Chapter 7

Four-Dimensional Variational Assimilation

With 3-dimensional methods of data assimilation, such as 3DVAR or OI, the problem is under-determined, so prior information is needed. This prior information or background often comes from a short term forecast. The background error covariance matrix determines the scales in the analysis. As modelled by OI, the smallest scales that can be represented with seasonal averaging of innovations is about 300 km. This scale reflects the smallest distance between radiosonde stations in North America and the fact that the flow has been averaged over 1 or more months. In addition, the background error correlation matrix is often modelled using homogeneous and isotropic assumptions. How can we relax some of these assumptions? Some fields such as ozone can exhibit very sharp gradients when it is merely passively transported in the upper troposphere. It can be shown that correlations are conserved between two points as they are advected. Thus the background errors can be very anisotropic, forming fine filaments. The background errors clearly have energy at small scales as evidenced by the sharp gradients. A more blatant example of the need for anisotropic correlations is the fact that the wind typically has a larger component in the zonal rather than meridional directions. Clearly, it would be useful to be able to analyse smaller scales in the analysis than 300 km since global NWP forecast models can represent scales to 50 km. It would also be useful to be able to have flow dependent background error covariances since the forecast error is flow dependent. If this were possible, then observations taken from within a high pressure centre could be analysed differently (barotropically) than those originating in a frontal zone (which could have baroclinic structures).

In this section, we shall see that 4D methods and in particular, 4DVAR, can provide both smaller scales in the analysis and flow dependent structures.

7.1 Introduction to 4DVAR

The general idea behind 4DVAR is to find the initial conditions which lead to the best fit to observations which are spread over a time interval. The notion of "best" is defined by a scalar cost function:

$$J = \frac{1}{2}\int_0^T < (\mathbf{x}(t) - \mathbf{z}(t)), (\mathbf{x}(t) - \mathbf{z}(t)) > dt. \tag{7.1}$$

As before, $\mathbf{x}(t)$ is the model state vector, but now it is varying continuously with time. Similarly, the observation vector is given by $\mathbf{z}(t)$ which varies continuously in time. Note that the model and observation vectors are the same dimension, in this discussion. The operator $< \mathbf{a}, \mathbf{b} >$ refers to some inner product. For model states in grid point space, the obvious choice of an inner product is the scalar or dot product. Let's define the initial model state vector by $\mathbf{x}(0) = \mathbf{x}_0$. For a deterministic model, once the initial conditions, boundary conditions and model parameters are specified, the state evolution can be determined. Thus the cost function may be viewed as a function of initial state, boundary conditions and model parameters only. If we assume the model parameters and boundary conditions are known, then J is a function of initial state alone. Therefore, the obvious question is: how does J change when the initial state is changed? Because the initial state is a vector of many dimensions, the answer depends on which components of the initial state vector are changed. An arbitrary change to the ith component of $\mathbf{x}_0$ will result in a change in J. This change divided by the perturbation to the ith component defines the *sensitivity* of J to that perturbation. In general, an arbitrary change to the initial state, $\delta\mathbf{x}_0$ will result in a change in J called the *directional derivative* of J at $\mathbf{x}_0$ in the direction $\delta\mathbf{x}_0$. The gradient of J is related to the directional derivative, δJ by

$$\delta J = < \nabla_{\mathbf{x}_0} J, \delta\mathbf{x}_0 >, \tag{7.2}$$

where

$$\nabla_{\mathbf{x}_0} J = \begin{bmatrix} \frac{\partial J}{\partial x_{0,1}} \\ \frac{\partial J}{\partial x_{0,2}} \\ \vdots \\ \frac{\partial J}{\partial x_{0,n}} \end{bmatrix}$$

Ultimately, the initial state we are interested in, is the one which minimizes J. The *adjoint method* involves the calculation of $\nabla_{\mathbf{x}_0} J$. This alone will not determine the minimum of J, but some iterative procedure can be defined which leads to a better guess (of initial state) which has lower cost (or better fits the observations). Thus, one can proceed iteratively until one is sufficiently close to the minimum.

The key step in 4DVAR is the calculation of the gradient of J about the initial state for an arbitrary direction. If one were to consider a brute force method, one could try finite differences. Thus for the ith component of the gradient vector, one could try

$$\frac{\partial J}{\partial x_{0,i}} = \frac{J(\mathbf{x}_0 + \alpha\mathbf{e}_i) - J(\mathbf{x}_0)}{\alpha},$$

where $\mathbf{e}_i$ is a vector of zeros except for the ith element which is 1, and where α is a small positive scalar. For realistic applications in which n is $\mathrm{O}(10^7)$, it is clear that this method is impractical, especially when one considers that the gradient is needed at every iteration of the descent method. The beauty of the adjoint method is that this gradient is obtained with a single integration of the *adjoint model* backward in time. As we shall see, the adjoint model is simply the adjoint of the Tangent Linear Model (TLM) with respect to a particular inner product. Because the adjoint model runs backward in time, people sometimes confuse it with the *inverse* model. However, for a weather forecast model, the inverse model doesn't exist. Processes such as diffusion are numerically unstable when the time direction is reversed. Moreover, processes such as precipitation are irreversible. The adjoint model does exist, however. This is clear when one realizes that the whole role of the adjoint model is simply to calculate the gradient of the cost function with respect to the initial state. This derivative exists except when there are discontinuous (or threshold) processes such as rainout. If such discontinuities can be smoothed, then the adjoint exists. (In practice, it is necessary to simplify or smooth physical processes with thresholds or on-off switches for the purpose of the adjoint calculation. The full nonlinear processes are used in the forward model however.)

In summary, the adjoint method is an iterative scheme which involves searching for the minimum of a scalar cost function with respect to a multi-dimensional initial state. The algorithm is called a descent method, and requires the derivative of the cost function with respect to arbitrary perturbations of the initial state. This derivative is obtained by running an adjoint model backward in time. Once the derivative is obtained, a direction which leads to lower cost has been identified, but the step size has not. Therefore, further calculations are needed to determine how far along this direction one needs to go to find a lower cost. Once this initial state is found, the next iteration is started. The algorithm proceeds until the minimum of the cost function is found.

It should be noted that the adjoint method is used in 4DVAR to find the initial conditions which minimize a cost function. However, one could equally well have chosen to find the boundary conditions, or model parameters. For the case of tropospheric pollutants, initial conditions are not so important in determining the chemical state. What is more important is the emissions or sources of pollution. Thus the data assimilation problem for tropospheric pollution is concerned with finding the sources (which can be viewed as model error) based on observations of constituents. Finally, for limited area models, variation of the cost function with respect to boundary conditions is also important.

Adjoint methods can be useful whenever the sensitivity of a scalar cost function with respect to model inputs is required, and has application in fields outside of data assimilation, such as predictability (using singular vectors) or ensemble forecasting, or targeting observations (finding where to place observations for most impact on the forecast).

7.2 A simple continuous problem

It is useful to first consider a 4DVAR problem for a low-order model. With a simple model, one can easily demonstrate that there are actually three ways of finding the derivatives of model output with respect to the input: (1) classical variational methods which involve the derivation of Euler-Lagrange equations, (2) the control theory approach of LeDimet and Talagrand (1986), and (3) the Lagrange multiplier approach of Thacker and Long (1988). The following discussion is from Lewis (1990).

Let us consider a highly truncated model of the atmosphere. We will use Platzman's (1964) truncated solution to Burger's equation (1D nonlinear advection-diffusion equation):

$$\frac{du_1}{dt} = -\frac{1}{2}u_1u_2 \tag{7.3}$$

$$\frac{du_2}{dt} = \frac{1}{2}u_1^2. \tag{7.4}$$

Here u_1 and u_2 are spectral amplitudes which are continuous functions of time. The model state is given by

$$\mathbf{x} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},$$

so the model can be written in vector notation as

$$\frac{d\mathbf{x}}{dt} = M(\mathbf{x}). \tag{7.5}$$

The problem is to find the optimal initial state (t=0) such that the functional

$$J = \frac{1}{2}\int_0^T < (\mathbf{x}(t) - \mathbf{z}(t)), (\mathbf{x}(t) - \mathbf{z}(t)) > dt. \tag{7.6}$$

is minimized, subject to the constraint, $d\mathbf{x}/dt = M(\mathbf{x})$. Here $\mathbf{z}(t)$ represents the observed spectral amplitudes which are continuous functions of time:

$$\mathbf{z} = \begin{bmatrix} u_1^{obs} \\ u_2^{obs} \end{bmatrix},$$

7.2.1 The classical variational method

Instead of minimizing J, we can form the Lagrangian using

$$L = J + \int_0^T \lambda_1(\frac{du_1}{dt} + \frac{1}{2}u_1u_2)dt + \int_0^T \lambda_2(\frac{du_2}{dt} - \frac{1}{2}u_1^2)dt, \tag{7.7}$$

where λ_1 and λ_2 are arbitrary Lagrange multipliers which are continuous functions of time. To minimize J, we require that the variation of L be 0. Thus,

$$\delta L = \delta J + \int_0^T \lambda_1(\frac{d\delta u_1}{dt} + \frac{1}{2}\delta u_1 u_2 + \frac{1}{2}u_1\delta u_2)dt + \int_0^T \lambda_2(\frac{d\delta u_2}{dt} - u_1\delta u_1)dt. \tag{7.8}$$

Now we can substitute for δJ and integrate the time derivative terms by parts to obtain:

$$\begin{aligned}\delta L &= \int_0^T [(u_1 - u_1^{obs})\delta u_1 + (u_2 - u_2^{obs})\delta u_2 - \delta u_1\frac{d\lambda_1}{dt} - \delta u_2\frac{d\lambda_2}{dt} + \frac{1}{2}\lambda_1\delta u_1 u_2 \\ &+ \frac{1}{2}\lambda_1 u_1\delta u_2 - \lambda_2 u_1\delta u_1]dt + (\lambda_1\delta u_1|_0^T + \lambda_2\delta u_2|_0^T).\end{aligned}$$

To get rid of the boundary terms, we must let $\lambda_1(T) = \lambda_1(0) = 0$ and $\lambda_2(T) = \lambda_2(0) = 0$ since δu_1 and δu_2 are arbitrary. In this case, we can simplify the above, and upon gathering terms, obtain:

$$\begin{aligned}\delta L &= \int_0^T \delta u_1(u_1 - u_1^{obs} - \frac{d\lambda_1}{dt} + \frac{1}{2}\lambda_1 u_2 - \lambda_2 u_1)dt \\ &+ \int_0^T \delta u_2(u_2 - u_2^{obs} - \frac{d\lambda_2}{dt} + \frac{1}{2}\lambda_1 u_1)dt = 0.\end{aligned}$$

Since δu_1 and δu_2 are arbitrary, δL vanishes when

$$\frac{d\lambda_1}{dt} + \lambda_2 u_1 - \frac{1}{2}\lambda_1 u_2 = u_1 - u_1^{obs} \tag{7.9}$$

$$\frac{d\lambda_2}{dt} - \frac{1}{2}\lambda_1 u_1 = u_2 - u_2^{obs}. \tag{7.10}$$

Thus to solve for the minimum, we must solve these two equations (Euler-Lagrange equations) along with the two constraints, (7.3) and (7.4) and the boundary conditions $\lambda_1(T) = \lambda_1(0) = 0$ and $\lambda_2(T) = \lambda_2(0) = 0$. This is not always easy to do, especially with a complex numerical model.

