Chapter 5
Reduced methods

5.1 ▪ Overview of reduction methods

Reduction methods aim at reducing the dimension of the workspace and therefore the computational cost of the problem. As DA often deals with large-scale problems, the tools offered by reduction can be very useful. This chapter gives an overview of the methods, as well as their applications within the DA framework.

The two following Chapters 6 and 7 focus more on specific widely used DA methods that employ such reductions.

5.1.1 ▪ What is problem reduction?

The general aim of reduced methods, or problem reduction, or dimension reduction, is to approximate a full-scale high-dimensional problem by a new, much smaller dimensional problem. This idea is widespread in applied mathematics. For example, in image compression, high-dimensional pixel representations of images are replaced by a reduced thresholded wavelet decomposition. Also, in signal processing, truncated Fourier series can be very good approximations of a given signal. Many other examples could be given, and the final idea would be similar: reduce the dimension of the problem to increase the computing efficiency and/or to allow larger problems to be considered.

The principle of reduced methods is to change variables from the vector space X of the system state into a reduced-dimensional state $\tilde{X} \subset X$. In general, the full solution (in X) is replaced by one of its projections on the reduced space $\tilde{X}$, and the reduced algorithms provide ways of computing this projection. The change of variable is usually performed via a change of basis, from a large-dimensional generic basis to a problem-adapted smaller basis, see, e.g., illustration in Figure 5.1.

The question is then how to choose an adequate basis, adapted to the problem, so that the subspace it spans is a good approximation of the full space. Before we explain how to choose a new basis, we review the various forms of reduction in DA below.

5.1.2 ▪ Some reduced bases used in DA

Various families of vectors can be used as reduced bases in DA: polynomial families; Lyapunov-type vectors; eigen/singular vectors; Fourier/sine/cosine families;

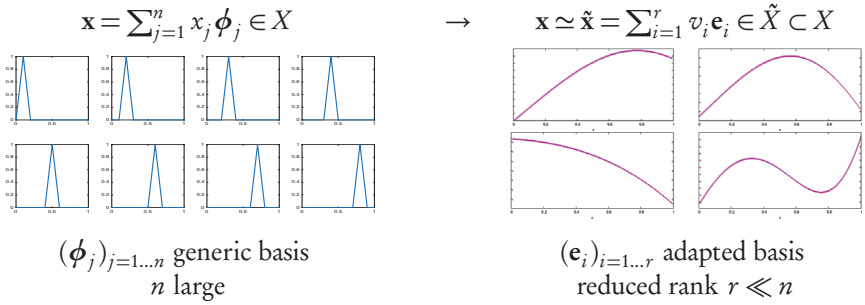

Figure 5.1. *Example of dimension reduction.*

and X-lets such as wavelets, curvelets, etc. We will give some details about each type of basis below.

5.1.2.1 ▪ Singular vectors and related bases

A widely used class of bases is related to eigen/singular value decompositions:

- forward/backward singular vectors,
- Lyapunov vectors,
- bred vectors,
- nonlinear singular vectors,
- principal component analysis (PCA), also called proper orthogonal decomposition (POD), empirical orthogonal functions (EOFs), etc.

Before we introduce forward/backward singular vectors, let us briefly recall the closely related linear algebra singular value decomposition (SVD). If $\mathbf{A}$ is an $n \times m$ real matrix, then it has a factorization, or SVD [Golub and van Loan, 2013], of the form

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}},$$

where Σ is an $n \times m$ rectangular diagonal matrix with nonnegative real numbers on the diagonal, called the singular values, and $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices of sizes $n \times n$ and $m \times m$ containing the left and right singular vectors, respectively. The nonzero singular values are the square roots of the nonzero eigenvalues of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{A}$, and $\mathbf{U}$ contains the eigenvectors of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ and $\mathbf{V}$ the eigenvectors of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$. Let us note here that both matrices $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ are symmetric positive semidefinite so that their eigenvalue decompositions exist and their eigenvalues are positive. Traditionally, the singular values are listed in decreasing order in the matrix Σ.

As we will see later, SVD provides a nice way to approximate a matrix $\mathbf{A}$ while reducing its computing storage requirements. First, choose a truncation index r so that we consider the first r singular values to be important and the others not. Then define $\mathbf{U}_r$ (respectively $\mathbf{V}_r$) as an $n \times r$ (resp. $m \times r$) matrix containing the first r left (resp. right) singular vectors, and Σ_r as an $r \times r$ diagonal matrix containing the r

largest singular values on the diagonal. Then the matrix $\mathbf{A}_r$ provides an approximation of $\mathbf{A}$:

$$\mathbf{A} \simeq \mathbf{A}_r = \mathbf{U}_r \Sigma_r \mathbf{V}_r.$$

This approximation gives the best approximation of $\mathbf{A}$ by a matrix of rank r,

$$||\mathbf{A} - \mathbf{A}_r||_{\mathrm{F}} = \min_{\mathrm{rank}(B) \leq r} ||\mathbf{A} - B||_{\mathrm{F}},$$

where $||\cdot||_{\mathrm{F}}$ is the Frobenius matrix norm. The rate of decrease of the singular values also gives information about the approximation error, $\mathbf{A} - \mathbf{A}_r$, but we will not go into details here. We refer to the reference book by Golub and van Loan [2013] for more details.

We can now define the classical bases used in DA relying on eigen/singular value decomposition. Let us denote by $\mathcal{M}(t_1, t_2)$ a nonlinear evolution model from time t_1 to t_2. If $\mathbf{x}(t_1)$ denotes the state of the system at time t_1, then this state evolves to $\mathbf{x}(t_2)$ at time t_2 according to the formula $\mathbf{x}(t_2) = \mathcal{M}(t_1, t_2)\mathbf{x}(t_1)$.

Let us now denote its tangent operator by $\mathbf{M}(t_1, t_2)$. This operator is a matrix, and if $\mathbf{z}(t_1)$ denotes a perturbation of the system at time t_1, then we have

$$\mathcal{M}(t_1, t_2)(\mathbf{x}(t_1) + \mathbf{z}(t_1)) = \mathcal{M}(t_1, t_2)(\mathbf{x}(t_1)) + \mathbf{M}(t_1, t_2)\mathbf{z}(t_1) + o(\mathbf{z}(t_1)). \tag{5.1}$$

We are interested in vectors for which the amplification of an initial perturbation is maximal. We can define the ratio

$$\rho(\mathbf{z}(t_1)) = \frac{||\mathcal{M}(t_1, t_2)(\mathbf{x}(t_1) + \mathbf{z}(t_1)) - \mathcal{M}(t_1, t_2)(\mathbf{x}(t_1))||}{||\mathbf{z}(t_1)||},$$

as well as its approximation when using the first-order Taylor formula (5.1),

$$\zeta(\mathbf{z}(t_1)) = \frac{||\mathbf{M}(t_1, t_2)\mathbf{z}(t_1)||}{||\mathbf{z}(t_1)||}.$$

Using the adjoint operator of $\mathbf{M}(t_1, t_2)$ and the scalar product $\langle \cdot, \cdot \rangle$ associated with the norm $||\cdot||$, we can rewrite ζ as

$$\zeta(\mathbf{z}(t_1)) = \frac{\sqrt{\langle \mathbf{z}(t_1), \mathbf{M}(t_1, t_2)^{\mathrm{T}}\mathbf{M}(t_1, t_2)\mathbf{z}(t_1) \rangle}}{||\mathbf{z}(t_1)||}. \tag{5.2}$$

An interesting type of vector is the kind that amplifies these ratios the most. Indeed, for numerical weather prediction (NWP), for example, a good and reliable forecast needs to be stable around the "correct" atmosphere trajectory. The most unstable/-variable vectors are the ones that need to be controlled and most precisely determined. From these ratios we obtain the following families of vectors:

Forward/backward singular vectors. We assume that t_1 and t_2 are both finite and the tangent linear hypothesis is valid.[31] We look for the maximizer of ζ, which is denoted by $\mathbf{z}_1^*(t_1)$ and called the first forward singular vector,

$$\zeta(\mathbf{z}_1^*(t_1)) = \max_{\mathbf{z}(t_1)} \zeta(\mathbf{z}(t_1)).$$

[31] This means that arbitrary perturbations to the initial state evolve approximately linearly in time.

