Introduction to Data Assimilation

Maëlle Nodet

team.inria.fr/moise/maelle

Université de Grenoble, INRIA, LJK

GdT Couplage, Montpellier
6 mars 2012

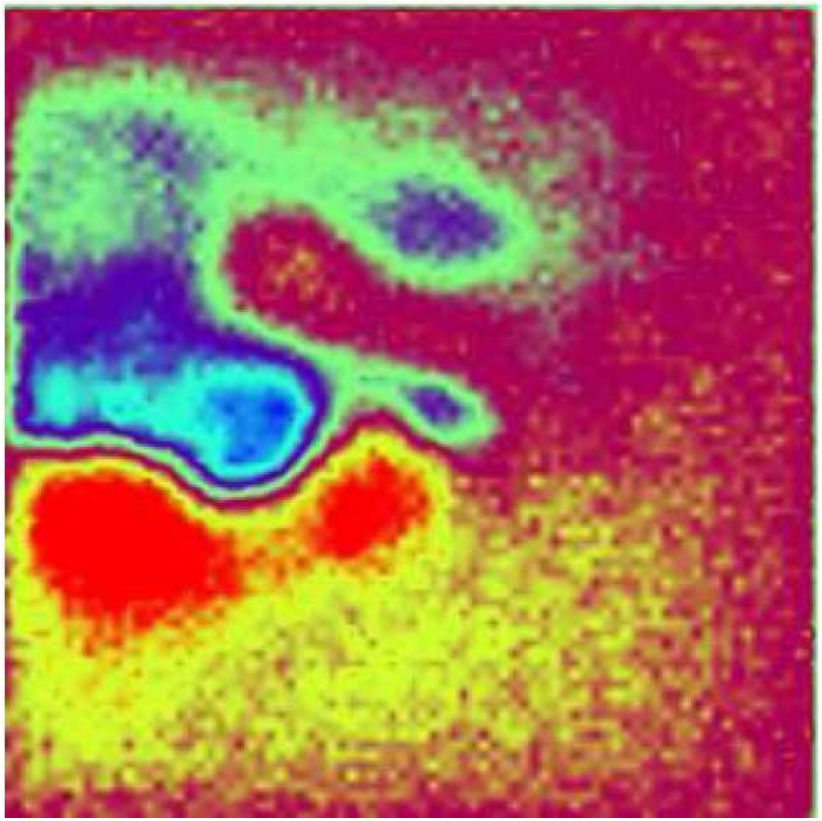
What is data assimilation?

Combine at best different sources of information to estimate the state of a system:

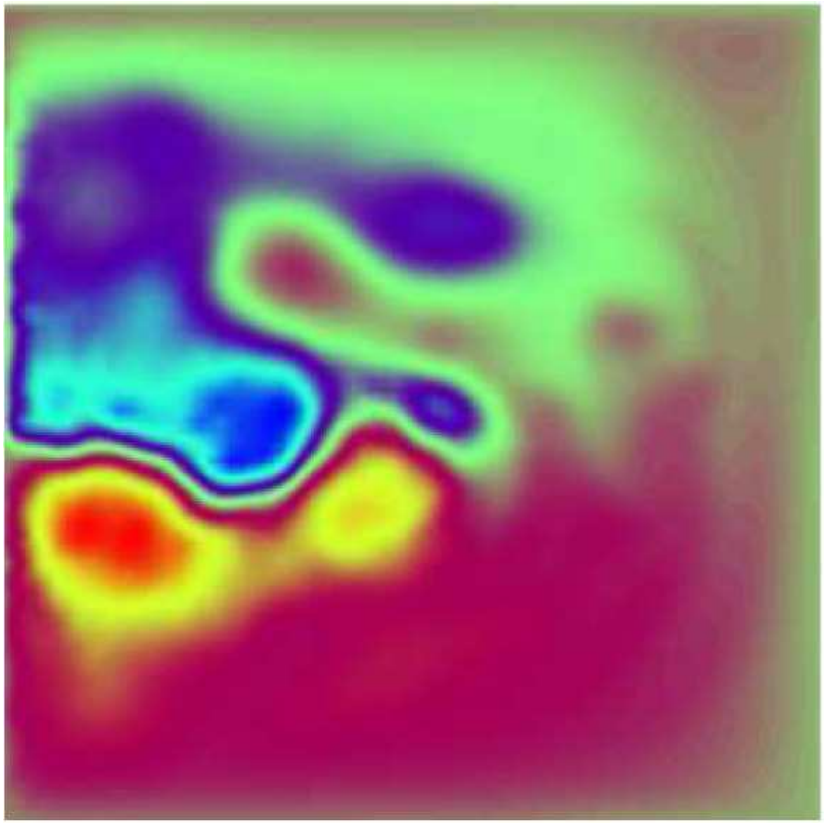
- model equations
- observations, data
- background, a priori information
- statistics