7.2.2 Optimal control theory

For optimal control theory, we first need to know how arbitrary perturbations of the intial condition evolve in time. Why? Ultimately, we want to compute the derivative of J with respect to the initial state, $\mathbf{x}_0$. However, we are going to do this using the chain rule:

$$\frac{dJ}{d\mathbf{x}_0} = \frac{dJ}{d\mathbf{x}}\frac{d\mathbf{x}}{d\mathbf{x}_0}.$$

The second term refers to the variation of the output of the model with respect to the input. This is given by the Tangent Linear Model or TLM. For the model given by (7.5), arbitrary perturbations evolve according to:

$$\frac{d\delta\mathbf{x}}{dt} = \frac{dM(\mathbf{x})}{d\mathbf{x}}\delta\mathbf{x}. \tag{7.11}$$

Let us consider the specific equations, (7.3) and (7.4). We shall expand the state in terms of a basic state and perturbation:

$$\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}.$$

Note that this mean state is evolving in time (unlike the case of usual stability analysis problems in atmospheric dynamics courses), and is often referred to as the mean or background trajectory. Expanding (7.3) and (7.4) yields:

$$\begin{aligned} \frac{d}{dt}(\bar{u}_1 + \delta u_1) &= -\frac{1}{2}(\bar{u}_1 + \delta u_1)(\bar{u}_2 + \delta u_2) && (7.12)\\ \frac{d}{dt}(\bar{u}_2 + \delta u_2) &= \frac{1}{2}(\bar{u}_1 + \delta u_1)^2. && (7.13) \end{aligned}$$

Now assuming the mean trajectory satisfies the equations:

$$\begin{aligned} \frac{d\bar{u}_1}{dt} &= -\frac{1}{2}\bar{u}_1\bar{u}_2 && (7.14)\\ \frac{d\bar{u}_2}{dt} &= \frac{1}{2}(\bar{u}_1)^2, && (7.15) \end{aligned}$$

we can subtract these from (7.12) and (7.13) to get:

$$\begin{aligned} \frac{d}{dt}(\delta u_1) &= -\frac{1}{2}(\bar{u}_1\delta u_2 + \bar{u}_2\delta u_1 + \delta u_1\delta u_2)\\ \frac{d}{dt}(\delta u_2) &= \bar{u}_1\delta u_1 + \frac{1}{2}(\delta u_1)^2 \end{aligned}$$

Now, in the definition of the derivative $d\mathbf{x}/d\mathbf{x}_0$, we are interested in changes to $\mathbf{x}$ in the limit of infinitessimally small changes in $\mathbf{x}_0$. So, for infinitessimally small $\delta\mathbf{x}_0$, the above system becomes

$$\begin{aligned} \frac{d}{dt}(\delta u_1) &= -\frac{1}{2}(\bar{u}_1\delta u_2 + \bar{u}_2\delta u_1) && (7.16)\\ \frac{d}{dt}(\delta u_2) &= \bar{u}_1\delta u_1. && (7.17) \end{aligned}$$

In matrix notation, we can write

$$\frac{d\delta\mathbf{x}}{dt} = \begin{bmatrix} -\frac{1}{2}\bar{u}_2 & -\frac{1}{2}\bar{u}_1 \\ \bar{u}_1 & 0 \end{bmatrix} \begin{bmatrix} \delta u_1 \\ \delta u_2 \end{bmatrix} = \mathbf{M}\delta\mathbf{x}. \tag{7.18}$$

This is the Tangent Linear Model or TLM. The TLM describes how arbitrary perturbations of the initial conditions evolve in time. Note that variations of the cost function are given by

$$\begin{aligned} \delta J &= \frac{1}{2}\int_0^T [< \bar{\mathbf{x}} + \delta\mathbf{x} - \mathbf{z}, \bar{\mathbf{x}} + \delta\mathbf{x} - \mathbf{z} > - < \bar{\mathbf{x}} - \mathbf{z}, \bar{\mathbf{x}} - \mathbf{z} >]dt \\ &= \int_0^T < \bar{\mathbf{x}} - \mathbf{z}, \delta\mathbf{x} > dt. \qquad (7.19) \end{aligned}$$

Again, because we are interested in infintessimally small perturbations (consistent with the definition of a derivative), we have dropped the second order terms.

Now consider the inner product of the Tangent Linear system with an arbitrary vector, $\mathbf{p}$. Then, integrate by parts.

$$\int_0^T < \frac{d\delta\mathbf{x}}{dt}, \mathbf{p} > dt \;=\; < \delta\mathbf{x}(T), \mathbf{p}(T) > - < \delta\mathbf{x}(0), \mathbf{p}(0) > - \int_0^T < \delta\mathbf{x}, \frac{d\mathbf{p}}{dt} > dt. \qquad (7.20)$$

Let us insist that $\mathbf{p}(T) = 0$. Then, using the TLM equation, we have that

$$\int_0^T < \mathbf{M}\delta\mathbf{x}, \mathbf{p} > dt \;=\; - < \delta\mathbf{x}(0), \mathbf{p}(0) > - \int_0^T < \delta\mathbf{x}, \frac{d\mathbf{p}}{dt} > dt. \qquad (7.21)$$

or on rearranging:

$$- < \delta\mathbf{x}(0), \mathbf{p}(0) >= \int_0^T < \mathbf{M}\delta\mathbf{x}, \mathbf{p} > dt + \int_0^T < \delta\mathbf{x}, \frac{d\mathbf{p}}{dt} > dt. \qquad (7.22)$$

Now, using the definition of an adjoint operator:

$$< \mathbf{x}, \mathbf{L}\mathbf{y} >=< \mathbf{L}^*\mathbf{x}, \mathbf{y} >$$

we can rewrite the first term to obtain

$$\begin{aligned} - < \delta\mathbf{x}(0), \mathbf{p}(0) > &= \int_0^T < \delta\mathbf{x}, \mathbf{M}^*\mathbf{p} > dt + \int_0^T < \delta\mathbf{x}, \frac{d\mathbf{p}}{dt} > dt. \\ &= \int_0^T < \delta\mathbf{x}, \mathbf{M}^*\mathbf{p} + \frac{d\mathbf{p}}{dt} > dt. \qquad (7.23) \end{aligned}$$

Let us require that

$$\frac{d\mathbf{p}}{dt} + \mathbf{M}^*\mathbf{p} = \bar{\mathbf{x}} - \mathbf{z} \qquad (7.24)$$

where $\mathbf{p}(T) = 0$. This linear equation defines the **adjoint model**. Then,

$$- < \delta\mathbf{x}(0), \mathbf{p}(0) >= \int_0^T < \delta\mathbf{x}, \bar{\mathbf{x}} - \mathbf{z} > dt = \delta J. \qquad (7.25)$$

Since, by definition,

$$\delta J =< \nabla_{\mathbf{x}_0} J, \delta\mathbf{x}_0 >,$$

it is clear that

$$\nabla_{\mathbf{x}_0} J = -\mathbf{p}(0).$$

Thus, we now know how to obtain the gradient of the cost function with respect to arbitrary changes in the initial condition: you simply integrate the adjoint equation (7.24) backward in time from the initial condition $\mathbf{p}(T) = 0$, with forcing by the misfit between the mean trajectory and the observations.

Using a brute force method to obtain the sensitivity of the model output with respect to the input would require running the TLM n times for an initial state of dimension n, This would give the n components of the gradient vector, $\nabla_{\mathbf{x}_0}$. The beauty of the adjoint method, is that you can obtain this derivative by simplying integrating the adjoint model backward in time, forced by the misfit between the background trajectory and the observations. Thus, obtaining the gradient is now tractable for large scale problems as when n is $O(10^7)$.

In the control theory derivation, the TLM and variations of the cost function involved linearization (or the dropping of nonlinear terms). Therefore, you may be wondering if some approximations have been made. The answer is no. The method is exact, because the linearization was done to obtain a derivative, and the definition of a derivative involves infinitessimal changes. Thus when we dropped nonlinear terms, it is absolutely consistent with the definition of a derivative.

Compared to the classical variational method, the adjoint method required the initial condition $\mathbf{p}(T) = 0$. The classical method required $\lambda_1(T) = \lambda_1(0) = 0$ and $\lambda_2(T) = \lambda_2(0) = 0$. For the adjoint method, $\mathbf{p}(0)$ is nonzero in general, since it only defines the gradient with respect to the initial condition. However, it is zero when the minimum of the cost function has been found. Thus the adjoint method uses the current estimate of the state (i.e. the background trajectory), whereas the Euler-Lagrange equations have a form which assumes the state at the minimum.

