A nudging-based data assimilation method: the Back and Forth Nudging (BFN) algorithm

Didier Auroux, Jacques Blum

Nonlin. Processes Geophys., 15, 305–319, 2008
www.nonlin-processes-geophys.net/15/305/2008/

Nonlinear Processes in Geophysics

A nudging-based data assimilation method: the Back and Forth Nudging (BFN) algorithm

D. Auroux[1] **and J. Blum**[2]

[1]Institut de Mathématiques de Toulouse, Université Paul Sabatier Toulouse 3, 31062 Toulouse Cedex 9, France
[2]Laboratoire J. A. Dieudonné, Université de Nice Sophia-Antipolis, Parc Valrose, 06108 Nice Cedex 2, France

Received: 17 August 2007 – Revised: 2 January 2008 – Accepted: 27 February 2008 – Published: 27 March 2008

Abstract. This paper deals with a new data assimilation algorithm, called Back and Forth Nudging. The standard nudging technique consists in adding to the equations of the model a relaxation term that is supposed to force the observations to the model. The BFN algorithm consists in repeatedly performing forward and backward integrations of the model with relaxation (or nudging) terms, using opposite signs in the direct and inverse integrations, so as to make the backward evolution numerically stable. This algorithm has first been tested on the standard Lorenz model with discrete observations (perfect or noisy) and compared with the variational assimilation method. The same type of study has then been performed on the viscous Burgers equation, comparing again with the variational method and focusing on the time evolution of the reconstruction error, i.e. the difference between the reference trajectory and the identified one over a time period composed of an assimilation period followed by a prediction period. The possible use of the BFN algorithm as an initialization for the variational method has also been investigated. Finally the algorithm has been tested on a layered quasi-geostrophic model with sea-surface height observations. The behaviours of the two algorithms have been compared in the presence of perfect or noisy observations, and also for imperfect models. This has allowed us to reach a conclusion concerning the relative performances of the two algorithms.

1 Introduction

Nudging is a data assimilation method that uses dynamical relaxation to adjust a model towards observations. The standard nudging algorithm consists in adding to the state equations of a dynamical system a feedback term proportional to the difference between the observation and the equivalent quantity computed by integration of the state equations. The model then appears as a weak constraint, and the nudging term forces the state variables to fit as well as possible to the observations. This forcing term in the model dynamics has a tunable coefficient that represents the relaxation time scale. This coefficient is chosen by numerical experimentation so as to keep the nudging terms small in comparison to the state equations, and large enough to force the model to the observations. The nudging term can also be seen as a penalty term, which penalizes the system if the model is too far from the observations.

The nudging method is a flexible assimilation technique, and computationally much more economical than variational data assimilation methods. First used in meteorology (Hoke and Anthes, 1976), the nudging method has been successfully introduced in oceanography in a quasi-geostrophic model (Verron, 1990; Verron and Holland, 1989; Blayo et al., 1994) and has been applied to a mesoscale model of the atmosphere with synoptic-scale data (Stauffer and Seaman, 1990). The nudging coefficients can be optimized by a variational method (Stauffer and Bao, 1993; Zou et al., 1992), where a parameter estimation approach is proposed to obtain optimal nudging coefficients, in the sense that the difference between the model solution and the observations is as small as possible. A comparison between optimal nudging and Kalman filtering can be found in Vidard et al. (2003). A drawback of this optimal nudging technique is that it requires the computation of the adjoint state of the model equations, which is not necessary in the standard nudging method.

The backward nudging algorithm consists in solving backwards in time the state equations of the model, starting from

Correspondence to: D. Auroux
(auroux@mip.ups-tlse.fr)

Published by Copernicus Publications on behalf of the European Geosciences Union and the American Geophysical Union.

the observation of the system state at the final time of the assimilation period. A nudging term, with the opposite sign compared to the standard nudging algorithm, is added to the state equations, and the final state computed in the backward integration is in fact an approximation of the initial state of the system (Auroux, 2003).

The Back and Forth Nudging (BFN) algorithm, introduced in Auroux and Blum (2005), consists in solving first the forward nudging equation, and then the model equation backwards in time with a relaxation term (with the opposite sign in comparison with the relaxation term introduced in the forward equation). The initial condition of this backward integration is the final state obtained by the standard nudging method. After integration of this backward equation, one obtains an estimate of the initial state of the system. We then repeat these forward and backward integrations (with the relaxation terms) until convergence of the algorithm. Such a forward-backward assimilation technique had already been introduced in Talagrand (1981a,b). In that algorithm, at each observation time, the values predicted by the model for the observed parameters were just replaced by the observed values. This corresponds to the particular case of our BFN algorithm where the nudging coefficients go to infinity.

The BFN algorithm can be compared to the four-dimensional variational algorithm (4D-VAR, see e.g. Le Dimet and Talagrand, 1986), which also consists in a sequence of forward and backward integrations. In our algorithm it is useless to linearize the system, even for nonlinear problems, and the backward system is not the adjoint equation but the model equations, with an extra feedback term that stabilizes the numerical integration of this ill-posed backward problem.

Let us finally mention another back and forth data assimilation method, called the quasi-inverse method (Kalnay et al., 2000). In that method, there are no nudging terms, and in the backward integration, the sign of the dissipation terms is changed for stability reasons. The idea of introducing relaxation (or nudging) terms in our algorithm enables us to keep the dissipation terms with the correct sign in the backward integration, as the nudging terms have a stabilizing role.

In Sect. 2 we first present the standard nudging algorithm for a non-linear model, then the nudging algorithm applied to the corresponding backward model, and finally we introduce the back and forth nudging algorithm. The end of Sect. 2 discusses some theoretical considerations about the choice of the nudging gain matrices, and some physical considerations that motivate our algorithm. Section 3 is devoted to the application of this algorithm to the Lorenz model, and to its comparison with the classical variational method. In Sect. 4, the one dimensional viscous Burgers' equation is considered and the two algorithms are again compared. In Sect. 5, we consider a quasi-geostrophic model, and study the effect of noisy observations and model errors. Finally, some conclusions are given in Sect. 6.

2 Description of the Back and Forth Nudging algorithm

2.1 Forward nudging

We assume that the model equations have been discretized in space by a finite difference, finite element, or spectral discretization method. The time continuous model satisfies dynamical equations of the form:

$$\frac{dX}{dt} = F(X), \quad 0 < t < T, \tag{1}$$

with an initial condition $X(0)=x_0$. We will denote by C the observation operator, allowing us to compare the observations $X_{\text{obs}}(t)$ with the corresponding $C(X(t))$, deduced from the state vector $X(t)$. We do not particularly assume that C is a linear operator. If we apply nudging to the model (1), we obtain

$$\begin{cases} \dfrac{dX}{dt} = F(X) + \mathbf{K}(X_{\text{obs}} - C(X)), \quad 0 < t < T, \\ X(0) = x_0, \end{cases} \tag{2}$$

where $\mathbf{K}$ is the nudging (or gain) matrix. The model then appears as a weak constraint, and the nudging term forces the state variables to fit as well as possible to the observations. In the linear case (where F is a matrix, and C is a linear operator), the forward nudging method is nothing else than the Luenberger observer (Luenberger, 1966), also called asymptotic observer, where the matrix $\mathbf{K}$ can be chosen so that the error goes to zero when time goes to infinity.

2.2 Backward nudging

We now assume that we have a final condition in Eq. (1) instead of an initial condition. This leads to the following backward equation:

$$\begin{cases} \dfrac{d\tilde{X}}{dt} = F(\tilde{X}), \quad T > t > 0, \\ \tilde{X}(T) = \tilde{x}_T. \end{cases} \tag{3}$$

The backward nudging algorithm (Auroux, 2003) consists in solving backwards in time the state equations of the model, starting from the observation of the system state at the final time. If we apply nudging to this backward model with a feedback term of the opposite sign (in order to have a well posed problem), we obtain

$$\begin{cases} \dfrac{d\tilde{X}}{dt} = F(\tilde{X}) - \mathbf{K}'(X_{\text{obs}} - C(\tilde{X})), \quad T > t > 0, \\ \tilde{X}(T) = \tilde{x}_T, \end{cases} \tag{4}$$

where $\mathbf{K}'$ is the backward nudging matrix.

The backward integration of this equation provides a state vector at time $t=0$, which can be seen as an identified initial condition for our data assimilation period.

2.3 Back and Forth Nudging (BFN) algorithm

The back and forth nudging algorithm, introduced in Auroux and Blum (2005), consists in first solving the forward nudging equation and then the backward nudging equation. The initial condition of the backward integration is the final state obtained after integration of the forward nudging equation. At the end of this process, one obtains an estimate of the initial state of the system. We repeat these forward and backward integrations (with the feedback terms) until convergence of the algorithm:

$$\begin{aligned} k \geq 1 \quad & \begin{cases} \dfrac{dX_k}{dt} = F(X_k) + \mathbf{K}(X_{\text{obs}} - C(X_k)), \\ X_k(0) = \tilde{X}_{k-1}(0), \end{cases} \\ k \geq 1 \quad & \begin{cases} \dfrac{d\tilde{X}_k}{dt} = F(\tilde{X}_k) - \mathbf{K}'(X_{\text{obs}} - C(\tilde{X}_k)), \\ \tilde{X}_k(T) = X_k(T), \end{cases} \end{aligned} \tag{5}$$

with the notation $\tilde{X}_0(0)=x_0$. Then, $X_1(0)=x_0$, and an integration of the direct model gives $X_1(T)$ and hence $\tilde{X}_1(T)$. An integration of the backward model gives $\tilde{X}_1(0)$, which is equal to $X_2(0)$, and so on.

