

4D-Variational Data Assimilation (4D-Var)

- 4DVAR, according to the name, is a four-dimensional variational method.
- 4D-Var is actually a direct generalization of 3D-Var to handle observations that are distributed in time. The cost function is the same, provided that the observation operators are generalized to include a forecast model that will allow a comparison between the model state and the observations at the appropriate time.
- 4D-Var seeks the *initial condition* such that the forecast best fits the observations within the assimilation interval.

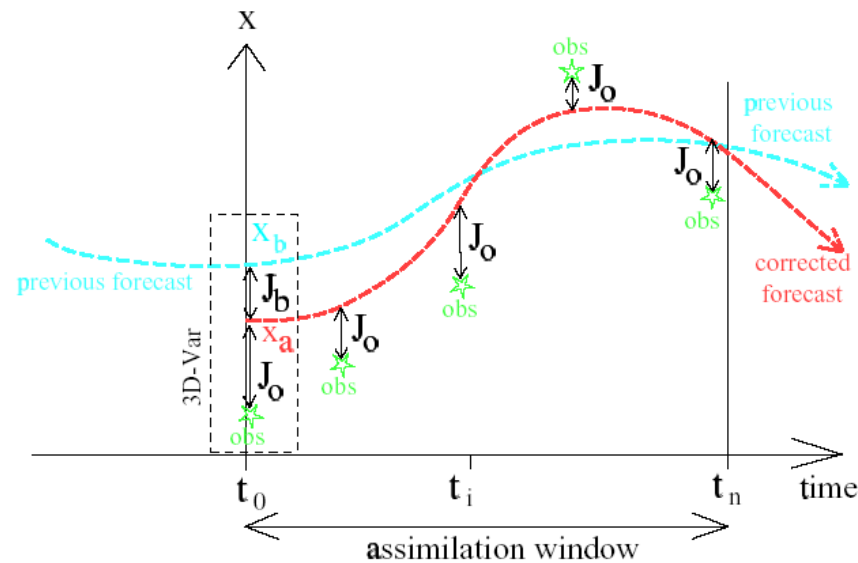


Figure 12. Example of 4D-Var intermittent assimilation in a numerical forecasting system. Every 6 hours a 4D-Var is performed to assimilate the most recent observations, using a segment of the previous forecast as background. This updates the initial model trajectory for the subsequent forecast.

Cost function for 4DVAR

Let $\mathbf{x}^f(t_i) = M_{i-1}[\mathbf{x}^a(t_{i-1})]$ represent the (nonlinear) model forecast that advances from the previous analysis time t_{i-1} to the current t_i .

Assume the observations distributed within a time interval (t_0, t_n) will be used. The cost function includes a term measuring the distance to the background *at the beginning of the interval*, and a summation over time of the cost function for each observational increment computed with respect to the model integrated to the time of the observation:

$$J(\mathbf{x}(t_0)) = \frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b(t_0))^T \mathbf{B}_0^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b(t_0)) + \frac{1}{2} \sum_{i=0}^N (H(\mathbf{x}_i) - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1} (H(\mathbf{x}_i) - \mathbf{y}_i^o) \quad (6.1)$$

where N is the number of observational vectors \mathbf{y}_i^o distributed over time.

- The control variable (the variable with respect to which the cost function is minimized) is the *initial* state of the model at the beginning of the time interval, $\mathbf{x}(t_0)$, whereas the analysis at the end of the interval is given by the *model integration* from the solution $\mathbf{x}(t_n) = M_0[\mathbf{x}(t_0)]$.
- In this sense, the model is used as *a strong constraint*, i.e., the analysis solution has to satisfy the model equations.

- In other words, *4D-Var seeks the initial condition such that the forecast best fits the observations within the assimilation interval.*
- The fact that the 4D-Var method assumes a perfect model is a disadvantage since (for example) it will give the same credence to older observations at the beginning of the interval as to newer observations at the end of the interval.