7.2.3 Lagrange multiplier method

Consider the Lagrange multiplier,

$$\boldsymbol{\Lambda}(t) = \begin{bmatrix} \lambda_1(t) \\ \lambda_2(t) \end{bmatrix}$$

and form the Lagrangian:

$$L = J + \int_0^T < \boldsymbol{\Lambda}, \frac{d\mathbf{x}}{dt} - M(\mathbf{x}) > dt. \tag{7.26}$$

Variations of the Lagrangian yield

$$\delta L = \delta J + \int_0^T < \delta\boldsymbol{\Lambda}, \frac{d\mathbf{x}}{dt} - M(\mathbf{x}) > dt + \int_0^T < \boldsymbol{\Lambda}, \frac{d\delta\mathbf{x}}{dt} - \mathbf{M}\delta\mathbf{x}) > dt. \tag{7.27}$$

The last term on the right hand side can be expanded as

$$\begin{aligned}
\int_0^T & < \boldsymbol{\Lambda}, \frac{d\delta\mathbf{x}}{dt} - \mathbf{M}\delta\mathbf{x}) > dt = \int_0^T < \boldsymbol{\Lambda}, \frac{d\delta\mathbf{x}}{dt} > dt - \int_0^T < \boldsymbol{\Lambda}, \mathbf{M}\delta\mathbf{x}) > dt \\
&= \ < \boldsymbol{\Lambda}(T), \delta\mathbf{x}(T) > - < \boldsymbol{\Lambda}(0), \delta\mathbf{x}(0) > - \int_0^T < \frac{d\boldsymbol{\Lambda}}{dt}, \delta\mathbf{x} > dt - \int_0^T < \boldsymbol{\Lambda}, \mathbf{M}\delta\mathbf{x}) > dt \\
&= \ < \boldsymbol{\Lambda}(T), \delta\mathbf{x}(T) > - < \boldsymbol{\Lambda}(0), \delta\mathbf{x}(0) > - \int_0^T < \frac{d\boldsymbol{\Lambda}}{dt}, \delta\mathbf{x} > dt - \int_0^T < \mathbf{M}^*\boldsymbol{\Lambda}, \delta\mathbf{x}) > dt \\
&= \ - \int_0^T < \frac{d\boldsymbol{\Lambda}}{dt} + \mathbf{M}^*\boldsymbol{\Lambda}, \delta\mathbf{x}) > dt
\end{aligned} \tag{7.28}$$

where we have used the natural boundary conditions, $\mathbf{\Lambda}(T) = \mathbf{\Lambda}(0) = 0$. Since we've already determined δJ from (7.19), we can write

$$\begin{aligned}\delta L &= \int_0^T < \bar{\mathbf{x}} - \mathbf{z}, \delta\mathbf{x} > dt + < \delta\mathbf{\Lambda}, \frac{d\mathbf{x}}{dt} - M(\mathbf{x}) > dt - \int_0^T < \frac{d\mathbf{\Lambda}}{dt} + \mathbf{M}^*\mathbf{\Lambda}, \delta\mathbf{x}) > dt \\ &= \int_0^T < -\frac{d\mathbf{\Lambda}}{dt} - \mathbf{M}^*\mathbf{\Lambda} + (\bar{\mathbf{x}} - \mathbf{z}), \delta\mathbf{x} > dt + < \delta\mathbf{\Lambda}, \frac{d\mathbf{x}}{dt} - M(\mathbf{x}) > dt \\ &= 0. \end{aligned} \tag{7.29}$$

At the minimum of L, for arbitrary variations in $\delta\mathbf{x}$ and $\delta\mathbf{\Lambda}$, we must have that

$$\begin{aligned}\frac{d\mathbf{x}}{dt} &= M(\mathbf{x}), \\ \frac{d\mathbf{\Lambda}}{dt} + \mathbf{M}^*\mathbf{\Lambda} &= \bar{\mathbf{x}} - \mathbf{z}.\end{aligned} \tag{7.30}$$

The first equation is the forward nonlinear model, and the second equation is exactly the adjoint model, (7.24), derived above. However, we have only defined the solution at the minimum. To get to the solution, the exact procedure of the previous section is used. Start with a first guess of the initial state, integrate to obtain the trajectory, then run the adjoint equation backwards in time, starting from $\mathbf{\Lambda}(T) = 0$. Then $\mathbf{\Lambda}(0)$ is obtained by directly integrating (7.30). The solution is obtained by assuming solutions of the form: $\mathbf{\Lambda} = \tilde{\mathbf{\Lambda}} e^{-\boldsymbol{M}^* t}$. Then

$$-\mathbf{\Lambda}(0) = \int_0^T e^{\boldsymbol{M}^* t} \nabla_{\mathbf{x}} J = \nabla_{\mathbf{x}_0} J.$$

Thus after the adjoint integration backward in time, $\mathbf{\Lambda}(0)$ gives us exactly what we want: the negative gradient of the cost function with respect to the initial state. Then using this, a step size must be determined and a new initial state with lower cost is obtained. Then the iterations are repeated.

The Lagrange multiplier method is identical to the adjoint method in practice since the iterative procedure and the adjoint model are the same. The only difference is in the derivation of the algorithm. By comparing the derivations, it is clear that the adjoint variable of the control theory method is actually the Lagrange multiplier.

Before we finish this section, we should write down the adjoint model equations. Let

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix},$$

The adjoint model is given by the transpose of the TLM transition equation, (7.18):

$$\frac{d\mathbf{\Lambda}}{dt} + \mathbf{M}^*\delta\mathbf{\Lambda} = \begin{bmatrix} d\lambda_1/dt \\ d\lambda_2/dt \end{bmatrix} + \begin{bmatrix} -\frac{1}{2}\bar{u}_2 & \bar{u}_1 \\ -\frac{1}{2}\bar{u}_1 & 0 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} \bar{u}_1 - \bar{u}_1^{obs} \\ \bar{u}_1 - \bar{u}_1^{obs} \end{bmatrix} \tag{7.31}$$

This is also the adjoint equation for the control theory method. Comparing to (7.9) and (7.10), we see that these are also identical to the adjoint equations derived in the classical variational method.

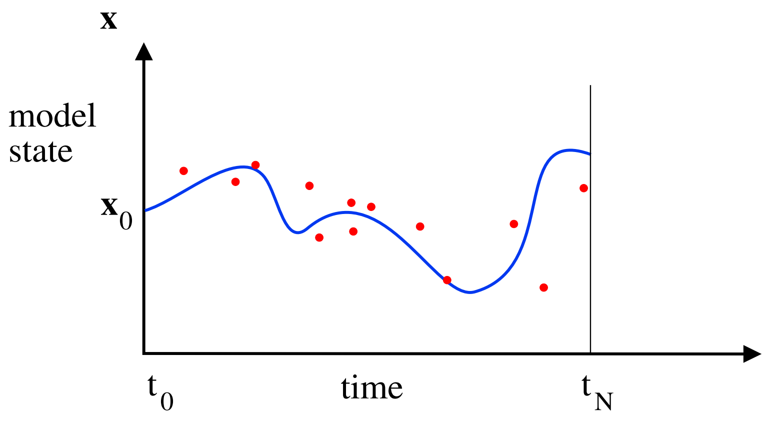

Figure 7.1: The 4DVAR algorithm. 4DVAR seeks the initial state which when integrated in time best fits the observations in a time interval.

7.3 The 4DVAR algorithm

Consider a set of observations distributed over a time interval or assimilation period from t_0 to t_N. This situation is depicted in Fig. 7.1. The model state has dimension n but is depicted in only 1 dimension, in this figure. The model state at t_0 is $\mathbf{x}_0$. The idea of the 4DVAR algorithm is to find the initial state, $\mathbf{x}_0$, which produces a model trajectory (when integrated in time using the forecast model) that "best" fits the observations. Here, "best" means that a scalar cost function is minimized. Mathematically, in 4DVAR, we want to find the state $\mathbf{x}^a$ that minimizes:

$$\begin{aligned} J(\mathbf{x}_0) &= \frac{1}{2}[\mathbf{x}_0 - \mathbf{x}_0^f]^{\mathrm{T}}(\mathbf{P}_0^f)^{-1}[\mathbf{x}_0 - \mathbf{x}_0^f] \\ &+ \frac{1}{2}\sum_{i=1}^{N}[\mathbf{z}_i - H(\mathbf{x}_i)]^{\mathrm{T}}\mathbf{R}^{-1}[\mathbf{z}_i - H(\mathbf{x}_i)]. \end{aligned} \tag{7.32}$$

At each model time step, the innovation (difference between model and observed states evaluated at observation locations) is squared and weighted by the inverse of the observation error variance. Thus accurate observations (small variances) are given greater weight than poor observations. The initial state $\mathbf{x}_0$ is called the *control variable* because it is the variable with respect to which the minimization will be performed.

At each time step, the model state, $\mathbf{x}_i$, is related to the model initial state, $\mathbf{x}_0$, through the forecast model itself:

$$\mathbf{x}_i = \mathrm{M}(\mathbf{x}_{i-1}) = M(M(\ldots M(\mathbf{x}_0))).$$

However, in reality, knowledge of the model state at t_0 does not give an accurate description of atmospheric state at a later time t_i, because the forecast model is imperfect. This departure from the truth is called model error and arises due to a number of sources: numerical discretization error, omission of processes, parameterization errors, representativeness errors, etc. Thus, in 4DVAR we use the forecast model as a *strong constraint* on state evolution. The advantage of doing so is that complex nonlinear relationships can develop between analysis variables. (Recall that in 3D schemes, we were limited to linear relationships to define physical balances between analysis variables.) Another advantage of relating model states in time using a forecast model is that instead of having N times n unknowns, we have only n unknowns, namely, $\mathbf{x}_0$. Thus the underdeterminacy problem (more unknowns than knowns/observations) is reduced. However, the drawback of this assumption is that we are essentially assuming the forecast model is perfect. This is never going to be the case. If we invoke a 4DVAR algorithm, then we are hoping that this is approximately the case.

Now consider the 4DVAR algorithm in more detail. The cost function, (7.32), is a scalar. This number is computed by summing over all time steps, the norm of the misfit between observations and model states and the norm of the departure from the initial state. This function is supplied to a packaged software routine designed for large scale problems. Typically, such software also required the gradient of the cost function. To calculate the gradient with respect to the control variable we first note the connection between model states at different time steps:

$$\mathbf{x}_{k+1} = \mathrm{M}(\mathbf{x}_k). \tag{7.33}$$

The variation in a state is related to that at another time step by:

$$\delta\mathbf{x}_{k+1} = \frac{d\mathrm{M}}{d\mathbf{x}}(\mathbf{x}_k)\delta\mathbf{x}_k = \mathbf{M}_k\delta\mathbf{x}_k. \tag{7.34}$$

The operator $\mathbf{M}_k$ is called the **Tangent Linear Model** and is a complete linearization of the forecast model with respect to the control variables (model state variables). This Tangent Linear Model (hereafter called the TLM) describes the evolution in time of perturbations about the initial state. The model is linear in terms of the variations since the nonlinear evolution of perturbations involve higher order terms:

$$\begin{aligned} \delta_{NL}\mathbf{x}_{k+1} &= \mathrm{M}(\mathbf{x}_k + \delta\mathbf{x}_k) - \mathrm{M}(\mathbf{x}_k) \\ &= \mathbf{M}_k\delta\mathbf{x}_k + \text{higher order terms.} \end{aligned} \tag{7.35}$$

Now the **adjoint** of a linear operator, $\mathbf{M}_k$, is defined as:

$$< \mathbf{M}_k\mathbf{x}, \mathbf{z} >=< \mathbf{x}, \mathbf{M}_k^*\mathbf{z} > . \tag{7.36}$$

In our case, the adjoint of the TLM is a full model called the **adjoint model**. If a nonlinear model is roughly quadratic in nonlinearity, then the Tangent Linear Model involves two terms for every 1 term in the forecast model and will thus be twice as expensive. It turns out that for NWP forecast models, the cost of the TLM is between 1 and 2 times the cost of the nonlinear model (or NLM). The cost of the adjoint model (or ADJ) is roughly the same as the cost of the TLM. Once we have developed the TLM and ADJ models of our forecast model, we can proceed with the 4DVAR algorithm.