The reader is referred to Auroux and Blum (2005) for the proof of convergence of this algorithm in a simple case (linear model and full observations). Moreover, if the observations are perfect (i.e. X_{obs} satisfies Eq. 1) and if $\mathbf{K}=\mathbf{K}'$ and F and $\mathbf{K}$ commute, then it is straightforward to see that $X_k(t)\rightarrow X_{\text{obs}}(t)$ when $k\rightarrow+\infty$.

If $\mathbf{K}=\mathbf{K}'$ and if the forward and backward trajectories $X_k(t)$ and $\tilde{X}_k(t)$ converge towards the same limit trajectory $X_\infty(t)$, then it is clear by adding the two equations of Eq. (5) that $X_\infty(t)$ also satifies the model Eq. (1), and that $\mathbf{K}(X_{\text{obs}}-C(X_\infty))=0$.

When the observations are discrete in time, i.e. the observation vector X_{obs} is only available at some times $(t_i)_{i=1\ldots N}$, then the nudging term is only added at these time steps:

$$\frac{dX}{dt} = F(X) + \sum_{i=1}^{N} \mathbf{K}(X_{\text{obs}} - C(X))\,\delta(t - t_i). \tag{6}$$

In the following numerical experiments, the observations are not available at each time step, and hence we solve this discrete nudging equation, instead of the continuous ones we previously described.

2.4 Choice of the nudging matrices

2.4.1 Forward nudging matrix **K**, variational interpretation of the nudging, and statistics of errors

The standard nudging method has been widely studied in the past decades (Hoke and Anthes, 1976; Verron, 1990; Stauffer and Seaman, 1990; Bao and Errico, 1997). Thus, there are several ways to choose the nudging matrix $\mathbf{K}$ in the forward part of the algorithm. One can for example consider the optimal nudging matrix $\mathbf{K}_{\text{opt}}$, as discussed in Zou et al. (1992) or Vidard et al. (2003). In such an approach, a variational data assimilation scheme is used in a parameter-estimation mode to determine the optimal nudging coefficients. This choice provides theoretically the best results for the forward part of the BFN scheme, but the computation of the optimal gain matrix is costly.

When $\mathbf{K}=0$, the forward nudging problem (2) simply becomes the direct model (1). On the other hand, setting $\mathbf{K}=+\infty$ forces the state variables to be equal to the observations at discrete times, as is done in Talagrand (1981a,b).

In order to correctly choose the forward nudging matrix, we can give a variational interpretation of the forward nudging. Let us assume that we know the statistics of errors on observations, and denote by $\mathbf{R}$ the covariance matrix of observation errors, which is usually assumed to be symmetric positive definite. We now set the nudging matrix to be

$$\mathbf{K} = C^T \mathbf{R}^{-1}, \tag{7}$$

and we assume the direct model to be linear (or linearized). We consider a temporal discretization of the forward nudging problem (2), using for example an implicit scheme. If we denote by X^n the solution at time t_n and X^{n+1} the solution at time t_{n+1}, and $\Delta t=t_{n+1}-t_n$, Eq. (2) becomes

$$\frac{X^{n+1} - X^n}{\Delta t} = FX^{n+1} + \mathbf{K}(X_{\text{obs}} - CX^{n+1}), \tag{8}$$

where F is assumed to be a symmetric linear model operator. Then, it is straightforward to see that X^{n+1} is solution of the following optimization problem

$$\begin{aligned} \min_X \Bigg[& \frac{1}{2}\langle X - X^n, X - X^n\rangle - \frac{\Delta t}{2}\langle FX, X\rangle \\ & + \frac{\Delta t}{2}\langle \mathbf{R}^{-1}(X_{\text{obs}} - CX), X_{\text{obs}} - CX\rangle \Bigg]. \end{aligned} \tag{9}$$

The first two terms correspond exactly to the energy of the discretized direct model, and the last term is the observation part of the variational cost function. This variational principle shows that at each time step, the nudging state is a compromise between minimizing the energy of the system and the distance to the observations. As a consequence, there is no need to consider an additional term ensuring an initial condition close to the background state like in variational algorithms, neither for stabilizing or regularizing the problem, nor from a physical point of view. One can simply initialize the BFN scheme with the background state, without any information on its statistics of errors. The nudging method naturally provides a correction to the model equations from the observations. The model equations are hence weak constraints in the BFN scheme. In some nonlinear cases, the $\langle FX, X\rangle$ term in Eq. (9) can be replaced by $-G(X)$, where G is the energy of the system at equilibrium.

In the following sections, all numerical experiments have been performed with an easy-to-implement nudging matrix:

$$\mathbf{K} = C^T(k.\mathbf{I}) = kC^T, \tag{10}$$

where k is a positive scalar gain, and $\mathbf{I}$ is the identity matrix of the observation space. This choice is motivated by the following remarks. First, the covariance matrix of observation errors is usually not well known (but if it is available, then one should consider Eq. (7) for the definition of $\mathbf{K}$). Secondly, this choice does not require a costly numerical integration of a parameter estimation problem for the determination of the optimal coefficients. Choosing $\mathbf{K}=C^T\mathbf{L}$, where $\mathbf{L}$ is a square matrix in the observation space, has another interesting property: if the observations are not located at a model grid point, or are a function of the model state vector, i.e. if the observation operator C involves interpolation/extrapolation or some change of variables, then the nudging matrix $\mathbf{K}$ will contain the adjoint operations, i.e. some interpolation/extrapolation back to the model grid points, or the inverse change of variable.

2.4.2 Backward nudging matrix $\mathbf{K}'$ and pole assignment method

The goal of the backward nudging term is both to have a backward data assimilation system and to stabilize the integration of the backward system (4), as this system is usually ill posed. The choice of the backward nudging matrix is then imposed by this stability condition. If we consider a linearized situation, in which the system and observation operators (F and C, respectively) are linear, and if we make the change of time variable $t'=T-t$, then the backward equation can be rewritten as

$$-\frac{d\tilde{X}}{dt'} = F\tilde{X} - \mathbf{K}'(X_{\text{obs}} - C\tilde{X}), \tag{11}$$

and then the matrix to be stabilized is $-F-\mathbf{K}'C$, i.e. the eigenvalues of this matrix should have negative real parts.

We now recall the pole assignment result (see e.g. Datta, 1987; Arnold and Datta, 1998; Bonnans and Rouchon, 2005; Trélat, 2005): if (F, C) is an observable system, where F is a $n\times n$ matrix and C is a $m\times n$ matrix (here n is the size of the control vector X and m is the size of the observation vector X_{obs}), then there exists at least one matrix $\mathbf{K}'$ such that $-F-\mathbf{K}'C$ is a Hurwitz matrix, i.e. all its eigenvalues are in the negative half-plane. We should also recall (see e.g. Arnold and Datta, 1998; Trélat, 2005) that (F, C) is an observable system if and only if the rank of $[C, CF, \ldots, CF^{n-1}]$ is equal to n.

Hence, we can assume that there exists at least one matrix $\mathbf{K}'$ such that the backward nudging system (4) is stable.

As in the forward part of the algorithm, for simplicity reasons we make the following choice for the backward nudging matrix $\mathbf{K}'$ in the next sections:

$$\mathbf{K}' = k'C^T\mathbf{I} = k'C^T, \tag{12}$$

where $k'>0$. The coefficient k' is usually chosen to be the smallest coefficient that makes the numerical backward integration stable.

2.5 Experimental approach

The same approach has been used for all the numerical experiments presented in the next sections. This approach consists in performing twin experiments with simulated data. First, a reference experiment is run and the corresponding data are extracted. From now on this reference trajectory will be called the exact solution. Experimental data are supposed to be obtained every n_x gridpoints of the model, and every n_t time steps. The simulated data are then optionally noised with a Gaussian white noise distribution, and provided as observations to the assimilation scheme. The first guess of the assimilation experiments is chosen to be either a constant field or the reference model state some time before the beginning of the assimilation period. Finally, the results of the assimilation process are compared with the exact solution.

3 Convergence of BFN and comparison with variational assimilation for the Lorenz equations

The BFN algorithm was first tested on Lorenz' chaotic system (Lorenz, 1963):

$$\begin{cases} \dfrac{dx}{dt} = 10\,(y-x), \\ \dfrac{dy}{dt} = 28\,x - y - xz, \\ \dfrac{dz}{dt} = -\dfrac{8}{3}\,z + xy. \end{cases} \tag{13}$$

3.1 Convergence of the BFN algorithm

We have performed twin experiments in order to prove the numerical convergence of the BFN algorithm. In this section, the assimilation period is $[0, 3]$, the time step is 0.001 and data are extracted every n_t=100 time steps (31 observations during the assimilation period). We assume that all three variables are observed.

We assume in this subsection that data are unnoised. The initialization of the BFN algorithm has been performed using a randomly noised state.

Figure 1 shows that the BFN iterates at time t=0 nearly converge towards the exact initial condition X_{true} in less than 10 iterations. Figure 2 shows that the successive BFN iterates are almost equal after 10 iterations. These two figures prove the numerical convergence of the BFN algorithm.

Figure 3 clearly shows that the BFN algorithm makes the trajectory converge towards the observations.

