Advanced data assimilation methods with evolving forecast error covariance

Four-dimensional variational analysis (4D-Var)

Shu-Chih Yang (with EK)

Find the optimal analysis

$T_1 = T_t + \varepsilon_1$ (forecast)

$T_2 = T_t + \varepsilon_2$ (observation)

⇨ Best estimate the true value

- Least squares approach

Find the optimal weights to minimize the analysis error covariance

$$T_a = \left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \right) T_1 + \left(\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \right) T_2$$

- Variational approach

Find the analysis that will minimize a cost function, measuring its distance to the background and to the observation

$$J(T) = \frac{1}{2} \left[\frac{(T - T_1)^2}{\sigma_1^2} + \frac{(T - T_2)^2}{\sigma_2^2} \right], \quad \frac{\partial J}{\partial T} = 0 \; for \; T = T_a$$

Both methods give the same T_a!

3D-Var

How do we find an optimum analysis of a 3-D field of model variable $\mathbf{x}^a$, given a background field, $\mathbf{x}^b$, and a set of observations, $\mathbf{y}^o$?

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}^b)^T \mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b) + \frac{1}{2}[\mathbf{y}^o - H(\mathbf{x})]^T \mathbf{R}^{-1}[\mathbf{y}^o - H(\mathbf{x})]$$

Distance to forecast (J_b) Distance to observations (J_o)

$$\nabla J(\mathbf{x}^a) = 0 \text{ at } J(\mathbf{x}^a) = J_{min}$$

- find the solution in 3D-Var

Directly set $\nabla J(\mathbf{x}^a)=0$ and solve

$$(\mathbf{B}^{-1}+\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})(\mathbf{x}^a-\mathbf{x}^b)=\mathbf{H}^T\mathbf{R}^{-1}[\mathbf{y}^o-H(\mathbf{x}^b)] \qquad \text{(Eq. 5.5.9)}$$

Usually solved as

$$(\mathbf{I}+ \mathbf{B}\,\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})(\mathbf{x}^a-\mathbf{x}^b)= \mathbf{B}\,\mathbf{H}^T\mathbf{R}^{-1}[\mathbf{y}^o-H(\mathbf{x}^b)]$$

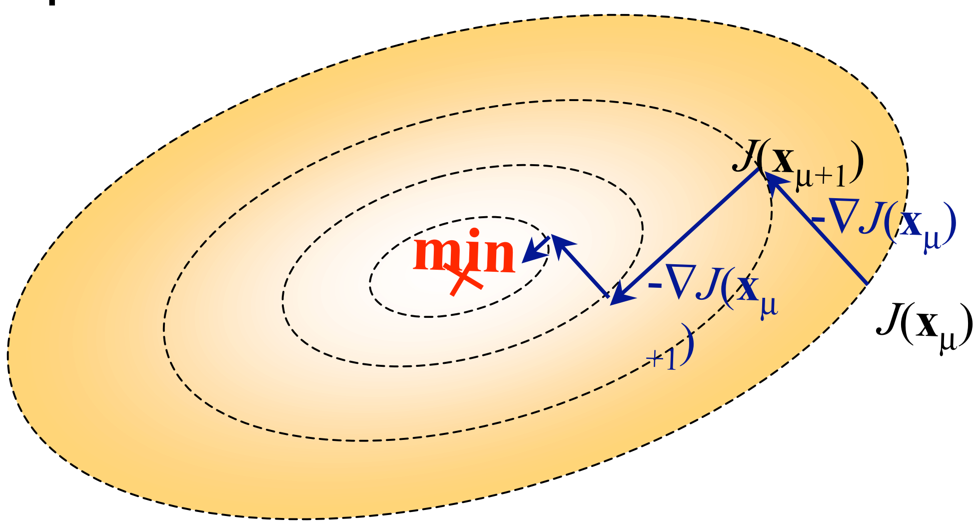
Minimize the cost function, $J(\mathrm{x})$

A descent algorithm is used to find the minimum of the cost function.

This requires the gradient of the cost function, ∇J.