Data only

data



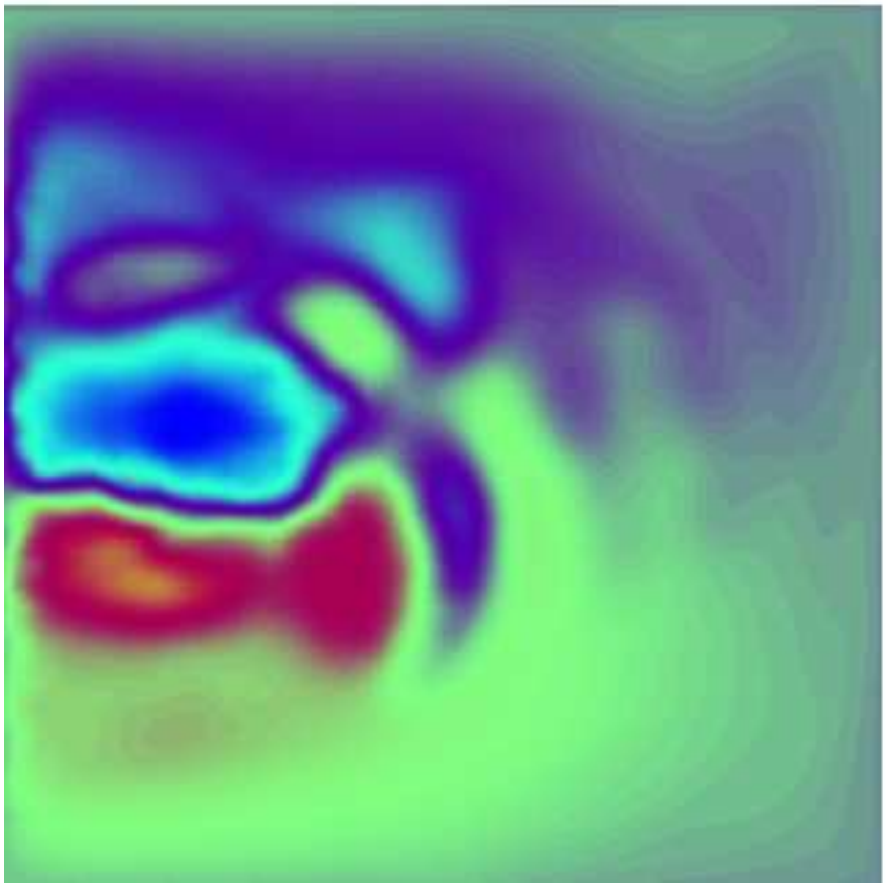
reference



(1 every 25 gripoint)

