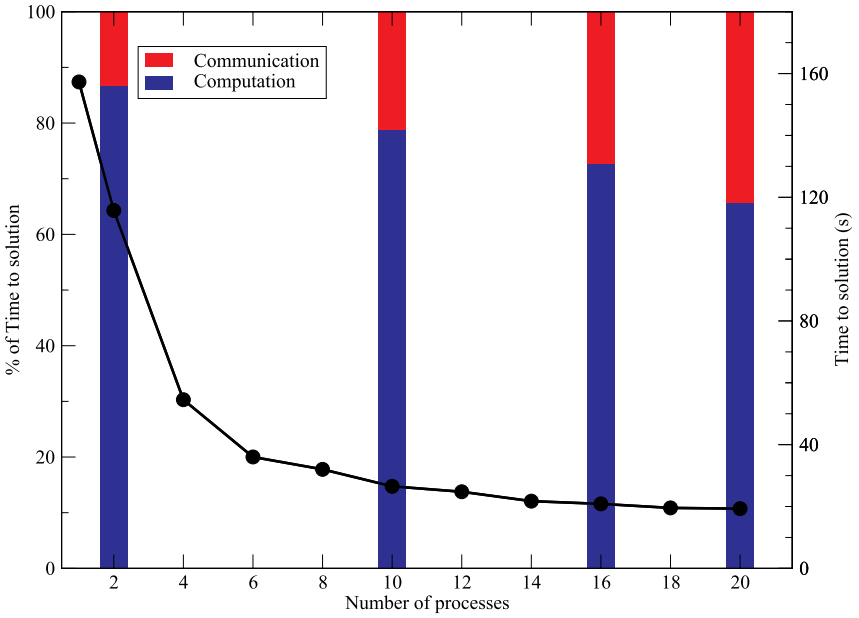


Fig. 7. Time-to-solution (black line and dots) and code profiling with different number of MPI processes shown as percentage of time-to-solution between computational time (blue) and MPI communication among the processes (red). Data refer to *DoubleApril* test. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

already applied in the MedBFM model system in which 3DVarBio is embedded, and by the suitability of MPI to be applied to eventual development of the codes (e.g., additional **V** operators).

Optimizing variational schemes in ocean science is a new field and the present approach is innovative in terms of the efficient parallel scheme implemented for the horizontal filtering. Parallel implementations of realistic applications of a 3D variational code with horizontal recursive filtering are proposed by Farina et al. (2015) and by D'Amore et al. (2012) using a static domain decomposition. In Farina et al. (2015) the communication among processes was performed exclusively using the MPI_send and MPI_recv functions, including the communication required to carry out filtering. Differently, in D'Amore et al. (2012) the parallelization of the filtering was based on independent operations calculated on 3 different quantities distributing data to 3 processors with maximum scalability nearly equal to 6 with 24 cores. The higher scalability obtained in our study shows that the use of dynamic domain decomposition allows to better exploit the full potential of parallelism with respect to a static domain decomposition approach in case of intrinsically serial algorithms, such as 3DVarBio filters, that may be handled using MPI_Alltoallv functionality (Jagode, 2006) for the reconstruction of the domain in row or column slices.

This study can be summarized as follows:

- The L-BFGS-B library used by the original version of the 3DVarBio code was updated with the minimization solver of the PETSc library, which allows parallel computation. Furthermore, a dynamic sliced domain decomposition was adopted, and implemented with the MPI_Alltoallv function.
- As a result, a maximum scaling of 18.7 (8.8) was obtained with 18 processes in single (double) precision.
- A comparison of single-node and multiple-node configurations showed that the execution time does not change significantly, demonstrating the memory bandwidth issues that affect the code.

