An easy-to-implement and efficient data assimilation method for the identification of the initial condition: the Back and Forth Nudging (BFN) algorithm

Didier Auroux[1], **Patrick Bansart**[2], **Jacques Blum**[2]

[1] Institut de Mathématiques, 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

E-mail: didier.auroux@math.univ-toulouse.fr

Abstract. This paper deals with a new data assimilation algorithm called the Back and Forth Nudging. The standard nudging technique consists in adding to the model equations a relaxation term, which is supposed to force the model to the observations. The BFN algorithm consists of repeating forward and backward resolutions of the model with relaxation (or nudging) terms, that have opposite signs in the direct and inverse resolutions, so as to make the backward evolution numerically stable. We then applied the Back and Forth Nudging algorithm to a simple non-linear model: the 1D viscous Burgers' equations. The tests were carried out through several cases relative to the precision and density of the observations. These simulations were then compared with both the variational assimilation (VAR) and quasi-inverse (QIL) algorithms. The comparisons deal with the programming, the convergence, and time computing for each of these three algorithms.

1. Introduction

Environmental scientists are increasingly turning to inverse methods for combining in an optimal manner all the sources of information coming from theory, numerical models and data. The aim of data assimilation is precisely to combine the observations and models, in order to retrieve a coherent and precise state of the system from a set of discrete space-time data.

Nudging is a data assimilation method that uses dynamical relaxation to adjust a model toward observations. The standard nudging algorithm consists in adding to the state equations of a dynamical system a feedback term, which is proportional to the difference between the observation and its equivalent quantity computed by the resolution of the state equations. The model appears then 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 term small in comparison with 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 backward nudging algorithm consists in solving the state equations of the model, backwards in time, starting from the observation of the system state at the final time. A nudging

term, with the opposite sign compared to the standard nudging algorithm, is added to the state equations, and the final obtained state in the backward resolution is in fact an approximation of the initial state of the system. The Back and Forth Nudging (BFN) algorithm, introduced in [1], consists in solving first the forward nudging equation and then the model equations backwards in time with a relaxation term which has the opposite sign to the one introduced in the forward equation. The initial condition of this backward resolution is the final state provided by the standard nudging method. One finally obtains an estimate of the initial state of the system after resolution of this backward equation.

We repeat these forward and backward resolutions (with the relaxation terms) until convergence of the algorithm. The BFN algorithm can be compared to the variational algorithm, which consists also in a sequence of forward and backward resolutions. But in our algorithm, even for nonlinear problems, it is useless to linearize the system and the backward system is not the adjoint equation but the model equation, with an extra feedback term that stabilizes the resolution of this ill-posed backward resolution.

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 some dynamical equations of the form

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

with an initial condition $X(0) = x_0$. In this equation, the function F represents the discretized model operator. We will denote by C the observation operator, allowing us to compare the observations $X_{obs}(t)$ with the corresponding $C(X(t))$, deduced from the state vector $X(t)$. If we apply the standard nudging method to the model, we obtain

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

where K is the nudging (or gain) matrix [3]. The model appears then 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 and C are matrices), the forward nudging method is nothing else as the Luenberger observer [6], also called asymptotic observer, where the matrix K can be chosen so that the error goes to zero when time goes to infinity.

2.2. Back and Forth Nudging (BFN) algorithm

In this algorithm, we repeat the forward and backward resolutions (with the feedback terms) until convergence of the algorithm:

$$\begin{cases} \dfrac{dX_k}{dt} = F(X_k) + K(X_{obs} - C(X_k)), \quad 0 < t < T, \quad k \geq 1, \\ X_k(0) = \overline{X}_{k-1}(0), \end{cases} \tag{3}$$

$$\begin{cases} \dfrac{d\overline{X}_k}{dt} = F(\overline{X}_k) - K'(X_{obs} - C(\overline{X}_k)), \quad T > t > 0, \quad k \geq 1, \\ \overline{X}_k(T) = X_k(T), \end{cases} \tag{4}$$

with the initial condition $\overline{X}_0(0) = x_0$.

