Assimilation Algorithms
Lecture 2: 3D-Var

Sébastien Massart

ECMWF

22 May 2023

ECMWF

Outline

ECMWF **EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS** ©ECMWF

Outline

1. **From Optimal Interpolation to 3D-Var**

2. The Maximum Likelihood Approach

3. Minimisation

4. Summary

ECMWF EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS
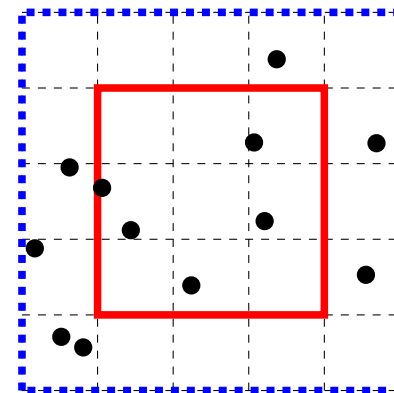
©ECMWF

From Optimal Interpolation to 3D-Var

✘ Previously in "Assimilation Algorithms": linear analysis equation

$$\mathbf{x}_a = \mathbf{x}_b + \mathbf{K}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$$

where

$$\mathbf{K} = \mathbf{P}^b\mathbf{H}^{\mathrm{T}}\left[\mathbf{H}\mathbf{P}^b\mathbf{H}^{\mathrm{T}} + \mathbf{R}\right]^{-1} \equiv \left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}$$

✘ Optimal Interpolation (OI) applies direct solution methods to invert the matrix $\left[\mathbf{H}\mathbf{P}^b\mathbf{H}^{\mathrm{T}} + \mathbf{R}\right]$.

✘ Data selection is applied to reduce the dimension of the matrix.

✘ Direct methods require access to the matrix elements. In particular, $\mathbf{H}\mathbf{P}^b\mathbf{H}^{\mathrm{T}}$ must be available in matrix form.

✘ This limits us to very simple observation operators.

Analysis box
Used observations
Selection area

From Optimal Interpolation to 3D-Var

- ✗ Linear analysis equation: $\mathbf{x}_a = \mathbf{x}_b + \mathbf{K}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$
- ✗ For $\mathbf{K} = \mathbf{P}^b\mathbf{H}^\mathrm{T}\left[\mathbf{H}\mathbf{P}^b\mathbf{H}^\mathrm{T} + \mathbf{R}\right]^{-1}$

we have $\mathbf{x}_a = \mathbf{x}_b + \mathbf{P}^b\mathbf{H}^\mathrm{T}\left[\mathbf{H}\mathbf{P}^b\mathbf{H}^\mathrm{T} + \mathbf{R}\right]^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$

if $\mathbf{z} = \left[\mathbf{H}\mathbf{P}^b\mathbf{H}^\mathrm{T} + \mathbf{R}\right]^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$

we have to solve $\left[\mathbf{H}\mathbf{P}^b\mathbf{H}^\mathrm{T} + \mathbf{R}\right]\mathbf{z} = \mathbf{y} - \mathcal{H}(\mathbf{x}_b)$

and then $\mathbf{x}_a = \mathbf{x}_b + \mathbf{P}^b\mathbf{H}^\mathrm{T}\mathbf{z}$

From Optimal Interpolation to 3D-Var

✗ Linear analysis equation: $\mathbf{x}_a = \mathbf{x}_b + \mathbf{K}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$

✗ For $\mathbf{K} = \left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}$

we have $\mathbf{x}_a = \mathbf{x}_b + \left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$

if $\delta\mathbf{x} = \left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$

we have to solve $\left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]\delta\mathbf{x} = \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$

and then $\mathbf{x}_a = \mathbf{x}_b + \delta\mathbf{x}$

From Optimal Interpolation to 3D-Var

✗ There are two forms to solve the linear analysis equation, depending which expression we adopt for **K**:

✗ For $\mathbf{K} = \mathbf{P}^b\mathbf{H}^{\mathrm{T}}\left[\mathbf{H}\mathbf{P}^b\mathbf{H}^{\mathrm{T}} + \mathbf{R}\right]^{-1}$ we have $\mathbf{x}_a = \mathbf{x}_b + \mathbf{P}^b\mathbf{H}^{\mathrm{T}}\,\mathbf{z}$ and:

$$\left[\mathbf{H}\mathbf{P}^b\mathbf{H}^{\mathrm{T}} + \mathbf{R}\right]\,\mathbf{z} = \mathbf{y} - \mathcal{H}(\mathbf{x}_b)$$

✗ For $\mathbf{K} = \left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}$, we have $\mathbf{x}_a = \mathbf{x}_b + \delta\mathbf{x}$ and:

$$\left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]\,\delta\mathbf{x} = \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$$

✗ The linear analysis equation could be solved as an equation of the form:

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

✗ The first of these alternatives is called PSAS (Physicalspace Statistical Analysis System)

✗ The second (although it may not look like it yet) is 3D-Var

From Optimal Interpolation to 3D-Var

Problem

✘ Find the solution $\mathbf{x}_a$ of the linear system:

$$\mathbf{Ax} = \mathbf{b}.$$

Direct methods

✘ Direct methods attempt to solve the problem by a finite sequence of operations.

✘ In the absence of rounding errors, direct methods would deliver an exact solution $\mathbf{x}_a$ of the linear system.