Variations in the cost function due to variations in the initial state (control variable) are given by:

$$\begin{aligned} \delta J &= \sum_{k=0}^{N} \langle \nabla_{\mathbf{x}_k} J, \delta\mathbf{x}_k \rangle \\ &= \sum_{k=0}^{N} \langle \nabla_{\mathbf{x}_k} J, \mathbf{M}_{k-1}\mathbf{M}_{k-2}\dots\mathbf{M}_0\delta\mathbf{x}_0 \rangle \end{aligned}$$

$$
\begin{aligned}
&= \sum_{k=0}^{N} \left\langle \mathbf{M}_0^* \mathbf{M}_1^* \dots \mathbf{M}_{k-1}^* \nabla_{\mathbf{x}_k} J, \delta \mathbf{x}_0 \right\rangle \\
&= \left\langle \sum_{k=0}^{N} \mathbf{M}_0^* \mathbf{M}_1^* \dots \mathbf{M}_{k-1}^* \nabla_{\mathbf{x}_k} J, \delta \mathbf{x}_0 \right\rangle \\
&= \left\langle \nabla_{\mathbf{x}_0} J, \delta \mathbf{x}_0 \right\rangle . \qquad (7.37)
\end{aligned}
$$

Equating the last two lines in the above yields:

$$
\nabla_{\mathbf{x}_0} J = \sum_{k=0}^{N} \mathbf{M}_0^* \mathbf{M}_1^* \dots \mathbf{M}_{k-1}^* \nabla_{\mathbf{x}_k} J. \qquad (7.38)
$$

The gradient of the cost function (7.32) with respect to the state at time t_k is very easy to calculate. However, (7.38) says that we can easily relate this gradient to the one we want– the gradient with respect to the initial state. We simply have to integrate the adjoint model backward in time with a forcing of $\nabla_{\mathbf{x}_k} J$ (which is easy to calculate). The output of the adjoint model is the gradient of the cost function with respect to the initial state. Because the TLM is a linear model, the adjoint model does not have to be run backwards for each time step t_k to compute the contribution of the innovation at t_k to the gradient at t_0. Instead, as we run backward in time, we can simply add in the gradient with respect to $\mathbf{x}_k$ and continue. Thus we need to integrate the adjoint model backward in time, only once, to get the impact of the initial perturbation on the fit to the observations at all time steps.

4DVAR can be seen to be an iterative algorithm. For iteration i, we will

1. Run the nonlinear model with initial conditions, $\mathbf{x}_0^i$, from t_0 to t_N.

2. Compute the cost, $J(\mathbf{x}_0^i)$.

3. Compute the gradient with respect to the initial state, $\nabla_{\mathbf{x}_0^i} J$, to find out the direction of steepest descent.

4. Choose a descent direction, $\mathbf{s}^i$, based on the direction of steepest descent, and choose a step size, ρ^i.

5. Modify the initial state: $\mathbf{x}_0^{i+1} = \mathbf{x}_0^i - \rho^i \mathbf{s}^i$.

This iteration is continued until the minimum of the cost function is found. The minimum occurs when the norm of the gradient of the cost function is zero. However for finite arithmetic with numerical error, we can only reduce the gradient norm to a small positive number. How small is small? This is not clear. Moreover, it may be too expensive to proceed until the minimum is found. Thus, an arbitrary stopping criterion is chosen, such as a preset number of iterations, or a reduction in the gradient norm by 2 orders of magnitude.

In summary, the 4DVAR algorithm searches for the initial state which, when integrated in time, produces a model trajectory that best fits the observations. “Best” here refers to the minimization of a scalar cost function. The number of unknown model states in the assimilation interval is N (where N is the number of model time steps in the assimilation period) but by insisting that each state be related to the initial state through the forecast model integration, the number of unknowns is reduced to 1 times the dimension of the model state, n. Thus forecast model is assumed to be perfect. 4DVAR is expensive to develop since two new, complex models must be coded: the Tangent Linear (TLM) and adjoint (ADJ) models. Thus the data assimilation scheme is now intimately tied to a particular forecast model and cannot be easily attached to a different forecast model. On the other hand, the future may bring solutions to this problem since adjoint compilers are being developed. Thus the TLM or ADJ model could be obtained by running the NLM through this type of compiler. Another possibility occurs when a forecast model is written in an object oriented language such as Fortran 90. In this case, adjoint and tangent linear operators can be easily defined (without taking the adjoint or derivative of the nonlinear model, line-by-line).

Clearly 4DVAR is expensive. Each iteration of the descent algorithm involves a forward model run (to evaluate the cost function), and an adjoint model run (to evaluate the gradient). If the ADJ model is 2 times the CPU of the nonlinear model, then one iteration of 4DVAR involves 3 times the cost of the nonlinear model. If there are 30 iterations to reach the minimum, then 4DVAR is 30x3=90 times the cost of running the forecast model alone during the assimilation period. Once the initial state is obtained, then the full suite of forecasts (from 1 to 14 days) can be issued.

Since 4DVAR is expensive, why is there such interest in 4DVAR for large scale problems? 4DVAR was implemented operationally at ECMWF (European Center for Medium Range Weather Forecasts) in Nov. 1997 and is being developed for operation NWP forecasting purposes at MSC, at the British Met Office and at NCEP (USA). The reason is that 4DVAR is very powerful. Like 3DVAR, observations need not be of model variables. But unlike, 3D schemes, temporal information of observations can be used to infer spatial information. Thus the influence of data is *flow dependent*. In addition, observations can be assimilated at the time of observation. This flow dependence means that increments from observations in a frontal zone can be baroclinically distributed while those from observations in a high center can be distributed barotropically. The greatest impact of 4DVAR operationally was found to be the better depiction of growing baroclinic disturbances. If a storm is missed, forecast scores over the whole month can be affected. With the implementation of 4DVAR, the likelihood of missing a storm was greatly reduced, improving forecast scores in the midlatitude regions.

7.3.1 Adjoint Operator for Euclidean norm

How do you compute an adjoint model? This was done by hand, (coding the adjoint line-by-line) for CMC's forecast model, GEM, by Monique Tanguay and myself. (Automatic adjoint compilers were not sufficiently developed by 1996). The adjoint of a linear operator is defined with respect to a specific inner product. For a grid point model, a convenient inner product is the Euclidean norm:

$$< \mathbf{a}, \mathbf{b} >= \mathbf{a}^{\mathrm{T}}\mathbf{b}.$$

In general, a TLM can be written as:

$$\mathbf{L}\mathbf{x} = \mathbf{y}.$$

The adjoint model is then:

$$\hat{\mathbf{x}} = \mathbf{L}^{*}\hat{\mathbf{y}}$$

The input variable of the TLM is $\mathbf{x}$ which is similar to the output variable of the ADJ. However, the ADJ variables are marked with hats to indicate that they are NOT the TLM variables. They do not even have the same units as the TLM variables. Rather, $\hat{\mathbf{x}}$ is better interpreted as the "sensitivity" of the cost function to the variable $\mathbf{x}$.

The adjoint operator is defined as:

$$< \mathbf{Lx}, \mathbf{y} >=< \mathbf{x}, \mathbf{L}^{*}\mathbf{y} > .$$

The adjoint operator is defined with respect to a particular linear operator, **L**, and a particular inner product, <>.

Now, for the Euclidean norm, $< \mathrm{a}, \mathrm{b} >= \mathbf{a}^{\mathrm{T}}\mathbf{b}$, then

$$< \mathbf{Lx}, \mathbf{y} >= \mathbf{x}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{y} =< \mathbf{x}, \mathbf{L}^{\mathrm{T}}\mathbf{y} > .$$

Thus, for the Euclidean norm,

$$\mathbf{L}^{*} = \mathbf{L}^{\mathrm{T}}.$$

For a particular subroutine of a TLM, each operation can be viewed as a matrix operation. Then the ADJ routine will involve taking the transpose of each matrix operation, but in the reverse order. Writing ADJ code may seem mysterious, but it is actually very mechanical (so compilers can be written to do this) and simple, once you've done a few examples. For a large model, an automatic procedure is recommended. Even for small models, adjoint compilers or tools such as AMC by Ralf Geiring could be useful. Older versions of AMC are available for free from Ralf, but newer versions are not.

7.3.2 Adjoint coding examples

Although I recommend using automatic routines to obtain adjoint code, it is sometimes simpler to do it by hand. The other main advantage of knowing how to write adjoint code is that it is no longer seen as mysterious. Furthermore, you would be able to check the output of automatic tools (which require some knowledge anyway. They cannot be used as black boxes).

The basic recipe for writing adjoint code line-by-line is to start with the TLM code and work module by module, testing each bit as you go. The general pattern is to copy the TLM code, reverse the order of steps, but keep the main trajectory calculation at the beginning, as in the TLM code. Then you simply transpose each statement. The idea of transposing statements will become obvious after a few examples.

