Variational Data Assimilation
Current Status

Yannick Trémolet
with contributions from Mike Fisher

ECMWF

JCSDA Summer Colloquium on Data Assimilation
July 2009

Outline

1. The Maximum Likelihood Approach
2. 4D-Var (and 3D-Var)
3. Minimization and Incremental 4D-Var
4. Preconditioning
5. Operational 4D-Var Setup
6. Summary

Outline

1. The Maximum Likelihood Approach
2. 4D-Var (and 3D-Var)
3. Minimization and Incremental 4D-Var
4. Preconditioning
5. Operational 4D-Var Setup
6. Summary

Maximum Likelihood

- We define the analysis $\mathbf{x}_a$ as the most probable state of the system given a background state $\mathbf{x}_b$ and observations $\mathbf{y}$:

$$\mathbf{x}_a = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y} \text{ and } \mathbf{x}_b)$$

- It will be convenient to define a cost function:

$$J(\mathbf{x}) = -\log p(\mathbf{x}|\mathbf{y} \text{ and } \mathbf{x}_b) + K$$

  where K is a constant.
- Since log is a monotonic function, $\mathbf{x}_a$ is also:

$$\mathbf{x}_a = \arg\min_{\mathbf{x}} J(\mathbf{x})$$

- Variational data assimilation comprises minimizing the cost function $J(\mathbf{x})$.

Maximum Likelihood and Bayes' Theorem

- Applying Bayes' theorem gives:

$$p(\mathbf{x}|\mathbf{y} \text{ and } \mathbf{x}_b) = \frac{p(\mathbf{y} \text{ and } \mathbf{x}_b|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y} \text{ and } \mathbf{x}_b)}$$

- $p(\mathbf{y} \text{ and } \mathbf{x}_b)$ is independent of $\mathbf{x}$ and *a priori* we know nothing about $\mathbf{x}$ (all values of $\mathbf{x}$ are equally likely) thus $p(\mathbf{x})$ is also independent of $\mathbf{x}$.
- Hence:

$$p(\mathbf{x}|\mathbf{y} \text{ and } \mathbf{x}_b) \propto p(\mathbf{y} \text{ and } \mathbf{x}_b|\mathbf{x})$$

- Finally, if observation errors and backgound errors are uncorrelated:

$$p(\mathbf{y} \text{ and } \mathbf{x}_b|\mathbf{x}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}_b|\mathbf{x})$$

$$\Rightarrow J(\mathbf{x}) = -\log p(\mathbf{y}|\mathbf{x}) - \log p(\mathbf{x}_b|\mathbf{x}) + K$$

Maximum Likelihood and Cost Function

- The maximum likelihood approach is applicable to any probability density functions $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x}_b|\mathbf{x})$.
- Consider the special case of Gaussian p.d.f's:

$$
\begin{aligned}
p(\mathbf{x}_b|\mathbf{x}) &= \frac{1}{(2\pi)^{N/2}|\mathbf{B}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{x}_b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}_b)\right] \\
p(\mathbf{y}|\mathbf{x}) &= \frac{1}{(2\pi)^{M/2}|\mathbf{R}|^{1/2}} \exp\left[-\frac{1}{2}[\mathcal{H}(\mathbf{x})-\mathbf{y}]^T\mathbf{R}^{-1}[\mathcal{H}(\mathbf{x})-\mathbf{y}]\right]
\end{aligned}
$$

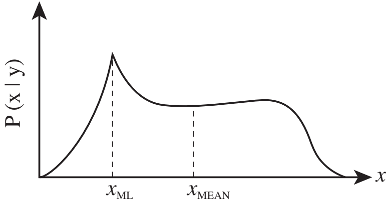
  where $\mathbf{B}$ and $\mathbf{R}$ are the background and observation error covariance matrices and $\mathcal{H}$ is the observation operator.
- With an appropriate choice of the constant:

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

- This is the variational data assimilation cost function.