Iterative methods

✘ Beginning with an approximation to the solution $\mathbf{x}_0$, an iterative method is a mathematical procedure that generates a sequence of improving approximate solutions $\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_n$.

✘ The n-th approximation is derived from the previous ones.

✘ The sequence of solutions converges to the exact solution.

From Optimal Interpolation to 3D-Var

Iterative methods have significant advantages over the direct methods used in Optimal Interpolation (OI):

- ✗ They can be applied to much larger problems than direct techniques, so we can avoid data selection.
- ✗ They do not require access to the matrix elements.
- ✗ Typically, to solve $\mathbf{Ax} = \mathbf{b}$, requires only the ability to calculate matrix-vector products: $\mathbf{Ax}$.
- ✗ This allows us to use operators defined by pieces of code rather than explicitly as matrices.
- ✗ Examples of such operators include radiative-transfer codes, numerical models, Fourier transforms, etc.

Example: Conjugate Gradients

To solve $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is real, symmetric and positive-definite:

$\mathbf{r}_0 := \mathbf{b} - \mathbf{Ax}_0 \qquad \mathbf{p}_0 := \mathbf{r}_0 \qquad k := 0$;

while $\mathbf{r}_{k+1}$ *is too large* **do**

    `/* Step in the direction of` $\mathbf{p}_k$ `*/`

    $\alpha_k := \dfrac{\mathbf{r}_k^{\mathrm{T}} \mathbf{r}_k}{\mathbf{p}_k^{\mathrm{T}} \mathbf{Ap}_k}$;

    `/* New state`   `*/`

    $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$;

    `/* New residual`   `*/`

    $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{Ap}_k$;

    `/* New direction of descent`   `*/`

    $\beta_k := \dfrac{\mathbf{r}_{k+1}^{\mathrm{T}} \mathbf{r}_{k+1}}{\mathbf{r}_k^{\mathrm{T}} \mathbf{r}_k}$;

    $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$;

    `/* Next iteration`   `*/`

    $k := k + 1$;

end

The result is x_{k+1}

Outline

ECMWF EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS

©ECMWF

3D-Var

✗ As we have seen, (linear) 3D-Var analysis can be seen as an application of iterative solution methods to the linear analysis equation.

✗ Historically, 3D-Var was not developed this way.

✗ We will now consider this alternative derivation.

✗ We will need to apply Bayes' theorem:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

where $p(A|B)$ is the probability of A given B, etc.

Maximum Likelihood

✗ We developed the linear analysis equation by searching for a linear combination of observation and background that minimised the variance of the error.

✗ An alternative approach is to look for the most probable solution $\mathbf{x}_a$, given the observations $\mathbf{Y}$ and having a prior knowledge $\mathbf{x}_b$ on the solution:

$$\mathbf{x}_a = \arg\max_{\mathbf{x}} \left[p(\mathbf{x}|\mathbf{y}) \right]$$

✗ It will be convenient to define a cost function

$$J(\mathbf{x}) = -\log \left[p(\mathbf{x}|\mathbf{y}) \right] + const.$$

✗ Then, since log is a monotonic function:

$$\mathbf{x}_a = \arg\min_{\mathbf{x}} \left[J(\mathbf{x}) \right]$$

Maximum Likelihood

✗ Applying Bayes' theorem gives:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})} \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$

✗ The maximum likelihood approach is applicable to any probability density functions $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x})$.

✗ However, let us consider the special case of Gaussian p.d.f's:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{N/2}|\mathbf{P}^b|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_b)^{\mathrm{T}}\mathbf{P}^{b^{-1}}(\mathbf{x}-\mathbf{x}_b)\right\}$$

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{(2\pi)^{M/2}|\mathbf{R}|^{1/2}} \exp\left\{-\frac{1}{2}[\mathbf{y}-\mathcal{H}(\mathbf{x})]^{\mathrm{T}}\mathbf{R}^{-1}[\mathbf{y}-\mathcal{H}(\mathbf{x})]\right\}$$

Maximum Likelihood

✘ Gaussian p.d.f's:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{N/2}|\mathbf{P}^b|^{1/2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{x}_b)^{\mathrm{T}}\mathbf{P}^{b^{-1}}(\mathbf{x}-\mathbf{x}_b)\right\}$$

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{(2\pi)^{M/2}|\mathbf{R}|^{1/2}}\exp\left\{-\frac{1}{2}[\mathbf{y}-\mathcal{H}(\mathbf{x})]^{\mathrm{T}}\mathbf{R}^{-1}[\mathbf{y}-\mathcal{H}(\mathbf{x})]\right\}$$

✘ Now,

$$J(\mathbf{x}) = -\log\left[p(\mathbf{x}|\mathbf{y})\right]+const.$$
$$= -\log\left[p(\mathbf{y}|\mathbf{x})\right]-\log\left[p(\mathbf{x})\right]+const$$

✘ Hence, with an appropriate choice of the constant *const*.:

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

✘ This is the 3D-Var cost function

Maximum Likelihood

✘ Let us introduce the dot product:

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \sum_{i=1}^{n} x_1(i)\, x_2(i) = \mathbf{x}_1^T \mathbf{x}_2$$

✘ The dot product is symmetric:

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \langle \mathbf{x}_2, \mathbf{x}_1 \rangle$$