By induction we define the following most variable vectors, called the forward singular vectors,

$$\zeta(\mathbf{z}_i^*(t_1)) = \max_{\mathbf{z}(t_1)\perp \mathrm{Span}(\mathbf{z}_1^*(t_1),\dots,\mathbf{z}_{i-1}^*(t_1))} \zeta(\mathbf{z}(t_1)), \quad i \geq 2.$$

The definition (5.2) allows us to say that the forward singular vectors are in fact the singular vectors of the matrix $\mathbf{M}(t_1, t_2)$, in other words, the eigenvectors associated with the largest eigenvalues of the positive operator $\mathbf{M}(t_1, t_2)^{\mathrm{T}}\mathbf{M}(t_1, t_2)$. Similarly, we can wonder which are the vectors at time t_2 that have been most amplified from time t_1, and we get the backward singular vectors, which are the eigenvectors associated with the largest eigenvalues of $\mathbf{M}(t_1, t_2)\mathbf{M}(t_1, t_2)^{\mathrm{T}}$.

Lyapunov vectors. The forward/backward singular vectors are associated with a finite time interval $[t_1, t_2]$. We can wonder what are the asymptotics of the eigenvectors of $\mathbf{M}^{\mathrm{T}}\mathbf{M}$ and $\mathbf{M}\mathbf{M}^{\mathrm{T}}$ when one of the bounds tends to infinity. The forward and backward Lyapunov vectors are defined as the limit of the singular vectors when $t_2 \to \infty$ (forward) or $t_1 \to -\infty$ (backward) [Legras and Vautard, 1996].

Nonlinear singular vectors. If we do not assume the tangent linear hypothesis, we can define the vectors that amplify the ratio ρ the most:

$$\rho(\mathbf{z}_1^*(t_1)) = \max_{\mathbf{z}(t_1)} \rho(\mathbf{z}(t_1)),$$

$$\rho(\mathbf{z}_i^*(t_1)) = \max_{\mathbf{z}(t_1)\perp \mathrm{Span}(\mathbf{z}_1^*(t_1),\dots,\mathbf{z}_{i-1}^*(t_1))} \rho(\mathbf{z}(t_1)), \quad i \geq 2.$$

These vectors are called the nonlinear singular vectors. Contrary to the singular vectors, we have few theoretical results about them [Mu, 2000], and they cannot be computed as eigenvectors. The forward nonlinear singular vectors can be computed fairly easily using constrained minimization algorithms, but the backward vectors are out of reach.

Bred modes, or breeding method. These vectors correspond to the backward Lyapunov vectors, but without the tangent linear hypothesis. They were introduced by Toth and Kalnay [1995] in a purely algorithmic DA framework for weather forecasting, without any mathematical theory.

Another family of vectors is related to the mathematical notion of eigenvectors/singular vectors, called principal components in the statistics community, empirical orthogonal functions (EOFs) in oceanography and DA, and proper orthogonal decomposition (POD) in automatics and engineering.

Principal components, or POD, EOF, or Karhunen–Loeve decomposition. We assume that we have a sample $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m)$ of system states, i.e., m vectors of dimension n. Principal component analysis (PCA) is a statistical method that consists of finding the largest inertial axes of the corresponding point cloud. To do so we consider the orthogonal projection of the sample $\{\mathbf{x}_i\}_{i=1,\dots,m}$ on a vector $\mathbf{u}_1$ and we look for $\mathbf{u}_1$ such that the variance of the projected sample is maximal. Then $\mathbf{u}_1$ is called the first principal component. The other principal components are found similarly and form an orthogonal family. More precisely, let us denote by $\mathbf{u}$ a unitary vector and by $\mathbf{p}_i$ the orthogonal projection of $\mathbf{x}_i$ on the line directed by $\mathbf{u}$. The inertia of the cloud with

respect to $\mathbf{u}$ is given by

$$I_{\mathbf{u}}(\mathbf{x}_1,\dots,\mathbf{x}_m)=\sum_{i=1}^{m}||\mathbf{x}_i-\mathbf{p}_i||^2.$$

This can be seen physically as the residual inertia around $\mathbf{u}$, and this quantity must therefore be minimal. Algebraic manipulations give

$$I_{\mathbf{u}}(\mathbf{x}_1,\dots,\mathbf{x}_m)=m||\overline{\mathbf{x}}-\overline{\mathbf{p}}||^2+\sum_{i=1}^{m}||\mathbf{x}_i-\overline{\mathbf{x}}||^2-\sum_{i=1}^{m}||\mathbf{p}_i-\overline{\mathbf{p}}||^2,$$

with $\overline{\mathbf{x}}=\frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$ and $\overline{\mathbf{p}}=\frac{1}{m}\sum_{i=1}^{m}\mathbf{p}_i$. As $\sum_{i=1}^{m}||\mathbf{x}_i-\overline{\mathbf{x}}||^2$ does not depend on $\mathbf{u}$, it is equivalent to minimizing

$$m||\overline{\mathbf{x}}-\overline{\mathbf{p}}||^2-\sum_{i=1}^{m}||\mathbf{p}_i-\overline{\mathbf{p}}||^2.$$

In other words, $\overline{\mathbf{p}}$ must be equal to $\overline{\mathbf{x}}$ and $\sum_{i=1}^{m}||\mathbf{p}_i-\overline{\mathbf{p}}||^2$ must be maximized. As $\mathbf{p}_i-\overline{\mathbf{p}}$ is the projection of $\mathbf{x}_i-\overline{\mathbf{x}}$, its norm is equal to $|\langle\mathbf{x}_i-\overline{\mathbf{x}},\mathbf{u}\rangle|$, where $\langle\cdot\rangle$ is the inner product. Let us denote by $\mathbf{X}$ the matrix whose ith line contains the vector $\mathbf{x}_i-\overline{\mathbf{x}}$. We thus have to maximize the quantity

$$||\mathbf{X}\mathbf{u}||^2=\mathbf{u}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{u}=\frac{\mathbf{u}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{u}}{\mathbf{u}^{\mathrm{T}}\mathbf{u}},$$

which is maximal when $\mathbf{u}=\mathbf{u}_1$ is the eigenvector of $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ associated with the largest eigenvalue, μ_1. In other words, $\mathbf{u}_1$ is the first right singular vector of $\mathbf{X}$. Similarly, the other principal components are the eigenvectors associated with the next largest eigenvalues. The larger the eigenvalue, the larger the part of the system variability represented by the associated principal component. Also, the variability explained by the first r principal components is given by

$$\frac{\sum_{i=1}^{r}\mu_i}{\sum_{i=1}^{m}\mu_i}. \tag{5.3}$$

In geosciences these are called EOFs, and the interested reader can consult, e.g., Hannachi et al. [2007] for a extensive review about EOFs and their variants, as well as their use in atmospheric sciences.

As we will see later, these bases are useful in DA, both for variational and filtering methods.

5.1.2.2 ▪ Multiscale analysis

Another class of families comes from multiscale analysis: Fourier series, spectral decomposition, wavelets, etc. The idea is to change variables into the space of the multiscale decomposition and to truncate the decomposition to reduce the working space dimension. In other words, instead of representing a state vector in the state space, we consider its multiscale series decomposition and we assume that a few modes are enough to get an accurate approximation,

$$\mathbf{x}\in X=\sum_{i=1}^{\infty}x_i\mathbf{e}_i\simeq\sum_{i=1}^{r}x_i\mathbf{e}_i\in\tilde{X},$$

where $\mathbf{e}_i$ are the multiscale basis vectors, such as Fourier modes, spectral modes, wavelets, etc.; x_i are the coefficients of $\mathbf{x}$ in this basis; and r is the reduced dimension. Fast algorithms, e.g., fast Fourier transform (FFT) or, similarly fast wavelet transform, are available to compute the coefficients x_i and to recompose the signal from the coefficient sequence.

Once again, these families are useful in DA, both in filtering and variational. As more details are not required for the remainder of this chapter, we simply refer the reader to classical works on this subject, e.g., by Mallat [1989, 2008], and to the vast online tutorial material at, e.g., `http://www.wavelet.org`.