Maximum Likelihood: Remarks

- The maximum-likelihood approach is general: as long as we know the p.d.f's, we can define the cost function.
  - Finding the global minimum may not be easy for non-Gaussian p.d.f's.
- In practice, background errors are usually assumed to be Gaussian (or a nonlinear transformation is applied to *make* them Gaussian).
- Non-Gaussian observation errors are taken into account.
  - Directionally-ambiguous wind observations from scatterometers,
  - Observations contaminated by occasional gross errors, which make outliers much more likely than implied by a Gaussian model.
- For Gaussian errors and linear observation operators, the maximum likelihood analysis coincides with the minimum variance solution. This is not the case in general:



Outline

1. The Maximum Likelihood Approach
2. 4D-Var (and 3D-Var)
3. Minimization and Incremental 4D-Var
4. Preconditioning
5. Operational 4D-Var Setup
6. Summary

3D-Var and 4D-Var

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

- We have not precisely defined the space over which the state variable $\mathbf{x}$ is defined or the observation operator $\mathcal{H}$.
- Depending on the choice of $\mathbf{x}$ and $\mathcal{H}$, the general approach described earlier will lead to different variational data assimilation methods.
- The simplest approach is to consider $\mathbf{x}$ as the state over the 3D spatial domain at analysis time, while $\mathcal{H}$ spatially interpolates this state and converts model variables to observed quantities: this is 3D-Var.
- Another more common approach is to consider $\mathbf{x}$ as the state over the 3D spatial domain and over the period for which observations are available, while $\mathcal{H}$ spatially and temporally interpolates this state and converts model variables to observed quantities: this is 4D-Var.

4D-Var

- We now discretize the assimilation window in time and define $\mathbf{x} = \{\mathbf{x}_i\}_{i=0,n}$ and $\mathbf{y} = \{\mathbf{y}_i\}_{i=0,n}$ where $\mathbf{x}_i$ and $\mathbf{y}_i$ are the state and observations at time t_i for $i = 0, \ldots, n$.
- Assuming that observation errors are uncorrelated in time, $\mathbf{R}$ is block diagonal, with blocks $\mathbf{R}_i$ corresponding to the observations at time t_i.
- The 4D-Var cost function is:

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_b)^T \mathbf{B}^{-1}(\mathbf{x}_0 - \mathbf{x}_b) + \frac{1}{2}\sum_{i=0}^{n}[\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i]^T \mathbf{R}_i^{-1}[\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i]$$

- $\mathcal{H}_i$ represents a spatial interpolation and transformation from model variables to observed variables (i.e. a 3D-Var-style observation operator).

Strong Constraint 4D-Var

- The states at various times are not independent: they are related through the forecast model:

$$\mathbf{x}_i = \mathcal{M}_i(\mathbf{x}_{i-1})$$

  where $\mathcal{M}_i$ is the forecast model integrated from time t_{i-1} to time t_i.

- By introducing the vectors $\mathbf{x}_i$, the unconstrained minimization problem:

$$\mathbf{x}_a = \arg\min_{\mathbf{x}} J(\mathbf{x})$$

  became a strong constraints minimization problem:

$$\begin{aligned} \mathbf{x}_a &= \arg\min_{\mathbf{x}_0} J(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k) \\ \text{subject to } \mathbf{x}_i &= \mathcal{M}_i(\mathbf{x}_{i-1}) \text{ for } i = 1, \ldots, n \end{aligned}$$

- This form of 4D-Var is called strong constraint 4D-Var.

Strong Constraint 4D-Var

- The 4D-Var cost function is:

$$
\begin{aligned}
J(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k) &= \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_b)^T \mathbf{B}^{-1} (\mathbf{x}_0 - \mathbf{x}_b) \\
&+ \frac{1}{2} \sum_{i=0}^{n} [\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i]^T \mathbf{R}_i^{-1} [\mathcal{H}_i(\mathbf{x}_i) - \mathbf{y}_i]
\end{aligned}
$$

- 4D-Var determines the analysis state at every gridpoint and at every time within the analysis window i.e. a four-dimensional analysis of the available asynoptic data.
- In deriving strong constraint 4D-Var, we have assumed that the observation operators and the model are perfect.
- As a consequence of the perfect model assumption, the analysis corresponds to a trajectory (i.e. an integration) of the forecast model.
- We will remove this assumption in the next lecture.

Outline

1. The Maximum Likelihood Approach
2. 4D-Var (and 3D-Var)
3. Minimization and Incremental 4D-Var
4. Preconditioning
5. Operational 4D-Var Setup
6. Summary

Minimizing the cost function

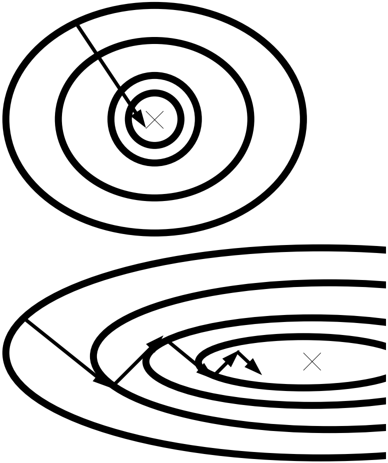
- We want to minimize the cost function:

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

- This is a very large-scale minimization problem ($\dim(\mathbf{x}) \approx 300 \times 10^6$ for the operational system at ECMWF.)
- Derivative-free algorithms are too slow (because each function evaluation gives very limited information about the shape of the cost function and in which direction the minimum might be).
- Practical algorithms for minimizing the cost function require its gradient.
- The simplest gradient-based minimization algorithm is called steepest descent:
  - Repeat until the gradient is sufficiently small:
  - Define a descent direction: $\mathbf{d}_k = -\nabla J(\mathbf{x}_k)$.
  - Find a step α_k, (line search) for which $J(\mathbf{x}_k + \alpha \mathbf{d}_k) < J(\mathbf{x}_k)$.
  - Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{d}_k$.

Minimizing the cost function

- Steepest descent can work well on very well conditioned problems in which the iso-surfaces of the cost function are nearly spherical.
- In this case, the steepest descent direction points towards the minimum.
- For poorly conditioned problems, with ellipsoidal iso-surfaces, steepest descent is not efficient.



Minimizing the cost function

- Steepest Descent is inefficient because it does not use information about the curvature (i.e. the second derivatives) of the cost function.
- The simplest algorithm that uses curvature is Newton's method.
- Newton's method uses a local quadratic approximation:

$$J(\mathbf{x} + \delta\mathbf{x}) \approx J(\mathbf{x}) + \delta\mathbf{x}^T \nabla J(\mathbf{x}) + \frac{1}{2}\delta\mathbf{x}^T J'' \delta\mathbf{x}$$

- Taking the gradient gives:

$$\nabla J(\mathbf{x} + \delta\mathbf{x}) \approx \nabla J(\mathbf{x}) + J'' \delta\mathbf{x}$$

- Since the gradient is zero at the minimum, Newton's method chooses the step at each iteration by solving:

$$J'' \delta\mathbf{x} = -\nabla J(\mathbf{x})$$

- Newton's method works well for cost functions that are well approximated by a quadratic function (i.e. for quasi-linear observation operators).

Minimizing the cost function

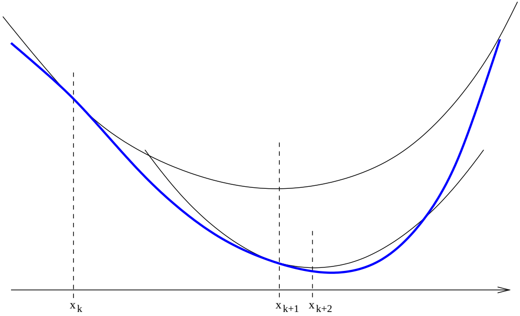
- Newton's method requires us to solve $J''\delta\mathbf{x}_k = -\nabla J(\mathbf{x}_k)$ at every iteration.
- J'' is a $\sim 10^8 \times 10^8$ matrix! Clearly, we cannot explicilty construct the matrix, or use direct methods to invert it.
- However, if we have a code that calculates Hessian-vector products, then we can use an iterative method (e.g. conjugate gradients) to solve for $\delta\mathbf{x}_k$.
- Such a code is called a second order adjoint. (See Wang, Navon, Le Dimet and Zou, 1992, Meteor. and Atmos. Phys.)
- Alternatively, some methods construct an approximation to J'' or $(J'')^{-1}$: these methods are called quasi-Newton methods.
- The most popular quasi-Newton method are the BFGS algorithm, (named after its creators Broyden, Fletcher, Goldfarb and Shanno) and its variant, the limited memory BFGS method.

Minimizing the cost function

- The methods presented so far apply to general nonlinear functions.
- An important special case occurs if the observation operator $\mathcal{H}$ is linear. In this case, the cost function is strictly quadratic, and the gradient is linear.
- In this case, it makes sense to determine the analysis by solving the linear equation $\nabla J(\mathbf{x}) = 0$.
- Since the matrix $J'' = \mathbf{B}^{-1} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}$ is symmetric and positive definite, the best algorithm to use is conjugate gradients.
- A good introduction to the method can be found online: *An Introduction to the Conjugate Gradient Method Without the Agonizing pain*, Shewchuk (1994).
- This will be useful in the incremental 4D-Var.

The Incremental Method

- A variant of the Newton method can be used: the nonlinear cost function is approximated by a quadratic cost function around the current guess. This quadratic cost function is minimized to provide an updated guess and the process is repeated.
- One complex problem is replaced by a series of (slightly) easier problems.