✘ Let us introduce the matrix $\mathbf{A}$

$$
\begin{aligned}
\langle \mathbf{x}_1, \mathbf{A}\mathbf{x}_2 \rangle &= \mathbf{x}_1^T \mathbf{A}\, \mathbf{x}_2 \\
&= \left(\mathbf{A}^T \mathbf{x}_1 \right)^T \mathbf{x}_2 \\
&= \langle \mathbf{A}^T \mathbf{x}_1, \mathbf{x}_2 \rangle
\end{aligned}
$$

✘ $\mathbf{A}^T$ is the adjoint of $\mathbf{A}$:

$$\langle \mathbf{x}_1, \mathbf{A}\mathbf{x}_2 \rangle = \langle \mathbf{A}^T \mathbf{x}_1, \mathbf{x}_2 \rangle$$

✘ If $\mathbf{A}$ is symmetric ($\mathbf{A}^T = \mathbf{A}$):

$$\langle \mathbf{x}_1, \mathbf{A}\mathbf{x}_2 \rangle = \langle \mathbf{A}\mathbf{x}_1, \mathbf{x}_2 \rangle$$

Maximum Likelihood

✗ The maximum likelihood analysis corresponds to the global minimum of the cost function (using the previously defined dot product):

$$J(\mathbf{x}) = \frac{1}{2}\left\langle \left[\mathbf{x}-\mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\left[\mathbf{x}-\mathbf{x}_b\right]\right\rangle + \frac{1}{2}\left\langle \left[\mathbf{y}-\mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\left[\mathbf{y}-\mathcal{H}(\mathbf{x})\right]\right\rangle$$

✗ At the minimum $\mathbf{x}_a$, the gradient of the cost function is zero:

$$\nabla J(\mathbf{x}) = 0$$

✗ The Taylor series of the cost function is (at the first order)

$$J(\mathbf{x}+\delta\mathbf{x}) = J(\mathbf{x}) + \left\langle \delta\mathbf{x}, \nabla J(\mathbf{x})\right\rangle$$

✗ Let introduce a perturbation $\delta\mathbf{x}$ of $\mathbf{x}$.

✗ If $\mathcal{H}$ is linear (or if we neglect second-order terms) then

$$\mathcal{H}(\mathbf{x}+\delta\mathbf{x}) = \mathcal{H}(\mathbf{x}) + \mathbf{H}\delta\mathbf{x}.$$

ECMWF EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS   ©ECMWF

Maximum Likelihood

✗ The cost function evaluated at $\mathbf{x} + \delta\mathbf{x}$ is

$$J(\mathbf{x} + \delta\mathbf{x}) = \frac{1}{2}\Big\langle \big[(\mathbf{x} - \mathbf{x}_b) + \delta\mathbf{x}\big], \mathbf{P}^{b^{-1}}\big[(\mathbf{x} - \mathbf{x}_b) + \delta\mathbf{x}\big]\Big\rangle$$

$$+ \frac{1}{2}\Big\langle \big[(\mathbf{y} - \mathcal{H}(\mathbf{x})) - \mathbf{H}\delta\mathbf{x}\big], \mathbf{R}^{-1}\big[(\mathbf{y} - \mathcal{H}(\mathbf{x})) - \mathbf{H}\delta\mathbf{x}\big]\Big\rangle$$

where we will neglect the second order terms $\langle \cdots \delta\mathbf{x}, \cdots \delta\mathbf{x}\rangle$

Maximum Likelihood

✘ The cost function evaluated at $\mathbf{x} + \delta\mathbf{x}$ is

$$
\begin{aligned}
J(\mathbf{x} + \delta\mathbf{x}) \approx{} & \frac{1}{2}\left\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \right\rangle \\
& + \frac{1}{2}\left\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\delta\mathbf{x} \right\rangle + \frac{1}{2}\left\langle \delta\mathbf{x}, \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \right\rangle \\
& + \frac{1}{2}\left\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \right\rangle \\
& - \frac{1}{2}\left\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\mathbf{H}\delta\mathbf{x} \right\rangle - \frac{1}{2}\left\langle \mathbf{H}\delta\mathbf{x}, \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \right\rangle
\end{aligned}
$$

Maximum Likelihood

✘ The cost function evaluated at $\mathbf{x} + \delta\mathbf{x}$ is

$$J(\mathbf{x} + \delta\mathbf{x}) \approx \frac{1}{2}\left\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \right\rangle$$

$$+ \frac{1}{2}\left\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\delta\mathbf{x} \right\rangle + \frac{1}{2}\left\langle \delta\mathbf{x}, \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \right\rangle$$

$$+ \frac{1}{2}\left\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \right\rangle$$

$$- \frac{1}{2}\left\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\mathbf{H}\delta\mathbf{x} \right\rangle - \frac{1}{2}\left\langle \mathbf{H}\delta\mathbf{x}, \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \right\rangle$$

Maximum Likelihood

✘ The cost function evaluated at $\mathbf{x} + \delta\mathbf{x}$ is

$$J(\mathbf{x} + \delta\mathbf{x}) \approx \frac{1}{2}\left\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \right\rangle$$

$$+ \frac{1}{2}\left\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\delta\mathbf{x} \right\rangle + \frac{1}{2}\left\langle \delta\mathbf{x}, \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \right\rangle$$

$$+ \frac{1}{2}\left\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \right\rangle$$

$$- \frac{1}{2}\left\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\mathbf{H}\delta\mathbf{x} \right\rangle - \frac{1}{2}\left\langle \mathbf{H}\delta\mathbf{x}, \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \right\rangle$$