5.1.2.3 ▪ Polynomials

Polynomial families, such as Taylor, Lagrange, and Hermite, have not been used in DA yet, but were used for a similar problem: the optimal control of PDEs, namely the Navier–Stokes equation, by Ito and Ravindran [1998]. In this paper the authors project the solution into a subspace spanned by various polynomial families and perform the optimization in the reduced space, thus gaining a lot of computing time without losing too much in terms of accuracy. This idea is similar to the reduced 4D-Var approach using singular vectors, EOFs, or bred modes, which will be detailed later on.

5.1.3 ▪ Reduced methods in DA

Different approaches to using reduction methods in DA can be mentioned: the first class of approaches reduces the model or the covariance matrices, but not the DA scheme per se, while the second class of approaches reduces the DA method, i.e., reduces the working dimension of the algorithm cores.

Reduction of the model or the covariance matrices. We can mention two approaches in this class. In the first approach we reduce the model, not the DA method. In this case the DA method remains identical, but the model computing time is much smaller, thus allowing time saving. However, the reduced dimension of the model might make it cruder and more prone to biases or errors.

In the second approach we use matrix rank reduction, for error covariance matrix modeling in both variational and filtering methods. In this case the full model is retained and the algorithm does not change dimension. In other words, if n is the dimension of the state vector, the minimization (for 4D-Var) and the inversion required in the Kalman gain matrix computation (for Kalman filter [KF]) are both performed in dimension n. What changes is essentially the rank of the covariance matrix ($\mathbf{B}$ or $\mathbf{P}^{\mathrm{f}}$), which is smaller.

Reduction of the DA method. In the filtering framework, reduction means covariance matrix rank reduction for the KF and rewriting of the algorithm so that the matrix inversion required to compute the gain, $\mathbf{K}$, is done in a reduced dimension.

In the variational framework, we reduce the control space/minimization space dimension to speed up the 4D-Var DA algorithm. In classical 4D-Var, the minimization space is equal to the parameter space, usually the model state space for the initial condition plus additional dimensions for the other poorly known parameters—boundary conditions or physical/numerical parameters. With reduction, the minimization space is a subspace of the full parameter space: the optimization problem is projected into a reduced subspace.

All of these approaches will be detailed in the following sections of this chapter. They will also, to a large extent, be the focus of the subsequent chapters.

5.2 ▪ Model reduction

The very first approach using model reduction techniques for DA consists of simply reducing the model. In other words, all the machinery of model reduction methods, algorithms, and libraries can be used as is. From the DA method point of view, nothing changes: regular algorithms are used with a different (reduced) model. Therefore, we will not go into too many details here, as model reduction per se goes well beyond the scope of this book, and we will briefly present examples of the various ways to use reduced models within DA schemes.

5.2.1 ▪ Reduction of the full model

5.2.1.1 ▪ Simple models

Model simplification through reduction has been used widely in the past, when DA methods were an emerging hot topic, and while computing power was not large. For example, Miller and Cane [1989] use a simple model of sea-level anomalies in the Pacific tropical ocean, essentially a superposition of linear wave equations, with a reduced number of waves, to demonstrate the use of a KF with a parametric modeling of the model error. At that time, the aim was to demonstrate the capability of the assimilation method to adequately make use of the observations, so that linear assumptions, allowing the use of the superposition principle and simplified models, sometimes oversimplified, were commonplace.

5.2.1.2 ▪ Proper orthogonal decomposition

Later on, the computing power increased as well as the complexity of the state-of-the-art models, and the efficiency of DA methods was well established. The focus was then on reducing the cost of the model to be able to handle either large-dimensional problems, where the size of the control vector is 10^7–10^9, or real-time DA applications such as, e.g., flow control applications. For example, in Cao et al. [2006b] POD was developed to build a reduced model of the Pacific tropical ocean, and then 4D-Var assimilation was performed with this reduced model and its adjoint. On the Kalman side, Farrell and Ioannou [2001] built a reduced model with balanced truncation (variant of the POD method using POD modes and a second kind of optimal vectors) and used it within a KF algorithm. On the real-time side, we can cite Artana et al. [2012], in which the authors combined POD and 4D-Var assimilation for fluid flow control applications. As in Cao et al. [2006a], they designed reduced direct and adjoint models to be used within the 4D-Var minimization algorithm.

5.2.1.3 ▪ Wavelets

Wavelets also provide a convenient way to reduce the dimension of the model. The idea is to retain only a small fraction of coefficients in the wavelet decomposition through a thresholding procedure that retains only the largest ones. One of the advantages of wavelets over the other reduction techniques is that they are local; in other words, the support of each individual wavelet is local in space. This is not the case for SVD, POD/EOF, Fourier, or polynomial bases, for which every basis member has a global

support. Wavelets can therefore be very good for local refinements or capturing fine local scales, while being cheap and sparse. This is why Tangborn and Zhang [2000] chose to build a reduced model using wavelets. They then proceeded to use the KF with the reduced model, similarly to what has been done for POD.

5.2.2 ▪ Reduction focused on covariance matrices

One of the major drawbacks of the DA algorithms in large dimensions is the prohibitive cost of the covariance matrices: they are too large to be fully estimated or even stored in memory, and too large to be used directly for numerical computations. Most of the solutions to the curse of dimensionality for these matrices rely on this simple principle: find a reduced and thus cheaper way to compute and represent or evolve the covariance matrices. In this subsection we will focus on methods dealing with covariance matrices only, which do not change the core dimension of the DA algorithm, i.e., the dimension of the space in which the inversion (KF)/optimization (Var) takes place.

5.2.2.1 ▪ B matrix modeling by factorization and spectral methods

The question of the initialization of the error covariance matrix ($\mathbf{B}$ or $\mathbf{P}_0^f$) is crucial for every DA method, sequential as well as variational. As the $\mathbf{B}$ matrix is generally huge in geosciences applications (up to $10^9 \times 10^9$), it is impossible to store it in memory as is. In other words, every modeling attempt aims at reducing its dimension. As the square root of the $\mathbf{B}$ matrix is often required in variational assimilation, factorizations of $\mathbf{B}^{1/2}$ using sparse matrices have been proposed, relying also on the symmetry property,

$$\mathbf{B} = \mathbf{B}^{1/2}\mathbf{B}^{T/2}.$$

Because every operational center for geosciences forecasting has its own factorization, the literature on this subject is extremely abundant, but we can cite Bannister [2008] for a review. Such factorizations include the spectral methods, which aim at reducing the covariances using independent spectral modes, e.g., Fourier on the sphere for atmospheric sciences. The idea is to decompose $\mathbf{B}^{1/2}$ into a product of sparse operators, e.g.,

$$\mathbf{B}^{1/2} = \mathbf{L}\Sigma\mathbf{C}^{1/2},$$

where $\mathbf{L}$ is a physical balance operator; Σ a spectral transform, e.g., Fourier series, spherical harmonics, etc.; and $\mathbf{C}$ a diagonal or block diagonal matrix containing the correlations in spectral space. As the spectral modes are global, there are also variants of this method using wavelets instead of Fourier modes to provide local inhomogeneity, as presented, e.g., in the review by Fisher [2003]. Let us also mention the diffusion equation method by Weaver and Courtier [2001], which, while not being a reduced method per se, models the $\mathbf{B}$ matrix thanks to a simple equation resolution.

As every operational center uses its own particular modeling for the $\mathbf{B}$ matrix, more details will not be given here, but the interested reader can refer to Fisher [2003], Fischer et al. [2005], Brousseau et al. [2011], and Rawlins et al. [2007], and references therein.

5.2.2.2 ▪ Covariance propagation using SVD

One of the most expensive steps of the (extended) KF is the error covariance matrix propagation (3.19),

$$\mathbf{P}^{\mathrm{f}}_{k+1} = \mathbf{M}_{k+1}\mathbf{P}^{\mathrm{a}}_{k}\mathbf{M}^{\mathrm{T}}_{k+1} + \mathbf{Q}_k, \tag{5.4}$$

where $\mathbf{M}_{k+1}$ is the linear model step from time k to time $k+1$, or its linearized version for the extended Kalman filter (EKF). Model reduction has been applied by Cohn and Todling [1996] to tackle this issue and render it computationally affordable. The idea was to perform an SVD of the matrix $\mathbf{M}_{k+1}$, to truncate the SVD to keep a small number, r, of significant singular values, while retaining most of the model dynamics. This allows us to simplify the matrix products of equation (5.4) and considerably reduce the operations count, and therefore the computational cost of the algorithm. However, this has to be done at every step of the model, thus adding a nonnegligible burden to the algorithm cost. This is why this idea is not sufficient in itself and has to be extended to the reduction of the full filter algorithm. We will see, e.g., the SEEK filter in Section 5.3.1, which provides an efficient way of doing so.