Computer code availability

- Name of code: 3DVarBio
- Developers: Anna Teruzzi, Pierluigi Di Cerbo
- Contact details: OGS - Istituto Nazionale di Oceanografia e di Geofisica Sperimentale, Via Beirut 2/4, 34151 Trieste, Italy; Tel. +39 040 2140622; email: ateruzzi@inogs.it
- Year first available: 2018
- Hardware required: 3DVarBio was run on 20 cores of the PICO cluster located at CINECA (Bologna, Italy)
- Software required: 3DVarBio code was compiled with intel compiler and needs MPI, PETSC and NETCDF libraries
- Program language: the code is written in Fortran90
- Program size: 7.5 Mb
- Details on how to access the source code: the source files of the 3DVarBio code can be downloaded from github (see details on the Instruction Guide)

Acknowledgements

This work was carried out as part of the Copernicus Marine Environment Monitoring Service (CMEMS) MASSIMILI project. CMEMS is implemented by Mercator Ocean in the framework of a delegation agreement with the European Union. The research reported herein was partially supported by the Italian Ministry of Education, Universities and Research (MIUR) through OGS and CINECA under HPC-TRES program grant number 2015-02 awarded to P. Di Cerbo. This study was conducted using E.U. Copernicus Marine Service Information.

Appendix A. Cost function formulation

The variational assimilation is based on the minimization of a cost function, that at each assimilation step is defined as

$$J(\delta\mathbf{x}) = \frac{1}{2}\delta\mathbf{x}^{\mathrm{T}}\mathbf{B}^{-1}\delta\mathbf{x} + \frac{1}{2}(\mathbf{d} - H\delta\mathbf{x})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{d} - H\delta\mathbf{x}), \tag{A.1}$$

where $\delta\mathbf{x} = \mathbf{x}^a - \mathbf{x}^f$ is the correction made by the assimilation to the model forecast and $\mathbf{d} = \mathbf{y} - H(\mathbf{x}^f)$ is the *misfit vector* between the observation and the model forecast. The term H is the observational operator, which transforms the state vector $\mathbf{x}$ into the observation space, and in the present application is assumed to be linear. The matrices $\mathbf{B}$ and $\mathbf{R}$ are the model and observation error covariance matrices, respectively.

Appendix B. Horizontal covariance operator

The horizontal covariance operator $\mathbf{V}_h$ has been designed as a Gaussian recursive filter operating along x and y directions:

$$\mathbf{V}_h = \frac{1}{2}((\mathbf{W}_y\mathbf{G}_y\mathbf{W}_x\mathbf{G}_x)_{dir} + (\mathbf{W}_x\mathbf{G}_x\mathbf{W}_y\mathbf{G}_y)_{rev}), \tag{B.1}$$

where $\mathbf{G}_x$ and $\mathbf{G}_y$ represent the recursive filter operators in the x and y directions, respectively, while $\mathbf{W}_x$ and $\mathbf{W}_y$ are diagonal matrices with normalization coefficients that may account for variable grid resolutions and correlation scales in the x and y directions, respectively. Eq. (B.1) shows that the filter is applied twice to overcome issues that are due to the extended grids and to ensure symmetry of the filter (see Dobricic and Pinardi (2008)). In particular, $(\mathbf{W}_y\mathbf{G}_y\mathbf{W}_x\mathbf{G}_x)_{dir}$ applies the filter first in the x direction and then in the y direction ("direct"), whereas $(\mathbf{W}_x\mathbf{G}_x\mathbf{W}_y\mathbf{G}_y)_{rev}$ applies the filter first in the y direction and then in the x direction ("reverse"). Both $\mathbf{G}_x$ and $\mathbf{G}_y$ are recursively applied in two steps, first forward and then backward. The total number of recursive filter applications is given by a namelist file and, in the present case, is equal to four.

For example, the recursive filter operator $\mathbf{G}$ along the x direction can be formulated as

$$\mathbf{G}_x = \begin{cases} B(x,y,z) = \alpha(x,y,z)B(x-1,y,z) + [1-\alpha(x,y,z)]A(x,y,z) \\ C(x,y,z) = \alpha(x,y,z)C(x+1,y,z) + [1-\alpha(x,y,z)]B(x,y,z) \end{cases}, \tag{B.2}$$

where $\alpha(x,y,z)$ is a filter parameter, $A(x,y,z)$ is the input of the filter, $B(x,y,z)$ the result of forward filtering, and $C(x,y,z)$ is the output after backward filtering. The correction at a given point depends on the previous point (in the case of the x direction, the correction at (x,y,z) depends on $(x-1,y,z)$ in the forward filter and on $(x+1,y,z)$ in the backward filter).