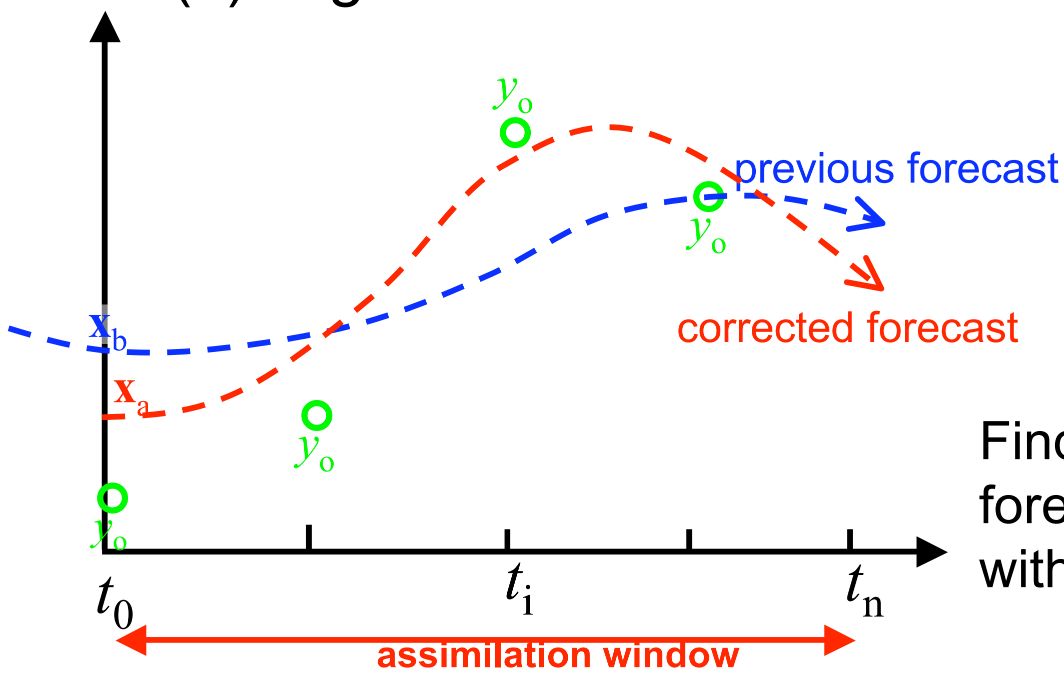
$$\delta J \approx \left[\frac{\partial J}{\partial x}\right]^T \cdot \delta x; \quad \nabla J = \frac{\partial J}{\partial x}$$

Ex: "steepest descent" method



4D-Var

$J(\mathbf{x})$ is generalized to include observations at different times.



Find the initial condition such that its forecast best fits the observations within the assimilation interval

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

Need to define $\nabla J(\mathbf{x}(t_0))$ in order to minimize $J(\mathbf{x}(t_0))$

Separate $J(x(t_0))$ into "background" and "observation" terms

$$J = J_b + J_o, \quad \frac{\partial J}{\partial \mathrm{x}(t_0)} = \frac{\partial J_b}{\partial \mathrm{x}(t_0)} + \frac{\partial J_o}{\partial \mathrm{x}(t_0)}$$

First, let's consider $J_b(\mathbf{x}(t_0))$

Given a symmetric matrix $\mathbf{A}$, and

a function $J = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}$, the gradient is given by $\frac{\partial J}{\partial \mathbf{x}} = \mathbf{A}\mathbf{y}$

$$J_b = \frac{1}{2}[\mathrm{x}(t_0) - \mathrm{x}^b(t_0)]^T \mathrm{B}^{-1}[\mathrm{x}(t_0) - \mathrm{x}^b(t_0)] \Rightarrow \frac{\partial J_b}{\partial \mathbf{x}(t_0)} = \mathbf{B}^{-1}[\mathbf{x}(t_0) - \mathbf{x}^b(t_0)]$$

∇J_o is more complicated, because it involves the integration of the model:

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

If $J = \mathbf{y}^T\mathbf{A}\mathbf{y}$ and $\mathbf{y} = \mathbf{y}(\mathbf{x})$, then $\frac{\partial J}{\partial \mathbf{x}} = \left[\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right]^T \mathbf{A}\mathbf{x}$,