Example 7.1 *DO LOOP 130*

```
      DO 130 I=1,N-1
  130 X(I)=a*Y(I+1)
```

Here X and Y are N dimensional vectors. The matrix form of the above DO LOOP is

$$\begin{pmatrix} X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{pmatrix} = \begin{pmatrix} 0 & a & 0 & \cdots & 0 & 0 \\ 0 & 0 & a & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a \end{pmatrix} \begin{pmatrix} Y(1) \\ Y(2) \\ \vdots \\ Y(N) \end{pmatrix} \tag{7.39}$$

Note that the matrix is rectangular with dimension $(N-1)\times N$. *The adjoint of this equation is easily obtained by transposing the matrix:*

$$\begin{pmatrix} Y(1) \\ Y(2) \\ \vdots \\ Y(N) \end{pmatrix} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ a & 0 & \cdots & 0 \\ 0 & a & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & a \end{pmatrix} \begin{pmatrix} X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{pmatrix} \tag{7.40}$$

The equivalent code is simply

```
      DO 130 I=1,N-1
  130 Y(I+1)=a*X(I)
```

Example 7.2 *DO LOOP 140*

```
      DO 140 I=1,N-1
  140 X(I)=X(I)+a*Y(I+1)
```

In matrix form, this DO LOOP is

$$\begin{pmatrix} X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & a & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 & a & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & & & & & & \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & a \end{pmatrix} \begin{pmatrix} X(1) \\ \vdots \\ X(N-1) \\ Y(1) \\ \vdots \\ Y(N) \end{pmatrix} \tag{7.41}$$

Note that the matrix is rectangular with dimension $(N-1)\times 2N-1$. *The adjoint of this equation is easily obtained by transposing the matrix:*

$$\begin{pmatrix} X(1) \\ \vdots \\ X(N-1) \\ Y(1) \\ \vdots \\ Y(N) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 0 \\ a & 0 & 0 & \cdots & 0 & 0 \\ 0 & a & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a \end{pmatrix} \begin{pmatrix} X(1) \\ X(2) \\ \vdots \\ X(N-1) \end{pmatrix} \tag{7.42}$$

The equivalent code is simply

```
      DO 140 I=1,N-1
  140 Y(I+1)=a*X(I)
      DO 141 I=1,N-1
  141 X(I)=X(I)
```

Of course the second DO LOOP could be omitted. Then the result of both examples 1 and 2 are identical. The adjoint of LOOP 130 has two solutions:

```
      DO 1000 I=1,N-1
 1000 Y(I+1)=a*X(I)
    or
      DO 2000 I=1,N-1
 2000 Y(I+1)=a*X(I) + Y(I+1)
```

If the values of Y(I) are reused after the DO loop 130 in example 1 in the TLM, the adjoint of the DO LOOP 130 should be DO LOOP 2000. On the other hand, if the values of Y(I) are not reused after DO LOOP 130 in the TLM, the adjoint loop should be 1000.

Example 7.3 *Recursion*

```
      DO 121 I=2,N
  121 X(I)=b*X(I-1)
```

To get the adjoint of LOOP 121, the order of the loop must be reversed. This can be done in general, but is not necessary when the order doesn't matter, as in the above examples. The adjoint is then

```
      DO 121 I=N,2,-1
  121 X(I-1)=X(I-1)+b*X(I)
```

Example 7.4 *Find the adjoint of the following TLM code:*

```
      DO 120 K=1,3
  120 D(K+1)=D(K)+A(K)*C(K)+E*B(K)
```

where A(K),B(K) are constants. The adjoint code is

```
      E = 0.0
      DO 101 K=1,3
  101 C(K)=0.0
      DO 120 K=3,1,-1
  120 D(K+1)=D(K)+A(K)*C(K)+E*B(K)
         E = E+B(K)*D(K+1)
         C(K) = C(K) + A(K)*D(K+1)
         D(K) = D(K) + D(K+1)
  120 CONTINUE
```

The initialization of E and C(K) is not needed if these variables exist in the adjoint code preceding loop 120.

7.3.3 Tangent Linear Hypothesis

In the 4DVAR algorithm, a TLM was defined (but never directly used). What does this linearization imply? By using a TLM, we are saying that arbitrary perturbations to the initial state evolve approximately linearly in time. This assumption can and should be tested before applying 4DVAR to any given problem.

For a nonlinear Model (NLM),

$$\frac{d\mathbf{x}}{dt} = \mathrm{M}(\mathbf{x}),$$

if $\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}$ satisfies the NLM then to first order:

$$\frac{d(\delta\mathbf{x})}{dt} = \frac{\partial \mathrm{M}(\bar{\mathbf{x}})}{\partial \mathbf{x}}\delta\mathbf{x} = \mathbf{M}(\bar{\mathbf{x}})\delta\mathbf{x}.$$

On the other hand, the nonlinear evolution of perturbations is given by

$$\mathrm{M}(\bar{\mathbf{x}} + \delta\mathbf{x}) - \mathrm{M}(\bar{\mathbf{x}}).$$

When the nonlinear evolution is similar to the linear evolution of perturbations, the tangent linear hypothesis is valid. Thus, the TLM is valid when:

$$\frac{||\mathrm{M}(\bar{\mathbf{x}} + \delta\mathbf{x}) - \mathrm{M}(\bar{\mathbf{x}}) - \mathbf{M}(\bar{\mathbf{x}})\delta\mathbf{x}||}{||\mathbf{M}(\bar{\mathbf{x}})\delta\mathbf{x}||} \ll 1.$$

The nonlinear evolution of perturbations involves second and higher order terms. These should be small compared to the linear term if the tangent linear hypothesis is to be valid.

The range of validity of a TLM must be tested for each particular problem. For NWP forecast models, the TLM was found to be valid for 2 days for an f-plane shallow water model (Lacarra and Talagrand 1988), 2 days for a T21L19 primitive equations model on a sphere (Rabier and Courtier 1992) and 1-1.5 days for a mesoscale primitive equations model (Vukicevic 1991).

Clearly, the assimilation period for 4DVAR must not exceed the range of validity of the TLM. The assimilation period must be chosen so that perturbations evolve approximately linearly. If the assimilation period is too long, the final state will be too sensitive to small perturbations in the initial state and we cannot uniquely determine an initial perturbation that moves the state closer to observations.

In choosing as assimilation period, does the principle, "the shorter the better" apply? Interestly, this is NOT the case. There is an optimal assimilation period for a given problem. This was demonstrated by Tanguay et al. (1995) using a barotropic vorticity equation model. In this model, there was a downscale energy cascade so that defining large scales of the model state would lead to some recovery of smaller scales that were not observed simply through the downscale cascade of information. Choosing a very short assimilation period does not allow for nonlinear scale interactions so that the recovery of smaller, unobserved scales is limited.

7.4 A scalar 4DVAR example

Consider the simple example illustrated in Fig. 7.2. The assimilation period is 8 time steps. There is a single observation at time step 2. The background is of course available at the initial time, time step 0. What is the analysis at time step 2?

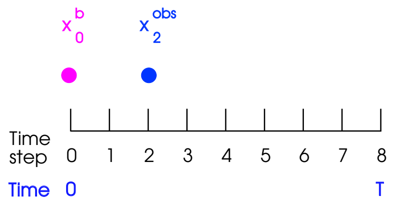

Figure 7.2: A scalar 4DVAR example

The scalar 4DVAR cost function is:

$$J(x_0) = \frac{1}{2}\frac{(x_0 - x_0^b)^2}{\sigma_b^2} + \frac{1}{2}\sum_{k=0}^{N}\frac{(x_k - x_k^{\text{obs}})^2}{\sigma_r^2}.$$

Since there is only 1 observation at time step 2, this simplifies to

$$J(x_0) = \frac{1}{2}\frac{(x_0 - x_0^b)^2}{\sigma_b^2} + \frac{1}{2}\frac{(x_2 - x_2^{\text{obs}})^2}{\sigma_r^2}.$$

The minimum occurs when J$\prime(x_0) = 0$. i.e. when

$$\frac{(x_0 - x_0^b)}{\sigma_b^2} + \frac{(x_2 - x_2^{\text{obs}})}{\sigma_r^2}\frac{dx_2}{dx_1}\frac{dx_1}{dx_0} = 0. \tag{7.43}$$

In order to determine, dx_2/dx_1 and dx_1/dx_0, we must define a specific forecast model. Therefore assume a simple, linear forecast model of the form:

$$\frac{dx}{dt} = -\gamma x.$$

This model can represent the chemical decay of a species, i.e. $x(t) = x(0)\mathrm{e}^{-\gamma t}$. Discretize using an upstream scheme:

$$x(t + \Delta t) - x(t) = \Delta t[-\gamma x(t + \Delta t)]$$

which leads to:

$$x(t + \Delta t) = \left(\frac{1}{1 + \gamma\Delta t}\right)x(t)$$

or

$$x_{k+1} = M(x_k) = cx_k. \tag{7.44}$$

where

$$c = \left(\frac{1}{1 + \gamma\Delta t}\right).$$

For the scalar, linear forecast model, $x_{k+1} = cx_k$, (7.43) becomes

$$\frac{(x_0 - x_0^b)}{\sigma_b^2} + \frac{(x_2 - x_2^{\text{obs}})}{\sigma_r^2}c^2 = 0. \tag{7.45}$$

Solving for x_0 and then deducing x_2 from (7.44) gives

$$x_0 = x_b + \frac{c^2\sigma_b^2}{c^4\sigma_b^2 + \sigma_r^2}[x_2^{\text{obs}} - c^2 x_0^b].$$

$$x_2 = c^2 x_b + \frac{c^4\sigma_b^2}{c^4\sigma_b^2 + \sigma_r^2}[x_2^{\text{obs}} - c^2 x_0^b].$$

Consider some special cases. Case 1: if $\gamma \to 0$, then c $\to$ 1. The model becomes a steady-state model, $x_{k+1} = x_k$. The solution then becomes the 3D solution, namely, $x_0 = x_b + \frac{\sigma_b^2}{\sigma_b^2+\sigma_r^2}[x_2^{\text{obs}} - x_0^b]$. If the model is steady state, one can simply use all observations whatever time they occur, as in the 3D case.

Another extreme is when $\gamma \to \infty$ and c $\to$ 0. The model then becomes $x_{k+1} = 0$ with the initial condition, $x_0 = x_b$. There is no dynamical link between states at two different timesteps. The initial chemical is depleted so quickly that it is removed by the first time step. Thereafter, its concentration is nil. Now, if the observation is perfect, σ_r^2=0, $x_2 = x_2^{\text{obs}}$ but x_0 is undefined. The information at x_0 cannot be spread backward in time because there is no dynamical connection between states at two different times.

7.4.1 A scalar Kalman filter example

Now for comparison, let us redo the simple scalar example with a Kalman filter. The stochastic-Dynamic model is

$$\begin{aligned} x_{k+1}^t &= M(x_k^t) + \epsilon_k^q \\ x_k^{\text{obs}} &= x_k^t + \epsilon_k^r \end{aligned}$$

where ϵ_k^q is N(0,σ_q^2), ϵ_k^r is N(0,σ_r^2), $x_0^t - x_0^b$ is N(0,σ_b^2)

The Kalman Filter steps are then:

1) Forecast step:

$$\begin{aligned} x_{k+1}^f &= M(x_k^a) = c x_k \\ P_{k+1}^f &= c^2 P_k^a + \sigma_q^2 \end{aligned}$$

2) Analysis step:

$$\begin{aligned} x_k^a &= x_k^f + K(x_k^{\text{obs}} - x_k^f) \\ K &= P_k^f (P_k^f + \sigma_r^2)^{-1} \\ P_k^a &= (1-K)P_k^f = (\frac{1}{P_k^f} + \frac{1}{\sigma_r^2})^{-1} \end{aligned}$$

3) Initial conditions:

$$\begin{aligned} x_0^a &= x_0^b \\ P_0^a &= \sigma_b^2 \end{aligned}$$

Now, let us begin at $k = 0$.