- The conjugate gradient algorithm can be used to solve efficiently the quadratic minimization problems.

The Incremental Method

- The cost function is written as a function of the correction to the first guess (the increment) $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}_g$:

$$\begin{aligned} J(\mathbf{x}_g + \delta\mathbf{x}) &= \frac{1}{2}(\mathbf{x}_g + \delta\mathbf{x} - \mathbf{x}_b)^T \mathbf{B}^{-1} (\mathbf{x}_g + \delta\mathbf{x} - \mathbf{x}_b) \\ &+ \frac{1}{2}[\mathcal{H}(\mathbf{x}_g + \delta\mathbf{x}) - \mathbf{y}]^T \mathbf{R}^{-1} [\mathcal{H}(\mathbf{x}_g + \delta\mathbf{x}) - \mathbf{y}] \end{aligned}$$

- The quadratic approximation of the cost function is obtained by linearizing around the curent guess:

$$J(\delta\mathbf{x}) = \frac{1}{2}(\delta\mathbf{x} + \mathbf{b})^T \mathbf{B}^{-1} (\delta\mathbf{x} + \mathbf{b}) + \frac{1}{2}(\mathbf{H}\delta\mathbf{x} + \mathbf{d})^T \mathbf{R}^{-1} (\mathbf{H}\delta\mathbf{x} + \mathbf{d})$$

where $\mathbf{b} = \mathbf{x}_g - \mathbf{x}_b$, $\mathbf{d} = \mathcal{H}(\mathbf{x}_g) - \mathbf{y}$ and $\mathbf{H}$ is the Jacobian of $\mathcal{H}$.

- The gradient is:

$$\nabla J(\delta\mathbf{x}) = \mathbf{B}^{-1}(\delta\mathbf{x} + \mathbf{b}) + \mathbf{H}^T \mathbf{R}^{-1} (\mathbf{H}\delta\mathbf{x} + \mathbf{d})$$

Calculating the Gradient: Tangent Linear and Adjoint

- To minimize the cost function, we must be able to calculate its gradient:

$$\nabla J(\delta \mathbf{x}) = \mathbf{B}^{-1}(\delta \mathbf{x} + \mathbf{b}) + \mathbf{H}^T \mathbf{R}^{-1}(\mathbf{H}\delta \mathbf{x} + \mathbf{d})$$

- The Jacobians $\mathbf{H}$ and $\mathbf{H}^T$ are much too large to be represented explicitly: we can only represent these as operators (subroutines) that calculate matrix-vector products.
- These codes are called the tangent linear code for $\mathbf{H}$ and the adjoint code for $\mathbf{H}^T$.
- For a good introduction about writing adjoints, see:
  X. Y. Huang and X. Yang, 1996, Variational Data Assimilation with the Lorenz model, HIRLAM Technical Report 26.

Writing the Adjoint Code

- Each line of the subroutine that applies $\mathcal{H}$ (including the forecast model) can be considered as a function h_k, so that

$$\mathcal{H}(\mathbf{x}) \equiv h_K \circ h_{K-1} \circ \cdots \circ h_1(\mathbf{x}).$$

- Each of the functions h_k can be linearized, to give the corresponding linear function $\mathbf{h}_k$. Each of these is extremely simple, and can be represented by one or two lines of code.
- The resulting code is called the tangent linear of $\mathcal{H}$ and:

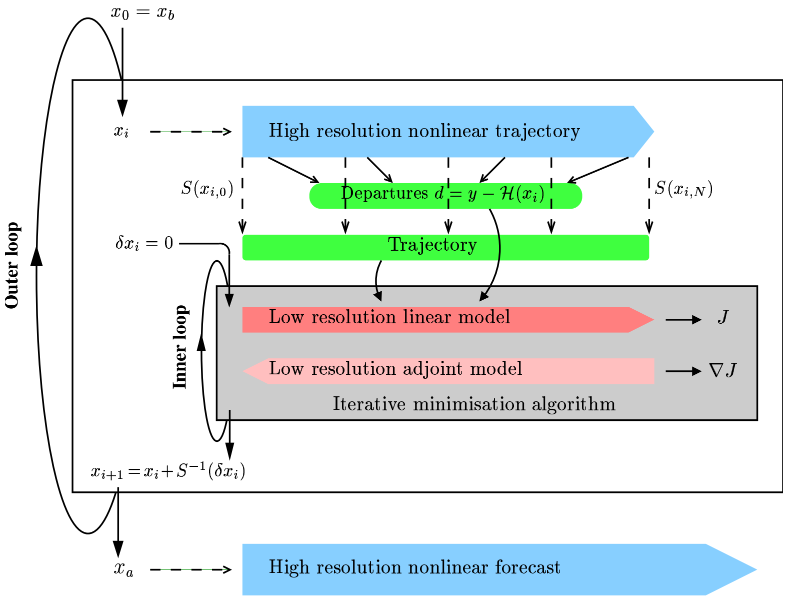
$$\mathbf{H}\delta\mathbf{x} \equiv \mathbf{h}_K \mathbf{h}_{K-1} \cdots \mathbf{h}_1 \delta\mathbf{x}$$