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

$\mathbf{x}_i = M_i[\mathbf{x}(t_0)]$

$$\frac{\partial(H(\mathrm{x}_i) - y_i^o)}{\partial \mathrm{x}_0} = \frac{\partial H}{\partial \mathrm{x}_i}\frac{\partial M_i}{\partial \mathrm{x}_0} = \mathbf{H}_i\mathbf{L}(t_0, t_i) = \mathbf{H}_i\mathbf{L}_{i\text{-}1}\mathbf{L}_{i\text{-}2}\cdots\mathbf{L}_0$$

$$[\mathbf{H}_\mathbf{i}\mathbf{L}_{\mathbf{i\text{-}1}}\mathbf{L}_{\mathbf{i\text{-}2}}...\mathbf{L}_\mathbf{0}]^T = \mathbf{L}_0^T\cdots\mathbf{L}_{i-2}^T\mathbf{L}_{i-1}^T\mathbf{H}_i^T = \mathbf{L}^T(t_i, t_0)\mathbf{H}_i^T$$

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

Adjoint model integrates increment backwards to t_0

weighted increment at observation time, t_i, in model coordinates

Simple example:
Use the adjoint model to integrate backward in time

$$
\begin{array}{ccccc}
t_0 & t_1 & t_2 & t_3 & t_4 \\
\bar{\mathbf{d}}_0 & \bar{\mathbf{d}}_1 & \bar{\mathbf{d}}_2 & \bar{\mathbf{d}}_3 & \bar{\mathbf{d}}_4
\end{array}
$$

$$\partial J_o / \partial \mathbf{x}_0 \quad \boxed{\bar{\mathbf{d}}_0 + \mathbf{L}_0^T(\bar{\mathbf{d}}_1 + \mathbf{L}_1^T(\bar{\mathbf{d}}_2 + \mathbf{L}_2^T(\bar{\mathbf{d}}_3 + \mathbf{L}_3^T \bar{\mathbf{d}}_4)))}$$

$$+$$

$$\partial J_b / \partial \mathbf{x}_0 \quad \boxed{\mathbf{B}_0^{-1}[\mathbf{x}(t_0) - \mathbf{x}^b(t_0)]}$$

$$\bar{\mathbf{d}}_i = \mathbf{H}_i^T \mathbf{R}_i^{-1}[H(\mathbf{x}_i) - \mathbf{y}_i^o]$$

Start from the end!

- In each iteration, ∇J is used to determine the direction to search the $J_{\min}$.
- 4D-Var provides the best estimation of the analysis state and error covariance is evolved implicitly.