Step 1: Forecast, k=0

$$\begin{aligned} x_1^f &= M(x_0^a) = cx_0^b \\ P_1^f &= c^2\sigma_b^2 + \sigma_q^2 \end{aligned}$$

Step 2: Analysis, k=1 (No obs, K = 0)

$$\begin{aligned} x_1^a &= x_1^f = cx_0^b \\ P_1^a &= P_1^f = c^2\sigma_b^2 + \sigma_q^2 \end{aligned}$$

Step 3: Forecast, k=1

$$\begin{aligned} x_2^f &= M(x_1^a) = c^2 x_0^b \\ P_2^f &= c^2 P_1^a + \sigma_q^2 = c^4\sigma_b^2 + (c^2+1)\sigma_q^2 \end{aligned}$$

Step 4: Analysis, k=2

$$\begin{aligned} x_2^a &= x_2^f + K(x_2^{\text{obs}} - x_2^f) \\ K &= P_2^f(P_2^f + \sigma_r^2)^{-1} = \frac{c^4\sigma_b^2 + (c^2+1)\sigma_q^2}{c^4\sigma_b^2 + (c^2+1)\sigma_q^2 + \sigma_r^2} \\ P_2^a &= (\frac{1}{P_2^f} + \frac{1}{\sigma_r^2})^{-1} \end{aligned}$$

To summarize, the solution is:

$$x_2^a = c^2 x_0^b + \frac{c^4\sigma_b^2 + (c^2+1)\sigma_q^2}{c^4\sigma_b^2 + (c^2+1)\sigma_q^2 + \sigma_r^2}(x_2^{\text{obs}} - c^2 x_0^b)$$

Case 1: Perfect model: $\sigma_q^2 = 0$, KF and 4DVAR solutions same at k=2

Case 2: $\gamma \to 0$, c $\to$ 1, $\to$ 3D sol'n: $x_2^a = x_b + \frac{\sigma_b^2 + 2\sigma_q^2}{\sigma_b^2 + 2\sigma_q^2 + \sigma_r^2}[x_2^{\text{obs}} - x_0^b]$.

Case 3: $\gamma \to \infty$, c $\to$ 0, $x_2^a = \frac{\sigma_q^2}{\sigma_q^2 + \sigma_r^2} x_2^{\text{obs}}$. If model is perfect, $\sigma_q^2 = 0$ and $x_2^a = 0$. If obs is perfect, $\sigma_r^2 = 0$ and $x_2^a = x_2^{\text{obs}}$.

7.5 4DVAR background error covariances

We have earlier claimed that 4DVAR allows for flow dependent influence of the data. This in turn can allow for the anisotropic, spreading of information and also the analysis of small scales. With 3D schemes, columns of the background error covariance matrix were plotted to get an idea of how the information of an observation is spread. However, in 4DVAR, we are actually running the forecast model forward in time and the adjoint model backward in time to spread the information of the observations. Thus we cannot simply examine the $\mathbf{P}^f$ matrix, which is modelled for stationary processes. However, we can see the influence of the data directly by running 4DVAR with a single observation.

Recall that 3DVAR solves:

$$\mathbf{x}^a = \mathbf{x}^b + \mathbf{P}\mathbf{H}^{\text{T}}(\mathbf{H}\mathbf{P}\mathbf{H}^{\text{T}} + \mathbf{R})^{-1}[\mathbf{z} - H(\mathbf{x}^b)]$$

Then for a single observation, $H = h$, a scalar function. Let $\mathbf{h} = dh/d\mathbf{x}$. In this case, $\mathbf{R}$ is a scalar, σ_r^2, and $\mathbf{HPH}^{\mathrm{T}} = \sigma_b^2$ is also a scalar. σ_b^2 is the background error variance at the observation location. The analysis increment can then be written as

$$\mathbf{x}^a - \mathbf{x}^b = \mathbf{Ph}^{\mathrm{T}} \left(\frac{z - h(\mathbf{x}^b)}{\sigma_b^2 + \sigma_r^2} \right). \tag{7.46}$$

In 4DVAR, H includes transport from t_0 to t_N by the TLM, $\mathbf{M}$:

$$\mathbf{x}^a(t_0) - \mathbf{x}^b(t_0) = \mathbf{PM}^{\mathrm{T}}\mathbf{h}^{\mathrm{T}} \left(\frac{z - h(\mathbf{x}^b(t_N))}{\sigma_b^2 + \sigma_r^2} \right) \tag{7.47}$$

where $\sigma_b^2 = \mathbf{hMPM}^{\mathrm{T}}\mathbf{h}^{\mathrm{T}}$. σ_b^2 is the background error variance at the time, location and variable of the observation. At t_N the analysis increments are:

$$\begin{aligned} \mathbf{x}^a(t_N) - \mathbf{x}^b(t_N) &= \mathbf{M}(\mathbf{x}^a(t_0) - \mathbf{x}^b(t_0)) \\ &= \mathbf{MPM}^{\mathrm{T}}\mathbf{h}^{\mathrm{T}} \left(\frac{z - h(\mathbf{x}^b(t_N))}{\sigma_b^2 + \sigma_r^2} \right). \end{aligned} \tag{7.48}$$

Note that $\mathbf{MPM}^{\mathrm{T}}$ appears in the direct propagation of the forecast error covariance matrix in the Kalman filter and represents the evolution of initial errors through the model dynamics. Thus in 4DVAR, we implicitly evolve the forecast error covariances through the assimilation period. However, because this calculation is not done explicitly, we do not have the updated covariance matrix at the end of the 4DVAR assimilation. Note also that unlike the Kalman filter, no model error term ($\mathbf{Q}$) is involved since the model was assumed to be perfect.

Thépaut et al. (1996) show some very nice examples of structure functions (the spreading of information of a single observation) in 4DVAR. Compared to 3DVAR, 4DVAR is shown to permit flow dependent spreading of information of observations. In particular, in a frontal zone, an observation can influence the analysis in a complex, dynamically consistent way. The analysis can then retain smaller scales compared to a 3DVAR analysis. As the assimilation period is decreased, there is less time for the flow to advect the initial 3DVAR covariances $\mathbf{P}^b$ so that the structure functions more closely resemble those of 3DVAR.

7.6 The conditional density viewpoint

Let us consider 4DVAR from a conditional density viewpoint. The problem is: what is the best estimate of a sequence of N model states, $\mathbf{X}_k = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k)^{\mathrm{T}}$, given that the observation sequence, $\mathbf{Z}_k = (\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_k)^{\mathrm{T}}$, has occurred? Let us assume that the true system obeys

$$\mathbf{x}_{k+1} = \boldsymbol{\phi}_k \mathbf{x}_k + \mathbf{w}_k, \tag{7.49}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k. \tag{7.50}$$

$\mathbf{w}_k$ is the model error and $\mathbf{v}_k$ is the observation error. The n-vector model state at time t_k is $\mathbf{x}_k$, while the observed state is $\mathbf{z}_k$, an m-vector. The transition matrix is $\boldsymbol{\phi}_k$ and the observation operator is $\mathbf{H}_k$. The model error and observation errors are assumed unbiased, white and independent of each other. i.e. For all k, l,

$$\begin{aligned} <\mathbf{w}_k>&=0, & <\mathbf{w}_k(\mathbf{w}_l)^{\mathrm{T}}>&=\mathbf{Q}_k\delta_l^k \\ <\mathbf{v}_k>&=0, & <\mathbf{v}_k(\mathbf{v}_l)^{\mathrm{T}}>&=\mathbf{R}_k\delta_l^k \\ <\mathbf{w}_k(\mathbf{v}_l)^{\mathrm{T}}>&=0. \end{aligned} \tag{7.51}$$

The observation, model and initial state errors are all assumed to be Gaussian.

As we know, we would like to estimate the conditional density of the trajectory $\mathbf{X}_k$ (during the assimilation period, $[t_0, t_N]$) given the observed trajectory, $\mathbf{Z}_k$:

$$p_{\mathbf{X}_k|\mathbf{Z}_k}(\mathbf{X}_k|\mathbf{Z}_k).$$

Let's write this p.d.f. in terms of quantities we can evaluate.

$$p_{\mathbf{X}_k|\mathbf{Z}_k}(\mathbf{X}_k|\mathbf{Z}_k) = \frac{p_{\mathbf{Z}_k|\mathbf{X}_k}(\mathbf{Z}_k|\mathbf{X}_k)p_{\mathbf{X}_k}(\mathbf{X}_k)}{p_{\mathbf{Z}_k}(\mathbf{Z}_k)}. \tag{7.52}$$

Now

$$\begin{aligned} p(\mathbf{Z}_k|\mathbf{X}_k) &= p(\mathbf{z}_k, \mathbf{Z}_{k-1}|\mathbf{X}_k) \\ &= p(\mathbf{z}_k|\mathbf{X}_k)p(\mathbf{Z}_{k-1}|\mathbf{X}_k) \\ &= p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{X}_{k-1})p(\mathbf{Z}_{k-1}|\mathbf{x}_k, \mathbf{X}_{k-1}) \\ &= p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{Z}_{k-1}|\mathbf{X}_{k-1}) \end{aligned} \tag{7.53}$$

where we used the fact that the observation error is white in time to get the 2nd and 4th lines. Thus $p(\mathbf{Z}_{k-1}|\mathbf{X}_{k-1})$ can be expanded by recursively applying (7.53):

$$p(\mathbf{Z}_k|\mathbf{X}_k) = p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{z}_{k-1}|\mathbf{x}_{k-1})\dots p(\mathbf{z}_0|\mathbf{x}_0) \tag{7.54}$$

Also, we can expand

$$\begin{aligned} p(\mathbf{X}_k) &= p(\mathbf{x}_k, \mathbf{X}_{k-1}) = p(\mathbf{x}_k|\mathbf{X}_{k-1})p(\mathbf{X}_{k-1}) \\ &= p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{X}_{k-1}). \end{aligned} \tag{7.55}$$