5.2.2.3 ▪ Ensemble/EOF estimations of the B matrix for variational assimilation

As we have said before, the question of the initialization of the error covariance matrix is crucial for every DA method. This is especially true for three-dimensional variational assimilation, where the background error covariance matrix $\mathbf{B}$ is not propagated in time and must therefore be correctly specified from the start. Indeed, the $\mathbf{B}$ matrix has multiple roles in the 3D-Var framework:

- information spreading: the $\mathbf{B}$ matrix spreads information from an observed geographical point to an unobserved one;
- information smoothing in space;
- balance properties: the $\mathbf{B}$ matrix spreads information from an observed variable to an unobserved one and helps preserve physical properties (if correctly specified, e.g., in the ocean geostrophy or temperature/salinity equilibrium);
- preconditioning of the assimilation: as the cost function is generally ill conditioned, a change of variable involving $\mathbf{B}^{1/2}$ usually helps reduce the condition number of the problem.

This paragraph deals with estimations of the $\mathbf{B}$ matrix using only a small number of well-chosen vectors, EOFs, or ensemble members.

EOFs have been used to reduce the rank of the $\mathbf{B}$ matrix and provide a low-cost approximation in a variational context [Robert et al., 2005]. Using the notation of Section 5.1.2.1, the authors computed the EOFs of their ocean model using a sequence $(\mathbf{x}_i)_i$ of model outputs computed with various perturbations of the input parameters. They assume that the covariance matrix of the centered samples is a good approximation of $\mathbf{B}$:

$$\mathbf{B} \simeq \frac{1}{m}\mathbf{X}\mathbf{X}^{\mathrm{T}}.$$

Then, keeping only the r leading EOFs, they provide a cheap way to approximate $\frac{1}{m}\mathbf{X}^{\mathrm{T}}\mathbf{X}$ and $\mathbf{B}$ using the truncated SVD of $\frac{1}{\sqrt{m}}\mathbf{X}$,

$$\mathbf{B} \simeq \frac{1}{m}\mathbf{X}\mathbf{X}^{\mathrm{T}} \simeq \mathbf{U}_r\boldsymbol{\Sigma}^2_r\mathbf{U}^{\mathrm{T}}.$$

Recently, an ensemble method has been applied in oceanography to estimate the **B** matrix in a variational framework. Daget et al. [2009] create an ensemble by applying perturbations to the forcing parameters, here surface fluxes such as wind stress, and use this ensemble to compute, using a Monte Carlo estimator, an estimation of the background error covariance matrix **B**. With a very small number of members, they still get encouraging results.

This method is also used at the European Center for Medium-range Weather Forecasts (ECMWF) to diagnose the background error covariances (see Fisher [2003] for a review). Contrary to the previous study, they do not produce a **B** matrix to be used in the DA algorithm, but they do provide a diagnosis. They also build an ensemble to produce an estimation of the **B** matrix, using forecasts produced by older and perturbed analyses.

5.3 ▪ Filtering algorithm reduction

This section deals with methods aiming to reduce the core dimension of the KF, without necessarily reducing the whole direct model. The dimensions involved are:

- n, the dimension of the system space (size of the model state vector);
- p, the number of observations (size of the observation vector); and
- r, a reduced dimension, usually such that $r \ll n$ and $r \ll p$.

The core dimension of the KF can be found in the Kalman gain matrix equation (3.20),

$$\mathbf{K} = \mathbf{P}^{\mathrm{f}}\mathbf{H}^{\mathrm{T}}(\mathbf{H}\mathbf{P}^{\mathrm{f}}\mathbf{H}^{\mathrm{T}} + \mathbf{R})^{-1}.$$

First attempts at reducing this equation were done by Cohn and Todling [1996]. In the framework of square root filters (see below and Chapter 6), they proposed to use a truncated eigenvalue decomposition of $\mathbf{M}\mathbf{P}^{\mathrm{a}}\mathbf{M}^{\mathrm{T}}$ to speed up the computation of **K**. However, there was no propagation of the eigenvectors in their filter, so the eigenvalue decomposition had to be performed at each time step, implying a noticeable computational burden.

A similar idea was proposed later on by Verlaan and Heemink [1997]. The forecast error covariance matrix $\mathbf{P}^{\mathrm{f}}$ was replaced by its truncated SVD. As the SVD was a costly step of the algorithm, it was performed only once, in the framework of a steady-state filter.

Either way, these approaches were a first step in the direction of filter reduction but not completely satisfactory because of the computational cost of the added SVD/EVD. The SEEK filter, in the next subsection, overcomes this difficulty by avoiding these computations and always providing a low-rank approximation of the covariance matrices.

5.3.1 ▪ The Singular Evolutive Extended Kalman Filter

The singular evolutive extended Kalman (SEEK) filter algorithm, which was first introduced by Pham et al. [1998], is a square-root type KF (see Chapter 6 for more details about these filters). Recent reviews about the SEEK filter can also be found in Brasseur and Verron [2006] and Rozier et al. [2007]. The idea is to replace the analysis error covariance matrix $\mathbf{P}^{\mathrm{a}}$ by a lower-rank SVD-induced approximation. The propagation

of the analysis covariance error of the KF is governed by (3.19)

$$\mathbf{P}^{\mathrm{f}}_{k+1} = \mathbf{M}_{k+1}\mathbf{P}^{\mathrm{a}}_{k}\mathbf{M}^{\mathrm{T}}_{k+1} + \mathbf{Q}_k, \tag{5.5}$$

but, as said before, the size of the involved matrices is prohibitive. The first idea of the SEEK filter is then to replace this costly step by a low-cost one.

The initialization of the filter is given by

$$\mathbf{x}^{\mathrm{f}}_0 = \mathbf{x}^{\mathrm{b}}, \quad \mathbf{P}^{\mathrm{f}}_0 = \mathbf{B} = \mathbf{S}^{\mathrm{f}}_0(\mathbf{S}^{\mathrm{f}}_0)^{\mathrm{T}},$$

where the matrix $\mathbf{S}^{\mathrm{f}}_0$ (size $n \times r$) contains the first r EOFs of the system. We also refer to page 144 for more details about the SEEK filter initialization.

We will now assume by induction that the matrix $\mathbf{P}^{\mathrm{a}}_{k+1}$ can be approximated by a low-rank square root factorization,

$$\mathbf{P}^{\mathrm{a}}_{k+1} \simeq \mathbf{S}^{\mathrm{a}}_{k+1}(\mathbf{S}^{\mathrm{a}}_{k+1})^{\mathrm{T}},$$

where $\mathbf{S}^{\mathrm{a}}_{k+1}$ (size $n \times r$) contains the r analyzed EOFs at time i. The propagation of the analysis of covariance step of the KF (5.5) is replaced by

$$\mathbf{P}^{\mathrm{f}}_{k+1} = \mathbf{M}_{k+1}\mathbf{S}^{\mathrm{a}}_{k}(\mathbf{S}^{\mathrm{a}}_{k})^{\mathrm{T}}\mathbf{M}^{\mathrm{T}}_{k+1} + \mathbf{Q}_k$$

or, in other words,

$$\mathbf{P}^{\mathrm{f}}_{k+1} = \tilde{\mathbf{S}}^{\mathrm{f}}_{k+1}(\tilde{\mathbf{S}}^{\mathrm{f}}_{k+1})^{\mathrm{T}} + \mathbf{Q}_k, \text{ with } \tilde{\mathbf{S}}^{\mathrm{f}}_{k+1} = \mathbf{M}_{k+1}\mathbf{S}^{\mathrm{a}}_{k}. \tag{5.6}$$

Let us remark here that the previous equations assumed the model $\mathbf{M}$ was linear, but the filter can also be written when it is not linear; see Verron et al. [1999]. In this case, $\tilde{\mathbf{S}}^{\mathrm{f}}_{k+1}$ is computed column by column, as follows. First, we propagate the analysis vector to get the forecast

$$\mathbf{x}^{\mathrm{f}}_{k+1} = \mathscr{M}_{k+1:k}(\mathbf{x}^{\mathrm{a}}_{k}).$$