Model only

reference

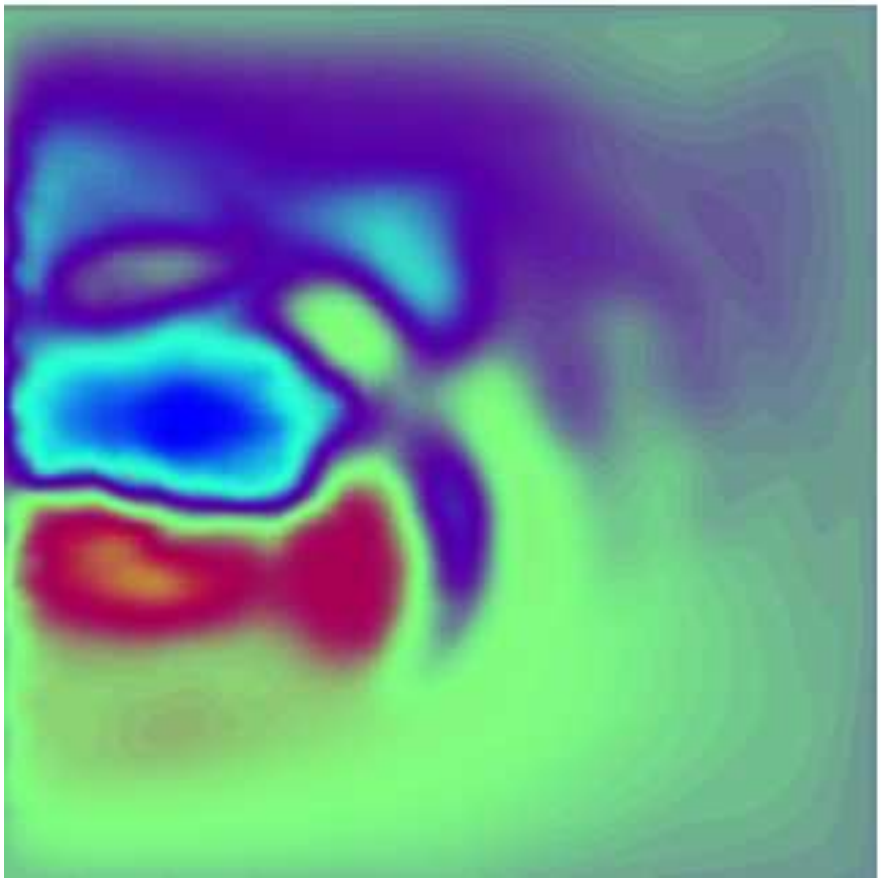


model

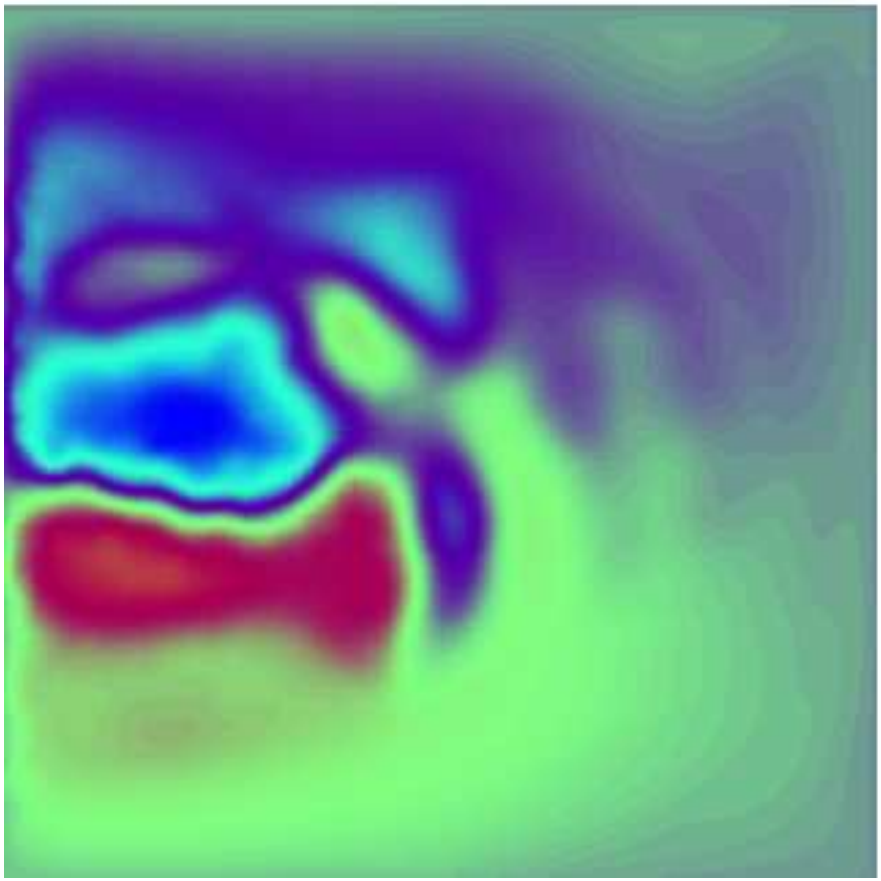
(after 4 months)

Model-Data coupling

reference



after data assimilation



What is data assimilation for?

Historically: meteorology. Later, oceanography.

Today, many other fields

- glaciology,
- seismology,
- nuclear fusion,
- medicine,
- agronomy,
- etc.