- The transpose, $\mathbf{H}^T \delta\mathbf{x} \equiv \mathbf{h}_1^T \mathbf{h}_2^T \cdots \mathbf{h}_K^T \delta\mathbf{x}$, is called the adjoint of $\mathcal{H}$.
- Again, each $\mathbf{h}_k^T$ is extremely simple – just a few lines of code.
- The difficulties in writing an adjoint can come from:
  - Non differentiable functions in the nonlinear cost function (model physics),
  - The length of the code (automatic tools can help).

The Incremental Method

- The 4D-Var cost function and its gradient can be evaluated for the cost of:
  - one integration of the forecast model,
  - one integration of the adjoint model.
- This cost is still prohibitive:
  - A typical minimization requires between 10 and 100 iterations,
  - The cost of the adjoint is typically 3 times that of the forward model.
  - The cost of the analysis would be roughly equivalent to between 20 and 200 days of model integration (with a 12h window).
- The incremental algorithm reduces the cost of 4D-Var by reducing the resolution of the model and using simplified physics (or by using a perturbation forecast model).
- The analysis increments are calculated at reduced resolution and must be interpolated to the high-resolution model's grid.
- The departures $\mathbf{d}$ are always evaluated using the full-resolution versions of $\mathcal{H}$ (and $\mathcal{M}$) i.e. the observations are always compared with the *full resolution* state.

Incremental 4D-Var



Outline

1. The Maximum Likelihood Approach
2. 4D-Var (and 3D-Var)
3. Minimization and Incremental 4D-Var
4. Preconditioning
5. Operational 4D-Var Setup
6. Summary

Preconditioning

- We noted that the steepest descent method works best if the iso-surfaces of the cost function are approximately spherical. This is generally true of all minimization algorithms.
- The degree of sphericity of the cost function can be measured by the eigenvalues of the Hessian. (Each eigenvalue corresponds to the curvature in the direction of the corresponding eigenvector.)
- In particular, the convergence rate will depend on the condition number:

$$\kappa = \lambda_{\max}/\lambda_{\min}$$

- The convergence can be accelerated by reducing the condition number of the Hessian.
- The Hessian of the 4D-Var cost function is $J'' = \mathbf{B}^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}$ (plus higher order terms).

Preconditioning

- We can speed up the convergence of the minimization by a change of variables $\chi = \mathbf{L}^{-1}\delta\mathbf{x}$ (i.e. $\delta\mathbf{x} = \mathbf{L}\chi$), where $\mathbf{L}$ is chosen to make the cost function more spherical.
- A common choice is $\mathbf{L} = \mathbf{B}^{1/2}$.
- The 4D-Var cost function becomes:

$$J(\chi) = \frac{1}{2}\chi^T\chi + \frac{1}{2}(\mathbf{HL}\chi - \mathbf{d})^T\mathbf{R}^{-1}(\mathbf{HL}\chi - \mathbf{d}).$$

- With this change of variables, the Hessian becomes:

$$J''_\chi = \mathbf{I} + \mathbf{L}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{HL}.$$

- The presence of the identity matrix in this expression guarantees that all eigenvalues are ≥ 1.
- There are no small eigenvalues to destroy the conditioning of the problem.

A case of poor convergence

- First experiments with direct assimilation of Meteosat radiance data showed significant analysis differences far away from observations.
- Differences disappeared when the number of iterations was increased: they were the result of insufficiant convergence.
- The conditioning of 4D-Var had been degraded by the inclusion of Meteosat radiance data.