In all these experiments, only 10 iterations are required to reach convergence. Recall that one iteration consists in one forward integration of the model (with a nudging term) and

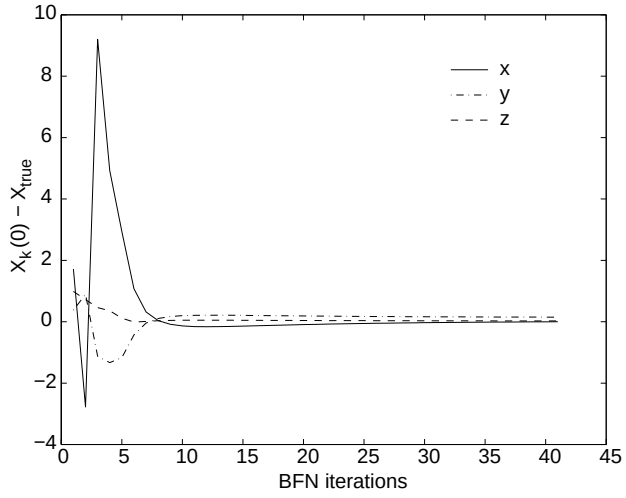

Fig. 1. Difference between the kth iterate $X_k(0)$ and the exact initial condition X_{true} for the 3 variables versus the number of BFN iterations.

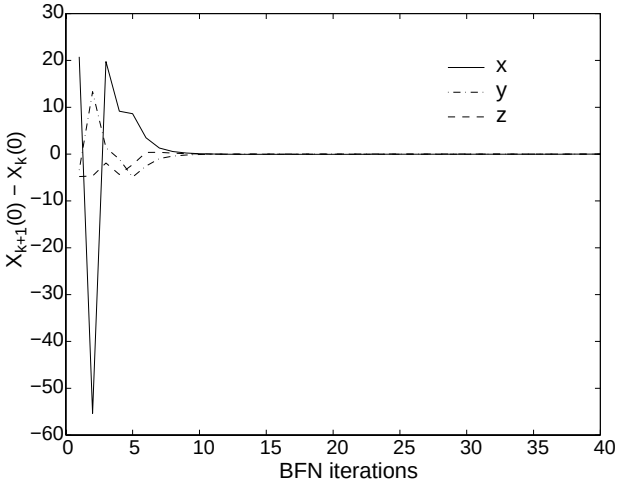

Fig. 2. Difference between two consecutive BFN iterates for the 3 variables versus the number of BFN iterations.

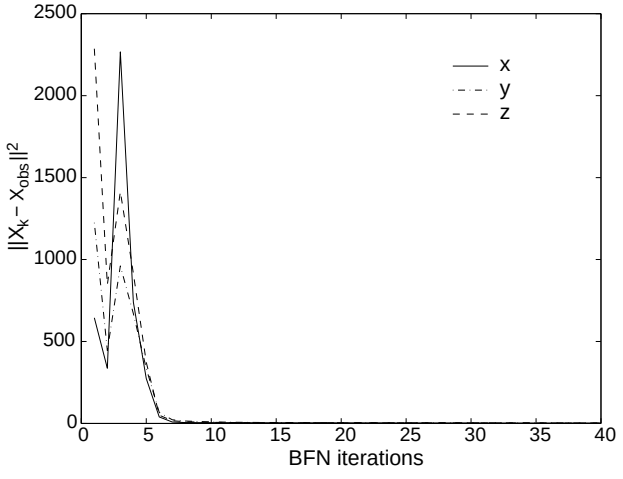

Fig. 3. Mean square difference between the observations and the BFN identified trajectory for the 3 variables versus the number of BFN iterations.

one backward integration of the same model (with an opposite sign nudging term). Finally, the nudging matrices $\mathbf{K}$ and $\mathbf{K}'$ are equal to kC^T and $k'C^T$ respectively (see Sect. 2.4), with k=50 and k'=100.

3.2 Comparison with the variational assimilation algorithm

We have compared the BFN and variational assimilation algorithms. The variational assimilation (VAR) algorithm is based on the minimization of a global cost function, which measures the discrepancy between the observations and the corresponding system states. The adjoint method allows one to compute the gradient of the cost function in a single numerical integration of the adjoint equation (see e.g. Le Dimet and Talagrand, 1986). One iteration of the minimization process consists then in one forward integration of the model (in order to compute the cost function) and one backward integration of the adjoint model (in order to compute its gradient). The computational costs of one BFN iteration and one VAR iteration are then nearly the same. We stopped both algorithms before convergence, with a maximum of 10 iterations. We have used a limited memory quasi-Newton technique (L-BFGS) for the minimization of the VAR cost function (Nocedal, 1980).

Figure 4 shows the trajectories identified by the variational (VAR) and BFN algorithms, using perfect observations (with the same time distribution as in the previous subsection). The reference trajectory is also shown for comparison. In order to make the figure clearer, only the first Lorenz variable x is represented. Recall that the assimilation period is [0, 3], and [3, 6] is a forecast period (no observations). We can see that the identified trajectories are very close to the reference trajectory all over the assimilation period. After that, the VAR trajectory becomes wrong before time t=4, whereas the BFN trajectory becomes wrong near t=5.

We have also studied the influence of observation errors on these algorithms. Figure 5 shows the same data as Fig. 4 in the case of 10%-noisy (Gaussian white noise) observations. The additional curve shows the perturbed trajectory derived from the noisy observations of the system at time t=0. Note that this curve diverges from the reference trajectory before the end of the assimilation period (near t=2.5). The conclusions concerning the difference between the VAR and BFN algorithms are almost the same as in the previous experiment (with perfect observations), even though the results are a little bit less accurate than before. It is clear that

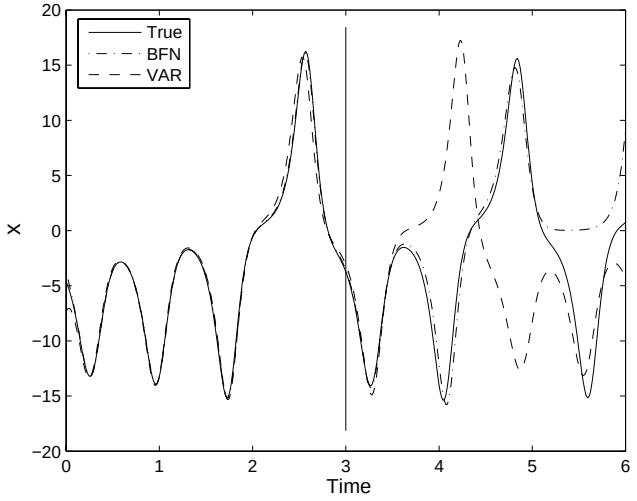

Fig. 4. Evolution in time of the reference trajectory (solid line), and of the trajectories identified by the variational assimilation (dashed line) and BFN (dash-dotted line) algorithms, in the case of perfect observations and for the first Lorenz variable x.

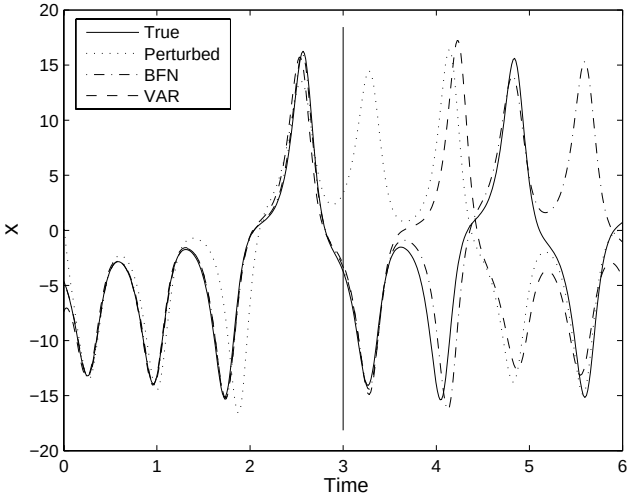

Fig. 5. Evolution in time of the reference trajectory (solid line), of the perturbed trajectory derived from the noisy observation at time t=0 (dotted line), and of the trajectories identified by the VAR (dashed line) and BFN (dash-dotted line) algorithms, in the case of noisy observations (with a 10% Gaussian white noise) and for the first Lorenz variable x.

for the same number of iterations, the BFN algorithm is in this case slightly better than the variational method.

4 Convergence and comparison with the VAR algorithm on the 1-D viscous Burgers' equation

4.1 Physical model

We consider in this section a very simple nonlinear geophysical model. The evolution model is the viscous Burgers' equation over a one-dimensional cyclic domain:

$$\frac{\partial X}{\partial t} + \frac{1}{2}\frac{\partial X^2}{\partial s} - \nu \frac{\partial^2 X}{\partial s^2} = 0, \tag{14}$$

where X is the state variable, s represents the distance in meters around the 45° N constant-latitude circle and t is the time. The period of the domain is roughly 28.3×10^6 m. The diffusion coefficient ν is set to $10^5\,\mathrm{m^2\,s^{-1}}$ (Fisher and Courtier, 1995). The time step is one hour, and the assimilation period is roughly one month (700 time steps). Data are available every n_x=5 gridpoints of the model, with a time sampling of 10 h (every n_t=10 time steps). This provides a spatial density similar to the longitudinal distribution of the mid-latitude radiosonde network. The observation noise distribution corresponds to a 5% root mean square (RMS) error. The first guess of the assimilation experiments is chosen to be a constant field (X=0 everywhere). Finally, in the following experiments, the nudging matrices $\mathbf{K}$ and $\mathbf{K}'$ are set to kC^T and $k'C^T$, respectively (see Sect. 2.4), with k=0.01 and k'=0.02.

4.2 Convergence of the BFN algorithm

We have first focused our interest on the numerical convergence of the BFN algorithm we have proposed, because the mathematical convergence result is currently only valid for linear models.

Figure 6 shows the RMS (root mean square) relative difference between two iterates of the BFN algorithm $\frac{\|X_{k+1}-X_k\|}{\|X_k\|}$ versus the number of iterations. We can clearly see that the relative difference between two iterates becomes smaller than 1% in fewer than 5 iterations. The numerical convergence of the algorithm is then obvious, and very quickly achieved.