What is data assimilation for?

Historically: initial state estimation, for weather forecasting.

Today, many other applications:

- initial conditions for predictions,
- calibration and validation,
- observing system design, monitoring and assessment,
- reanalysis,
- better understanding (model errors, data errors, physical process interactions, parameters, etc),
- etc.

Example from ocean forecasting

Short term ocean forecasting (a few weeks) of currents, temperature, salinity, for fisheries, coastal economy, sailing...

We aim at producing an estimate $\mathbf{x}^a$ of the true state $\mathbf{x}^t = (u, v, T, S)$ of the ocean at initial time, to initialize forecasts.

We are given:

- a background estimate $\mathbf{x}^b = (u^b, v^b, T^b, S^b)$, which is either a previous forecast or comes from climatology,
- partial observations $\mathbf{y}^o = \mathcal{H}(\mathbf{x}^t) + \epsilon^o$, distributed over a given time window, e.g. temperatures from buoys, sea surface elevation from satellites, currents from moorings, ...

Observation operator $\mathcal{H}$ contains the dynamical model mapping the initial state of the ocean to the actual temperature, currents, sea surface height at given points in space and time.

Example from glaciology

Short term ice dynamics forecasting (100 years), to estimate Antarctica and Greenland contribution to sea level change.

We aim at producing an estimate $\mathbf{x}^a$ of the true input parameters $\mathbf{x}^t = \beta$ of the basal drag coefficient (function of space, constant over time) at the bottom of the ice cap.

We are given:

- a background estimate $\mathbf{x}^b = \beta^b$, which is roughly inferred from surface velocities,
- partial observations $\mathbf{y}^o = \mathcal{H}(\mathbf{x}^t) + \epsilon^o$, e.g. surface velocities, ice surface elevation, approximate bedrock topography ...

Observation operator $\mathcal{H}$ contains the dynamical model mapping the basal drag of the ice cap to the surface variables at given points in space and time.

Outline

1. Stochastic data assimilation
   - Best linear unbiased estimator (BLUE)
   - Kalman filter algorithm
2. Variational Data Assimilation
   - Principle of variational methods
   - Gradient-based optimization
   - Variational algorithms
3. Implementation issues
   - Non linearities
   - High dimensional problems
   - Gradient computation: adjoint method

Errors statistics

Mean:

$$\mathrm{E}(x) =< x > \text{ scalar}, \quad \mathrm{E}(\mathbf{x}) = (\mathrm{E}(x_1), \mathrm{E}(x_2), ..., \mathrm{E}(x_n)) \text{ vector-valued}$$

Variance, covariance (x, y scalar):

$$\mathrm{Var}(x) = \mathrm{E}((x - \mathrm{E}(x))^2), \quad \mathrm{Cov}(x, y) = \mathrm{E}((x - \mathrm{E}(x))(y - \mathrm{E}(y)))$$

We say that errors are:

- unbiased if $\mathrm{E}(\epsilon) = 0$;
- uncorrelated if $\mathrm{E}(\epsilon_1 \epsilon_2^T) = 0$;
- non trivial if $\mathrm{Cov}(\epsilon)$ is positive-definite ;

Covariance matrix

Covariance matrix ($\mathbf{x}$ vector-valued):

$$\begin{aligned} \mathrm{Cov}(\mathbf{x}) &= \mathrm{E}((\mathbf{x} - \mathrm{E}(\mathbf{x}))(\mathbf{x} - \mathrm{E}(\mathbf{x}))^T) \\ (\mathrm{Cov}(\mathbf{x}))_{i,j} = \mathrm{Cov}(x_i, x_j) &= \mathrm{E}((x_i - \mathrm{E}(x_i))(x_j - \mathrm{E}(x_j))) \end{aligned}$$

E.g. for $\mathbf{x} = (x_1, x_2, x_3)$:

$$\mathrm{Cov}(\mathbf{x}) = \begin{pmatrix} \mathrm{Var}(x_1) & \mathrm{Cov}(x_1, x_2) & \mathrm{Cov}(x_1, x_3) \\ \mathrm{Cov}(x_1, x_2) & \mathrm{Var}(x_2) & \mathrm{Cov}(x_2 x_3) \\ \mathrm{Cov}(x_1, x_3) & \mathrm{Cov}(x_2, x_3) & \mathrm{Var}(x_3) \end{pmatrix}$$