Then we propagate the r columns of $\tilde{\mathbf{S}}^{\mathrm{f}}_{k+1}$ using the nonlinear model $\mathscr{M}_{k+1:k}$,

$$(\tilde{\mathbf{S}}^{\mathrm{f}}_{k+1})_j = \mathscr{M}_{k+1:k}(\mathbf{x}^{\mathrm{a}}_{k} + (\mathbf{S}^{\mathrm{a}}_{k})_j) - \mathscr{M}_{k+1:k}(\mathbf{x}^{\mathrm{a}}_{k}), \quad j = 1, \ldots, r,$$

where $(\tilde{\mathbf{S}}^{\mathrm{f}}_{k+1})_j$ is the jth column vector of the reduced matrix $\tilde{\mathbf{S}}^{\mathrm{f}}_{k+1}$. Of course, if the model is linear or if the tangent linear hypothesis is valid, then this equation is equivalent to (5.6). This can be seen as an evolution equation, namely the forecast step, for the jth EOF. In this respect, the SEEK filter is very similar to the ensemble Kalman filter (EnKF; see Chapter 6) in the sense that the computation of the forecast error covariance matrix is replaced by the propagation of a reduced number of state vectors: EOFs here, ensemble members for the EnKF.

To avoid multiple SVD computations, because they take place at each filter step, the SEEK filter requires $\mathbf{P}^{\mathrm{f}}$ and $\mathbf{P}^{\mathrm{a}}$ to keep the same rank, r, so this calls for some adjustment, essentially hypotheses about the $\mathbf{Q}$ matrix, so that $\mathbf{P}^{\mathrm{f}}$ and $\tilde{\mathbf{S}}^{\mathrm{f}}$ still have the same rank. More details about this point can be found in Brasseur and Verron [2006]. We assume that $\mathbf{P}^{\mathrm{f}}$ can therefore be written as

$$\mathbf{P}^{\mathrm{f}}_{k+1} = \mathbf{S}^{\mathrm{f}}_{k+1}(\mathbf{S}^{\mathrm{f}}_{k+1})^{\mathrm{T}},$$

where $\mathbf{S}^{\mathrm{f}}$ and $\tilde{\mathbf{S}}^{\mathrm{f}}$ have the same reduced rank, r.

Algorithm 5.1 SEEK filter equations.

Analysis step:

1. $\mathbf{K}_k = \mathbf{S}_k^{\mathrm{f}} \left[\mathbf{I}_r + (\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})\right]^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}$,
2. $\mathbf{x}_k^{\mathrm{a}} = \mathbf{x}_k^{\mathrm{f}} + \mathbf{S}_k^{\mathrm{f}} \left[\mathbf{I}_r + (\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})\right]^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{y}_k^{\mathrm{o}} - \mathbf{H}_k\mathbf{x}_k^{\mathrm{f}})$,
3. $\mathbf{P}^{\mathrm{a}} = \mathbf{S}_k^{\mathrm{a}}(\mathbf{S}_k^{\mathrm{a}})^{\mathrm{T}}$, with $\mathbf{S}_k^{\mathrm{a}} = \mathbf{S}_k^{\mathrm{f}} \left[\mathbf{I}_r + (\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})\right]^{-1/2}$.

Forecast step:

4. $\mathbf{x}_{k+1}^{\mathrm{f}} = \mathcal{M}_{k+1:k}\mathbf{x}_k^{\mathrm{a}}$,
5. $(\tilde{\mathbf{S}}_{k+1}^{\mathrm{f}})_j = \mathcal{M}_{k+1:k}(\mathbf{x}_k^{\mathrm{a}} + (\mathbf{S}_k^{\mathrm{a}})_j) - \mathcal{M}_{k+1:k}(\mathbf{x}_k^{\mathrm{a}}), \quad j = 1, \ldots, r$,
6. $\mathbf{P}_{k+1}^{\mathrm{f}} = \tilde{\mathbf{S}}_{k+1}^{\mathrm{f}}(\tilde{\mathbf{S}}_{k+1}^{\mathrm{f}})^{\mathrm{T}} + \mathbf{Q}_k$.

Similarly, the analysis step of the KF can be then replaced by a low-cost one, using reduced-rank matrices

$$\mathbf{K}_k = \mathbf{S}_k^{\mathrm{f}} \left[\mathbf{I}_r + (\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})\right]^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}.$$

Let us remark here that, as $\mathbf{H}$ has p rows and n columns, $\mathbf{S}^{\mathrm{f}}$ has n rows and r columns, and $\mathbf{R}$ has p rows and p columns, the matrix to be inverted has r rows and r columns. This is to be compared to the full Kalman gain formula, where the inversion takes place in the observation space (size $p \times p$). The update of the analysis vector is then

$$\mathbf{x}_k^{\mathrm{a}} = \mathbf{x}_k^{\mathrm{f}} + \mathbf{S}_k^{\mathrm{f}} \left[\mathbf{I}_r + (\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})\right]^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{y}_k^{\mathrm{o}} - \mathbf{H}_k\mathbf{x}_k^{\mathrm{f}}).$$

We can see that the analysis increment, $\mathbf{x}_k^{\mathrm{a}} - \mathbf{x}_k^{\mathrm{f}}$, is a linear combination of the columns of $\mathbf{S}_k^{\mathrm{f}}$. In other words, it is a linear combination of the EOFs at time k. Finally, the updated analysis covariance matrix becomes

$$\mathbf{P}^{\mathrm{a}} = \mathbf{S}_k^{\mathrm{a}}(\mathbf{S}_k^{\mathrm{a}})^{\mathrm{T}}, \text{ with } \mathbf{S}_k^{\mathrm{a}} = \mathbf{S}_k^{\mathrm{f}} \left[\mathbf{I}_r + (\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{H}_k\mathbf{S}_k^{\mathrm{f}})\right]^{-1/2},$$

where the inverse of the square root also takes place in dimension r. This equation can be seen as the analysis step for the EOFs' update.

To recapitulate, Algorithm 5.1 summarizes the SEEK filter steps.

SEEK filter initialization. A few words should be said here about the initialization of the filter, which requires a background and a background error covariance matrix,

$$\mathbf{x}_0^{\mathrm{f}} = \mathbf{x}^{\mathrm{b}}, \quad \mathbf{P}_0^{\mathrm{f}} = \mathbf{B}.$$

The choice of $\mathbf{x}^{\mathrm{b}}$ is a classical problem, not specific to the SEEK filter. The $\mathbf{B}$ matrix, however, can also be chosen through reduced-order modeling. Indeed, as we have seen before, the first r EOFs span a subspace for which the projected variance of the sample

is maximal, that is to say the error is minimal. Therefore, the first r EOFs provide an optimal low-rank approximation of the covariance matrix of a given sample of system state vectors. The initialization process of the SEEK filter is implemented as follows:

1. Design an experiment to compute a sample of N system states, using the direct model $\mathcal{M}$, $(\mathbf{x}_j)_{j=1..N}$. Form the matrix $\mathbf{X}$ that has $\mathbf{x}_j - \frac{1}{N}\sum_{l=1}^{N}\mathbf{x}_l$ as columns. This step should be done carefully, as the variability contained in the $\mathbf{x}_j$'s determines the variability of the EOFs and the $\mathbf{B}$ matrix. The sensitive parameters of the model, as well as a "correct" number of model runs, N, should be sampled and tuned carefully.

2. Perform the SVD of matrix $\mathbf{X}$.

3. Choose an adequate number, r, for the reduced rank, i.e., the total number of EOFs. This step is both crucial and complex, as r will be a compromise between the final accuracy of the filter and its computing requirements. The original work by Pham et al. [1998] states that the singular value associated with the $(r+1)$th EOF should be smaller than one, as well as those of all the EOFs not chosen, to assure the filter's stability. However, this criterion is difficult to measure in practice. Instead, the behavior of the ratio (5.3) giving the percentage of variability explained by the first r EOFs, as a function of r, can be a good indicator to find a suitable r. Usually, in large-scale oceanography, satisfactory results are obtained with r equal to a few dozens to a few hundreds.