Maximum Likelihood

✘ The cost function evaluated at $\mathbf{x} + \delta\mathbf{x}$ is

$$
\begin{aligned}
J(\mathbf{x} + \delta\mathbf{x}) \approx\ & \frac{1}{2}\Big\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \Big\rangle \\
& + \frac{1}{2}\Big\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\delta\mathbf{x} \Big\rangle + \frac{1}{2}\Big\langle \delta\mathbf{x}, \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \Big\rangle \\
& + \frac{1}{2}\Big\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \Big\rangle \\
& - \frac{1}{2}\Big\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\mathbf{H}\delta\mathbf{x} \Big\rangle - \frac{1}{2}\Big\langle \mathbf{H}\delta\mathbf{x}, \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \Big\rangle
\end{aligned}
$$

Maximum Likelihood

✘ The cost function evaluated at $\mathbf{x} + \delta\mathbf{x}$ is

$$J(\mathbf{x} + \delta\mathbf{x}) \approx \frac{1}{2}\left\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \right\rangle$$

$$+ \frac{1}{2}\left\langle \left[\mathbf{x} - \mathbf{x}_b\right], \mathbf{P}^{b^{-1}}\delta\mathbf{x} \right\rangle + \frac{1}{2}\left\langle \delta\mathbf{x}, \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \right\rangle$$

$$+ \frac{1}{2}\left\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \right\rangle$$

$$- \frac{1}{2}\left\langle \left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right], \mathbf{R}^{-1}\mathbf{H}\delta\mathbf{x} \right\rangle - \frac{1}{2}\left\langle \mathbf{H}\delta\mathbf{x}, \mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \right\rangle$$

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

$$+ \left\langle \delta\mathbf{x}, \mathbf{P}^{b^{-1}}\left[\mathbf{x} - \mathbf{x}_b\right] \right\rangle$$

$$- \left\langle \delta\mathbf{x}, \mathbf{H}^T\mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x})\right] \right\rangle$$

Maximum Likelihood

✘ The maximum likelihood analysis corresponds to the global minimum of the cost function (using the previously defined dot product):

$$J(\mathbf{x}) = \frac{1}{2}\left\langle [\mathbf{x} - \mathbf{x}_b]\,,\, \mathbf{P}^{b^{-1}}[\mathbf{x} - \mathbf{x}_b] \right\rangle + \frac{1}{2}\left\langle [\mathbf{y} - \mathcal{H}(\mathbf{x})]\,,\, \mathbf{R}^{-1}[\mathbf{y} - \mathcal{H}(\mathbf{x})] \right\rangle$$

✘ The cost function evaluated at $\mathbf{x} + \delta\mathbf{x}$ is (at the first order)

$$J(\mathbf{x} + \delta\mathbf{x}) = J(\mathbf{x}) + \left\langle \delta\mathbf{x}\,,\, \mathbf{P}^{b^{-1}}[\mathbf{x} - \mathbf{x}_b] - \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}[\mathbf{y} - \mathcal{H}(\mathbf{x})] \right\rangle.$$

✘ The Taylor series of the cost function is (at the first order)

$$J(\mathbf{x} + \delta\mathbf{x}) = J(\mathbf{x}) + \left\langle \delta\mathbf{x}\,,\, \nabla J(\mathbf{x}) \right\rangle$$

✘ We deduce the gradient of the cost function

$$\nabla J(\mathbf{x}) = \mathbf{P}^{b^{-1}}[\mathbf{x} - \mathbf{x}_b] + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}[\mathcal{H}(\mathbf{x}) - \mathbf{y}]$$

Maximum Likelihood

✘ At the minimum $\mathbf{x}_a$, the gradient of the cost function ($\nabla J(\mathbf{x})$) is zero:

$$\nabla J(\mathbf{x}_a) = \mathbf{P}^{b^{-1}}\left[\mathbf{x}_a - \mathbf{x}_b\right] + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathcal{H}(\mathbf{x}_a) - \mathbf{y}\right] = \mathbf{0}$$

✘ Now, if $\mathcal{H}$ is linear (or if we neglect second-order terms) then

$$\mathcal{H}(\mathbf{x}_a) = \mathcal{H}(\mathbf{x}_b) + \mathbf{H}\delta\mathbf{x}_a \quad \text{where} \quad \delta\mathbf{x}_a = \mathbf{x}_a - \mathbf{x}_b$$

✘ Hence:

$$\mathbf{P}^{b^{-1}}\delta\mathbf{x}_a + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathcal{H}(\mathbf{x}_b) - \mathbf{y}\right] + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\delta\mathbf{x}_a = \mathbf{0}$$

✘ Rearranging a little gives:

$$\left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]\delta\mathbf{x}_a = \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$$

✘ This is exactly the equation for the minimum-variance analysis we derived at the start of the lecture!

Maximum Likelihood

✘ We have shown that the maximum likelihood approach is naturally expressed in terms of a cost function representing minus the log of the probability of the analysis state.

✘ The minimum of the cost function corresponds to the maximum likelihood (probability) solution.