Appendix C. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.cageo.2019.01.003.

References

Arcucci, R., D'Amore, L., Carracciuolo, L., Scotti, G., Laccetti, G., 2017a. A decomposition of the tikhonov regularization functional oriented to exploit hybrid multilevel parallelism. Oct. Int. J. Parallel Program. 45 (5), 1214–1235. https://doi.org/10.1007/s10766-016-0460-3.

Arcucci, R., D'Amore, L., Pistoia, J., Toumi, R., Murli, A., 2017b. On the variational data assimilation problem solving and sensitivity analysis. Apr. J. Comput. Phys. 335, 311–326. http://www.sciencedirect.com/science/article/pii/S0021999117300505.

Balay, S., Gropp, W., McInnes, L., Smith, B., 1997. Efficient management of parallelism in object oriented numerical software libraries. In: Arge, E., Bruaset, A.M., Langtangen, H.P. (Eds.), Modern Software Tools in Scientific Computing. Birkhäuser Press, pp. 163–202.

Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., McInnes, L., Rupp, K., Smith, B., Zampini, S., Zhang, H., Zhang, H., 2016a. PETSc Users Manual. Tech. Rep. ANL-95/11 - Revision 3.7. Argonne National Laboratory URL. http://www.mcs.anl.gov/petsc.

Balay, S., Abhyankar, S., Adams, M., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W., Kaushik, D., Knepley, M., McInnes, L., Rupp, K., Smith, B., Zampini, S., Zhang, H., Zhang, H., 2016b. PETSc Web Page. URL. http://www.mcs.anl.gov/petsc http://www.mcs.anl.gov/petsc.

Bannister, R., 2017. A review of operational methods of variational and ensemble-variational data assimilation. Q. J. R. Meteorol. Soc. 143 (703), 607–633. URL. https://doi.org/10.1002/qj.2982.

Bennett, A., Chua, B., Pflaum, B., Erwig, M., Fu, Z., Loft, R., Muccino, J., 2008. The inverse ocean modeling system. Part I: Implementation. J. Atmos. Ocean. Technol. 25 (9), 1608–1622.

Bertino, L., Evensen, G., Wackernagel, H., Aug 2003. Sequential data assimilation techniques in oceanography. Int. Stat. Rev. 71 (2), 223–241. URL. http://projecteuclid.org/euclid.isr/1069172299.

Bolzon, G., Cossarini, G., Lazzari, P., Salon, S., Teruzzi, A., Crise, A., Solidoro, C., 2017. Mediterranean Sea Biogeochemical Analysis and Forecast (CMEMS MED AF-Biogeochemistry 2013-2017). Copernicus Monitoring Environment Marine Service. https://doi.org/10.25423/MEDSEA_ANALYSIS_FORECAST_BIO_006_006.

Ciavatta, S., Kay, S., Saux-Picart, S., Butenschoen, M., Allen, J.I., 2016. Decadal reanalysis of biogeochemical indicators and fluxes in the North West European shelf-sea ecosystem. J. Geophys. Res.: Oceans 121 (3), 1824–1845. URL. https://doi.org/10.1002/2015JC011496.

Cossarini, G., Lermusiaux, P.F.J., Solidoro, C., 2009. Lagoon of Venice ecosystem: seasonal dynamics and environmental guidance with uncertainty analyses and error subspace data assimilation. J. Geophys. Res.: Oceans 114 (C6) n/a–n/a, c06026. URL. https://doi.org/10.1029/2008JC005080.

Cossarini, G., Lazzari, P., Solidoro, C., Mar 2015. Spatiotemporal variability of alkalinity in the Mediterranean Sea. Biogeosciences 12, 1647–1658.

D'Amore, L., Arcucci, R., Carracciuolo, L., Murli, A., Nov 2014. A scalable approach for variational data assimilation. J. Sci. Comput. 61 (2), 239–257. URL. https://doi.org/10.1007/s10915-014-9824-2.