The K and K' matrices are often chosen as simple scalar gains. In the forward part of the algorithm, the coefficient K is chosen as in the literature of the standard nudging method,

whereas the coefficient K' is usually chosen as being the smallest coefficient that makes the backward resolution stable. One can see [1] for the proof of convergence of this algorithm in a simple case (linear model and full observations).

3. Numerical experiments on Burgers' equation

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

$$\begin{cases} \dfrac{\partial u}{\partial t} + \dfrac{\partial}{\partial x}(\dfrac{u^2}{2}) = \nu \dfrac{\partial^2 u}{\partial x^2}, \quad 0 < x < 1, \quad 0 < t < T, \\ u(0,t) = u(1,t) = 0, \quad 0 \leq t \leq T, \\ u(x,0) = u_0(x), \quad 0 < x < 1, \end{cases} \tag{5}$$

where $u(x,t)$ is the state variable and $\nu > 0$ is the dissipation coefficient.

3.1. Discretization scheme

We have implemented the following discretization scheme:

$$u_j^{n+1} = u_j^n + dt \left[-\frac{1}{2} \frac{(u_{j+1}^n)^2 - (u_{j-1}^n)^2}{2dx} + \nu \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{dx^2} \right], \ 0 < j < J, \ 0 \leq n < N, \tag{6}$$

where u_j^n represents the discrete solution at time step n, $0 \leq n \leq N$, and at space point j, $0 \leq j \leq J$. More explicitely, u_j^n represents an approximation of $u(x_j, t^n)$, where $x_j = j \times dx$ and $t^n = n \times dt$. Finally, J and N are respectively the number of sub-intervals in which the space $[0;1]$ and time $[0;T]$ intervals have been split, and $dx = \frac{1}{J}$ and $dt = \frac{T}{N}$.

The nonlinear term is discretized with a time explicit scheme and the diffusive term with a time implicit scheme. Finally, we have used centered finite difference schemes for both space derivatives. This scheme has been validated on an analytical solution.

The discretized boundary and initial conditions are the following:

$$u_0^n = u_J^n = 0, \ \forall \ 0 \leq n \leq N; \qquad u_j^0 = u_0(x_j), \ \forall \ 0 < j < J. \tag{7}$$

3.2. Application of the BFN algorithm to Burgers' equation

We first consider the continuous equations of the BFN algorithm applied to Burgers' equation:

$$\begin{cases} \dfrac{\partial u_k}{\partial t} + \dfrac{\partial}{\partial x}(\dfrac{u_k^2}{2}) = \nu \dfrac{\partial^2 u_k}{\partial x^2} + K(u_{obs} - u_k), \quad 0 < t < T, \quad 0 < x < 1, \quad k \geq 1, \\ u_k(x,0) = \overline{u}_{k-1}(x,0), \quad 0 < x < 1, \qquad K > 0, \end{cases} \tag{8}$$

$$\begin{cases} \dfrac{\partial \overline{u_k}}{\partial t} + \dfrac{\partial}{\partial x}(\dfrac{\overline{u}_k^2}{2}) = \nu \dfrac{\partial^2 \overline{u}_k}{\partial x^2} - K'(u_{obs} - \overline{u}_k), \quad T > t > 0, \quad 0 < x < 1, \quad k \geq 1, \\ \overline{u}_k(x,T) = u_k(x,T), \quad 0 < x < 1, \qquad K' > 0, \end{cases} \tag{9}$$

with $\overline{u}_0(x,0) = u_b(x)$, where u_b is the first guess of the initial condition (for example the background state coming from the simulation of a previous period), and where K and K' are respectively the forward and backward nudging scalar gains. Note that for simplicity reasons, we assume here the observations to be complete in space, i.e. C is the identity operator.

The discretization of equations $(8-9)$ has been done with the same discretization schemes as for equation (5) (see equation (6) for example), the nudging terms being discretized in a time explicit and space centered way.