✘ For Gaussian errors and linear observation operators, the maximum likelihood analysis coincides with the minimum variance solution.

✘ This is not the case in general:

ECMWF
EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS
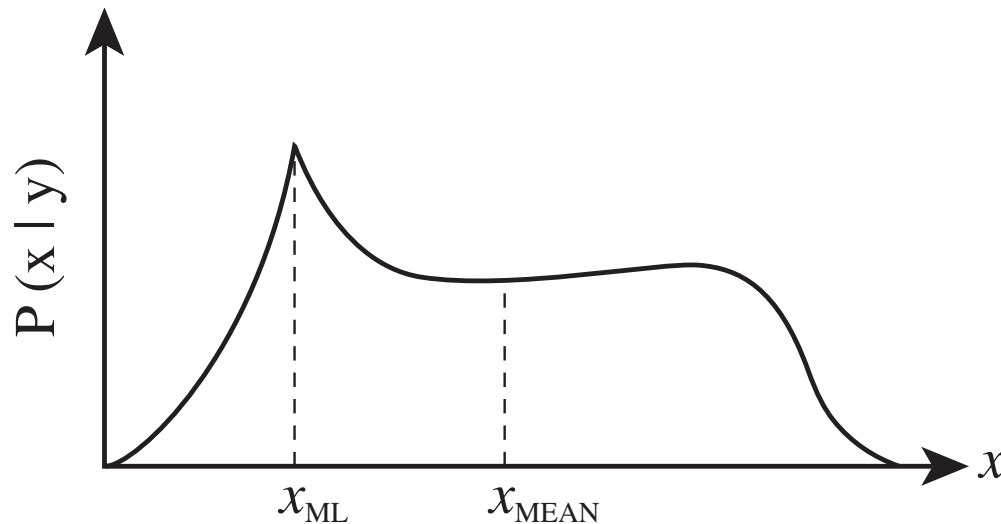
Maximum Likelihood

✗ In the nonlinear case, the minimum variance approach is difficult to apply.

✗ The maximum-likelihood approach is much more generally applicable

✗ As long as we know the p.d.f's, we can define the cost function
  ⇨ However, finding the global minimum may not be easy for highly 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).
  ⇨ This makes the background-error term of the cost function quadratic.

✗ However, non-Gaussian observation errors are taken into account. For example:
  ⇨ Directionally-ambiguous wind observations from scatterometers
  ⇨ Observations contaminated by occasional gross errors, which make outliers much more likely than implied by a Gaussian model.

Outline

Minimisation

✗ In 3D-Var, the analysis is found by minimising the cost function:

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

✗ This is a very large-scale $(\dim(\mathbf{x}) \approx 10^8)$ minimisation problem.

✗ The size of the problem restricts on the algorithms we can use.

✗ Derivative-free algorithms (which require only the ability to calculate $J(\mathbf{x})$ for arbitrary $\mathbf{x}$) are too slow.

✗ This is because each function evaluation gives very limited information about the shape of the cost function.
  ↪ E.g. a finite difference, $J(\mathbf{x} + \delta\mathbf{v}) - J(\mathbf{x}) \approx \delta\mathbf{v}^{\mathrm{T}} \nabla J(\mathbf{x})$, gives a single component of the gradient.
  ↪ We need $O(10^8)$ components to work out which direction is "downhill".

Minimisation

✗ Practical algorithms for minimising the 3D-Var cost function require us to calculate its gradient:

$$\nabla J(\mathbf{x}) = \mathbf{P}^{b^{-1}}(\mathbf{x} - \mathbf{x}_b) + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathcal{H}(\mathbf{x}) - \mathbf{y}\right]$$

✗ The simplest gradient-based minimisation algorithm is called steepest descent:

Let $\mathbf{x}_0$ be an initial guess of the analysis;

while *gradient is not sufficiently small* **do**

```
    /* Define a descent direction                                    */
```
$\mathbf{d}_k = -\nabla J(\mathbf{x}_k)$;
```
    /* Find a step αₖ, e.g.  by line minimisation of the
```
$\qquad$ function $J(\mathbf{x}_k + \alpha\mathbf{d}_k)$, `for which` $J(\mathbf{x}_k + \alpha\mathbf{d}_k) < J(\mathbf{x}_k)$ `*/`
$\alpha_k = \cdots$;
```
    /* Compute the new estimate                                      */
```
$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{d}_k$;
```
    /* Next step                                                     */
```
$k = k + 1$

end

Minimisation

✘ Steepest descent can work well on problems in which the iso-surfaces of the cost function are nearly spherical.

⇨ In this case, the steepest descent direction points towards the minimum.

⇨ They are very well conditioned problems.

✘ For problems with ellipsoidal iso-surfaces, steepest descent is not efficient.