3D-Var vs. 4D-Var

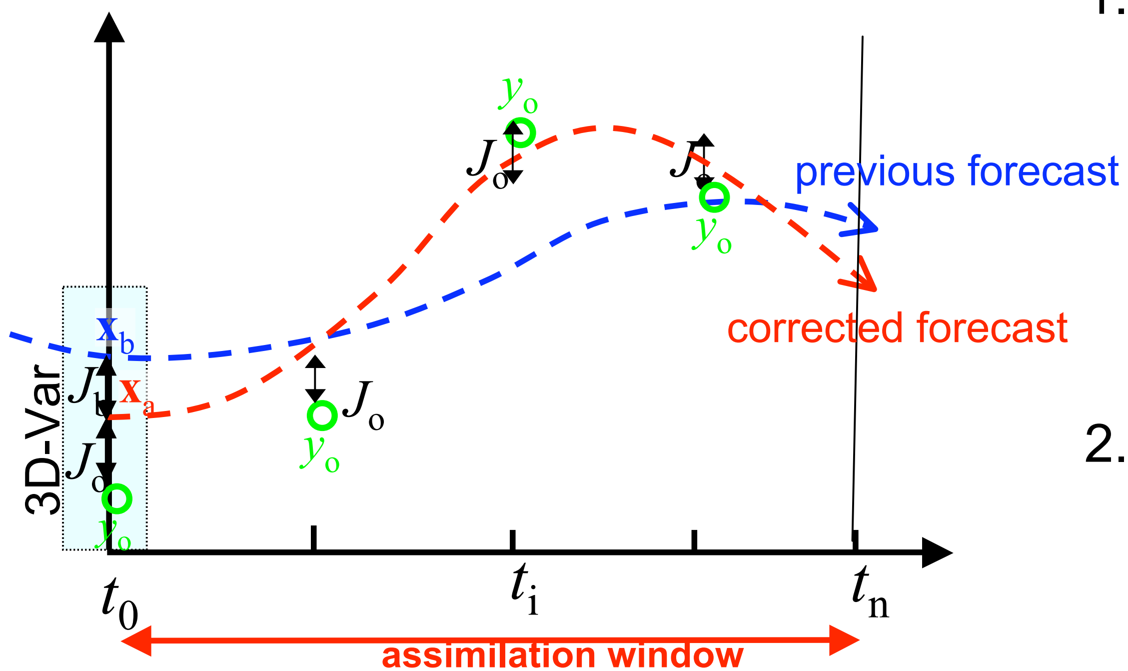


Figure from http://www.ecmwf.int/

1. 4D-Var assumes a perfect model. It will give the same credence to older observations as to newer observations.
   - algorithm modified by Derber (1989)
2. Background error covariance is time-independent in 3D-Var, but evolves implicitly in 4D-Var.
3. In 4D-Var, the adjoint model is required to compute ∇J.

Practical implementation: use the incremental form

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

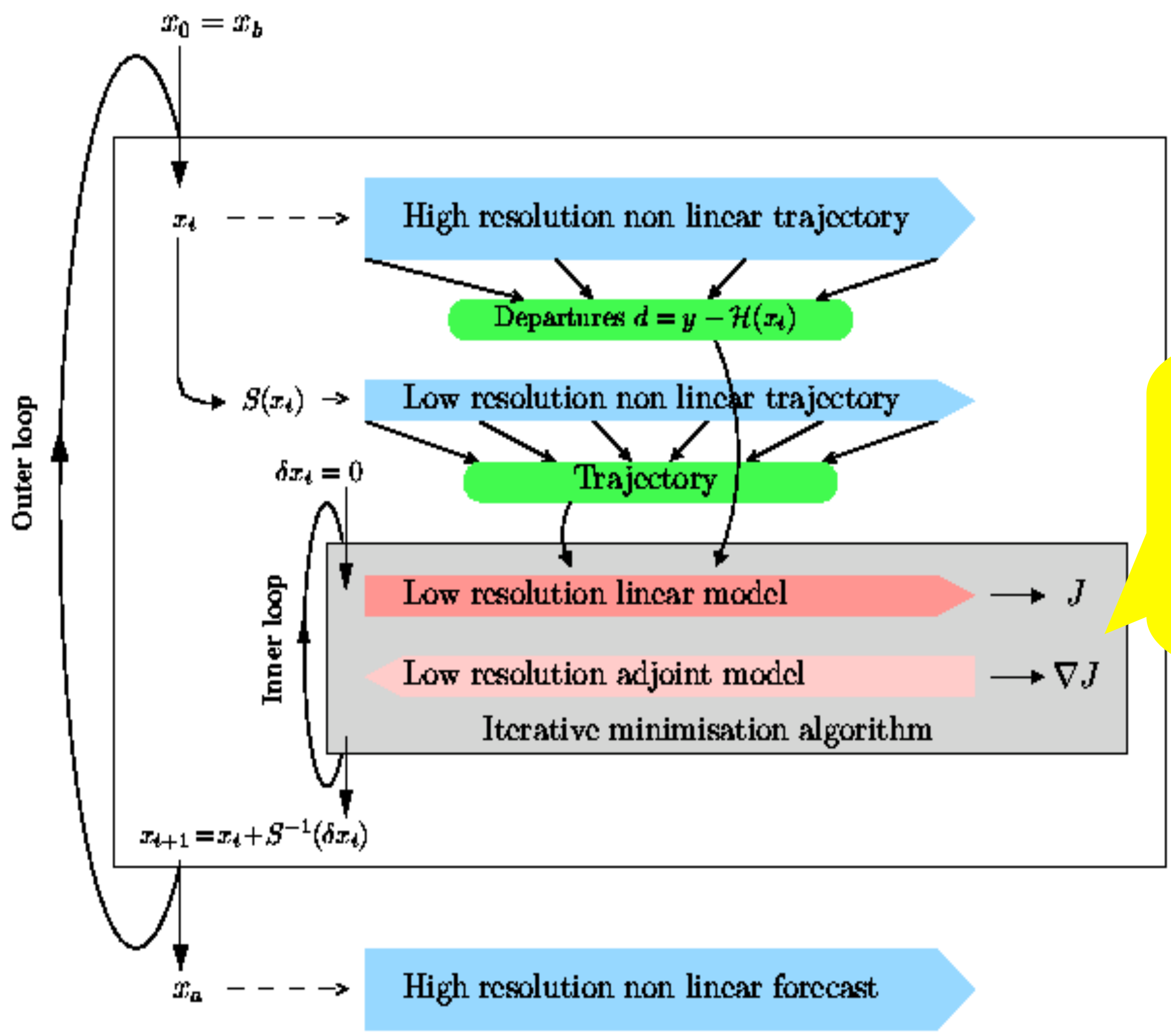
where $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}_b$ and $\mathbf{d} = \mathbf{y}_\mathbf{o} - H(\mathbf{x})$