- The 4DVAR tries to use all observations in the assimilation time interval as well as possible. The following is an example of how information is propagated in time by a simple advection model, so that the observations can be compared with the first guess.

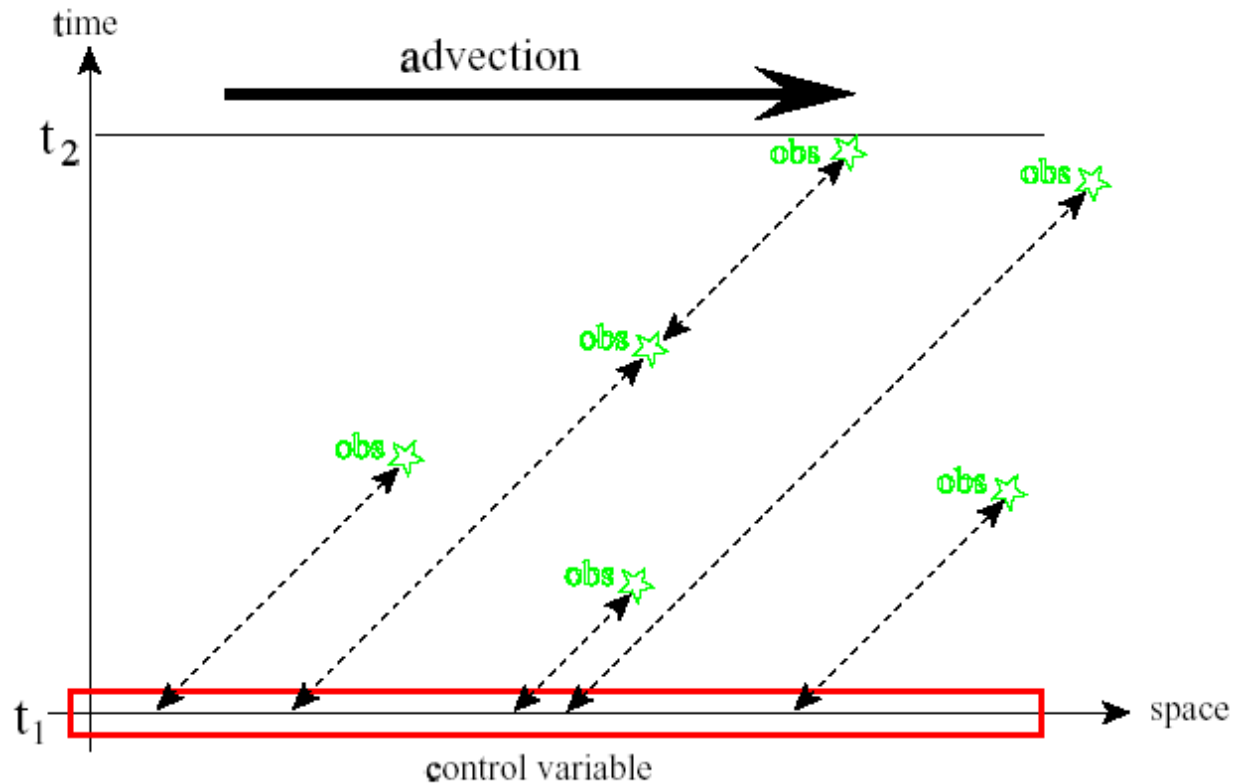


Figure 13. Example of propagation of the information by 4D-Var (or, equivalently, a Kalman filter) in a 1-D model with advection (i.e. transport) of a scalar quantity. All features observed at any point within the 4D-Var time window (t_1, t_2) will be related to the correct upstream point of the control variable by the tangent linear and adjoint model, along the characteristic lines of the flow (dashed).