3.3. Application of the VAR algorithm to Burgers' equation

The variational assimilation (VAR) algorithm is based on the minimization of a cost function, which measures the discrepancy between the observations and the corresponding model states [5]. In order to find the solution that minimizes this cost function, we have to compute its gradient, using the adjoint equations. We present in the following this discretized algorithm.

First, the discrete cost function is: $J(V) = \frac{1}{2}\sum_{n=0}^{N}\sum_{j=1}^{J-1}(u_j^n - {u_{obs}}_j^n)^2$, where $V = \left(u_0^0, \dots u_J^0\right)^T$ is the unknown initial condition.

We now have to compute the gradient of the cost function J. For this purpose, we use the lagrangian method. The lagrangian is:

$$L(u,V,p) \ = J(V) + \sum_{n=0}^{N-1}\sum_{j=1}^{J-1} p_j^{n+1}\left[\frac{u_j^{n+1}-u_j^n}{dt} + \frac{1}{2}\frac{(u_{j+1}^n)^2-(u_{j-1}^n)^2}{2dx} - \nu\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{dx^2}\right]. \tag{10}$$

Using formulas of discrete integration by parts, and assuming that p satisfies the same boundary conditions as u, we can rewrite the lagrangian as follows:

$$L(u,V,p) = J(V) + \frac{1}{dt}\sum_{j=1}^{J-1} p_j^N u_j^N - \frac{1}{dt}P^0.V - \sum_{n=0}^{N-1}\sum_{j=1}^{J-1} u_j^n\left(\frac{p_j^{n+1}-p_j^n}{dt}\right)$$
$$- \sum_{n=0}^{N-1}\sum_{j=1}^{J-1}\frac{1}{2dx}\frac{(u_j^n)^2}{2}(p_{j+1}^{n+1}-p_{j-1}^{n+1}) - \sum_{n=0}^{N-1}\sum_{j=1}^{J-1}\frac{\nu}{dx^2}u_j^n(p_{j+1}^n - 2p_j^n + p_{j-1}^n),$$

where $P^0 = (p_0^0, \dots, p_J^0)^T$. By derivating the lagrangian at the optimum with respect to the control variable V, we obtain the gradient of the cost function:

$$\frac{\partial L(u,V,p)}{\partial V} = \nabla J(V) = -\frac{1}{dt}P^0. \tag{11}$$

Then, we consider the derivative of the lagrangian with respect to u, in order to get the adjoint equation:

$$-\frac{p_j^{n+1}-p_j^n}{dt} - u_j^n\frac{p_{j+1}^{n+1}-p_{j-1}^{n+1}}{2dx} - \nu\frac{p_{j+1}^n - 2p_j^n + p_{j-1}^n}{dx^2} + (u_j^n - {u_{obs}}_j^n) = 0, \quad N > n \geq 0, \tag{12}$$

the final condition being: $p_j^N = -dt(u_j^N - {u_{obs}}_j^N)$.

3.4. Application of the QIL algorithm to the Burgers' equation

The Quasi-Inverse (QI) and Quasi-Inverse Linear (QIL) algorithms were introduced by Pu, Kalnay, Sela and Szunyogh in 1997 [7]. They were then tested on the Burgers' equation and compared to the variational assimilation in Kalnay, Park, Pu and Gao's article in 2000 [4].

Like the BFN and VAR algorithms, the initialization of the quasi-inverse algorithm is done with a wrong initial condition, e.g. a background state provided by a data assimilation on a previous period. Then, it computes the forecast error at the final time, between the final observation and the final state corresponding to the wrong initial condition. In order to find the best estimation of the true initial condition, the algorithm uses an approximation of the inverse of the tangent linear model (TLM). The inverse TLM approximation consists of simply running the TLM backward in time and changing the sign of the dissipative terms in order to avoid computational blow-up. We will see later the consequences of this choice by comparing

the QIL method with the BFN scheme. Then, the QIL consists of finding an estimation of the initial error from the forecast error thanks to this quasi-inverse TLM.