With this form, it is possible to choose a "simplification operator, $\mathbf{S}$" to solve the cost function in a low dimension space (change the control variable).
Now, $\delta\mathbf{w}=\mathbf{S}\delta\mathbf{x}$ and minimize $J(\delta\mathbf{w})$

The choice of the simplification operator

- Lower resolution
- Simplification of physical process

Example of using simplification operator



Both TLM and ADJ use a low resolution and also simplified physics due to the limitation of the computational cost.

Example with the Lorenz 3-variable model

Nonlinear model

$\mathbf{x}=[x_1,x_2,x_3]$

$$\frac{dx_1}{dt} = -px_1 + px_2$$

$$\frac{dx_2}{dt} = rx_1 - x_1x_3 - x_2$$

$$\frac{dx_3}{dt} = x_1x_2 - bx_3$$

Tangent linear model

$\delta\mathbf{x}=[\delta x_1, \delta x_2, \delta x_3]$

$$\mathbf{L} = \frac{\partial M}{\partial \mathbf{x}} = \frac{\partial M}{\partial x_i} = \begin{bmatrix} -p & p & 0 \\ r - \mathbf{x_3} & -1 & -\mathbf{x_1} \\ \mathbf{x_2} & \mathbf{x_1} & -b \end{bmatrix}$$

Adjoint model

$\delta\mathbf{x}^*=[\delta x^*_1, \delta x^*_2, \delta x^*_3]$

$$\mathbf{L}^T = \left[\frac{\partial M}{\partial x_i}\right]^T = \begin{bmatrix} -p & r - \mathbf{x_3} & \mathbf{x_2} \\ p & -1 & \mathbf{x_1} \\ 0 & -\mathbf{x_1} & -b \end{bmatrix}$$

- The background state is needed in both $\mathbf{L}$ and $\mathbf{L}^T$ (need to save the model trajectory)
- In a complex NWP model, it is impossible to write explicitly this matrix form

Example of tangent linear and adjoint codes (1)

use forward scheme to integrate in time

In tangent linear model

$$\frac{\delta x_3(t+\Delta t)-\delta x_3(t)}{\Delta t}=x_2(t)\delta x_1(t)+x_1(t)\delta x_2(t)-b\delta x_3(t)$$

$$\delta x_3(t+\Delta t)=\delta x_3(t)+[x_2(t)\delta x_1(t)+x_1(t)\delta x_2(t)-b\delta x_3(t)]\Delta t$$

forward in time

We will see that in the adjoint model the above line becomes

$$\delta x_3^*(t)=\delta x_3^*(t)+(1-b\Delta t)\delta x_3^*(t+\Delta t)$$

$$\delta x_2^*(t)=\delta x_2^*(t)+(x_1(t)\Delta t)\delta x_3^*(t+\Delta t)$$

$$\delta x_1^*(t)=\delta x_1^*(t)+(x_2(t)\Delta t)\delta x_3^*(t+\Delta t)$$

$$\delta x_3^*(t+\Delta t)=0$$

backward in time

* Try an example in Appendix B (B.1.15)