A variation in the cost function when the control variable $\mathbf{x}(t_0)$ is changed by a small amount $\delta\mathbf{x}(t_0)$ is given by

$$\delta J = J[\mathbf{x}(t_0) + \delta\mathbf{x}(t_0)] - J(\mathbf{x}(t_0)) \approx \frac{\partial J}{\partial x_1} \delta x_1 + \frac{\partial J}{\partial x_2} \delta x_2 + \dots + \frac{\partial J}{\partial x_n} \delta x_n = \left[\frac{\partial J}{\partial \mathbf{x}(t_0)} \right]^T \delta\mathbf{x}(t_0) \quad (6.2)$$

where the gradient of the cost function $\left[\frac{\partial J}{\partial \mathbf{x}(t_0)} \right]_j = \frac{\partial J}{\partial x_j(t_0)}$ is a column vector.

- Iterative minimization schemes require the estimation of the cost function gradient w.r.t. control variables.
- Similar to the 3DVAR case, the steepest descent method is the simplest scheme in which the change in the control variable after each iteration is chosen to be opposite to the gradient $\delta\mathbf{x}(t_0) = -a \nabla_{\mathbf{x}(t_0)} J = -a \frac{\partial J}{\partial \mathbf{x}(t_0)}$.
- Other, more efficient methods, such as conjugate gradient or quasi-Newton also require the use of the gradient.
- To solve this minimization problem efficiently, we need to be able to compute the gradient of J with respect to the elements of the control variable.

Calculation of the gradient of 4DVAR cost function

As we saw earlier, given a symmetric matrix \mathbf{A} and a functional $J = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x}$, the gradient is given by $\frac{\partial J}{\partial \mathbf{x}} = \mathbf{A} \mathbf{x}$.

If $J = \frac{1}{2} \mathbf{y}^T \mathbf{A} \mathbf{y}$, and $\mathbf{y} = \mathbf{y}(\mathbf{x})$, then (by applying the chain rule)

$$\frac{\partial J}{\partial \mathbf{x}} = \left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right]^T \mathbf{A} \mathbf{y} \quad (6.3)$$

where $\left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right]_{k,l} = \frac{\partial y_k}{\partial x_l}$ is a matrix.

We can write (6.1) as $J = J_b + J_o$, and from the rules discussed above, the **gradient of the background component of the cost function** $J_b = \frac{1}{2} (\mathbf{x}(t_0) - \mathbf{x}^b(t_0))^T \mathbf{B}_0^{-1} (\mathbf{x}(t_0) - \mathbf{x}^b(t_0))$ with respect to $\mathbf{x}(t_0)$ is given by

$$\frac{\partial J_b}{\partial \mathbf{x}(t_0)} = \mathbf{B}_0^{-1} (\mathbf{x}(t_0) - \mathbf{x}^b(t_0)). \quad (6.4)$$

This is the same as in the 3DVAR case.

Before we proceed to determine the gradient of the observation term, let's first define **Tangent Linear Model and Its Adjoint**.

A **linear tangent model** (or tangent linear model, TLM as it's often called) is obtained by linearizing the model about the nonlinear trajectory of the model between t_{i-1} and t_i , so that if we introduce a perturbation in the initial conditions, the final perturbation is given by

$$\mathbf{x}(t_i) + \delta\mathbf{x}(t_i) = M_{i-1}[\mathbf{x}(t_{i-1}) + \delta\mathbf{x}(t_{i-1})] = M_{i-1}[\mathbf{x}(t_{i-1})] + \mathbf{L}_{i-1}\delta\mathbf{x}(t_{i-1}) + O(|\delta\mathbf{x}|^2) \quad (6.5)$$

The **linear tangent model** \mathbf{L}_{i-1} is a matrix that transforms an small initial perturbation at t_{i-1} to the final perturbation at t_i .