Theoretical example

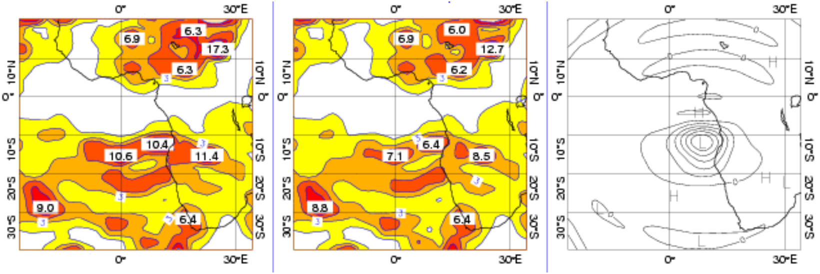
- Analysis of one variable at two locations.
- Background error σ_b with correlation α between the two points.
- n observations at each point with observation error σ_o ($\mathbf{R} = \sigma_o^2 \mathbf{I}$).
- Observation operator $\mathbf{H}$ is a $2n \times 2$ matrix with n rows equal to $(1 \;\; 0)$ and n rows equal to $(0 \;\; 1)$. This gives $\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} = n\mathbf{I}/\sigma_o^2$.
- The condition number is:

$$\kappa = \frac{\sigma_b^2(1+\alpha) + \sigma_o^2/n}{\sigma_b^2(1-\alpha) + \sigma_o^2/n}.$$

- If the grid points are close, $\alpha \approx 1$ and $\kappa = 2n(\sigma_b/\sigma_o)^2 + 1$.
- The conditioning of the problem deteriorates with:
  - increasing data density (larger n),
  - larger background error (σ_b),
  - more accurate data (smaller σ_o).

A case of poor convergence

- The lack of convergence with Meteosat data was traced back to too large humidity background error in very dry areas.



Condition Number	Original bg error	Modified bg error
Without Meteosat data	2229	2208
With Meteosat data	4495	2232

- Is there a more systematic way of achieving good preconditioning?

Hessian Preconditioning

- The incremental cost function can be written as:

$$J(\delta\mathbf{x}) = \frac{1}{2}\delta\mathbf{x}^T J'' \delta\mathbf{x} + \mathbf{g}^T \delta\mathbf{x} + c.$$

- Preconditioning replaces $J(\mathbf{x})$ by:

$$J_\chi(\chi) = \frac{1}{2}\chi^T \mathbf{L}^T J'' \mathbf{L}\chi + \mathbf{g}^T \mathbf{L}\chi + c.$$

- The difficulty is to find $\mathbf{L}$ such that $\kappa(\mathbf{L}^T J'' \mathbf{L}) \ll \kappa(J'')$
- A perfect preconditioner would be $\mathbf{L} = (J'')^{-1/2}$.
  (If $\kappa(\mathbf{L}^T J'' \mathbf{L}) = 1$, the minimisation converges in one iteration.)
- Is there an approximation of the Hessian that can easily be inverted?

Hessian Eigenvectors Preconditioning

- The Hessian can be written as: $J'' = \sum_{k=1}^{N} \lambda_k \mathbf{v}_k \mathbf{v}_k^T$
  where λ_k and $\mathbf{v}_k$ are its eigenvalues and eigenvectors.
- Preconditioning based on the K leading eigenvectors of J'':

$$\mathbf{L}^{-1} = \mathbf{I} + \sum_{k=1}^{K} (\mu_k^{1/2} - 1) \mathbf{v}_k \mathbf{v}_k^T$$

$$\text{gives} \quad J''_\chi = \sum_{k=1}^{K} \mu_k \lambda_k \mathbf{v}_k \mathbf{v}_k^T + \sum_{k=K+1}^{N} \lambda_k \mathbf{v}_k \mathbf{v}_k^T .$$

- Choose μ_k verifying $\mu_k \lambda_k < \lambda_{K+1}$, then

$$\kappa(J''_\chi) = \lambda_{K+1} / \lambda_N = \lambda_{K+1}.$$

- The eigenvectors can be computed using the Lanczos method, which is very closely related to the conjugate gradient algorithm.

Conjugate Gradient and Lanczos Algorithm

- The relation between Conjugate Gradient and Lanczos Algorithms allows to simultaneously:
  - Minimise the cost function,
  - Compute the eigenvectors and eigenvalues of the Hessian.
- The connection can be exploited to improve the minimisation:
  - At each outer loop iteration, the eigenvectors and eigenvalues of the Hessian can be computed,
  - They are used to precondition the minimisation in the following outer loop iteration.
- The 4D-Var eigenvectors are large scale:
  - They can be computed at low resolution and used to precondition a higher resolution miminization.
  - This leads to multi-incremental algorithm.
- It only applies to strictly quadratic inner loop cost functions: Var QC, QuickScat ambiguous winds in outer loop.