Example of tangent linear and adjoint codes (2)

use forward scheme to integrate in time

Tangent linear model,

$$\delta x_3(t+\Delta t) = \delta x_3(t) + [x_2(t)\delta x_1(t) + x_1(t)\delta x_2(t) - b\delta x_3(t)]\Delta t$$

forward in time

$$\begin{bmatrix} \delta x_3(t+\Delta t) \\ \delta x_1(t) \\ \delta x_2(t) \\ \delta x_3(t) \end{bmatrix} = \begin{bmatrix} 0 & x_2(t)\Delta t & x_1(t)\Delta t & (1-b\Delta t) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_3(t+\Delta t) \\ \delta x_1(t) \\ \delta x_2(t) \\ \delta x_3(t) \end{bmatrix}$$

We have to write for each statement all the “active” variables.

Then we transpose it to get the adjoint model

Example of tangent linear and adjoint codes (3)

Tangent linear model,

$$\delta x_3(t+\Delta t) = \delta x_3(t) + [x_2(t)\delta x_1(t) + x_1(t)\delta x_2(t) - b\delta x_3(t)]\Delta t$$

forward in time

$$\begin{bmatrix} \delta x_3(t+\Delta t) \\ \delta x_1(t) \\ \delta x_2(t) \\ \delta x_3(t) \end{bmatrix} = \begin{bmatrix} 0 & x_2(t)\Delta t & x_1(t)\Delta t & (1-b\Delta t) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_3(t+\Delta t) \\ \delta x_1(t) \\ \delta x_2(t) \\ \delta x_3(t) \end{bmatrix}$$

$$\begin{bmatrix} \delta x_3^*(t+\Delta t) \\ \delta x_1^*(t) \\ \delta x_2^*(t) \\ \delta x_3^*(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ x_2(t)\Delta t & 1 & 0 & 0 \\ x_1(t)\Delta t & 0 & 1 & 0 \\ (1-b\Delta t) & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_3^*(t+\Delta t) \\ \delta x_1^*(t) \\ \delta x_2^*(t) \\ \delta x_3^*(t) \end{bmatrix}$$

Adjoint model: transpose of the linear tangent, backward in time

Execute in reverse order

Example of tangent linear and adjoint codes (4)

$$\delta x_3(t+\Delta t) = \delta x_3(t) + [x_2(t)\delta x_1(t) + x_1(t)\delta x_2(t) - b\delta x_3(t)]\Delta t$$

Adjoint model: transpose of the linear tangent, backward in time

Execute in reverse order

$$\begin{bmatrix} \delta x_3^*(t+\Delta t) \\ \delta x_1^*(t) \\ \delta x_2^*(t) \\ \delta x_3^*(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ x_2(t)\Delta t & 1 & 0 & 0 \\ x_1(t)\Delta t & 0 & 1 & 0 \\ (1-b\Delta t) & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x_3^*(t+\Delta t) \\ \delta x_1^*(t) \\ \delta x_2^*(t) \\ \delta x_3^*(t) \end{bmatrix}$$