⇨ They are poorly conditioned problems.
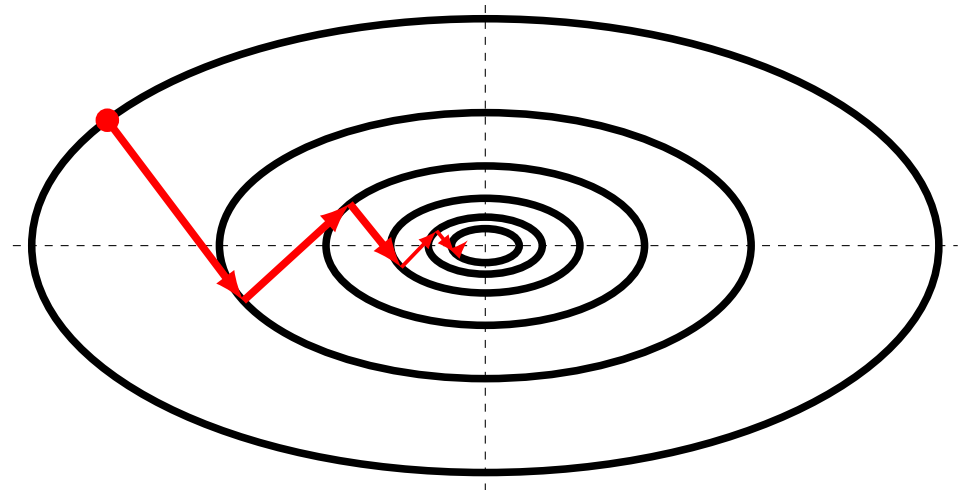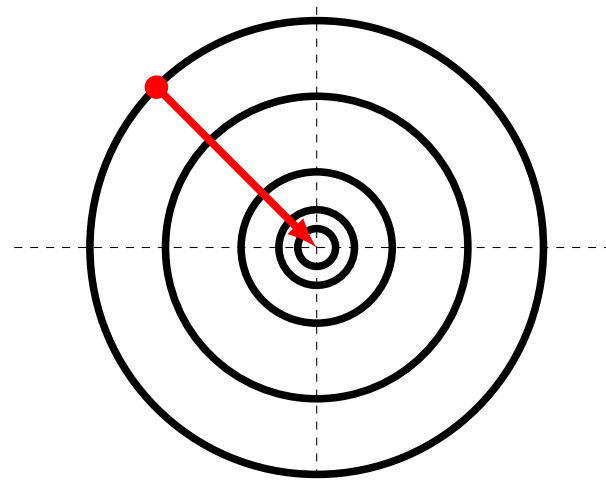
Curvature

✘ We define the curvature as the amount by which a line deviates from being straight.
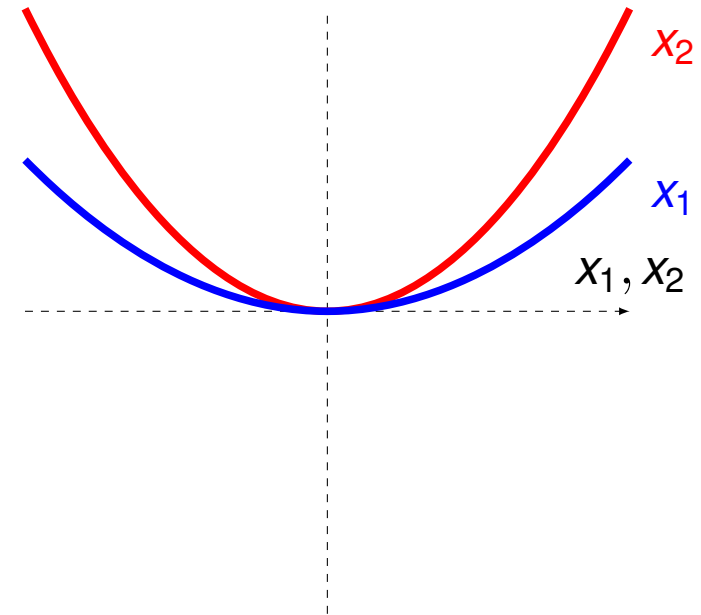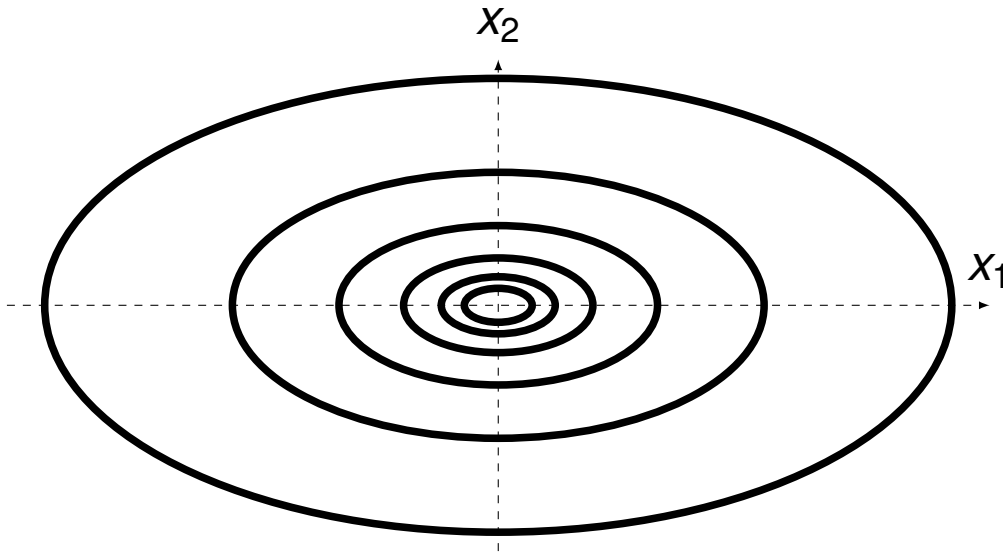
✘ The degree of sphericity of the cost function can be measured by the eigenvalues of the Hessian (matrix J'' of second derivatives of J).