The model M_t describes the evolution of the system from initial time to time t. We denote by X_0 the initial condition, and the corresponding solution at time t is given by

$$X_t = M_t(X_0) \tag{13}$$

The corresponding tangent linear model L describes the evolution of the initial perturbation δX_0 in time:

$$M_t(X_0 + \delta X_0) = M_t(X_0) + L_t(\delta X_0) + O(\delta X_0)^2. \tag{14}$$

Besides, we look for a perturbation δX_0 such that the final state is close to the final observation:

$$M_t(X_0 + \delta X_0) \simeq M_t(X_0) + L_t(\delta X_0) \simeq X_t^{obs} \tag{15}$$

where X_t^{obs} is the observation of the system at the final time. Then, we use the quasi-inverse TLM in order to find an estimation of the initial condition error:

$$\delta X_0 = L_t^{-1}[X_t^{obs} - M_t(X_0)] \tag{16}$$

where L_t^{-1} is the quasi-inverse TLM. Then we add this estimation error to the initial condition, and we get a new initial condition which should be closer to the true one. This process is then repeated iteratively.

We now consider the TLM for Burgers' equation, in which we denote by $\widehat{u}$ the perturbation of the trajectory:

$$\frac{\partial \widehat{u}}{\partial t} = -u\frac{\partial \widehat{u}}{\partial x} - \widehat{u}\frac{\partial u}{\partial x} + \nu\frac{\partial^2 \widehat{u}}{\partial x^2}. \tag{17}$$

The quasi-inverse is then simply given by the same equation, with an opposite diffusion coefficient:

$$\frac{\partial \widehat{u}}{\partial t} = -u\frac{\partial \widehat{u}}{\partial x} - \widehat{u}\frac{\partial u}{\partial x} - \nu\frac{\partial^2 \widehat{u}}{\partial x^2}, \tag{18}$$

with the following final condition, at time $t = T$: $\widehat{u}(T) = u_{obs}(T) - u(T)$. The QIL will give us $\widehat{u}(0)$, and the new initial condition will be $u(0) + \widehat{u}(0)$.

We used the same numerical scheme as the authors [4]: a leap-frog scheme for the advection and a Dufort-Frankel scheme for the dissipation. The corresponding discretization of the direct Burgers' equation is:

$$\frac{u_j^{n+1} - u_j^{n-1}}{2dt} = -u_j^n\frac{u_{j+1}^n - u_{j-1}^n}{2dx} + \nu\frac{u_{j+1}^n - (u_j^{n+1} + u_j^{n-1}) + u_{j-1}^n}{dx^2}, \quad 0 < n < N. \tag{19}$$

This scheme is a two-step scheme. So, we used the previously mentioned scheme (see the beginning of section 3) to compute the second time step or the penultimate one for the backward resolution. Therefore the discrete QIL is given by

$$\frac{\hat{u}_j^{n+1} - \hat{u}_j^{n-1}}{2dt} = -u_j^n\frac{\hat{u}_{j+1}^n - \hat{u}_{j-1}^n}{2dx} - \hat{u}_j^n\frac{u_{j+1}^n - u_{j-1}^n}{2dx} - \nu\frac{\hat{u}_{j+1}^n - (\hat{u}_j^{n+1} + \hat{u}_j^{n-1}) + \hat{u}_{j-1}^n}{dx^2}, \tag{20}$$

with the same boundary conditions as u, and with the final condition $\hat{u}_j^N = {u_{obs}}_j^N - u_j^N$, where the solution (u_j^n) comes from the forward resolution of the discretized Burgers' equation. After resolution, we get an estimation of the initial error $(\hat{u}_j^0)$, and then this process is repeated iteratively.

3.5. Numerical results

The numerical tests were performed with a gaussian initial condition. We consider a discretization grid with $N = 250$ time steps and $J = 100$ space steps, and $T = 5$. The dissipation coefficient is set to $\nu = 0.001$. The initialization of all algorithms is done with the exact initial condition multiplied by 0.25.

3.5.1. Comparison between VAR and BFN

Full and perfect observations We first assume that the observations are unnoised, and that we fully observe the state. The observations are represented in figure 1.