The TLM equation is then

$$\delta\mathbf{x}(t_i) = \mathbf{L}_{i-1}\delta\mathbf{x}(t_{i-1}).$$

For example, the nonlinear advection $\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = 0$ is integrated using the forward-in-time centered-in-space finite difference scheme (this scheme is actually computationally unstable – it's given here for illustration purpose only),

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{2\Delta x} u_i^n (u_{i+1}^n - u_{i-1}^n).$$

The TLM model for perturbation u is

$$\delta u_i^{n+1} = \delta u_i^n + \frac{\Delta t}{2\Delta x} \left[u_i^n (\delta u_{i+1}^n - \delta u_{i-1}^n) + \delta u_i^n (u_{i+1}^n - u_{i-1}^n) \right] = -\mu u_i^n \delta u_{i-1}^n + \left[1 + \mu (u_{i+1}^n - u_{i-1}^n) \right] \delta u_i^n + \mu u_i^n \delta u_{i+1}^n$$

where $\mu \equiv \frac{\Delta t}{2\Delta x}$. Therefore

$$\delta \mathbf{u}^{n+1} = \begin{bmatrix} \delta u_1^{n+1} \\ \delta u_2^{n+1} \\ \vdots \\ \delta u_N^{n+1} \end{bmatrix} = \begin{bmatrix} 1 + \mu(u_2^n - u_0^n) & \mu u_1^n & \dots & 0 \\ -\mu u_1^n & 1 + \mu(u_3^n - u_1^n) & \mu u_3^n & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & -\mu u_{N-1}^n & 1 + \mu(u_{N+1}^n - u_{N-1}^n) \end{bmatrix} \begin{bmatrix} \delta u_1^n \\ \delta u_2^n \\ \vdots \\ \delta u_N^n \end{bmatrix} = \mathbf{L} \delta \mathbf{u}^n.$$

If there are several steps in a time interval $t_0 - t_i$, the tangent linear model that advances a perturbation from t_0 to t_i is given by the product of the tangent linear model matrices that advance it over each step:

$$\mathbf{L}(t_0, t_i) = \prod_{j=i-1}^0 \mathbf{L}(t_j, t_{j+1}) = \prod_{j=i-1}^0 \mathbf{L}_j = \mathbf{L}_{i-1} \mathbf{L}_{i-2} \dots \mathbf{L}_0 \quad (6.6)$$

The **adjoint model**, defined as the transpose of the linearized forward model, is given by

$$\mathbf{L}(t_i, t_0)^T = \prod_{j=0}^{i-1} \mathbf{L}(t_{j+1}, t_j)^T = \prod_{j=0}^{i-1} \mathbf{L}_j^T = \mathbf{L}_0^T \mathbf{L}_1^T \dots \mathbf{L}_{i-1}^T \quad (6.7)$$

Eq. (6.7) shows that the adjoint model “advances” a perturbation backwards in time, from the final to the initial time, because the right most \mathbf{L}^T in the equation is that of the final time interval (from t_{i-1} to t_i) and the left most \mathbf{L}^T is that of the first time interval (from t_0 to t_1) and matrix product is evaluated in the right-to-left order.

The **gradient of the observational term**,

$$J_o = \frac{1}{2} \sum_{i=0}^N (H(\mathbf{x}_i) - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1} (H(\mathbf{x}_i) - \mathbf{y}_i^o)$$

is more complicated because $\mathbf{x}_i = M_i(\mathbf{x}(t_0))$.

If we introduce a perturbation to the initial state, then $\delta \mathbf{x}_i = \mathbf{L}(t_0, t_i) \delta \mathbf{x}_0$, so that

$$\frac{\partial (H(\mathbf{x}_i) - \mathbf{y}_i^o)}{\partial \mathbf{x}(t_0)} = \frac{\partial H}{\partial \mathbf{x}_i} \frac{\partial M}{\partial \mathbf{x}_o} = \frac{\partial H}{\partial \mathbf{x}_i} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_o} = \mathbf{H}_i \mathbf{L}(t_0, t_i) = \mathbf{H}_i \prod_{j=i-1}^0 \mathbf{L}(t_j, t_{j+1}) \quad (6.8)$$

As indicated by (6.8), the matrices \mathbf{H}_i and \mathbf{L}_i are the linearized Jacobians, $\frac{\partial H}{\partial \mathbf{x}_i}$ and $\frac{\partial M}{\partial \mathbf{x}_o}$, respectively.