4. Form the matrix $\mathbf{S}_0^{\mathrm{f}}$ (size $n \times r$) containing the first r EOFs in columns.

5. Finally, set
$$\mathbf{x}_0^{\mathrm{f}} = \mathbf{x}^{\mathrm{b}}, \quad \mathbf{P}_0^{\mathrm{f}} = \mathbf{B} = \mathbf{S}_0^{\mathrm{f}}(\mathbf{S}_0^{\mathrm{f}})^{\mathrm{T}}.$$

This filter has known some evolutions and generalizations in the past, e.g., the interpolated version [Pham et al., 1998] to replace linearization by interpolation and the associated smoother [Cosme et al., 2010]. We refer to the chapter about ensemble filters, Chapter 6, for more details about reduced ensemble smoothers.

The SEEK filter is an efficient example of reduced-rank square root filters (RRSQRT) [Verlaan and Heemink, 1997], which make the most of the square root factorization of the covariance matrices and reformulate the KF steps using mostly the square roots instead of the full matrices. The Singular Evolutive Interpolated Kalman (SEIK) filter [Pham, 2001] is an interpolated variant of the SEEK filter that can also be interpreted as an ensemble filter with a reduced ensemble. This general class of filters will be detailed in Chapter 6.

5.3.2 ▪ Wavelet rank reduction for Kalman filtering

As we have seen in Section 5.2.1.3, wavelets have been used to produce reduced models. Tangborn and Zhang [2000] performed wavelet model reduction for the forecast propagation but changed variables back into physical space for the analysis, so that the Kalman gain was still very costly to compute. Tangborn [2004] and Auger and Tangborn [2004] upgraded the algorithm to do all computations, analysis included, in the possibly truncated wavelet space, thus reducing the computational cost of the KF.

We can also cite Chin and Mariano [1999] and Chui and Chen [1999], who propose a reduction of the error space by projection onto wavelet space. This is quite

similar to the SEEK, which projects onto a space spanned by EOFs instead. As before, the authors argue that using wavelets allows local inhomogeneity, contrary to global functions such as EOFs or Fourier modes.

5.3.3 ▪ The EnKF

The ensemble Kalman filter (EnKF), which will be described in more detail in Chapter 6, is also a reduced method in the sense that it requires the propagation and analysis of a small number of ensemble members, instead of full-rank covariance matrices. The idea behind the EnKF is simply a Monte Carlo estimator of the covariance matrix, which we recall here briefly.

Given a sample of independent observations, $\epsilon_1, \ldots, \epsilon_p$, of the n-dimensional vector $\boldsymbol{\varepsilon}$ corresponding, e.g., to the background error, then we can compute with $\epsilon_1, \ldots, \epsilon_p$ an unbiased estimation of the covariance matrix of $\boldsymbol{\varepsilon}$,

$$\mathrm{Cov}(\boldsymbol{\varepsilon}) = \mathrm{E}\left[(\boldsymbol{\varepsilon} - \mathrm{E}(\boldsymbol{\varepsilon}))(\boldsymbol{\varepsilon} - \mathrm{E}(\boldsymbol{\varepsilon}))^{\mathrm{T}}\right] \simeq \frac{1}{p-1}\sum_{k=1}^{p}(\epsilon_k - \overline{\epsilon})(\epsilon_k - \overline{\epsilon})^{\mathrm{T}},$$

where $\overline{\epsilon}$ is the mean of the sample $\{\epsilon_k\}_{k=1,\ldots,p}$, defined by

$$\overline{\epsilon} = \frac{1}{p}\sum_{k=1}^{p}\epsilon_k.$$

As for the SEEK filter, but with ensemble members instead of EOFs, the idea is then to change the filter equations so that the covariance matrices do not have to be explicitly computed, but so that the forecast and analysis can be done solely on the sample.

5.4 ▪ Reduced methods for variational assimilation

This section deals with reduced methods within variational assimilation, mostly four-dimensional. Contrary to the simple use of reduced models, in this section we will see methods for which at least two direct models coexist: the "full" model and a reduced version with which the optimization is performed.

5.4.1 ▪ Multigrid and (multi)incremental approaches

5.4.1.1 ▪ Incremental 4D-Var

Incremental 4D-Var is a variation of 4D-Var that takes into account small nonlinearities while still using quadratic optimization. The hypothesis "small nonlinearities" can be stated as

$$\mathcal{M}_{k+1:k}\ldots\mathcal{M}_{1:0}(\mathbf{x}) - \mathcal{M}_{k+1:k}\ldots\mathcal{M}_{1:0}(\mathbf{x}^{\mathrm{b}}) \simeq \mathbf{M}_{k+1}\ldots\mathbf{M}_1(\mathbf{x}-\mathbf{x}^{\mathrm{b}}),$$

where $\mathcal{M}_{k+1:k}$ represents the full nonlinear model time step and $\mathbf{M}_{k+1}$ is a linear approximation of $\mathcal{M}_{k+1:k}$. Similarly, one asks for

$$\begin{aligned}&\mathcal{H}_{k+1}\mathcal{M}_{k+1:k}\ldots\mathcal{M}_{1:0}(\mathbf{x}) - \mathcal{H}_{k+1}\mathcal{M}_{k+1:k}\ldots\mathcal{M}_{1:0}(\mathbf{x}^{\mathrm{b}})\\ \simeq\ &\mathbf{H}_{k+1}\left(\mathcal{M}_{k+1:k}\ldots\mathcal{M}_{1:0}(\mathbf{x}) - \mathcal{M}_{k+1:k}\ldots\mathcal{M}_{1:0}(\mathbf{x}^{\mathrm{b}})\right),\end{aligned}$$

where $\mathbf{H}_{k+1}$ is required to be a good linear approximation of the possibly nonlinear observation operator $\mathscr{H}_{k+1}$. Then we define the *increment* $\boldsymbol{\delta}\mathbf{x}$ as

$$\boldsymbol{\delta}\mathbf{x} = \mathbf{x} - \mathbf{x}^{\mathrm{b}},$$

and the incremental 4D-Var cost function as a function of $\boldsymbol{\delta}\mathbf{x}$ is

$$\tilde{J}(\boldsymbol{\delta}\mathbf{x}) = \tilde{J}^{\mathrm{b}}(\boldsymbol{\delta}\mathbf{x}) + \tilde{J}^{\mathrm{o}}(\boldsymbol{\delta}\mathbf{x}),$$

$$\tilde{J}^{\mathrm{b}}(\boldsymbol{\delta}\mathbf{x}) = \boldsymbol{\delta}\mathbf{x}^{\mathrm{T}}\mathbf{B}^{-1}\boldsymbol{\delta}\mathbf{x},$$

$$\tilde{J}^{\mathrm{o}}(\boldsymbol{\delta}\mathbf{x}) = \sum_{k=1}^{K}(\mathbf{d}_k - \mathbf{H}_k\mathbf{M}_k \ldots \mathbf{M}_1\boldsymbol{\delta}\mathbf{x})^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{d}_k - \mathbf{H}_k\mathbf{M}_k \ldots \mathbf{M}_1\boldsymbol{\delta}\mathbf{x}),$$

where $\mathbf{d}_k$ is the *innovation vector*

$$\mathbf{d}_k = \mathbf{y}_k^{\mathrm{o}} - \mathscr{H}_k\mathscr{M}_k \ldots \mathscr{M}_1(\mathbf{x}^{\mathrm{b}}).$$

Thus, the incremental cost function, $\tilde{J}(\boldsymbol{\delta}\mathbf{x})$, is an approximation of the full 4D-Var cost function, and it is quadratic, thus easier to minimize. The incremental 4D-Var algorithm allows us to take into account small nonlinearities as it periodically updates the linearized operators $\mathbf{M}_k$ and $\mathbf{H}_k$. Algorithm 5.2 sums up the procedure, and Figure 5.2 provides a synthetic sketch.

Incremental 4D-Var is a widely used method in variational assimilation for two main reasons. First, it requires the minimization of only quadratic functionals that are not nonconvex. Second, it allows the use of a simplified model in the inner loop.

The second point is relevant to reduced methods, because it is often a reduced, cheaper, version of the model that is used in the inner loop, as illustrated by Figure 5.2. The variety of such applications is huge, as numerous teams did and still do incremental 3D-/4D-Var. The cheaper, reduced, version of the tangent linear model (TLM) can be obtained in various ways. First, it can be done by simplification of the physics, in particular of the nondifferentiable processes. Second, it can be achieved using a coarser resolution, i.e., fewer physical points on the grid. Finally, the tangent model can be approximated by an actual reduced model, e.g., using truncated spherical harmonics or POD/EOFs.