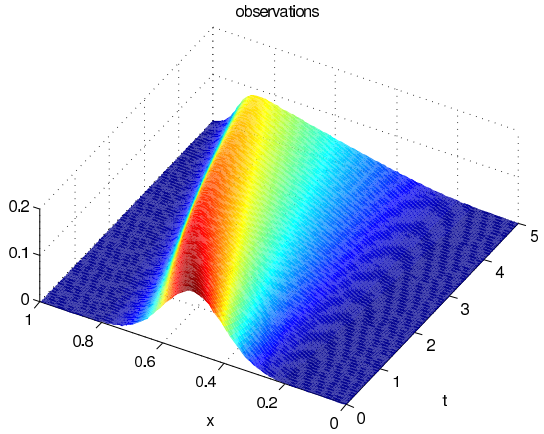

Figure 1. Perfect observations.

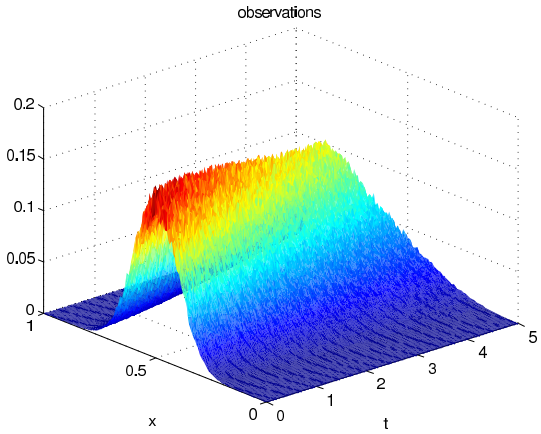

Figure 2. 25% noised observations.

The results we obtained with these perfect observations have been summarized in table 1. The "**C**" symbol means that the algorithm converged for the corresponding number of iterations, the convergence criterion being a 10^{-3} relative difference between two successive iterates. K is the nudging coefficient of the forward equation and K' is the one of the backward equation. The RMS relative difference is the discrepancy between the initial condition found by the algorithm and the exact one, using the L^2 norm: $\frac{\|u_{true}(x,0) - u(x,0)\|}{\|u_{true}(x,0)\|}$.

Table 1. Comparison VAR-BFN in the case of full perfect observations.

Number of iterations	2	12
RMS rel. diff. for VAR (in %)	32.2	0.088 **C**
RMS rel. diff. for BFN (in %, K=0.5, K'=100)	0.028 **C**	

Partial observations The results we obtained for partial (unnoised) observations have been summarized in table 2. In this case, the nudging terms are only applied at the instants where observations exist. dt_{obs} and dx_{obs} represent respectively the number of space steps and the number of time steps between two observations. For example, in the last column, we have an observation every 4 time and space steps. In order to improve the efficiency of the BFN algorithm, we have interpolated the observations in space. In this table, we also give the number of iterations the algorithms needed to achieve convergence.

Table 2. Comparison VAR-BFN in the case of partial perfect observations.

Algorithm		$dx_{obs} = 1$ $dt_{obs} = 4$	$dx_{obs} = 4$ $dt_{obs} = 1$	$dx_{obs} = 4$ $dt_{obs} = 4$
VAR	Number of iterations	10	10	10
	RMS relative difference (in %)	1.43	1.51	2.05
BFN	Number of iterations	2	2	2
	RMS relative difference (in %)	0.019	0.013	0.047
	$K = 0.5$; K':	6000	500	12000

Full and noised observations In this part, we now consider full and noised observations. We applied two amplitudes of noise: 10% and 25%. Figure 2 represents the observations with 25% noise. The results have been summarized in table 3.

Table 3. Comparison VAR-BFN in the case of noised observations.