D'Amore, L., Arcucci, R., Li, Y., Montella, R., Moore, A., Phillipson, L., Toumi, R., 2018. Performance assessment of the incremental strong constraints 4dvar algorithm in ROMS. In: Wyrzykowski, R., Dongarra, J., Deelman, E., Karczewski, K. (Eds.), Parallel Processing and Applied Mathematics. Lecture Notes in Computer Science. Springer International Publishing, pp. 48–57.

D'Amore, L., Arcucci, R., Marcellino, L., Murli, A., 2012. Hpc computation issues of the incremental 3d variational data assimilation scheme in oceanvar software. J. Numer. Anal. Ind. Appl. Math. 7 (3–4), 91–105.

Dobricic, S., Pinardi, N., 2008. An oceanographic three-dimensional variational data assimilation scheme. Ocean Model. 22 (3–4), 89–105. URL. http://www.sciencedirect.com/science/article/pii/S1463500308000176.

Farina, R., Dobricic, S., Storto, A., Masina, S., Cuomo, S., Mar 2015. A revised scheme to compute horizontal covariances in an oceanographic 3D-VAR assimilation system. J. Comput. Phys. 284 (C), 631–647. URL. https://doi.org/10.1016/j.jcp.2015.01.003.

Ford, D., Edwards, K., Lea, D., Barciela, R., Martin, M., Demaria, J., 2012. Assimilating globcolour ocean colour data into a pre-operational physical-biogeochemical model. Ocean Sci. 8 (5), 751–771.

Fuhrer, O., Osuna, C., Lapillonne, X., Gysi, T., Bianco, M., Schulthess, T., 2013. Towards gpu-accelerated operational weather forecasting. In: The GPU Technology Conference.

Ghil, M., Malanotte-Rizzoli, P., 1991. Data assimilation in meteorology and oceanography. Adv. Geophys. 33, 141–266.

Guest, M., Aloisio, G., Blügel, S., Orozco, M., Ricoux, P., Schäfer, A., Kenway, R., Downes, T., Lippert, T., Ramalho, M., Erbacci, G., 2012. The Scientific Case for HPC in Europe 2012 - 2020. Insight Publisher Ltd., Bristol, UK.

Jagode, H., 2006. Fourier Transforms for the BlueGene/L Communication Network. University of Edinburgh, Master's thesis.

Keyes, D., 1998. How scalable is domain decomposition in practice. In: Proceedings of the 11th International Conference on Domain Decomposition Methods. Citeseer, pp. 286–297.

Krishnamoorthy, S., Baskaran, M., Bondhugula, U., Ramanujam, J., Rountev, A., Sadayappan, P., Jun 2007. Effective automatic parallelization of stencil computations. SIGPLAN Not. 42 (6), 235–244. URL. http://doi.acm.org/10.1145/1273442.1250761.

Lazzari, P., Teruzzi, A., Salon, S., Campagna, S., Calonaci, C., Colella, S., Tonani, M., Crise, A., 2010. Pre-operational short-term forecasts for Mediterranean Sea biogeochemistry. Ocean Sci. 6 (1), 25–39. URL. http://www.ocean-sci.net/6/25/2010/.

Lazzari, P., Solidoro, C., Ibello, V., Salon, S., Teruzzi, A., Béranger, K., Colella, S., Crise, A., 2012. Seasonal and inter-annual variability of plankton chlorophyll and primary production in the Mediterranean Sea: a modelling approach. Biogeosciences 9 (1), 217–233. URL. https://www.biogeosciences.net/9/217/2012/.

Lazzari, P., Solidoro, C., Salon, S., Bolzon, G., Feb 2016. Spatial variability of phosphate and nitrate in the Mediterranean Sea: a modeling approach. Deep-Sea Res. I: Oceanogr. Res. 108, 39–52.

Li, S., Zhang, Y., Hoefler, T., 2018. Cache-oblivious mpi all-to-all communications based on morton order. IEEE Trans. Parallel Distr. Syst. 29 (3), 542–555.

Lorenc, A., 1986. Analysis methods for numerical weather prediction. Q. J. R. Meteorol. Soc. 112 (474), 1177–1194. URL. https://doi.org/10.1002/qj.49711247414.