Regarding this last item, we can mention Lawless et al. [2008], who apply the balanced truncation method, which is a variation of the POD method, within an incremental 4D-Var. They compare the balanced truncation with an equivalently costly reduced model consisting of halving the grid-point spacing and show the superiority of the former.

5.4.1.2 ▪ Multi-incremental 4D-Var

The incremental 4D-Var usually carries two models: the full nonlinear model for the outer loop and a reduced linear version for the inner loop. The multi-incremental version of the algorithm is a refinement consisting of using various reduced models for the various inner loops [Veersé and Thepaut, 1998]. This method is used operationally in Europe (Météo-France, ECMWF, UK MetOffice) with various ways of reducing the model, e.g., varying resolution, truncated harmonics or Fourier series, simplified physics, etc. During the first outer loop iteration, the inner loop is performed with a coarse, simplified, cheap model, so that an approximate minimum is reached quickly. Then, during the second outer iteration, a more accurate but still reduced model is

Algorithm 5.2 Incremental 4D-Var.

Initialize: $\mathbf{x}^{\mathrm{r}} = \mathbf{x}^{\mathrm{g}}$
($\mathbf{x}^{\mathrm{r}}$ is called the *reference state*; $\mathbf{x}^{\mathrm{g}}$ is the first guess.)

OUTER LOOP BEGINS

1. Integrate the full nonlinear model: $\mathbf{x}_k^{\mathrm{r}} = \mathcal{M}_{k:0}[\mathbf{x}^{\mathrm{r}}]$.
2. Compute the innovation vector $\mathbf{d}_k$ thanks to the nonlinear observation operator.

   INNER LOOP BEGINS

   (a) Compute the incremental cost function, $\tilde{J}(\boldsymbol{\delta}\mathbf{x})$, using the linear approximations $\mathbf{M}$ and $\mathbf{H}$, approximations performed around the reference trajectory $\mathbf{x}^{\mathrm{r}}$.

   (b) Compute the gradient, $\nabla\tilde{J}(\boldsymbol{\delta}\mathbf{x})$, thanks to the adjoint operators $\mathbf{M}^{\mathrm{T}}$ and $\mathbf{H}^{\mathrm{T}}$.

   (c) Minimize $\tilde{J}$, e.g., with a gradient descent method, such as conjugate gradient, quasi-Newton, etc.

   INNER LOOP ENDS

3. Update the analysis increment $\boldsymbol{\delta}\mathbf{x}^{\mathrm{a}} = \boldsymbol{\delta}\mathbf{x}$.
4. Update the reference trajectory $\mathbf{x}^{\mathrm{r}} = \mathbf{x}^{\mathrm{r}} + \boldsymbol{\delta}\mathbf{x}^{\mathrm{a}}$.

OUTER LOOP ENDS

Compute the analysis: $\mathbf{x}^{\mathrm{a}} = \mathbf{x}^{\mathrm{r}}$, $\mathbf{x}_k^{\mathrm{a}} = \mathcal{M}_{k:0}[\mathbf{x}^{\mathrm{a}}]$.

used, and so on. This approach allows the use of very coarse approximations for the first iterations, thus reducing the computational cost of the overall algorithm.

However, contrary to incremental 4D-Var, there is no convergence theorem, and divergence is well known after a few outer loops [Tremolet, 2007a]. In a simplified framework, it was even shown by Neveu [2011] that this approach first converges quickly to the solution but then quickly diverges, and that the optimal number of iterations was difficult to grasp. It is nevertheless used operationally because convergence is very fast during the first iterations and in practice few of them are performed anyway, because of the numerical cost of the optimization.

5.4.1.3 ▪ Multigrid methods

Multigrid methods have been developed for decades in linear algebra, and extended to nonlinear systems. Let us recall briefly the principle, and consider for simplicity a linear equation,

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

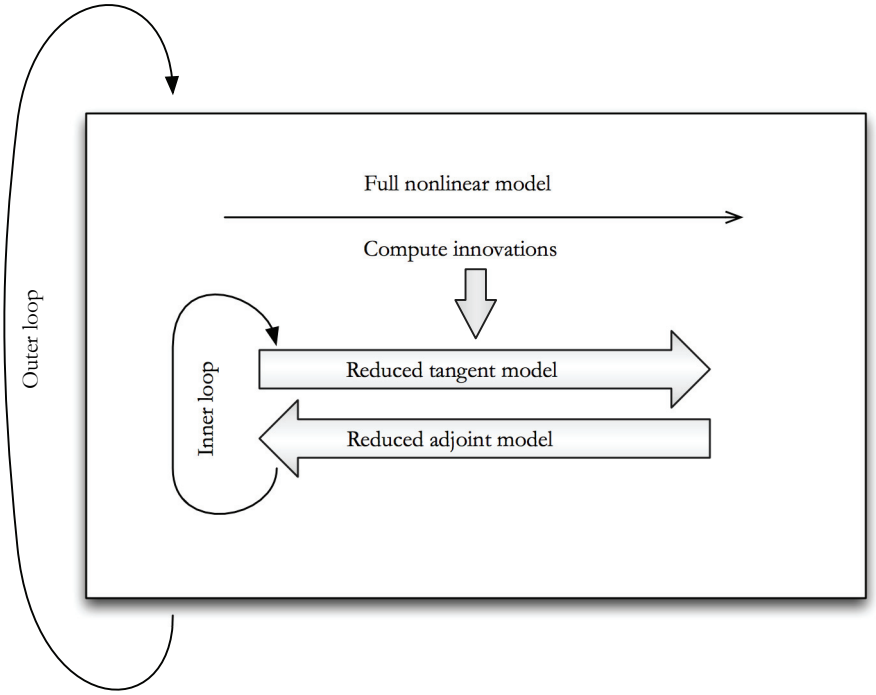

Figure 5.2. *Incremental 4D-Var with reduced models.*

where $\mathbf{A}$ is a given square matrix, $\mathbf{b}$ a vector, and $\mathbf{x}$ the unknown. This equation may come, e.g., from the discretization of a PDE, so that the vector $\mathbf{x}$ is a model state on a given grid in space and/or time. This system can be solved using iterative methods, such as Jacobi or Gauss–Seidel (see, e.g., Golub and van Loan [2013]). It has been shown that these methods have good rates of convergence for small scales/high frequencies but quite poor convergence for large scales/low frequencies. The idea of the multigrid method is to provide corrections of this problem by coarsening the grids: the residual error is projected onto a coarser resolution grid so that low frequencies on the fine original grid become high frequencies on the coarse grid, and a correction is provided from the coarse grid. This can be illustrated as follows with two grids:

1. Perform a few iterations to solve $\mathbf{A}_\mathrm{f}\mathbf{x} = \mathbf{b}_\mathrm{f}$ on the fine original grid; obtain a first guess $\mathbf{x}_\mathrm{f}$ of the solution on the fine grid.
2. Compute the fine-grid residual error: $\mathbf{r}_\mathrm{f} = \mathbf{A}_\mathrm{f}\mathbf{x}_\mathrm{f} - \mathbf{b}_\mathrm{f}$.
3. Project the error, $\mathbf{r}_\mathrm{f}$, onto the coarse grid to obtain $\mathbf{r}_\mathrm{c}$. As the error is essentially located on the large scales, one should have $\mathbf{r}_\mathrm{c} \simeq \mathbf{r}_\mathrm{f}$.
4. Perform a few iterations with the coarse grid solver, $\mathbf{A}_\mathrm{c}\mathbf{y} = \mathbf{r}_\mathrm{c}$, to obtain a coarse grid correction $\mathbf{y}_\mathrm{c}$.
5. Update $\mathbf{x} = \mathbf{x}_\mathrm{f} + \mathbf{y}_\mathrm{c}$, and perform a few more iterations on the fine grid.

Of course the multigrid method can be extended to more than two grids, and multiple cycles can be done by alternating resolutions on the various grids. This has also been adapted to the Newton and Gauss–Newton methods to solve a DA-type equation, $\nabla J(\mathbf{x}) = 0$. More details can be found in Debreu et al. [2015] and Neveu [2011]. This approach is quite promising. However, it needs to be implemented on more realistic cases. Moreover, it still requires a few costly iterations on the fine grid, so that for now it is not competitive compared to the multi-incremental approach, which has an excellent convergence rate.