Superlinear Convergence and Rounding Error

- When λ_{max} or λ_{min} has converged in the Lanczos process, $J(\mathbf{x})$ has been *fully minimised* in the direction of the eigenvector.
- The minimisation then behaves as if it were minimising a problem with a smaller condition number $\kappa = \lambda_{max}/\lambda_{min}$.
- Conjugate Gradients *should* converge superlinearly.
- In practice, rounding errors spoil things, making the convergence linear.
- Rounding error causes the Lanczos algorithm to produce spurious multiple copies of eigenvectors.
- The two effects are connected.
- A well-known method of preventing spurious multiple eigenvalues in the Lanczos algorithm is to explicitely orthogonalise the gradient vectors. This method also restores superlinear convergence in CG.

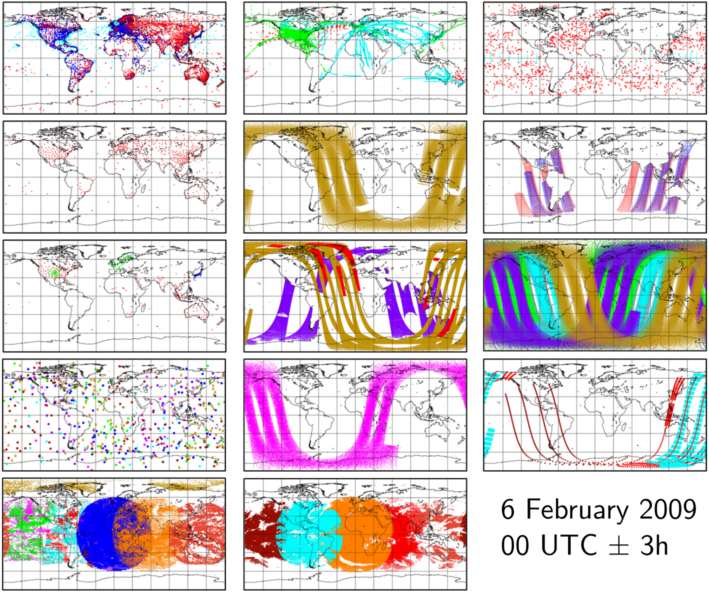
Outline

1. The Maximum Likelihood Approach
2. 4D-Var (and 3D-Var)
3. Minimization and Incremental 4D-Var
4. Preconditioning
5. Operational 4D-Var Setup
6. Summary

Operational 4D-Var Setup

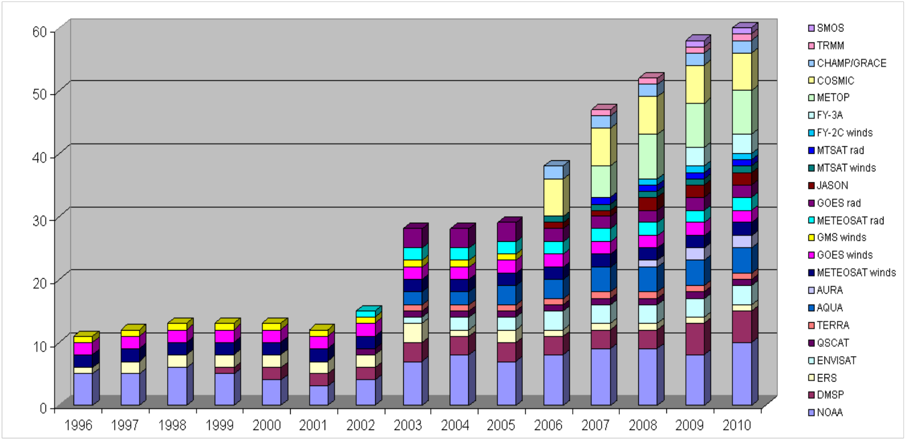
- Strong constraint 4D-Var has been used in operations at ECMWF since 25 Nov. 1997.
- The current configuration uses a 12h cycling window.
- The outer loop (and forecast) resolution is T799 (25km).
- The inner loops resolutions are T95, T159 and T255 (200, 125 and 80 km).
- On average, 9 million observations are assimilated per 12h cycle.
- 96% of assimilated data is from satellites.
- On average, 4D-Var runs on 1536 CPUs in 1h10.