Algorithm		noise = 10%	noise = 25%
VAR	Number of iterations	12	13
	RMS relative difference (in %)	6.32	14.1
BFN	Number of iterations	2	2
	RMS relative difference (in %)	8.65	14.8
	K	0.5	0.5
	K'	100	100

3.5.2. Comparison between QIL and BFN The original version of the QIL algorithm only uses the observation at the final time as we have seen above, whereas the BFN is more conceived to use observations all along the time period thanks to the nudging term in the resolution [4, 1]. Another main difference between these two algorithms is in their own process. Indeed, the BFN is a compromise between the model equations and the observations. So, the backward equation in the algorithm corresponds to the true backward Burgers' equation stabilized by a nudging term, whereas the QIL is not the true backward linear tangent model but a "quasi-inverse" one, since the stability of this backward equation is obtained by changing the sign of the dissipation term. This choice can be a source of issues in the case of a high physical dissipation or when the time period is too long.

Furthermore, we can notice that for the QIL, as for the BFN, the results and the convergence are highly sensitive to the number of time steps between the initial time and the first observation (the only one for the QIL). In order to make a precise comparison between the QIL and the BFN, an implementation of the QIL using several observations in time would be needed. Nevertheless, we can make a few first comparisons using one and only final observation on a short period of time (so that the QIL algorithm is expected to converge). The results are summarized in table 4.

We can clearly observe that the QIL algorithm gives very good results on a short period of time whereas the BFN is more efficient on a longer period. Furthermore, we must say that in the former case, the BFN only uses one observation while it is more conceived to use several

Table 4. Comparison between QIL and BFN

Algorithm		$T = 0.1$	$T = 0.3$	$T = 0.5$
QIL	Number of iterations	4	6	11
	RMS relative difference (in %)	0.0074	0.096	0.56
BFN	Number of iterations	3	5	6
	RMS relative difference (in %, $K = 0.5$, $K' = 800$)	0.042	0.11	0.20

observations all along the time period. Several tests were also performed on noised observations and the results show the same efficiency towards noise.

4. Conclusions

It clearly appears that the BFN provides very interesting results concerning the precision of the identified initial condition, even when the observations are sparse in time or are perturbed. Yet, we have to notice that the BFN is sensitive to the location of the first avalaible observation.

Another positive aspect of the BFN is that the computing time is much shorter than the variational assimilation algorithm, which needs at least ten iterations to converge.

The comparison between the BFN and QIL algorithms highligths that the QIL is only efficient on a quite short period of time. When we increase the time period, the BFN became more efficient than the QIL.

The comparison between the VAR and BFN algorithms shows that the variational assimilation gives better results when the observations are sparsed in space whereas the BFN needs an interpolation to be efficient. The VAR algorithm is very precise but it needs much more iterations and is a lot harder to implement than the BFN.

Indeed, the BFN is very easy to implement, in particular because it does not require neither any linearization nor minimization, and it only requires the discretization of the model equations with the extra nudging terms. Hence it is a very promising data assimilation algorithm, which has also been tested on other chaotic systems (Lorenz equations, quasi-geostrophic model) [2].

References

[1] Auroux D and Blum J 2005 Back and forth nudging algorithm for data assimilation problems *C. R. Acad. Sci. Ser. I* **340** pp 873–78

[2] Auroux D and Blum J 2008 A nudging-based data assimilation method: the Back and Forth Nudging (BFN) algorithm *Nonlin. Processes Geophys.* **15** pp 305–19

[3] Hoke J and Anthes A 1976 The initialization of numerical models by a dynamic initialization technique *Mon. Wea. Rev.* **104** pp 1551–56

[4] Kalnay E, Ki Park S, Pu Z-X and Gao J 2000 Application of the quasi-inverse method to data assimilation *Month. Weather Rev.* **128** pp 864–75

[5] Le Dimet F-X and Talagrand O Variational algorithms for analysis and assimilation of meteorogical observations: theoretical aspects *Tellus* **38A** pp 97–110

[6] Luenberger D 1966 Observers for multivariable systems *IEEE Trans. Autom. Contr.* **11** pp 190–97

[7] Pu Z-X, Kalnay E, Sela J and Szunyogh I 1997 Sensitivity of forecast errors to initial conditions with a quasi-inverse linear method *Month. Weather Rev.* **125** pp 2479–2503