The assumption that $\mathbf{x}_k$ is Markov was used to obtain the second line. Applying (7.55) recursively yields

$$\begin{aligned} p(\mathbf{X}_k) &= p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{x}_{k-2})\dots p(\mathbf{x}_1|\mathbf{x}_0)p(\mathbf{X}_0) \\ &= p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{x}_{k-2})\dots p(\mathbf{x}_1|\mathbf{x}_0)p(\mathbf{x}_0) \end{aligned} \tag{7.56}$$

Because the observation, model and initial state errors are Gaussian we can write the complete p.d.f.s we require. Then from chapter 4 (Todling p. 65) we have that the conditional p.d.f. (7.54) is

$$p(\mathbf{z}_k|\mathbf{x}_k) \sim N(\mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)$$

so that

$$p(\mathbf{Z}_k|\mathbf{X}_k) \sim N(\mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)N(\mathbf{H}_{k-1}\mathbf{x}_{k-1}, \mathbf{R}_{k-1})\dots N(\mathbf{H}_0\mathbf{x}_0, \mathbf{R}_0).$$

Similarly, we can deduce that

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) \sim N(\boldsymbol{\phi}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1})$$

so that

$$p(\mathbf{X}_k) \sim N(\boldsymbol{\phi}_{k-1}\mathbf{x}_{k-1}, \mathbf{Q}_{k-1})N(\boldsymbol{\phi}_{k-2}\mathbf{x}_{k-2}, \mathbf{Q}_{k-2}) \ldots N(\boldsymbol{\phi}_0\mathbf{x}_0, \mathbf{Q}_0)N(\hat{\mathbf{x}}_0^f, \mathbf{P}_0^f).$$

The numerator of the *a posteriori* p.d.f. (7.52) can thus be written as:

$$\begin{aligned} p_{\mathbf{Z}_k|\mathbf{X}_k}(\mathbf{Z}_k|\mathbf{X}_k)p_{\mathbf{X}_k}(\mathbf{X}_k) &\propto \exp\Bigg\{-\frac{1}{2}\sum_{k=0}^{N}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k)^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k) \\ &- \frac{1}{2}\sum_{k=0}^{N-1}(\mathbf{x}_k - \boldsymbol{\phi}_{k-1}\mathbf{x}_{k-1})^{\mathrm{T}}\mathbf{Q}_{k-1}^{-1}(\mathbf{x}_k - \boldsymbol{\phi}_{k-1}\mathbf{x}_{k-1}) \\ &- \frac{1}{2}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f)^{\mathrm{T}}(\mathbf{P}_0^f)^{-1}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f)^{\mathrm{T}}\Bigg\}. \end{aligned} \tag{7.57}$$

To maximize (7.52) with respect to $\mathbf{X}_k$, we can simply maximize the numerator of (7.52) and this is equivalent to minimizing the negative of the argument of the exponent in the above. Thus, the mode estimation of the *a posteriori* p.d.f., for Gaussian errors leads to the minimization of the cost function:

$$\begin{aligned} J(\mathbf{X}_k) &= \frac{1}{2}\sum_{k=0}^{N}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k)^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k) \\ &+ \frac{1}{2}\sum_{k=0}^{N-1}(\mathbf{x}_k - \boldsymbol{\phi}_{k-1}\mathbf{x}_{k-1})^{\mathrm{T}}\mathbf{Q}_{k-1}^{-1}(\mathbf{x}_k - \boldsymbol{\phi}_{k-1}\mathbf{x}_{k-1}) \\ &+ \frac{1}{2}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f)^{\mathrm{T}}(\mathbf{P}_0^f)^{-1}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f)^{\mathrm{T}}. \end{aligned} \tag{7.58}$$

(7.58) is the general 4DVAR cost function. The control variable is $(N+1)n$ where $N+1$ is the number of time steps in the assimilation interval and n is the dimension of the model state. If n is typically O(10^7) and N is O(10), clearly, this problem is too big to solve. However, if the forecast model is very good, then the time evolution of the true state is known:

$$\mathbf{x}_{k+1} = \boldsymbol{\phi}_k\mathbf{x}_k.$$

Thus all states, $\mathbf{x}_k$ can be related to the initial state $\mathbf{x}_0$ using this relationship. The number of unknowns is thus reduced to simply n- the same as in the case of 3DVAR, but now we have observations in the whole assimilation period. For the case of a perfect model, $\mathbf{w}_k$=0 in (7.49) and (7.58) reduces to:

$$\begin{aligned} J(\mathbf{x}_0) &= \frac{1}{2}\sum_{k=0}^{N}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k)^{\mathrm{T}}\mathbf{R}_k^{-1}(\mathbf{z}_k - \mathbf{H}_k\mathbf{x}_k) \\ &+ \frac{1}{2}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f)^{\mathrm{T}}(\mathbf{P}_0^f)^{-1}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f). \end{aligned} \tag{7.59}$$

The minimization problem (7.58) is called weak constraint 4DVAR, while (7.59) is the 4DVAR problem using the forecast model as a strong constraint. It is 4DVAR with a strong constraint (perfect model assumption) that was implemented operationally at ECMWF (European Center for Medium Range Weather Forecasting) in 1997 and is planned to be implemented operationally at CMC (Canadian Meteorological Centre).

An interesting thing about 4DVAR is that the minimization process requires the cost function and its gradient with respect to the state vector. Thus for nonlinear models and observation operators we need the derivatives of the transition matrix and $\mathbf{H}_k$ with respect to the state. These derivatives are called the "Tangent Linear" Models (TLMs). The adjoint of this linear operator, with respect to a given inner product is called the "adjoint" model. The adjoint of the Tangent Linear Forecast model runs backward in time, but is not an inverse model. Rather, the adjoint model is the derivative of J with respect to the initial state. Or, the adjoint model can be viewed as the sensitivity (as measured by a particular cost function) of the model to the initial state.

Note that 3DVAR is a special case of 4DVAR, for which $N{=}0$ in (7.59). Thus observations are available only at the same time as the analysis and background fields.

REFERENCES

1. Klinker, E., F. Rabier, G. Kelly, J.-F. Mahfouf, 2000: The ECMWF operational implementation of four dimensional variational assimilation. Part III: Experimental results and diagnostics with operational configuration. *Q. J. R. Meteorol. Soc.*, **126**, 1191-1215.

2. Lacarra, J.-F., and O. Talagrand, 1988: Short-range evolution of small perturbations in a barotropic model. *Tellus*, **40**, 91-95.

3. Ledimet, F. X. and O. Talagrand, 1986: Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects. *Tellus*, **38A**:97-110.

4. Lewis, J., 1990: Introduction to the adjoint method of data assimilation: Parts I and II. In *Mesoscale Data Assimilation*, 1990 NCAR Summer Colloquium, p. 297-319.

5. Platzman, G. W., 1964: An exact integral of complete spectral equations for unsteady one-dimensional flow. *Tellus*, **16**: 422-431.

6. Rabier, F. and Ph. Courtier, 1992: Four-dimensional assimilation in the presence of baroclinic instability, *Q. J. R. Meteorol. Soc.*, **118**(506), 649-672.

7. Rabier, H. Järvinen, E. Klinker, J.-F. Mahfouf, A. Simmons, 2000: The ECMWF operational implementation of four dimensional variational assimilation. Part I: Experimental results with simplified physics. *Q. J. R. Meteorol. Soc.*, **126**, 1143-1170.

8. Tanguay, M., P. Bartello, P. Gauthier, 1995: Four-dimensional data assimilation with a wide range of scales *Tellus*, **47A**, 974-997.

9. Thacker, W. C. and B. R. Long, 1988: Fitting dynamics to data. *J. Geophys. Res.*, **93**:1227-1240.

10. Thépaut, J.-N., P. Courtier, G. Belaud and G. Lemaître, 1996: Dynamical structure functions in a four-dimensional variational assimilation: A case study. *Q. J. R. Meteorol. Soc.*, **122**, 535-561.

11. Vukicevic, T., 1991: Nonlinear and linear evolution of initial forecast errors. *Mon. Wea. Rev.*, **119**(7), 1602-1611.

7.7 Problem Set 7

1. (a) Consider the 3-component Lorenz equations:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y-x), \\ \frac{dy}{dt} &= \rho x - y - xz, \\ \frac{dz}{dt} &= xy - \beta z, \end{aligned} \tag{7.60}$$

where σ, ρ and β are real constants. The cost function which defines the misfit between model state and observations is given by

$$J = \frac{1}{2}[(x - x^{obs})^2 + (y - y^{obs})^2 + (z - z^{obs})^2.]$$

What is the adjoint system of equations? First write the Tangent Linear system, then take the transpose of the transition matrix and add forcing by the gradient of the cost function, J.

(b) Consider the nonlinear advection-diffusion in one-dimension:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0 \tag{7.61}$$

over the domain $[0, L] \times [0, T]$. The cost function is given by:

$$J = \frac{1}{2}\int_0^T \int_0^L (u - u^{obs})^2 dx dt.$$

We are now going to derive the adjoint equation following the control theory method of section 7.2.2.

(i) Write the TLM equation.
(ii) Write the expression for δJ.
(ii) Multiply the TLM equation by an arbitrary variable p(x,t) and integrate over the domain in x and t. For the time derivative term, integrate by parts first in time. For the x derivative term, integrate by parts in x first. For the term involving a second derivative, you will need to integrate by parts twice. Eliminate some terms using the following boundary conditions on p:

$$p(x,T) = 0, \qquad p(0,t) = p(L,t), \qquad \frac{\partial p}{\partial x}(0,t) = \frac{\partial p}{\partial x}(L,t)$$

Define the adjoint equation including forcing by the gradient of J.

2. Find the adjoint of **dum.m** by reversing the code. Test the adjoint routine and verify that it is correct to at least 14 digits. You will receive **dum.m** by email, but here are the contents of this file:

```
function y=dum(x,n)

y=zeros(1,n);
z=zeros(1,2*n);

for i=1:n-1
   y(i)=x(i+1);
end

for i=1:n
   z(i)=x(i);
end

for i=1:n
   z(i+n)=x(i);
end

for i=1:n
   y(i)=y(i) + 11*z(i) + 6*z(n+i);
end
```

3. Scalar 4DVAR. Consider the following scalar system:

$$\mathrm{x}_{k+1} = m\mathrm{x}_k + \mathrm{w}_k \tag{7.62}$$

$$\mathrm{z}_k = \mathrm{H}\mathrm{x}_k + \mathrm{v}_k \tag{7.63}$$

Let P_0 be the initial state error variance and R be the obs error variance. Then the cost function is given by:

$$J(\mathrm{x}_0) = \frac{1}{2\mathrm{P}_0}(\mathrm{x}_0 - \mathrm{x}_0^f)^2 + \frac{1}{2\mathrm{R}}\sum_{k=1}^{K}(\mathrm{z}_k - \mathrm{H}\mathrm{x}_k)^2.$$

The gradient is given by

$$\nabla_{\mathrm{x}_0} J = \frac{1}{\mathrm{P}_0}(\mathrm{x}_0 - \mathrm{x}_0^f) - \sum_{k=1}^{K} m_0 m_1 \ldots m_{k-1}\mathrm{H}\mathrm{R}^{-1}(\mathrm{z}_k - \mathrm{H}\mathrm{x}_k)$$

where we used the fact that the transpose of a scalar is the scalar itself.