Therefore the gradient of the observation cost function is given by (remember $\frac{\partial J}{\partial \mathbf{x}} = \left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right]^T \mathbf{A} \mathbf{y}$)

$$\left[\frac{\partial J_o}{\partial \mathbf{x}(t_0)} \right] = \sum_{i=0}^N \mathbf{L}(t_i, t_0)^T \mathbf{H}_i^T \mathbf{R}_i^{-1} (H(\mathbf{x}_i) - \mathbf{y}_i^o) \quad (6.9)$$

Equation (6.9) shows that every iteration of the 4D-Var minimization requires the computation of the gradient, i.e., computing the increments $(H(\mathbf{x}_i) - \mathbf{y}_i^o)$ at the observation times t_i during a forward integration, multiplying them by $\mathbf{H}_i^T \mathbf{R}_i^{-1}$ and integrating these weighted increments back to the initial time using the adjoint model.

Denote $\bar{\mathbf{d}}_i = \mathbf{H}_i^T \mathbf{R}_i^{-1} (H(\mathbf{x}_i) - \mathbf{y}_i^o) = -\mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{d}_i$ which we call the weighted observational increment for observations at time t_i . Since parts of the backward adjoint integration are common to several time intervals, the summation in (6.9) can be arranged more conveniently:

$$\begin{aligned} \left[\frac{\partial J_o}{\partial \mathbf{x}(t_0)} \right] &= \sum_{i=0}^N \mathbf{L}(t_i, t_0)^T \bar{\mathbf{d}}_i = \sum_{i=0}^N \mathbf{L}_0^T \mathbf{L}_1^T \dots \mathbf{L}_{i-1}^T \bar{\mathbf{d}}_i \\ &= \bar{\mathbf{d}}_0 + \mathbf{L}_0^T \bar{\mathbf{d}}_1 + \dots + \mathbf{L}_0^T \mathbf{L}_1^T \dots \mathbf{L}_{i-2}^T \bar{\mathbf{d}}_{i-1} + \mathbf{L}_0^T \mathbf{L}_1^T \dots \mathbf{L}_{i-1}^T \bar{\mathbf{d}}_i \\ &= \bar{\mathbf{d}}_0 + \mathbf{L}_0^T (\bar{\mathbf{d}}_1 + \mathbf{L}_1^T (\bar{\mathbf{d}}_2 + \dots + \mathbf{L}_{i-2}^T (\bar{\mathbf{d}}_{i-1} + \mathbf{L}_{i-1}^T \bar{\mathbf{d}}_i))) \end{aligned}$$

Assume, for example that the interval of assimilation is from 00Z to 12Z, and that there are observations every 3 hours (Fig. 5.6).

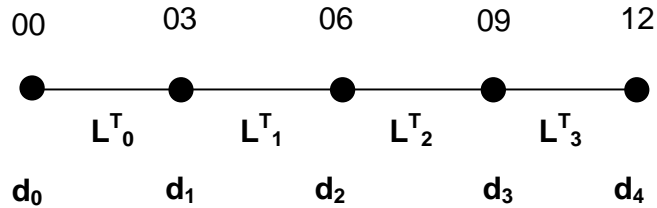


Figure: Schematic of the computation of the gradient of the observational cost function for a period of 12 hours, observations every 3 hours and the adjoint model that integrates backwards within each interval.