We have then compared the BFN iterates with the exact solution (or reference trajectory) with the aim of quantifying the identification of the true initial state.

Figure 7a shows the RMS relative difference between the BFN iterates at time t=0 and the exact initial condition $\frac{\|X_k(0)-X_{\mathrm{true}}\|}{\|X_{\mathrm{true}}\|}$ versus the number of iterations. We again observe convergence in fewer than 5 iterations. The identification error is nearly 12% at the end of the process. This seems huge, but compared to some other data assimilation techniques, the BFN algorithm is not supposed to identify precisely the initial condition but rather the reference trajectory as a whole. Figure 7b shows the RMS difference between the BFN iterates and the exact final condition (i.e. the reference trajectory at the end of the assimilation period) versus the number of iterations. We can see that the relative

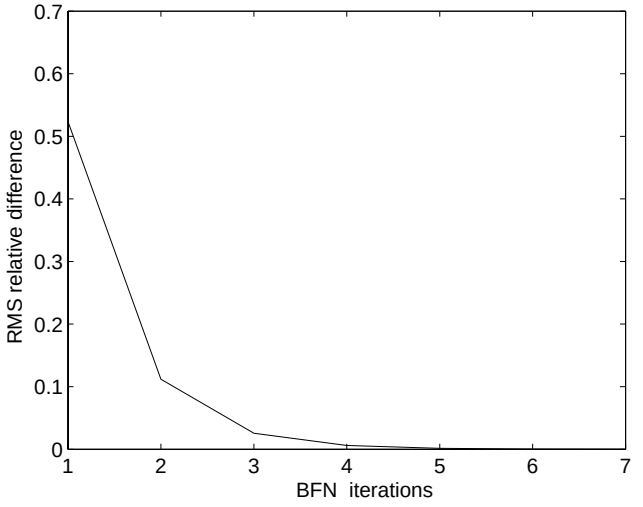

Fig. 6. RMS relative difference between two consecutive iterates of the BFN algorithm versus the number of iterations.

difference between the true final solution and the identified final solution is about 5%.

4.3 Comparison with the VAR algorithm

In this subsection, we focus our interest on the evolution of the system forecast after the assimilation period. This is the most frequent application of data assimilation. We have considered on one hand the initial condition provided by the BFN algorithm or the variational assimilation (VAR) algorithm, and on the other hand an interpolation in the state space of the first available observation (at time $t=0$). We have used these states as initial conditions for the exact model (14) and computed the corresponding trajectories over a 4 month time period, corresponding to the assimilation period followed by a 3 month prediction period.

We have first compared our algorithm with the standard VAR algorithm in the case of perfect observations. The spatial and time distributions of the observations remain unchanged, and the VAR cost function is still minimized using a limited-memory quasi-Newton algorithm (Nocedal, 1980).

Figure 8 shows the RMS (root mean square) relative difference between the reference trajectory and the identified trajectories for the BFN (dotted line) and VAR (dash-dotted line) algorithms. We recall that the assimilation period $[0, 700]$ is followed by a three times longer forecast period ($[700, 2800]$). Both algorithms were initialized with the same initial condition, corresponding to the true state vector at a previous time. Both algorithms were stopped after at most 10 iterations, and therefore, probably before convergence. One can see that at the beginning of the prediction period ($t=700$), the identification error of the BFN algorithm is smaller than 1%, whereas that of the VAR algorithm is

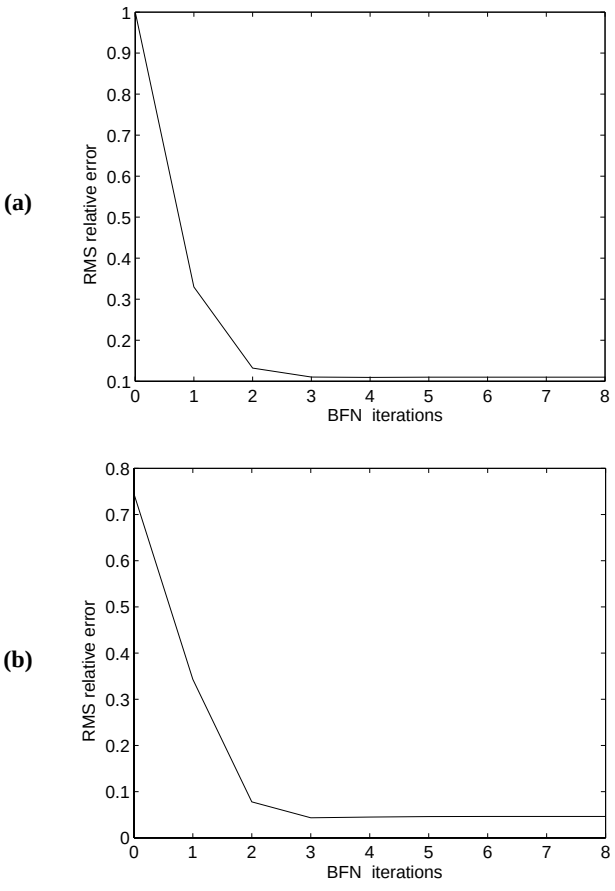

Fig. 7. RMS relative difference between the BFN iterates and the exact solution versus the number of iterations, at time $t=0$ **(a)** and at time $t=T$ **(b)**.

greater than 2%, the difference being even more significant at the end of the prediction period.

However, allowing a larger number of iterations before stopping these algorithms does not affect much the output of the BFN algorithm, as the convergence is nearly achieved in fewer than 10 iterations, whereas the VAR algorithm provides a more accurate trajectory. It simply needs nearly 30 iterations (3 times more than for the BFN) to produce comparable results in this case.

We now consider noisy observations (using the same noise distribution as in the previous subsections). Figure 9 shows the RMS relative difference between the BFN trajectory (computed using the last BFN state at time $t=0$ as an initial condition) and the reference trajectory (dotted line), between the VAR trajectory (computed using the last initial state produced by the minimization process) and the reference trajectory (dash-dotted line), and between the perturbed trajectory (derived from the noisy observations of the system at time $t=0$) and the reference trajectory (solid line), versus time.

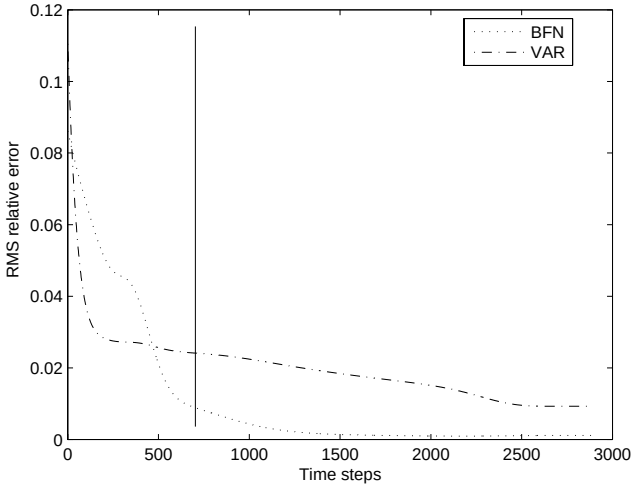

Fig. 8. Evolution in time of the RMS difference between the reference trajectory and the identified trajectories for the BFN (dotted line) and VAR (dash-dotted line) algorithms, in the case of perfect observations.

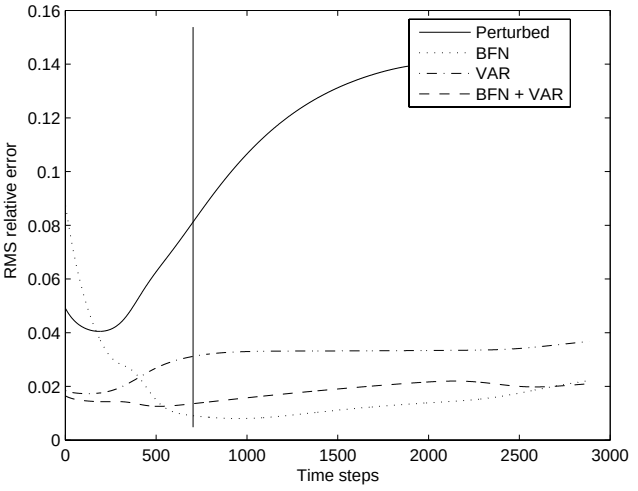

Fig. 9. Evolution in time of the RMS difference between the reference trajectory and the perturbed trajectory derived from the noisy observations of the system at time t=0 (full line), and between the reference trajectory and the identified trajectories for the BFN (dotted line), VAR (dash-dotted line) and BFN-preprocessed VAR (dashed line) algorithms, in the case of noisy observations (with a 5% RMS error).

The fourth curve (dashed line) will be discussed at the end of this subsection.

We can see that the perturbed trajectory presents an error of about 5% at the beginning of the assimilation period. The stable modes of the model cause the error to decrease at first, but after a few days (200 time steps, nearly 6 days) the unstable modes cause the error to increase over time (as observed in Pires et al., 1996). If we consider the BFN trajectory, we can see that at the beginning of the assimilation period (time t=0), the error is much higher than for the perturbed trajectory, nearly 9%, but after 500 time steps, the error is smaller than 2%. Even after 4 months, the error is still smaller than 3%, whereas the error of the perturbed trajectory has nearly reached 15% by that time. This clearly proves the efficiency of the BFN algorithm, allowing one to identify a trajectory over a 4 month period with an assimilation period of only one month, with less than 2% RMS error using 5% noisy observations. The VAR trajectory produces quite a good approximation of the initial condition (much more accurate than the BFN algorithm), but then, after a short decrease, the error increases slightly and remains nearly constant all over the prediction period, close to 4%. One should keep in mind that both algorithms have been stopped after only 10 iterations, and the convergence of the VAR algorithm has not been reached.