In adjoint model the line above becomes

$$\delta x_3^*(t) = \delta x_3^*(t) + (1-b\Delta t)\delta x_3^*(t+\Delta t)$$

$$\delta x_2^*(t) = \delta x_2^*(t) + (x_1(t)\Delta t)\delta x_3^*(t+\Delta t)$$

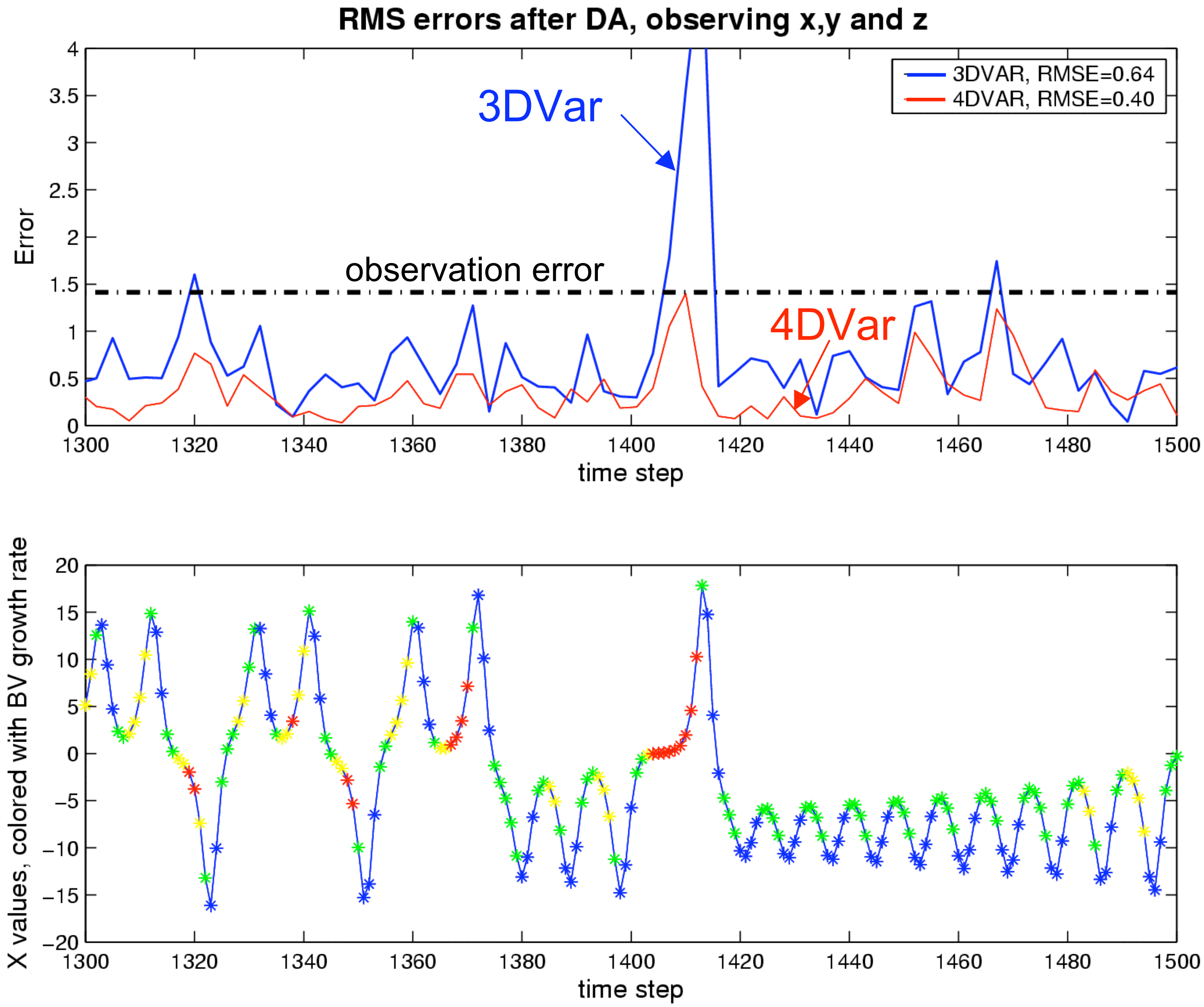
$$\delta x_1^*(t) = \delta x_1^*(t) + (x_2(t)\Delta t)\delta x_3^*(t+\Delta t)$$

$$\delta x_3^*(t+\Delta t) = 0$$

backward in time

RMS error of 3D-Var and 4D-Var in Lorenz model

Experiments: DA cycle and observations: $8\Delta t$, $\mathbf{R}=2*\mathbf{I}$
4D-Var assimilation window: $24\Delta t$