In this problem, we will write a MATLAB script to run the 4DVAR scheme. The assimilation period will be timesteps.

1. What is the Hessian of J? You may use the fact that m is independent of timestep k.

2. Show by hand that the adjoint test is passed. In other words, let x be the model input and y be the model output after k time steps, while $\hat{x}$ is the adjoint model output and $\hat{y}$ is the adjoint model input. Then show that

$$< x, \hat{x} >=< y, \hat{y} > .$$

3. Here are some steps to help you write the 4DVAR script. Note that the first 2 steps (initialization statements) are the same as in the scalar KF problem of the last chapter. Note also that the instructions may seem laborious (given that simplifications are possible due to the specific model used) but this will give you the basic 4DVAR coding recipe for any model.

(a) Define parameters and constants.

(b) Define initial conditions for the background and truth.

(c) Loop in time for nk timesteps. First generate the obs by perturbing the truth. Then propagate the truth in time

(d) Compute the cost function. To do this, define a separate function to evaluate the cost given an initial state. This will involve integrating the background state in time.

(e) Compute the gradient.

   i. First copy your script to evaluate the cost function, renaming it to the gradient function.

   ii. Take the derivative of the background term.

   iii. Set the adjoint variable at nk+1 to zero. Set the gradient equal to the adjoint variable at nk+1.

   iv. run the model to get the background trajectory.

   v. loop backwards in time from nk to 1. Within the loop (i) add the contribution to the gradient for time step k, then (ii) run the adjoint model backward one timestep using the current gradient as the initial condition. Overwrite the gradient with the output.

(f) Obtain the solution using Newton's method. Use the Hessian obtained in (a). Because the cost function is purely quadratic, Newton's method will obtain the minimum in 1 iteration.

(g) Integrate the solution to get the final trajectory (the analysis).

(h) Plot background, true and obs as a function time step. Plot the actual error ($|\mathrm{x}_k^a - \mathrm{x}_k^t|$) as a function of time.

(i) Be prepared to hand-in a printout of your code, or email it to Lisa.

4. Run the 4DVAR script for R=H=P$_0$=m=1. Let Q=0 and nk=50. Explain the results. Now try Q=1. Explain what is happening in the plots and how the solution differs from the scalar KF example for the same parameters. Note that the model is assumed to be perfect.

5. Now see what happens when the model is not perfect. When you generate the obs, use $m^t = m + 0.05 = 1.05$. Explain the results. Keep the parameter values used in part 4 with Q=1.

6. Finally, imagine what would happen with a different assimilation period, say np=10. Then there would be 5 cycles during the 50 time steps. (You may assume the model is perfect.) Without actually coding anything (it would be too much work I think), draw a schematic of what you think the plot of analysis, truth and obs would look like. After each cycle, the initial background would be taken from the analysis at the last time step of the previous cycle.

4. MATLAB: Passive advection in 1D using 4DVAR. Let us return to the advection problem of Ch. 3, prob. 4. and Ch. 5 prob. 3. This time we will run 4DVAR instead of a KF or OI algorithm. To run 4DVAR you will need the additional MATLAB scripts: testadj.m, cost.m, gradcost.m, upwindad.m, 4dvar.m.

   (a) Note that the forecast model (for the 1D passive advection) is linear. Thus, there is no need to linearize the model. However, we need an adjoint model for 4DVAR. This is provided in upwindad.m . Once an adjoint is developed, we need to test it to see that it is correct to machine precision. To test the adjoint you will need to run testadj.m. However, you must first complete this code. Two lines near the bottom have been commented out. Uncomment these and complete the equations. Run testadj.m. Provide a hardcopy of your results when you have verified that the adjoint is correct.

   (b) Cost function and gradient. Because our model is linear, the cost function is purely quadratic:

$$J(\mathbf{x}_0) \quad = \quad \frac{1}{2}\sum_{k=0}^{N}(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k)^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k) + (\mathbf{x}_0 - \hat{\mathbf{x}}_0^f)^{\mathrm{T}}(\mathbf{P}_0^f)^{-1}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f). \quad (7.64)$$

   The cost function is coded in cost.m. Read through this code and then complete the missing two lines. The gradient of this cost function is

$$\nabla J(\mathbf{x}_0) \quad = \quad -\sum_{k=0}^{N}\mathbf{M}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\ldots\mathbf{M}_{k-1}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k) + (\mathbf{P}_0^f)^{-1}(\mathbf{x}_0 - \hat{\mathbf{x}}_0^f). \quad (7.65)$$

   where we assumed the model was linear and given by:

$$\begin{aligned}\mathbf{x}_k &= \mathbf{M}_{k-1}\mathbf{x}_{k-1}\\ &= \mathbf{M}_{k-1}\mathbf{M}_{k-2}\ldots\mathbf{M}_0\mathbf{x}_0 \end{aligned} \quad (7.66)$$

   and the adjoint model is

$$\delta\mathbf{x}_{k-1} = \mathbf{M}_{k-1}^{\mathrm{T}}\delta\mathbf{x}_k \quad (7.67)$$

   or

$$\delta\mathbf{x}_0 = \mathbf{M}_0^{\mathrm{T}}\mathbf{M}_1^{\mathrm{T}}\ldots\mathbf{M}_{k-2}^{\mathrm{T}}\mathbf{M}_{k-1}^{\mathrm{T}}\delta\mathbf{x}_k \quad (7.68)$$

   for a Euclidean norm (adjoint = transpose). Read through gradcost.m and complete the missing two lines of code.

   (c) Gradient test. Note that the gradient test is already in the var4d.m code. Run $\boxed{\text{var4d(0,1,0.95,0)}}$. Try various observation frequencies:20, 10, 5, 2, 1. Choose an observation pattern of 1, (obs every grid point), and an obs error of 0.02. Look for the output of the gradient test. Mathematically explain what this test is doing. (Hint: Consider a Taylor expansion of $J(\mathbf{x}+\alpha\delta\mathbf{x})-J(\mathbf{x})$. Then choose $\delta\mathbf{x} = \nabla J(\mathbf{x})$.) Why do we need this test, if we already tested the adjoint for accuracy?

   (d) 4DVAR minimization. Normally, we can use some packaged software (based on a quasi-Newton or conjugate gradient method for example) for the minimization. MATLAB's basic package does not have an optimization routine that also uses the gradient information. However, because our cost function is purely quadratic, we can write our own minimization algorithm based on Newton's method. Read the section below on Newton's method. Newton's method requires knowledge of the Hessian of the cost function. To approximate the Hessian, we simply perturbed the gradient. We approximate the jth column of the Hessian matrix by a finite difference:

$$(J'')e_j = \frac{1}{\alpha}[\nabla J(\mathbf{x}_g + \alpha e_j) - \nabla J(\mathbf{x}_g)] \tag{7.69}$$

where $\mathbf{x}_g$ is the guess or background state. e_j is the jth column of the identity matrix. I used α=10^{-3}.

(e) Run the 4DVAR by typing $\boxed{\text{var4d(0,1,0.95,0)}}$. The Courant number will now be fixed at 0.95 and T_{final}=1 as in the OI problem. The same three questions will be asked. Hitting "return" will give the default. First enter observation frequency of 5 (obs every 5 time steps), an observation sparsity of 1 (obs at every gridpoint), and "return" for the obs error standard deviation. This will give the default value of 0.02. How does the analysis compare with the truth? Note that there is no error estimate. A bit more work is required to come up with an error estimate and this was not done. Unlike the KF, the analysis error covariance matrix is not part of the algorithm for 4DVAR.

(f) Now let's make the problem a little harder. Again type $\boxed{\text{var4d(0,1,0.95,0)}}$, but provide an obs error of 0.2. Keep the obs frequency of 5 and the obs sparsity of 1. Compare your results with those of the Kalman filter.

(g) Now let's see what happens when there are data gaps. Type $\boxed{\text{var4d(0,1,0.95,0)}}$, but answer "return" to all questions. This gives an obs every time step, over the left half of the domain with a std deviation of 0.02. How the analysis fare? Now decrease the observation frequency by typing first 2 then 5 and keeping the obs pattern and error std dev the same as before. Now what happens to the solution? Why? Again compare your results to the KF solution and discuss what you see.

Newton method solution for a quadratic cost function

Consider a quadratic cost function:

$$J(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} + \mathbf{b}^{\mathrm{T}}\mathbf{x} + c. \tag{7.70}$$

The gradient is

$$\nabla J(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}. \tag{7.71}$$

The Hessian or second derivative is

$$J''(\mathbf{x}) = \mathbf{A}. \tag{7.72}$$

Since we want to minimize (7.70), we want to solve for $\nabla J(\mathbf{x}) = \mathbf{0}$. Now for an initial guess $\mathbf{x}_g$, we have that

$$\nabla J(\mathbf{x}_g) = \mathbf{A}\mathbf{x}_g + \mathbf{b} \tag{7.73}$$

while at the solution, $\hat{\mathbf{x}}$, the gradient is zero, i.e.

$$\nabla J(\hat{\mathbf{x}}) = 0 = \mathbf{A}\hat{\mathbf{x}} + \mathbf{b}. \tag{7.74}$$

Subtracting (7.73) - (7.74) yields:

$$\nabla J(\mathbf{x}_g) = \mathbf{A}(\mathbf{x}_g - \hat{\mathbf{x}}). \tag{7.75}$$

Solving for $\hat{\mathbf{x}}$, we obtain

$$\hat{\mathbf{x}} = \mathbf{x}_g - \mathbf{A}^{-1}\nabla J(\mathbf{x}_g) \tag{7.76}$$

or on substituting the Hessian for $\mathbf{A}$:

$$\hat{\mathbf{x}} = \mathbf{x}_g - (J'')^{-1}\nabla J(\mathbf{x}_g). \tag{7.77}$$