It is particularly interesting to see that, at the end of the assimilation period (or at the beginning of the prediction period), after 700 time steps, the difference between the BFN trajectory and the exact trajectory has nearly reached its minimum. This shows how efficient the BFN algorithm can be for the prediction step, whereas the reconstruction of the initial state is not very efficient. Since the error on the BFN trajectory decreases sharply and remains small afterwards, one can say that the identified initial condition is part of the stable manifold of the model, and the reconstruction error on the unstable manifold is extremely small. This is due to the fact that the BFN initial condition comes from a backward integration of the model, at which point the trajectory has been smoothed by the stable manifold of the backward model, which is the unstable manifold of the forward model.

We have also studied the possibility of using the BFN algorithm as a preprocessing tool for the VAR algorithm. For this purpose, we stopped the BFN after only 5 iterations, and we used the resulting initial state vector as an initialization vector for the VAR minimization process, which we also stopped after 5 iterations. The computational cost is then the same as for 10 iterations of the BFN or VAR algorithms alone. The fourth curve (dashed line) of Fig. 9 shows the RMS relative difference between the reference trajectory and the trajectory identified by this BFN-preprocessed VAR algorithm. One can see that the use of the BFN algorithm as a preconditioner for the VAR algorithm gives a slightly better initial state, and the error on the identified trajectory remains smaller than 2% over the entire assimilation period, which is not the case for either algorithm considered alone. During the prediction period, the error remains nearly constant, and is smaller than for the VAR algorithm, but a little bit larger than for the BFN algorithm, even though at the end of the period it is exactly of the same order.

We must finally mention that, after convergence, the VAR algorithm usually identifies a better trajectory than the BFN

algorithm, as can be seen in Fig. 10. However the VAR algorithm needs many more iterations than the BFN scheme to reach convergence. In this experiment for example, the numbers of iterations at convergence are 6 and 29 for the BFN and VAR algorithms respectively. In nearly all the experiments we performed, starting from the same initialization state, the VAR algorithm needed at least four times more iterations than the BFN algorithm to reach convergence, but most of the time, it identified a slightly better state, and hence gave a better prediction. This also confirms that the BFN algorithm is much more powerful in the very first iterations.

5 Convergence and comparison with 4D-VAR on a layered quasi-geostrophic ocean model

5.1 Quasi-geostrophic ocean model

We consider here a layered quasi-geostrophic ocean model (Holland, 1978; Verron et al., 1992; Blayo et al., 1994). In this model the ocean is supposed to be stratified in n layers, each of them having a constant fluid density. The quasi-geostrophic model is obtained by taking a first order expansion of the Navier-Stokes equation with respect to the Rossby number. The model system is then composed of n coupled equations resulting from the conservation law of the potential vorticity. The equations can be written as:

$$\frac{D_1\,(\theta_1(\Psi)+f)}{Dt} + A_4\nabla^6\Psi_1 = F_1 \quad \text{in } \Omega\times]0,T[, \tag{15}$$

at the surface layer (k=1);

$$\frac{D_k\,(\theta_k(\Psi)+f)}{Dt} + A_4\nabla^6\Psi_k = 0 \quad \text{in } \Omega\times]0,T[, \tag{16}$$

at the intermediate layers (k=2, ..., n−1);

$$\frac{D_n\,(\theta_n(\Psi)+f)}{Dt} + A_1\Delta\Psi_n + A_4\nabla^6\Psi_n = 0, \tag{17}$$

in $\Omega\times]0,T[$, at the bottom layer (k=n).

The notations are as follows:

– $\Omega \subset \mathbb{R}^2$ is the circulation basin and $]0,T[$ is the time interval;

– n is the number of layers;

– Ψ_k is the stream function at layer k, Ψ is the vector $(\Psi_1,\ldots,\Psi_n)^T$;

– θ_k is the sum of the dynamical and thermal vorticities at layer k:

$$\theta_k(\Psi) = \Delta\Psi_k - (W\Psi)_k,$$

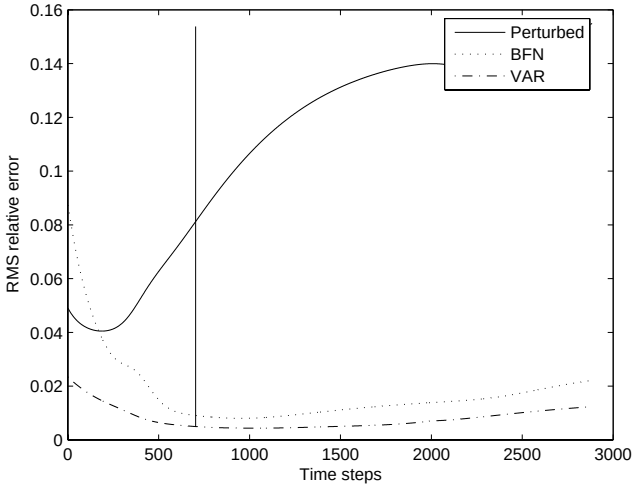

Fig. 10. Evolution in time of the RMS difference between the reference trajectory and the perturbed trajectory derived from the noisy observations of the system at time t=0 (solid line, same as solid line in Fig. 9), and between the reference trajectory and the trajectories identified by the BFN (dotted line, same as dotted line in Fig. 9) and VAR (dash-dotted line) algorithms after convergence, in the case of noisy observations (with a 5% RMS error).

with $-(W\Psi)_k = \frac{f_0^2\rho}{H_k g}\left(\frac{\Psi_{k+1}-\Psi_k}{\rho_{k+1}-\rho_k} - \frac{\Psi_k-\Psi_{k-1}}{\rho_k-\rho_{k-1}}\right)$, $1<k<n$, and $-(W\Psi)_1 = \frac{f_0^2\rho}{H_1 g}\left(\frac{\Psi_2-\Psi_1}{\rho_2-\rho_1}\right)$ and $-(W\Psi)_n = \frac{f_0^2\rho}{H_n g}\left(-\frac{\Psi_n-\Psi_{n-1}}{\rho_n-\rho_{n-1}}\right)$.

– f is the Coriolis force (f_0 is the Coriolis force at the reference latitude of the ocean).
In the β-plane approximation, the Coriolis force varies linearly with respect to the latitude.

– g represents the constant of gravity, ρ_k the fluid density at layer k (and ρ the average fluid density), and H_k the depth of the k-th layer;

– $\frac{D_k}{Dt}$ is the Lagrangian particular derivative:

$$\frac{D_k}{Dt} = \frac{\partial}{\partial t} + J(\Psi_k,.),$$

where J is the Jacobian operator

$$J(f,g) = \frac{\partial f}{\partial x}\frac{\partial g}{\partial y} - \frac{\partial f}{\partial y}\frac{\partial g}{\partial x}.$$

– $\Delta\Psi_n$ represents the bottom friction dissipation, $\nabla^6\Psi_k$ represents the lateral friction (of biharmonic type)

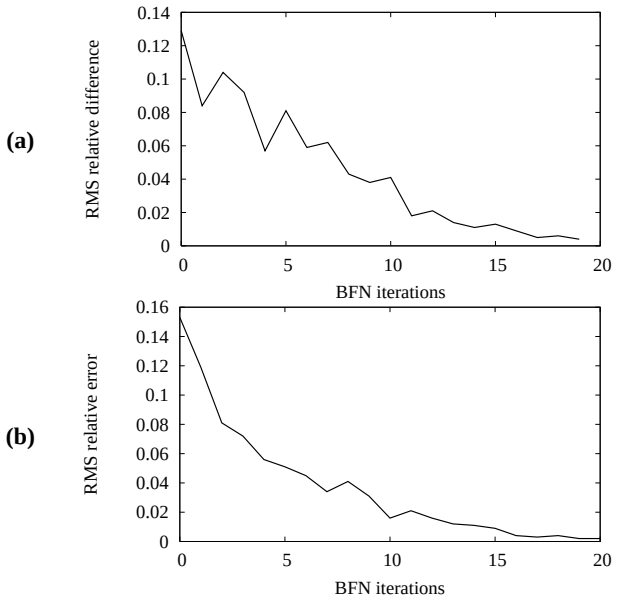

Fig. 11. RMS relative difference between two consecutive BFN iterates **(a)** and between the BFN iterates and the exact solution **(b)** versus the number of BFN iterations.

dissipation, and A_1 and A_4 are respectively the bottom and lateral friction dissipation coefficients;

– and F_1 is the forcing term, the wind stress applied to the ocean surface.

The initial conditions $\Psi_k(0)$ and some boundary conditions resulting from the mass conservation law (Holland, 1978; Luong et al., 1998; Auroux and Blum, 2004) complete the equations of the direct model.

We suppose that the data we want to assimilate come from satellite measurements of the sea-surface height h, which is directly related to the upper layer stream function Ψ_1 by $h=\frac{f_0}{g}\Psi_1$. Thus, we assume that we have an observational stream function Ψ_1^{obs}. These observations are only available at times t_i, $i=1\dots N$, over the data assimilation period $[0,T]$, and are also discrete in space. We consider then that the vector $\Psi_1^{\text{obs}}(t_i)$ represents the set of available observations of the ocean surface at time t_i.

The control vector u (which has to be determined) is the initial state of the stream functions at all layers $(\Psi_k(0))_{k=1\dots n}$.