Malouf, R., 2002. A comparison of algorithms for maximum entropy parameter estimation. In: Proceedings of the 6th Conference on Natural Language Learning - Volume 20. COLING-02. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1–7. URL. https://doi.org/10.3115/1118853.1118871.

Moore, A.M., Arango, H.G., Broquet, G., Powell, B.S., Weaver, A.T., Zavala-Garay, J., 2011. The regional ocean modeling system (roms) 4-dimensional variational data assimilation systems: Part i–system overview and formulation. Prog. Oceanogr. 91 (1), 34–49.

Munson, T., Sarich, J., Wild, S., Benson, S., McInnes, L., 2015. Tao 3.6 Users Manual.

Tech. rep. .
Pascolo, E., Salon, S., Melaku Canu, D., Solidoro, C., Cavazzoni, C., Umgiesser, G., 2016. OpenMP tasks: asynchronous programming made easy. In: International Conference on High Performance Computing & Simulation. HPCS 2016, Innsbruck, Austria July 18-22, 2016. pp. 901–907. URL. https://doi.org/10.1109/HPCSim.2016.7568430.
Rabier, F., 2005. Overview of global data assimilation developments in numerical weather-prediction centres. Q. J. R. Meteorol. Soc. 131 (613), 3215–3233. URL. https://doi.org/10.1256/qj.05.129.
Schulz, R., 2008. 3D FFT with 2D decomposition. CS project report:¡ http://cmb.ornl.gov/Members/z8g/csproject-report.pdf .
Teruzzi, A., Dobricic, S., Solidoro, C., Cossarini, G., 2014. A 3-D variational assimilation scheme in coupled transport-biogeochemical models: forecast of Mediterranean biogeochemical properties. J. Geophys. Res.: Oceans 119 (1), 200–217. URL. https://doi.org/10.1002/2013JC009277.
Teruzzi, A., Cossarini, G., Lazzari, P., Salon, S., Bolzon, G., Crise, A., Solidoro, C., 2016. Mediterranean Sea Biogeochemical Reanalysis (CMEMS MED REA-Biogeochemistry 1999-2015). Copernicus Monitoring Environment Marine Service. https://doi.org/10.25423/MEDSEA_REANALYSIS_BIO_006_008.
Teruzzi, A., Bolzon, G., Salon, S., Lazzari, P., Solidoro, C., Cossarini, G., 2018. Assimilation of coastal and open sea biogeochemical data to improve phytoplankton simulation in the mediterranean sea. Ocean Model. 132, 46–60. URL. http://www.sciencedirect.com/science/article/pii/S1463500318303184.
Tsiaras, K., Hoteit, I., Kalaroni, S., Petihakis, G., Triantafyllou, G., Apr 2017. A hybrid ensemble-OI Kalman filter for efficient data assimilation into a 3-D biogeochemical model of the Mediterranean. Ocean Dynam. 67 (6), 673–690 responsible Editor: Dieter Wolf-Gladrow).
Weaver, A.T., Tshimanga, J., Piacentini, A., Jan 2016. Correlation operators based on an implicitly formulated diffusion equation solved with the Chebyshev iteration. Q. J. R. Meteorol. Soc. 142 (694), 455–471. URL https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.2664.
Xu, S., Huang, X., Zhang, Y., Hu, Y., Fu, H., Yang, G., 2014. Porting the princeton ocean model to GPUs. In: Sun, X., Qu, W., Stojmenovic, I., Zhou, W., Li, Z., Guo, H., Min, G., Yang, T., Wu, Y., Liu, L. (Eds.), Algorithms and Architectures for Parallel Processing: 14th International Conference, ICA3PP 2014, Dalian, China, August 24-27, 2014. Proceedings, Part I. Springer International Publishing, Cham, pp. 1–14. URL. https://doi.org/10.1007/978-3-319-11197-1_1.
Zhu, C., Byrd, R., Lu, P., Nocedal, J., Dec 1997. Algorithm 778: L-BFGS-B: fortran subroutines for large-scale bound-constrained optimization. ACM Trans. Math Software 23 (4), 550–560. URL. http://doi.acm.org/10.1145/279232.279236.