⇨ Each eigenvalue corresponds to the curvature in the direction of the corresponding eigenvector.

Preconditioning

✗ The steepest descent method works best if the iso-surfaces of the cost function are approximately spherical.

✗ This is generally true of all minimisation algorithms.

✗ In particular, the convergence rate will depend on the condition number:

$$\kappa = \frac{\lambda_{\max}}{\lambda_{\min}},$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum eigenvalues respectively.

✗ In general, expressing the cost function directly in terms of **x** will not lead to spherical iso-surfaces.

Preconditioning

✘ We can speed up the convergence of the minimisation by a change of variables $\chi = \mathbf{L}^{-1}(\mathbf{x} - \mathbf{x}_b)$, where $\mathbf{L}$ is chosen to make the cost function more spherical.

✘ A common choice is $\mathbf{L} = \mathbf{P}^{b1/2}$. The cost function becomes:

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

✘ With this change of variables, the Hessian becomes:

$$J''_{\chi} = \mathbf{I} + \mathbf{L}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{L} \text{ (plus higher order terms)}$$

✘ The presence of the identity matrix in this expression guarantees that the minimum eigenvalue is ≥ 1.

✘ There are no small eigenvalues to destroy the conditioning of the problem.

Newton's methods

✗ Steepest Descent is inefficient because it does not use information about the curvature of the cost function.

✗ The simplest algorithms that use curvature are in the family of Newton's methods.

✗ Newton's methods use a local quadratic approximation:

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

✗ 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 methods

✘ Newton's method:
Start with an initial guess, $\mathbf{x}_0$;
while *gradient is not sufficiently small* **do**

> /* Solve $J''\delta\mathbf{x}_k = -\nabla J(\mathbf{x}_k)$ */
> $\delta\mathbf{x}_k = \cdots$;
> /* Compute the new estimate */
> $\mathbf{x}_{k+1} = \mathbf{x}_k + \delta\mathbf{x}_k$;
> /* Next step */
> $k = k+1$

end

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

✘ However, it suffers from several problems . . .

⇝ There is no control on the step length $\|\delta\mathbf{x}\|$.

⇝ The method can make huge jumps into regions where the local quadratic approximation is poor.

✘ This can be controlled using line searches, or by trust region methods that limit the step size to a region where the approximation is valid.

Newton's methods

✗ Newton's method requires us to solve $J'' \delta \mathbf{x}_k = -\nabla J(\mathbf{x}_k)$ at every iteration.

✗ Now, J'' is a $\sim 10^8 \times 10^8$ matrix! Clearly, we cannot explicitly 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 call a second order adjoint. See Wang, Navon, LeDimet, Zou, 1992 Meteor. and Atmos. Phys. 50, pp3-20 for details.

✗ Alternatively, we can use a method that constructs an approximation to $(J'')^{-1}$.

✗ Methods based on approximations of J'' or $(J'')^{-1}$ are called quasi-Newton methods.

Newton's methods

✗ By far the most popular quasi-Newton method is the BFGS algorithm, named after its creators Broyden, Fletcher, Goldfarb and Shanno.

✗ The BFGS method builds up an approximation to the Hessian:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k \mathbf{s}_k^{\mathrm{T}}} - \frac{\mathbf{B}_k \mathbf{s}_k (\mathbf{B}_k \mathbf{s}_k)^{\mathrm{T}}}{\mathbf{s}_k \mathbf{B}_k \mathbf{s}_k^{\mathrm{T}}}$$

where $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \nabla J(\mathbf{x}_{k+1}) - \nabla J(\mathbf{x}_k)$.

✗ The approximation is symmetric and positive definite, and satisfies

$$\nabla J(\mathbf{x}_{j+1}) - \nabla J(\mathbf{x}_j) = J''(\mathbf{x}_{j+1} - \mathbf{x}_j) \qquad \text{for } j = 0, 1, \cdots, k$$

✗ There is an explicit expression for the inverse of $\mathbf{B}_k$, which allows Newton's equation to be solved at the cost of $O(Nk)$ operations.

Newton's methods

✘ The BFGS quasi-Newton method:

Start with an initial guess, $\mathbf{x}_0$;

Start with an initial approximation of the Hessian (typically, $\mathbf{B}_0 = \mathbf{I}$);

while *gradient is not sufficiently small* **do**

```
/* Solve the approximate Newton's equation,
   Bₖδxₖ = −∇J(xₖ), to determine the search direction.
   */
```

$\delta\mathbf{x}_k = \cdots;$

```
/* Perform a line search to find a step αₖ for which for
   which J(xₖ + αₖδxₖ) < J(xₖ)                          */
```

$\alpha_k = \cdots;$

```
/* Compute the new estimate                            */
```

$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\delta\mathbf{x}_k;$

```
/* Generate an updated approximation to the Hessian    */
```

$\mathbf{B}_{k+1} = \cdots;$

```
/* Next step                                           */
```

$k = k + 1$

end

Newton's methods

The BFGS quasi-Newton method

✗ As k increases, the cost of storing and applying the approximate Hessian increases linearly.