The numerical experiments have been made on a three-layered square ocean. The basin has horizontal dimensions of 4000 km×4000 km and its depth is 5 km. The layers' depths are 300 m for the surface layer, 700 m for the intermediate layer, and 4000 m for the bottom layer. The ocean is discretized by a Cartesian mesh of 200×200×3 grid points. The time step is 1.5 h. The initial conditions are chosen equal to zero for a six-year ocean spin-up phase, the final state of which becoming the initial state of the data assimilation period. Then the assimilation period starts (time $t=0$) with this initial condition ($\Psi_k(0)$), and lasts 5 days (time $t=T$), i.e. 80 time steps. Data are available every $n_x=5$ gridpoints of the model, with a time sampling of 7.5 h (i.e. every $n_t=5$ time steps). The first guess of the assimilation experiments is chosen to be the reference state of the ocean one year before the beginning of the assimilation period. Finally, the nudging matrices $\mathbf{K}$ and $\mathbf{K}'$ are set to kC^T and $k'C^T$, respectively (see Sect. 2.4), where $k=1.8\times10^{-6}$ and $k'=7.4\times10^{-6}$.

5.2 Convergence of the BFN algorithm in the case of perfect observations

We first focus our interest on the numerical convergence of the BFN algorithm. In this part, we have used the exact (i.e. unnoised) observations.

Figure 11 shows the evolution of the difference between two consecutive BFN iterates (a), and that of the difference between the BFN iterates and the exact solution (b), versus the number of iterations of the BFN scheme. We can see that after 20 iterations, the successive iterates are almost equal to each other, and they nearly converge towards the exact initial condition. This proves the numerical convergence of the BFN algorithm in this case.

5.3 Comparison with 4D-VAR in the case of perfect observations

The 4D-VAR algorithm requires the computation of the adjoint state, which consists in n vectors (here $n=3$), representing the components of the adjoint state in each layer. The gradient of the cost function is itself made of n components, representing the derivatives of the cost function with respect to the initial state of the stream function at each layer. Each of these components is computed from the value of the adjoint state at time $t=0$ (final time of the backward integration of the adjoint equations). The detailed expressions of the n components of the gradient of the cost function are given in Luong et al. (1998); Auroux and Blum (2004). The 4D-VAR functional contains both a regularization term, depending on the potential vorticity, and an observation term, quantifying the difference between the observations and the state function (see e.g. Luong et al., 1998). In all the following discussion, we only refer to the observation part of the cost function and its gradients. The minimization of the cost function is performed using a limited-memory BFGS algorithm (M1QN3 routine, from the MODULOPT library: Gilbert and Lemaréchal, 1989).

In this section, we have compared the BFN and 4D-VAR algorithms. Figure 12 shows the true initial state (a), the initialization vector used for both 4D-VAR and BFN algorithms

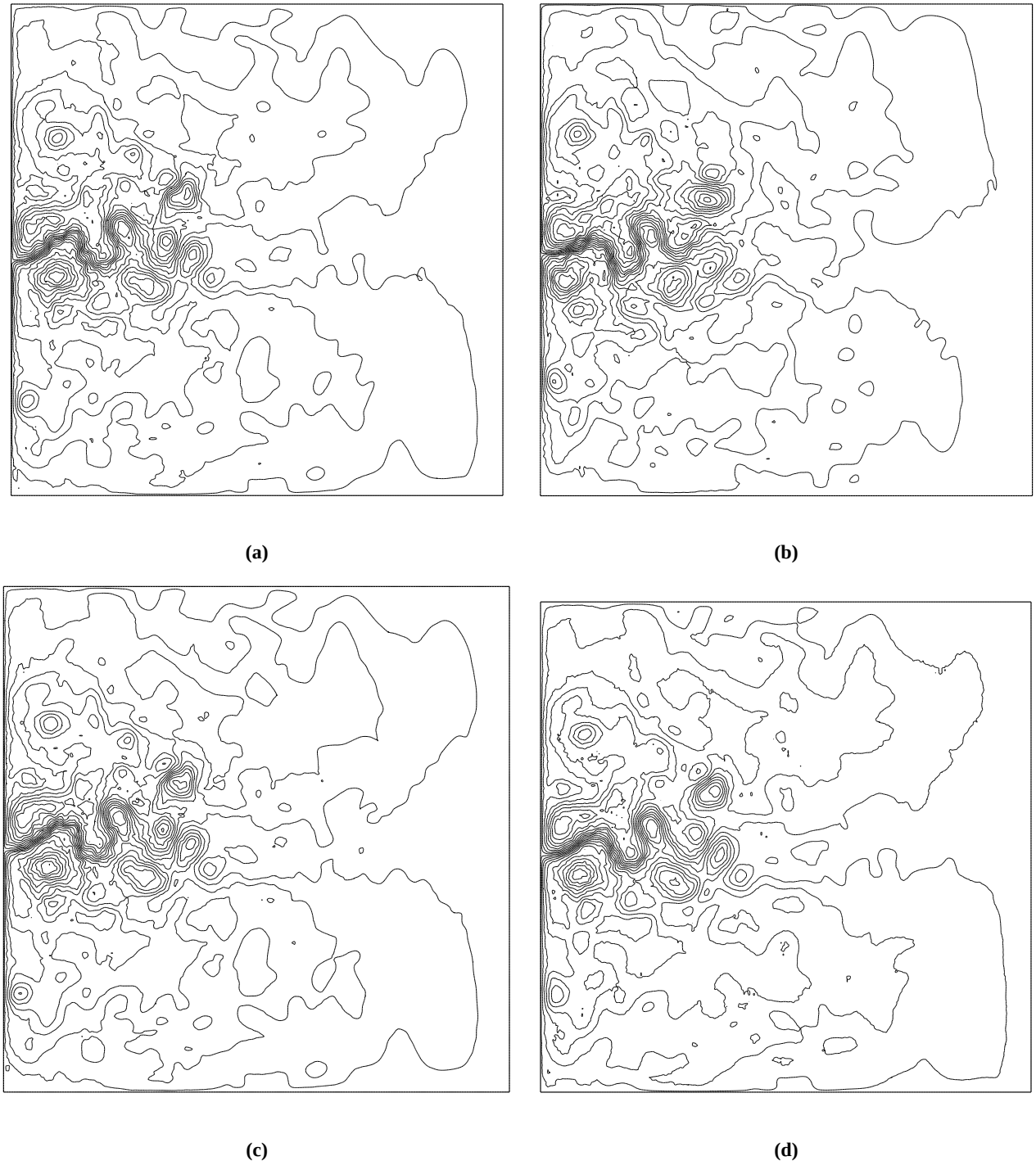

Fig. 12. Upper layer of: true initial state **(a)**, initialization (or background) state for both 4D-VAR and BFN algorithms **(b)**, and initial states identified by the BFN **(c)** and 4D-VAR **(d)** algorithms.

(b), and the initial states identified by the BFN (c) and 4D-VAR (d) algorithms respectively. Only the first (uppermost) of the three layers is represented.

In order to quantify these results, Fig. 13 shows the evolution of the 4D-VAR cost function and the 3 components of its gradient versus the number of iterations for the BFN (a) and 4D-VAR (b) algorithms. It can be seen that both the cost function and its gradients decrease more rapidly with successive iterations using the BFN scheme than using 4D-VAR. Recall that the computational cost of a BFN iteration is

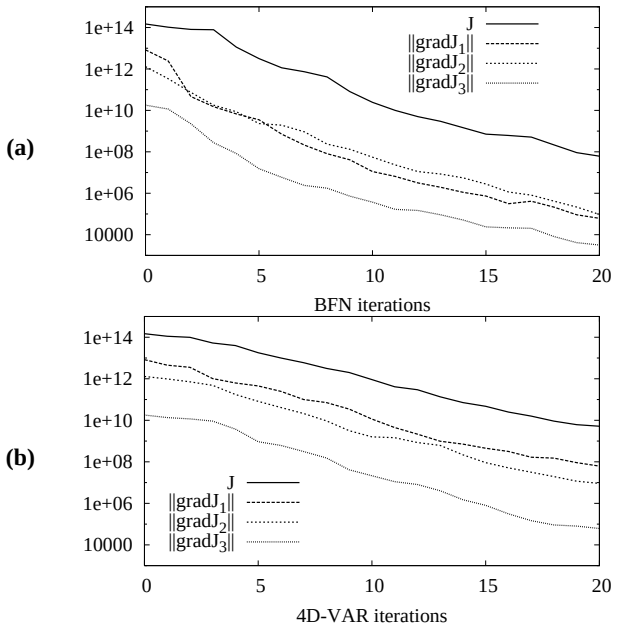

Fig. 13. Evolution of the 4D-VAR cost function and of the 3 gradients of the cost function in the 3 ocean layers for the BFN iterates versus the number of iterations **(a)** and for the 4D-VAR iterates versus the number of iterations **(b)**.

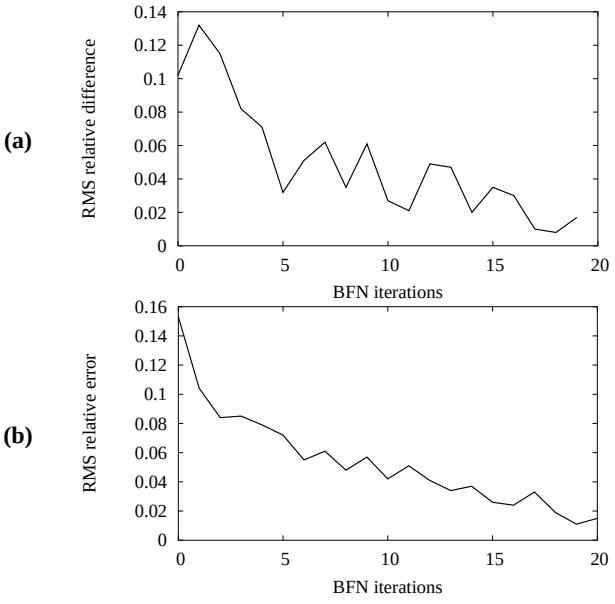

Fig. 14. RMS relative difference between two consecutive BFN iterates **(a)** and between the BFN iterates and the exact solution **(b)** versus the number of BFN iterations in the case of noisy observations.

almost equivalent to the cost of a 4D-VAR iteration, as both algorithms consist in a forward model integration and a backward model integration (and the models involved have the same sizes). In 20 iterations, the cost function has been divided by 4 orders with the 4D-VAR, and by 6 orders with the BFN. From this, we can conclude that the BFN algorithm is more efficient than the 4D-VAR algorithm in minimizing, in the same computing time, the quadratic difference between the observations and the corresponding state variables. This is all the more remarkable as, unlike the case of 4D-VAR, this is not the primary goal of the BFN scheme. The next point will be to check if these findings remain valid when the observations are noisy.