Procedure of 4DVAR minimization

- 1) First integrate the (nonlinear) forward model and save the nonlinear trajectory (i.e., model state at every time step) in the assimilation window. This 4D state is needed for defining \mathbf{L}_{i-1}^T and for calculating the observational increments.
- 2) Compute the weighted negative observation increments $\bar{\mathbf{d}}_i = \mathbf{H}_i^T \mathbf{R}_i^{-1} (H(\mathbf{x}_i) - \mathbf{y}_i^o) = -\mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{d}_i$.
- 3) The adjoint model $\mathbf{L}^T(t_i, t_{i-1}) = \mathbf{L}_{i-1}^T$ applied on a vector “advances” it from t_i to t_{i-1} . In our case, this vector is $\bar{\mathbf{d}}_i$ or the combination of $\bar{\mathbf{d}}$'s at times later than t_{i-1} . This vector is the ‘adjoint’ variable that the adjoint equation ‘advances’.
- 4) Then we can write (6.9) as
$$\frac{\partial J_o}{\partial \mathbf{x}_0} = \bar{\mathbf{d}}_0 + \mathbf{L}_0^T (\bar{\mathbf{d}}_1 + \mathbf{L}_1^T (\bar{\mathbf{d}}_2 + \dots + \mathbf{L}_{i-2}^T (\bar{\mathbf{d}}_{i-1} + \mathbf{L}_{i-1}^T \bar{\mathbf{d}}_i))) \quad (6.10)$$
- 5) From (6.4) plus (6.10) we obtain the gradient of the cost function, and the minimization algorithm is then called to modify appropriately the control variable $\mathbf{x}(t_0)$.
- 6) After this change, a new forward integration and new observational increments are computed and the process is repeated until convergence.

Therefore, each minimization iteration involves one forward integration of the nonlinear prediction model and one ‘backward’ integration of the adjoint model. Because everything in the adjoint model is reversed in order, and the definition of the adjoint operator requires the value of the variables in the forward model (even within the individual steps of a single time step), many recalculations are often involved in order to restore the values of these variables (unless they are saved in during the forward integration step), the adjoint model is often 2-3 times as expensive as the forward model. But still, only one integration is needed instead of N of them as Eq. (6.9) might suggest. This is so because of the linear nature of Eq. (6.9).

10.2 Theorem: minimization of the 4D-Var cost function

The evaluation of the 4D-Var observation cost function and its gradient, $\mathcal{J}_o(\mathbf{x})$ and $\nabla \mathcal{J}_o(\mathbf{x})$, requires one direct model integration from times 0 to n and one suitably modified *adjoint integration* made of transposes of the tangent linear model time-stepping operators \mathbf{M}_i .

Proof:

The first stage is the direct integration of the model from \mathbf{x} to \mathbf{x}_n , computing successively at each observation time \tilde{i} :

- 1) the forecast state $\mathbf{x}_{\tilde{i}} = \mathbf{M}_{\tilde{i}}\mathbf{M}_{\tilde{i}-1}\dots\mathbf{M}_1\mathbf{x}_1$,
- 2) the “normalized departures” $\mathbf{d}_{\tilde{i}} = \mathbf{R}_{\tilde{i}}^{-1}(\mathbf{y}_{\tilde{i}} - \mathbf{H}_{\tilde{i}}[\mathbf{x}_{\tilde{i}}])$ which are stored,
- 3) the contributions to the cost function $\mathcal{J}_{o\tilde{i}}(\mathbf{x}) = (\mathbf{y} - \mathbf{H}_{\tilde{i}}[\mathbf{x}_{\tilde{i}}])^T \mathbf{d}_{\tilde{i}}$
- 4) And finally $\mathcal{J}_o(\mathbf{x}) = \sum_{\tilde{i}=0}^n \mathcal{J}_{o\tilde{i}}(\mathbf{x})$.

To compute $\nabla \mathcal{J}_o$ it is necessary to perform a slightly complex factorization:

$$\begin{aligned} -\frac{1}{2}\nabla \mathcal{J}_o &= -\frac{1}{2} \sum_{\tilde{i}=0}^n \nabla \mathcal{J}_{o\tilde{i}} \\ &= \sum_{\tilde{i}=0}^n \mathbf{M}_1^T \dots \mathbf{M}_{\tilde{i}}^T \mathbf{H}_{\tilde{i}}^T \mathbf{d}_{\tilde{i}} \\ &= \mathbf{H}_0^T \mathbf{d}_0 + \mathbf{M}_1^T [\mathbf{H}_1^T \mathbf{d}_1 + \mathbf{M}_2^T [\mathbf{H}_2^T \mathbf{d}_2 + \dots + \mathbf{M}_n^T \mathbf{H}_n^T \mathbf{d}_n] \dots] \end{aligned}$$

and the last expression is easily evaluated from right to left using the following algorithm:

- 5) initialize the so-called *adjoint variable* $\tilde{\mathbf{x}}$ to zero at final time: $\tilde{\mathbf{x}} = \mathbf{0}$
- 6) for each time step $\tilde{i} - 1$ the variable $\tilde{\mathbf{x}}_{\tilde{i}-1}$ is obtained by adding the *adjoint forcing* $\mathbf{H}_{\tilde{i}}^T \mathbf{d}_{\tilde{i}}$ to $\tilde{\mathbf{x}}_{\tilde{i}}$ and by performing the *adjoint integration* by multiplying the result by $\mathbf{M}_{\tilde{i}}^T$, i.e. $\tilde{\mathbf{x}}_{\tilde{i}-1} = \mathbf{M}_{\tilde{i}}^T (\tilde{\mathbf{x}}_{\tilde{i}} + \mathbf{H}_{\tilde{i}}^T \mathbf{d}_{\tilde{i}})$
- 7) at the end of the recurrence, the value of the adjoint variable $\tilde{\mathbf{x}}_0 = -(1/2)\nabla \mathcal{J}_o(\mathbf{x})$ gives the required result.

The terminology employed in the algorithm reflects the fact that the computations look like the integration of an *adjoint model* backward in time with a time-stepping defined by the transpose time-stepping operators \mathbf{M}_i^T and an external forcing $\mathbf{H}_i^T \mathbf{d}_i$, which depends on the distance between the model trajectory and the observations. In this discrete presentation it is just a convenient way of evaluating an algebraic expression²⁵.

Incremental Form of 4DVAR

4D-Var can also be written in an incremental form with the cost function defined by

$$J(\delta \mathbf{x}_0) = \frac{1}{2} (\delta \mathbf{x}_0)^T \mathbf{B}_0^{-1} (\delta \mathbf{x}_0) + \frac{1}{2} \sum_{i=0}^N \left[\mathbf{H}_i \mathbf{L}(t_0, t_i) \delta \mathbf{x}_0 - \mathbf{d}_i^o \right]^T \mathbf{R}_i^{-1} \left[\mathbf{H}_i \mathbf{L}(t_0, t_i) \delta \mathbf{x}_0 - \mathbf{d}_i^o \right] \quad (6.11)$$

and the observational increment is defined as $\mathbf{d}_i^o = \mathbf{y}_i^o - H(\mathbf{x}^f(t_i))$.

Because

$$H(\mathbf{x}_i) = H(\mathbf{x}_0) + \frac{\partial H(\mathbf{x}_i)}{\partial \mathbf{x}(t_0)} \delta \mathbf{x}_0 = H(\mathbf{x}_0) + \mathbf{H}_i \mathbf{L}(t_0, t_i) \delta \mathbf{x}_0$$

which is plugged into (6.1) to give (6.11).

Within the incremental formulation, it is possible to choose a “*simplification operator*” that solves the problem of minimization in a lower dimensional space for \mathbf{w} than that of the original model variables \mathbf{x} :

$$\delta \mathbf{w} = \mathbf{S} \delta \mathbf{x}$$

\mathbf{S} is meant to be rank deficient (as would be the case, for example, if a lower resolution spectral truncation or a low-resolution grid was used for \mathbf{w} than for \mathbf{x}). In the latter case, \mathbf{w} can be \mathbf{x} at every other grid point, in this case, \mathbf{w} is a vector that about $\frac{1}{4}$ the length of \mathbf{x} .