5.4.2 ▪ Reduced 4D-Var

Like reduced Kalman filtering (see Section 5.3), reduced 4D-Var [Durbiano, 2001] aims at reducing the dimension in the core part of the algorithm, namely the minimization. The control variable $\mathbf{x} \in \mathbb{R}^n$, which can contain as always the initial state of the system, numerical parameters, boundary conditions, forcing parameters, etc., is projected onto a lower-dimension space around the background

$$\mathbf{x} = \mathbf{x}^{\mathrm{b}} + \sum_{i=1}^{r} c_i \mathbf{l}_i,$$

where r is the reduced dimension, $\{c_1, \dots, c_r\}$ are real coefficients, and $\{\mathbf{l}_1, \dots, \mathbf{l}_r\}$ is a suitable reduced basis of vectors in $\mathbb{R}^n$. In the framework of incremental 4D-Var, the control variable would be simply

$$\delta\mathbf{x} = \sum_{i=1}^{r} c_i \mathbf{l}_i,$$

so that now $\delta\mathbf{x} \in \mathrm{Span}(\mathbf{l}_1, \dots, \mathbf{l}_r)$. The principle of reduced 4D-Var is then to minimize the reduced incremental cost function

$$\tilde{J}_r(c_1, \dots, c_r) = \tilde{J}_r^{\mathrm{b}}(c_1, \dots, c_r) + \tilde{J}_r^{\mathrm{o}}(c_1, \dots, c_r),$$

$$\tilde{J}_r^{\mathrm{b}}(c_1, \dots, c_r) = \left(\textstyle\sum_{i=1}^{r} c_i \mathbf{l}_i\right)^{\mathrm{T}} \mathbf{B}_r^{-1} \left(\textstyle\sum_{i=1}^{r} c_i \mathbf{l}_i\right),$$

$$\tilde{J}_r^{\mathrm{o}}(c_1, \dots, c_r) = \sum_{k=1}^{K} \left[\mathbf{d}_k - \mathbf{H}_k \mathbf{M}_k \dots \mathbf{M}_1 \left(\textstyle\sum_{i=1}^{r} c_i \mathbf{l}_i\right)\right]^{\mathrm{T}} \mathbf{R}_k^{-1} \left[\mathbf{d}_k - \mathbf{H}_k \mathbf{M}_k \dots \mathbf{M}_1 \left(\textstyle\sum_{i=1}^{r} c_i \mathbf{l}_i\right)\right],$$

where $\mathbf{d}_k$ is the innovation vector, as in incremental 4D-Var (see Algorithm 5.2). The background error covariance matrix $\mathbf{B}_r$ is also reduced using the EOFs,

$$\mathbf{B}_r = \mathbf{L}\boldsymbol{\Delta}_r \mathbf{L}^{\mathrm{T}},$$

where $\mathbf{L}$ is the matrix containing the $\mathbf{l}_i$'s in columns and $\boldsymbol{\Delta}_r$ is a diagonal matrix.

The reduced cost function $\tilde{J}_r$ is now a function from $\mathbb{R}^r$ to $\mathbb{R}$. In other words, the minimization is now done in a space of dimension r, usually between a few tens and a few hundreds, instead of n, usually around millions or billions, so that it is much cheaper computationally.

This algorithm was introduced by Durbiano [2001], and various bases were considered for $\mathbf{l}_1, \dots, \mathbf{l}_r$: EOFs, forward/backward singular vectors, Lyapunov vectors, bred vectors (see Section 5.1.2.1 for more details about the bases).

The algorithm was implemented for a 2D shallow-water model, and the study concluded the superiority of the EOFs. A later article [Robert et al., 2005] implemented with success the method in a more realistic primitive equations ocean model, but still in the idealized twin experiment framework, with simulated data instead of real observations.

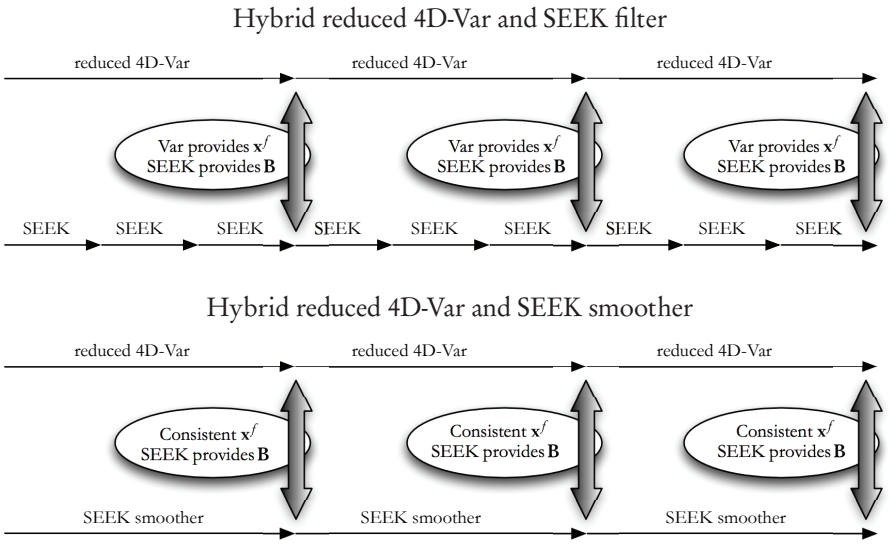

Figure 5.3. *Hybridization of the reduced 4D-Var and SEEK filter/smoother algorithms.*

5.4.2.1 ▪ Hybrid reduced/incremental 4D-Var

Reduced 4D-Var has been applied to a more realistic ocean experiment with real data in Hoteit and Köhl [2006] and Robert et al. [2006a]. Reduced 4D-Var was shown to have a tremendous impact on the convergence rate. However, it was unable to reach a good enough minimum, contrary to the classical unreduced incremental 4D-Var. This seems to be due mostly to the presence of a significant model error, which is more important when using only a few EOFs than with the full model. It was therefore proposed to hybridize both methods: first perform a few iterations of reduced 4D-Var, as a preprocessing tool, and then perform a few iterations of classical incremental 4D-Var. This method, called two-step 4D-Var by Robert et al. [2006a] and the hybrid reduced adjoint method by Hoteit and Köhl [2006], combined the advantages of both methods: good minimization and computational savings, so that the same final result was obtained with a computational time divided by two.

5.4.2.2 ▪ Hybrid reduced 4D-Var/SEEK filter

Hybrid methods coupling variational and filtering methods are numerous and will be treated in more detail in Chapter 7. This subsection is restricted to the hybridization of *reduced* variational and filtering methods.

A full comparison between the reduced 4D-Var and SEEK filter was done by Robert et al. [2006b] in an idealized, perfect model, twin experiment framework. The authors also proposed a new algorithm based on both methods. The idea is simply to perform in parallel a reduced 4D-Var assimilation and three cycles of the SEEK filter on the same assimilation window, starting from the same background information, as shown in Figure 5.3. At the end of the assimilation window,

1. the reduced 4D-Var analysis provides the new background state, and
2. the SEEK filter provides the new background error covariance matrix.

As the SEEK filter and the reduced 4D-Var did not provide strictly the same results, both were required to run fully on the same time window, resulting in doubling the computing time compared to using only one or the other. However, as both are reduced methods, the time savings are still significant.

An updated version of this algorithm was proposed with the SEEK smoother instead of the SEEK filter by Krysta et al. [2011], based on a idea of Veersé et al. [2000]—see Figure 5.3 for a schematic presentation. The SEEK smoother and the reduced 4D-Var methods provide consistent results at the end of the time window, i.e., the same analysis; covariance of the SEEK consistent with the reduced cost function, when using the same background state at the initial time; the same background error covariance matrix; and the same set of EOFs for the reduction. The coupling then works as follows:

1. run a reduced 4D-Var analysis and use it to compute the new background state, and
2. in parallel, perform a SEEK smoother analysis *only for the analysis error covariance matrix* on the same time window, and feed it to the reduced 4D-Var.

This method offers significantly improved results compared to the classical 4D-Var, due to the background error covariance matrix evolving cycle after cycle.