✗ Moreover, the vectors $\mathbf{s}_k$ and $\mathbf{y}_k$ generated many iterations ago no longer provide accurate information about the Hessian.

✗ It is usual to construct $\mathbf{B}_k$ from only the $O(10)$ most recent iterations.

✗ The algorithm is then called the limited memory BFGS method.

Newton's methods

✘ 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:

$$
\begin{aligned}
\nabla J(\mathbf{x}) &= \mathbf{P}^{b^{-1}} \delta\mathbf{x} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathbf{H}\mathbf{x}_b + \mathbf{H}\,\delta\mathbf{x} - \mathbf{y}\right] \\
&= \left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]\delta\mathbf{x} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\left[\mathbf{H}\mathbf{x}_b - \mathbf{y}\right]
\end{aligned}
$$

✘ In this case, it makes sense to determine the analysis by solving the linear equation $\nabla J(\mathbf{x}) = \mathbf{0}$.

✘ Since the matrix $\left[\mathbf{P}^{b^{-1}} + \mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\right]$ is symmetric and positive definite, the best algorithm to use is conjugate gradients. The algorithm was presented earlier in this lecture.

✘ A good introduction to the method can be found online: Shewchuk (1994) "An Introduction to the Conjugate Gradient Method Without the Agonizing pain".

ECMWF Lanczos algorithm

✘ We precondition using $\mathbf{L} = \mathbf{P}^{b^{1/2}}$, the gradient (with respect to χ) is:

$$\nabla_\chi J(\chi) = \chi + \mathbf{L}^{\mathrm{T}} \mathbf{H}^{\mathrm{T}} \mathbf{R}^{-1} \left(\mathbf{y} - \mathcal{H}(\mathbf{x}_b + \mathbf{L}\chi) \right)$$

✘ We use the conjugate gardient to solve

$$\left[\mathbf{I} + \mathbf{L}^{T} \mathbf{H}^{\mathrm{T}} \mathbf{R}^{-1} \mathbf{H} \mathbf{L} \right] \chi = \mathbf{L}^{T} \mathbf{H}^{\mathrm{T}} \mathbf{R}^{-1} \left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b) \right]$$

✘ This allows the hessian matrix to be approximated by

$$J'' = \mathbf{I} + \sum_{i=1}^{K} \left(\lambda_i - 1 \right) \mathbf{v}_i \mathbf{v}_i^{T}$$

where λ_i and $\mathbf{v}_i$ are the eigenvalues and eigenvectors determined by the congugate gradient and the Lanczos algorithm

✘ We can use as a new preconditioning

$$(J'')^{-1/2} = \mathbf{I} + \sum_{i=1}^{K} \left(\lambda_i^{-1/2} - 1 \right) \mathbf{v}_i \mathbf{v}_i^{T}$$

Calculating the operators

✘ We use the conjugate gardient to solve

$$\left[\mathbf{I} + \mathbf{L}^T\mathbf{H}^\mathrm{T}\mathbf{R}^{-1}\mathbf{H}\mathbf{L}\right]\chi = \mathbf{L}^T\mathbf{H}^\mathrm{T}\mathbf{R}^{-1}\left[\mathbf{y} - \mathcal{H}(\mathbf{x}_b)\right]$$

✘ Typically, $\mathbf{R}$ is diagonal — observation errors are treated as being mutually uncorrelated.

✘ However, the matrices $\mathbf{H}^\mathrm{T}$, $\mathbf{L}^\mathrm{T}$ and $\mathbf{L}$ are not diagonal, and are much too large to be represented explicitly.

✘ We must represent these as operators (subroutines) that calculate matrix-vector products.

Calculating the operators

✗ Take $\mathcal{H}$ as an example. Each line of the subroutine that applies $\mathcal{H}$ can be considered as a function h_k, so that

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

✗ Each of the functions h_k can be linearised, to give the corresponding linear function $\mathbf{h}_k$. The resulting code is called the tangent linear of $\mathcal{H}$.

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

✗ The transpose is called the adjoint of $\mathcal{H}$

$$\mathbf{H}^{\mathrm{T}}(\mathbf{x}) \equiv \mathbf{h}_1^{\mathrm{T}} \mathbf{h}_2^{\mathrm{T}} \cdots \mathbf{h}_K^{\mathrm{T}} \mathbf{x}$$

✗ Each $\mathbf{h}_k$ and $\mathbf{h}_k^{\mathrm{T}}$ is extremely simple — just to a few lines of code.

Tangent Linear and Adjoints

There is a whole 1-hour lecture on tangent linear and adjoint operators Tuesday when you will learn to derive tangent linear and adjoint equations for a simple nonlinear equation.

ECMWF **EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS** ©ECMWF

Outline

ECMWF EUROPEAN CENTRE FOR MEDIUM-RANGE WEATHER FORECASTS

©ECMWF

Summary

✗ We showed that 3D-Var can be considered as an iterative procedure for solving the linear (minimum variance) analysis equation.

✗ We also derived 3D-Var from the maximum likelihood principle.

✗ The Maximum Likelihood approach can be applied to non-Gaussian, nonlinear analysis.

✗ We introduced the 3D-Var cost function.

✗ We considered how to minimise the cost function using algorithms based on knowledge of its gradient.

✗ We looked at a simple preconditioning.

✗ Finally, we saw how it is possible to write code that computes the gradient.