After the minimum of the problem is obtained for $J(\delta\mathbf{w})$, $\mathbf{x}_0^b = \mathbf{x}_0^g + \mathbf{S}^{-1} \delta\mathbf{w}_0$ and a new “*outer iteration*” at the full model resolution can be carried out (Lorenz 1997). Here the inversion operator \mathbf{S}^{-1} can be spatial interpolation from the coarser resolution grid of \mathbf{w} to the higher-resolution grid of \mathbf{x} .

In essence, the gridded fields associated with the high-resolution trajectory is first interpolated to a coarser resolution grid, then a cost function defined on this coarse grid is minimized using an iterative minimization algorithm that involves forward and backward integration for the TLM and adjoint models in every iteration. When an analysis increment is found that minimizes this cost function, it is interpolated to the higher-resolution grid and added to the initial condition estimate of the previous outer iteration. Starting from this updated initial condition, a nonlinear forward integration is performed on the high resolution grid to produce a more accurate nonlinear trajectory. This trajectory is again interpolated to the coarse resolution grid, the nonlinear model is linearized around this new trajectory to obtain the more accurate TLM and adjoint models. Another ‘outer’ iteration loop, that continues the minimization ‘inner loops’ then begins.

The iteration process can also be accelerated through the use of “*preconditioning*”, a change of control variables that makes the cost function more “spherical”, and therefore each iteration can get closer to the center (minimum) of the cost function (e.g., Parrish and Derber 1992; Lorenz 1997).

4DVAR versus 3D analysis methods (3DVAR, IO)

When compared to a 3-D analysis algorithm in a sequential assimilation system, the (strong-constraint) 4D-Var has the following characteristics:

- it works under the assumption that the model is perfect. Problems can be expected if model errors are large.
- it requires the implementation of the rather special \mathbf{L}_i^T operators, the so-called adjoint model. This can be a lot of work if the forecast model is complex.
- in a real-time system it requires the assimilation to wait for the observations over the whole 4D-Var time interval to be available before the analysis procedure can begin, whereas sequential systems can process observations shortly after they are available. This can delay the availability of analysis.
- \mathbf{x}_a is used as the initial state for a forecast, then by construction of 4D-Var one is sure that the forecast will be completely consistent with the model equations and the four-dimensional distribution of observations until the end of the 4D-Var time interval (the *cut off time*). This makes intermittent 4D-Var a very suitable system for numerical forecasting.
- (standard strong-constraint) 4D-Var is an optimal (subjecting to a number of assumptions) and relatively efficient (relative to e.g., traditional Kalman filter or weak-constraint 4DVAR) assimilation algorithm over its time period thanks to a theorem that says that “The evaluation of the 4D-Var observation cost function and its gradient, $J(\mathbf{x}_0)$ and $\nabla J_0(\mathbf{x})$, requires one direct model integration from times t_0 to t_n and one suitably modified *adjoint integration* made of transposes of the tangent linear model time-stepping operators \mathbf{M}_i .”. In fact it is the application of this procedure that made 4DVAR practical.

4DVAR versus Kalman filter

- The most important advantage of 4D-Var is that if we assume that the model is perfect, and that the a priori error covariance at the initial time \mathbf{B}_0 is correct, *it can be shown that the 4D-Var analysis at the final time is identical to that of the extended Kalman Filter* (Lorenc, 1986, Daley, 1991).
- This means that *implicitly 4D-Var is able to evolve the forecast error covariance from \mathbf{B}_0 to the final time (because Kalman filter explicitly evolves \mathbf{B})*.
- Unfortunately, this implicit covariance is not available at the end of the cycle, and neither is the new analysis error covariance. In other words, 4D-Var is able to find the BLUE (Best Linear Unbiased Estimate) but not its error covariance.
- To mitigate this problem, a simplified Kalman Filter algorithm has been proposed to estimate the evolution of the analysis errors in the subspace of the dynamically most unstable modes (Fisher and Courtier, 1995, Cohn and Todling, 1996).
- 4DVAR is much cheaper than the conventional Kalman Filter.
- 4DVAR is, however, comparable in cost to ensemble-based Kalman filter, which uses a forecast ensemble to estimate the evolving forecast error covariance.