Observation Coverage



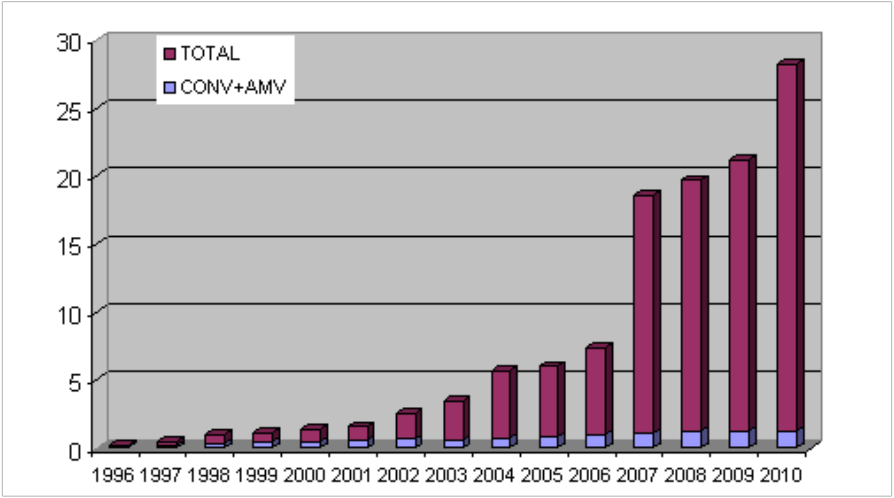
6 February 2009
00 UTC ± 3h

Observation Sources



Assimilating new data types requires a lot of ressources (developments and computer time).

Observation Numbers



Observation numbers have increased regularly and will increase even faster in the future.

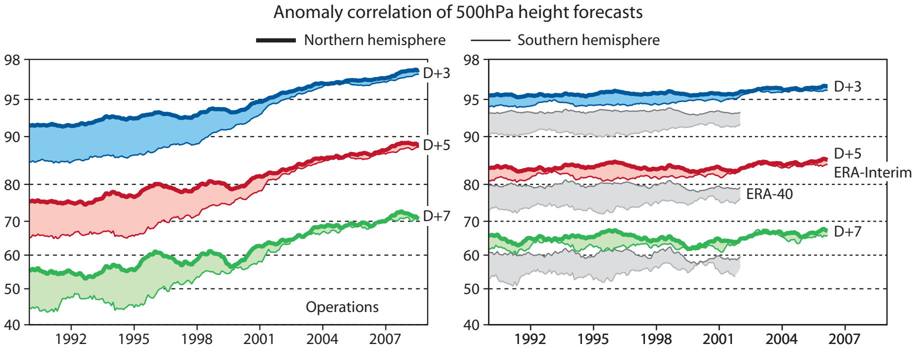
Observation Usage

Data count for one 12h 4D-Var cycle
(0900-2100 UTC, 3 March 2008)

	Screened		Assimilated	
Synop:	450,000	0.3%	64,000	0.7%
Aircraft:	434,000	0.3%	215,000	2.4%
Dribu:	24,000	0.02%	7,000	0.1%
Temp:	153,000	0.1%	76,000	0.8%
Pilot:	86,000	0.1%	39,000	0.4%
AMV's:	2,535,000	1.6%	125,000	1.4%
Radiance data:	150,663,000	96.9%	8,207,000	91.0%
Scat:	835,000	0.5%	149,000	1.7%
GPS radio occult.	271,000	0.2%	137,000	1.5%
TOTAL:	155,448,000	100.0%	9,018,000	100.0%

We are still far from using all available observations.

Performance



Forecast performance has increased regularly over the years.

Outline

1. The Maximum Likelihood Approach
2. 4D-Var (and 3D-Var)
3. Minimization and Incremental 4D-Var
4. Preconditioning
5. Operational 4D-Var Setup
6. Summary

Summary

- The Maximum Likelihood approach is general and can be in principle be applied to non-Gaussian, nonlinear analysis.
- 3D-Var and 4D-Var derive from the maximum likelihood principle.
- 4D-Var is an extension of 3D-Var to the case where observations are distributed in time.
- The cost function is minimized using algorithms based on knowledge of its gradient.
- The incremental method with appropriate preconditioning allows the computational cost to be reduced to acceptable levels.
- In strong constraint 4D-Var the model is assumed to be perfect, so that the four-dimensional analysis state corresponds to an integration (trajectory) of the model.

Summary

- Most (all?) of the main NWP centres run variational data assimilation schemes operationally.
  - ECMWF, United Kingdom, France, Germany, Canada, USA (NCEP, NRL, GMAO), Japan, Korea, Taiwan, China, Australia, HIRLAM countries, ALADIN countries...
- Forecast performance has improved over the years, in particular because of the ability of variational systems to adapt to and benefit from the varying components of the global observing system.
- Other aspects are important but were not covered in this talk:
  - Modelling of $\mathbf{B}$ and balance considerations,
  - Definition of the observation operators,
  - Observation variational bias correction.