5.4 Convergence and comparison with 4D-VAR in the case of noisy observations

We now consider noisy observations (see previous subsections for details about the noise distribution). We have performed the same experiments as in the previous subsections and first studied the convergence of the BFN algorithm, and then compared it with the 4D-VAR algorithm.

Figure 14 shows the evolution of the relative difference between two consecutive BFN iterates (a) and of the difference between the BFN iterates and the exact solution (b), versus the number of iterations in the case of noisy observations. We can see that after 20 iterations, the successive iterates are nearly equal, and they are relatively close to the exact initial condition. Even if the convergence is a little bit less obvious than in the case of perfect observations, these two figures show that the BFN algorithm converges.

As in the previous subsections in the case of perfect observations, we have compared the BFN and 4D-VAR algorithms. Figure 15 shows the evolution of the 4D-VAR cost function and gradients for the BFN (a) and 4D-VAR (b) algorithms. We still observe a decrease of both the cost function and its gradients with the BFN iterates, but contrary to the previous case (perfect observations), the decrease is almost the same as with the 4D-VAR algorithm, even though after 10 iterations the BFN provides better results. The 4D-VAR algorithm seems more efficient at smoothing the observations (Gaussian white noise) than the BFN algorithm, but this is highly related to the choice of the regularization term in the 4D-VAR cost function.

Figure 16 shows, for each of the three layers, the RMS relative difference between the BFN trajectory (computed using the last BFN state at time t=0 as initial condition) and the reference trajectory as a solid line, and between the 4D-VAR trajectory (computed using the last initial state produced by the minimization process) and the reference trajectory as a dotted line, versus time. The first 5 days correspond to the assimilation period, and the next 15 days correspond to the forecast period.

The first point is that the BFN and 4D-VAR reconstruction errors have similar global behaviours: first decreasing at the

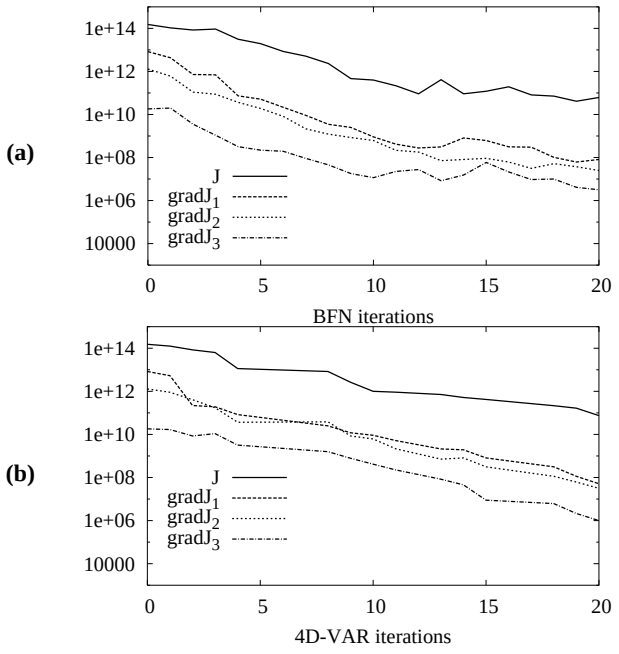

Fig. 15. Evolution of the 4D-VAR cost function and of the 3 gradients of the cost function in the 3 ocean layers for the BFN iterates versus the number of BFN iterations **(a)** and for the 4D-VAR iterates versus the number of 4D-VAR iterations **(b)** in the case of noisy observations.

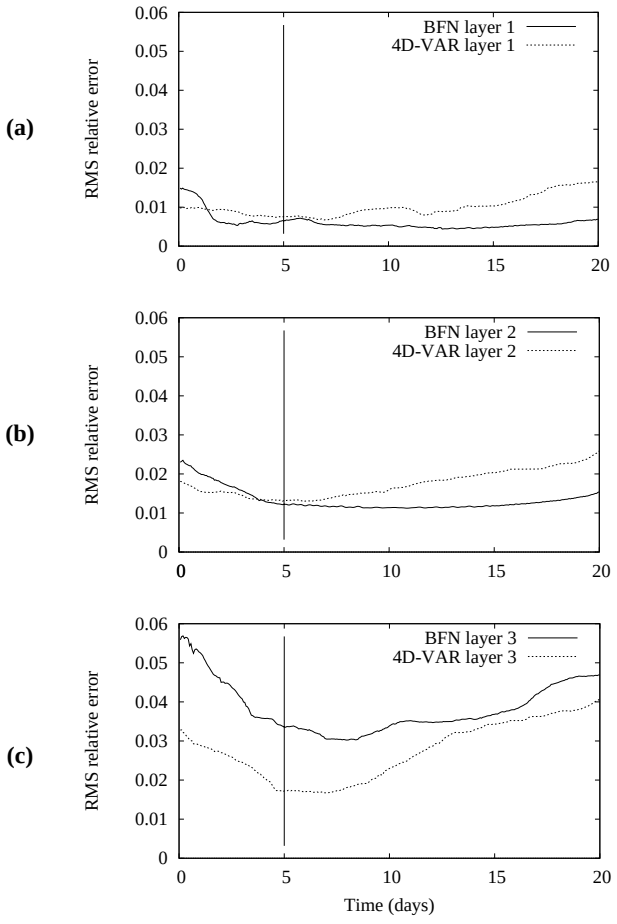

Fig. 16. Evolution in time of the RMS relative difference between the reference trajectory and the identified trajectories for the BFN (solid line) and 4D-VAR (dotted line) algorithms, versus time, for each layer: from surface **(a)** to bottom **(c)**.

beginning, during the assimilation period, and then increasing over the forecast period (see e.g. Pires et al., 1996, for a detailed study of the 4D-VAR estimation error). Even if the reconstruction error on the initial condition is much higher with the BFN algorithm than with 4D-VAR, the BFN error decreases much more steeply and for a longer time than the 4D-VAR one, and increases less quickly at the end of the forecast period. This remark should be compared with the last paragraphs of Sect. 4.3, describing the same behaviour of the BFN algorithm on Burgers' equation. The quality of the initial condition reconstruction is better using the 4D-VAR algorithm, but the BFN algorithm provides a comparable final estimation, and even a better forecast. Another interesting point is that, even if only surface observations are assimilated, the identification of the intermediate and bottom layers is quite good. The 4D-VAR algorithm was already known to propagate surface information to all layers (Luong et al., 1998), but it is also the case for the BFN algorithm, even if this algorithm is less efficient on the bottom layer.

5.5 Comparison with 4D-VAR in the case of an imperfect model

In this subsection, we have performed twin experiments with the aim of identifying the initial condition using observations generated by a different model than the one used for the assimilation. There are no more observation errors in this experiment. We still have the same exact initial condition as previously, and an experiment is run with a biased model (with a 2% model error), from which surface data are extracted every 5 grid points of the model and every 5 time steps. These simulated data are not noised. The model error enables us to generate observations derived from a different model, corrected by some a priori estimations of neglected effects in the theoretical model. From now on, we forget the model error, and we want to identify the initial condition using the assimilating model (without any additional term) and the data we have extracted from the reference (biased) model. Assimilating these observations, generated by the reference model, in the assimilating model allows us to study the impact of an imperfect model on the BFN scheme.

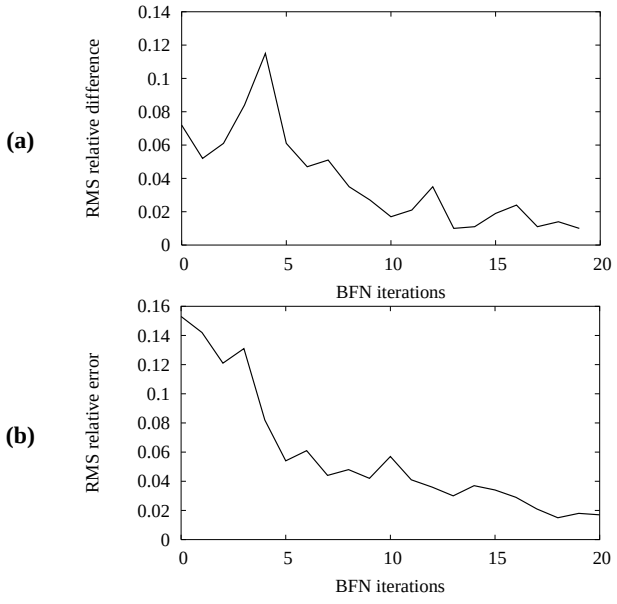

Fig. 17. RMS relative difference between two consecutive BFN iterates **(a)** and between the BFN iterates and the exact solution **(b)** versus the number of BFN iterations in the case of an imperfect model.