Notations

State

- $\mathbf{x}$ state vector or input parameters
- $\mathbf{x}^t$ true state (unknown)
- $\mathbf{x}^b$ background state (a priori information), background error $\epsilon^b = \mathbf{x}^b - \mathbf{x}^t$, covariance matrix $\mathbf{B}$
- $\mathbf{x}^a$ analyzed state (result of the assimilation process), analysis error $\epsilon^a = \mathbf{x}^a - \mathbf{x}^t$, covariance matrix $\mathbf{A}$

Observations

- observation vector $\mathbf{y}^o$
- observation operator $\mathcal{H}$, mapping state space to observation space: $\mathbf{y}^o = \mathcal{H}(\mathbf{x}^t) + \epsilon^o$
- observation error ϵ^o, covariance matrix $\mathbf{R}$

Problem position: what we have

We aim at producing an estimate $\mathbf{x}^a$ of the true input parameters $\mathbf{x}^t$ of the system.

We are given:

- a background estimate $\mathbf{x}^b$, whose error ϵ^b are assumed unbiased and non trivial, with covariance matrix $\mathbf{B}$ given,
- partial observations $\mathbf{y}^o = \mathcal{H}(\mathbf{x}^t) + \epsilon^o$, where ϵ^o are unbiased and non trivial, with covariance matrix $\mathbf{R}$ given.
  Observation operator $\mathcal{H}$ maps the input parameters to the observation variables (can contain complex laws, PDEs, non linear physics, ...).

We also assume that:

- $\mathcal{H} = \mathbf{H}$ is a linear operator,
- ϵ^o and ϵ^b are not correlated.

Problem position: what we look for

We aim at producing an estimate $\mathbf{x}^a$ of the true state $\mathbf{x}^t$ of the system.

The best estimate is searched for as a linear combination of the background estimate and the observation:

$$\mathbf{x}^a = \mathbf{L}\,\mathbf{x}^b + \mathbf{K}\,\mathbf{y}^o$$

Optimality criterium

We look for an unbiased estimate $\mathbf{x}^a$, with minimal variance $\mathrm{tr}(\mathbf{A})$.

Best linear unbiased estimator, or least squares analysis

1. BLUE analysis:

$$\begin{cases} \mathbf{x}^a = (\mathbf{I} - \mathbf{K}\mathbf{H})\,\mathbf{x}^b + \mathbf{K}\,\mathbf{y}^o = \mathbf{x}^b + \mathbf{K}(\mathbf{y}^o - \mathbf{H}(\mathbf{x}^b)) \\ \mathbf{K} = \mathbf{B}\mathbf{H}^T(\mathbf{H}\mathbf{B}\mathbf{H}^T + \mathbf{R})^{-1} \end{cases}$$

$\mathbf{K}$: gain, or weight matrix, $\mathbf{y}^o - \mathbf{H}(\mathbf{x}^b)$ innovation.

2. Analysis covariance matrix: $\mathbf{A} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{B}$
3. Equivalent variational optimization problem: (optimal least squares)

$$\begin{cases} \mathbf{x}^a = \arg\min \mathcal{J} \\ \mathcal{J}(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^b)^T \mathbf{B}^{-1} (\mathbf{x} - \mathbf{x}^b) + (\mathbf{y}^o - \mathbf{H}(\mathbf{x}))^T \mathbf{R}^{-1} (\mathbf{y}^o - \mathbf{H}(\mathbf{x})) \end{cases}$$

$\mathcal{J}$: cost function.