Evans et al., BAMS, 2004

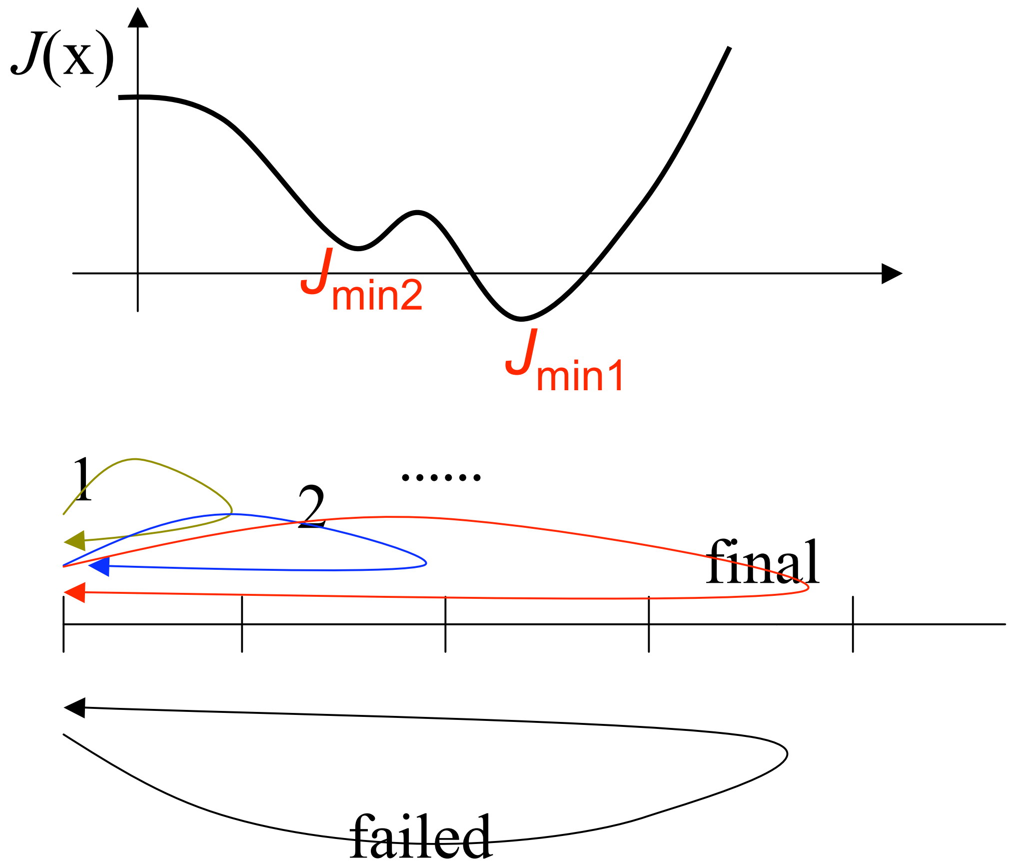
4D-Var in the Lorenz model (Kalnay et al., 2005)

	Win=8	16	24	32	40	**48**	56	64	72
Fixed window	0.59	0.59	0.47	0.43	0.62	**0.95**	0.96	0.91	0.98
Start with short window	0.59	0.51	0.47	0.43	0.42	**0.39**	0.44	0.38	0.43

Impact of the window length

- Lengthening the assimilation window reduces the RMS analysis error up 32 steps.
- For the long windows, error increases because the cost function has multiple minima.
- This problem can be overcome by the quasi-static variational assimilation approach (Pires et al, 1996), which needs to start from a shorter window and progressively increase the length of the window.

Schematic of multiple minima and increasing window size (Pires et al, 1996)



Dependence of the analysis error on B_0

Win=8	B=∞	$B_{3D\text{-}Var}$	50% $B_{3D\text{-}Var}$	40% $B_{3D\text{-}Var}$	30% $B_{3D\text{-}Var}$	20% $B_{3D\text{-}Var}$	10% $B_{3D\text{-}Var}$	5% $B_{3D\text{-}Var}$
RMSE	0.78	0.59	0.53	0.52	0.50	0.51	0.65	>2.5

Dependence of the analysis error on the B_0

•Since the forecast state from 4D-Var will be more accurate than 3D-Var, the amplitude of B should be smaller than the one used in 3D-Var.

• Using a covariance proportional to $B_{3D\text{-}Var}$ and tuning its amplitude is a good strategy to estimate B.