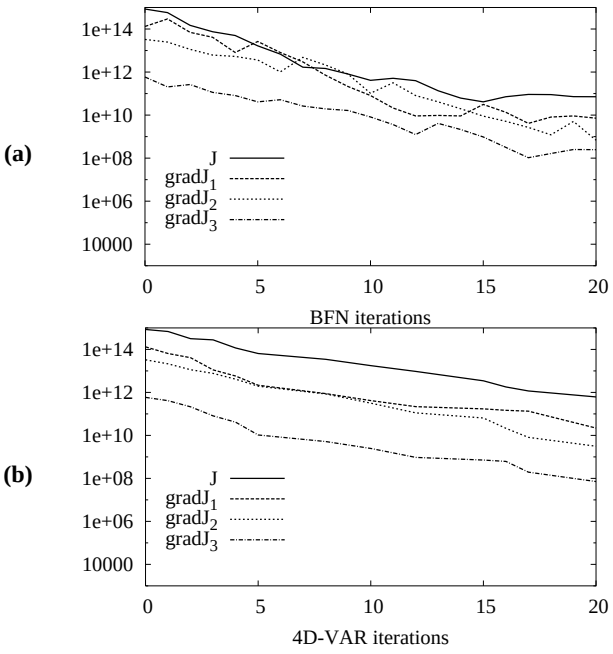

Fig. 18. Evolution of the 4D-VAR cost function and of the 3 gradients of the cost function in the 3 ocean layers for the BFN iterates versus the number of iterations **(a)** and for the 4D-VAR iterates versus the number of iterations **(b)** in the case of an imperfect model.

Figures 17 and 18 are the analogues of Figs. 14 and 15 for this new experiment. The BFN algorithm still converges quite well. Although the decrease of the gradients of the cost function is faster with 4D-VAR than with BFN, the cost function itself decreases more rapidly with BFN, and after 20 iterations, it is 10 times smaller than with the 4D-VAR algorithm.

The 4D-VAR algorithm used in this experiment does not take into account any model error term (which exists in reality), and then gives worse results than the BFN. The slower decrease of the gradients can be easily explained by the fact that the goal of the BFN algorithm is not to minimize the cost function and its gradients, but to identify a trajectory, whereas the 4D-VAR algorithm explicitly aims to decrease the gradients of the cost function. On the other hand, the BFN algorithm is taking advantage of the feedback terms directly added into the model equations, and these additional terms can consequently be considered as a corrective term in the model, and thus as a model error term: the nudging method implicitly uses the observations in the partial reconstruction of the model error.

6 Conclusions

The BFN algorithm appears to be a very promising data assimilation method. It is extremely easy to implement: no linearization of the model equations, no computation of the adjoint state, no optimization algorithm. The only necessary work is to add a relaxation term to the model equations. The key point in the backward integration is that the nudging term (with the opposite sign to the forward integration one) makes it numerically stable. Hence the nudging (or relaxation) term has a double role: it forces the model to the observations and it stabilizes the numerical integration. It is simultaneously a penalization and regularization term.

The BFN algorithm has been compared with the variational method on several types of non-linear systems: Lorenz, Burgers, quasi-geostrophic model. The conclusion of the various experiments performed in Sects. 3 to 5 is that the BFN algorithm is better than the variational method for the same number of iterations (and hence for the same computing time). It converges in a small number of iterations. Of course the initial condition is usually poorly identified by the BFN scheme, but on the other hand, the final state of the assimilation period is much better identified by the BFN algorithm than by the variational assimilation algorithm, which is a key point for the prediction phase that starts at the end of the assimilation period. Hence the prediction phase is usually better when it comes after an assimilation period treated by the BFN algorithm, rather than by a variational assimilation method.

The two algorithms can be combined, in the sense that one can perform several BFN iterations before switching to the

variational method and this will considerably accelerate the convergence of the variational method. Finally the BFN algorithm enables one to consider the problem of imperfect models at no additional cost, as the model equations are not strong constraints in this nudging method (as they are usually in a variational method) and the relaxation term can be seen as a model error term.

The determination of the nudging coefficients (or matrices) should still be improved, particularly by a numerical stability study of the backward integration, which will give the optimal nudging coefficients that make the backward integration stable. Moreover the algorithm will be tested on more sophisticated models (shallow-water, primitive equations, ...) with various types of observations (satellite measurements, in situ data, ...), in order to determine whether this algorithm can be used in more realistic conditions.

Acknowledgements. This work has been supported by a French national programme of INSU (Institut National des Sciences de l'Univers du CNRS) called LEFE (Les Enveloppes Fluides et l'Environnement) and especially by its Data Assimilation part. This work has been partly done within the MOISE (Modelling, Observations, Identification for Environmental Sciences) research project of INRIA Rhône-Alpes. The authors would like to thank O. Talagrand for his useful comments.

Edited by: O. Talagrand
Reviewed by: two anonymous referees

References

Arnold, M. and Datta, B. N.: Single-input eigenvalue assignment algorithms: A close look, SIAM J. Matrix Anal. Appl., 19(2), 444–467, 1998.

Auroux, D.: Etude de différentes méthodes d'assimilation de données pour l'environnement, Ph.D. thesis, University of Nice Sophia-Antipolis, 2003.

Auroux, D. and Blum, J.: Data assimilation methods for an oceanographic problem, vol. XVI of Mathematics in Industry Series: Multidisciplinary methods for analysis, optimization and control of complex systems, pp. 179–194, Springer-Verlag, 2004.

Auroux, D. and Blum, J.: Back and forth nudging algorithm for data assimilation problems, C. R. Acad. Sci. Ser. I, 340, 873–878, 2005.

Bao, J.-W. and Errico, R. M.: An adjoint examination of a nudging method for data assimilation, Mon. Weather Rev., 125, 1355–1373, 1997.

Blayo, E., Verron, J., and Molines, J.-M.: Assimilation of TOPEX/POSEIDON altimeter data into a circulation model of the North Atlantic, J. Geophys. Res., 99(C12), 24 691–24 705, 1994.

Bonnans, K. and Rouchon, P.: Commande et optimisation de systèmes dynamiques, Les Éditions de l'École Polytechnique, Palaiseau, 2005.

Datta, B. N.: An algorithm to assign eigenvalues in a Hessenberg matrix: Single-input case, IEEE Trans. Automatic Control, 32(5), 414–417, 1987.

Fisher, M. and Courtier, P.: Estimating the covariance matrix of analysis and forecast error in variational data assimilation, Tech. Rep. 220, ECMWF, 1995.

Gilbert, J.-C. and Lemaréchal, C.: Some numerical experiments with variable-storage quasi-Newton algorithms, Math. Prog., 45, 407–435, 1989.

Hoke, J. and Anthes, R. A.: The initialization of numerical models by a dynamic initialization technique, Mon. Weather Rev., 104, 1551–1556, 1976.

Holland, W. R.: The role of mesoscale eddies in the general circulation of the ocean, J. Phys. Ocean., 8(3), 363–392, 1978.

Kalnay, E., Ki Park, S., Pu, Z.-X., and Gao, J.: Application of the quasi-inverse method to data assimilation, Mon. Weather Rev., 128, 864–875, 2000.

Le Dimet, F.-X. and Talagrand, O.: Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects, Tellus, 38A, 97–110, 1986.

Lorenz, E. N.: Deterministic non periodic flow, J. Atmos. Sci., 20, 130–141, 1963.

Luenberger, D.: Observers for multivariable systems, IEEE Trans. Autom. Contr., 11, 190–197, 1966.

Luong, B., Blum, J., and Verron, J.: A variational method for the resolution of a data assimilation problem in oceanography, Inverse Problems, 14, 979–997, 1998.

Nocedal, J.: Updating quasi-Newton matrices with limited storage, Math. Comput., 35, 773–782, 1980.

Pires, C., Vautard, R., and Talagrand, O.: On extending the limits of variational assimilation in nonlinear chaotic systems, Tellus, 48A, 96–121, 1996.

Stauffer, D. R. and Bao, J. W.: Optimal determination of nudging coefficients using the adjoint equations, Tellus, 45A, 358–369, 1993.

Stauffer, D. R. and Seaman, N. L.: Use of four dimensional data assimilation in a limited area mesoscale model – Part 1: Experiments with synoptic-scale data, Mon. Weather Rev., 118, 1250–1277, 1990.

Talagrand, O.: A study of the dynamics of four-dimensional data assimilation, Tellus, 33A, 43–60, 1981a.

Talagrand, O.: On the mathematics of data assimilation, Tellus, 33A, 321–339, 1981b.

Trélat, E.: Contrôle optimal: théorie et applications, Vuibert, Paris, 2005.

Verron, J.: Altimeter data assimilation into an ocean circulation model: sensitivity to orbital parameters, J. Geophys. Res., 95(C7), 443–459, 1990.

Verron, J. and Holland, W. R.: Impact de données d'altimétrie satellitaire sur les simulations numériques des circulations générales océaniques aux latitudes moyennes, Ann. Geophys., 7, 31–46, 1989,
http://www.ann-geophys.net/7/31/1989/.

Verron, J., Molines, J.-M., and Blayo, E.: Assimilation of Geosat data into a quasigeostrophic model of the North Atlantic between 20^oN and 50^oN: preliminary results, Oceanol. Acta, 15(5), 575–583, 1992.

Vidard, P.-A., Le Dimet, F.-X., and Piacentini, A.: Determination of optimal nudging coefficients, Tellus, 55A, 1–15, 2003.

Zou, X., Navon, I. M., and Le Dimet, F.-X.: An optimal nudging data assimilation scheme using parameter estimation, Q. J. Roy. Meteorol. Soc., 118, 1163–1186, 1992.