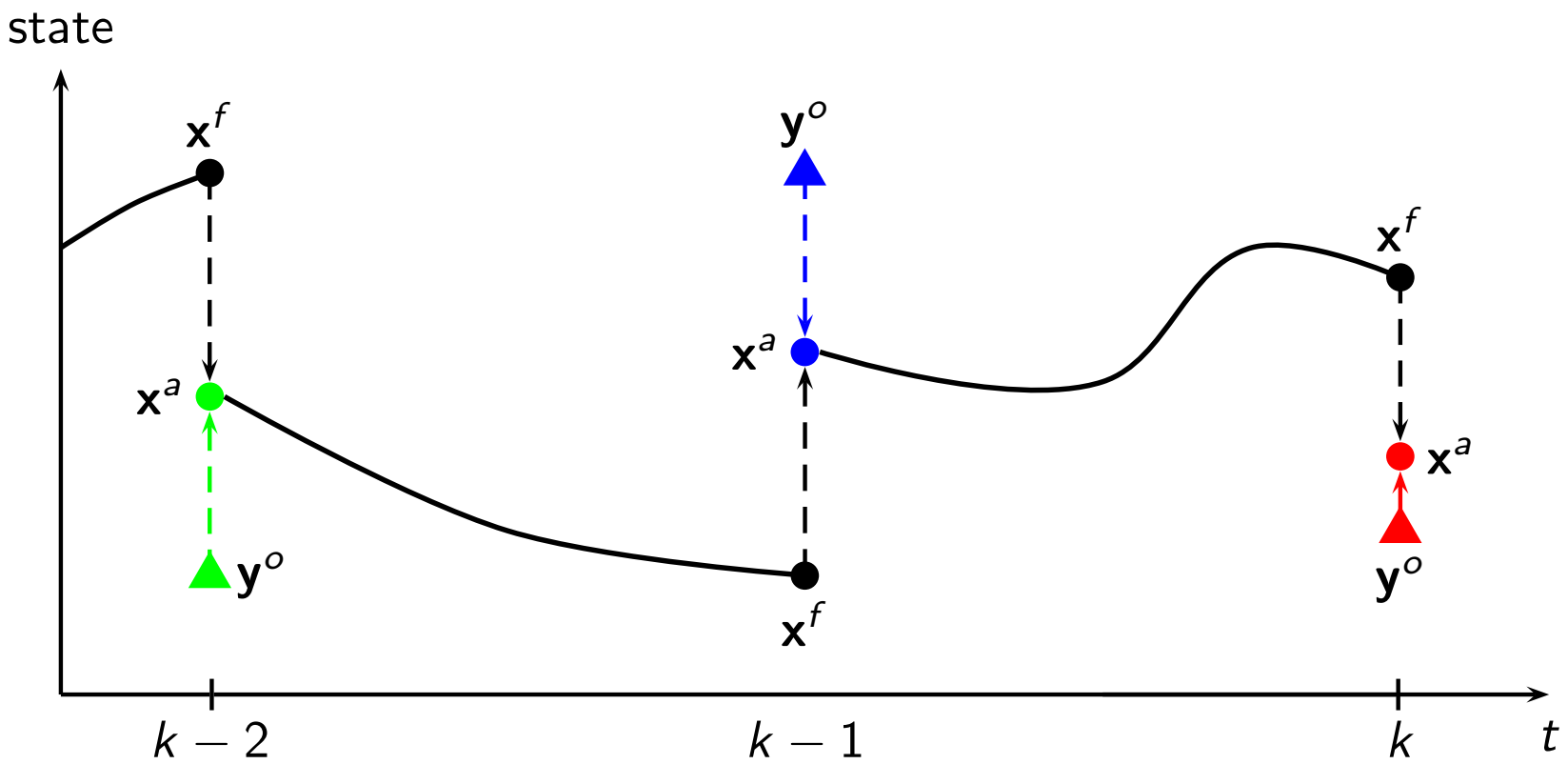
Data assimilation methods

Two types of methods:

1. Direct computation of the BLUE, and the gain matrix $\mathbf{K}$.
   Main algorithm: Kalman filter
   $\longrightarrow$ stochastic data assimilation, this section.

2. Minimization of the cost function $\mathcal{J}$ using optimization and adjoint methods.
   Main algorithm: 4D-Var
   $\longrightarrow$ variational data assimilation, next section.

Time-dependant problems: the Kalman filter sequence



Back to notations

Vectors:

- k time index
- $\mathbf{x}_k^f$ forecast state (background), forecast error covariance matrix $\mathbf{P}_k^f$
- $\mathbf{x}_k^a$ analyzed state (result of the assimilation process), analysis error covariance matrix $\mathbf{P}_k^a$

Operators:

- model operator $\mathbf{x}_{k+1}^t = \mathbf{M}_{k,k+1}(\mathbf{x}_k^t) + \eta_{k,k+1}$, model error $\eta_{k,k+1}$, covariance matrix $\mathbf{Q}_k$
- observation operator $\mathbf{y}_k^o = \mathbf{H}_k(\mathbf{x}^t) + \epsilon_k^o$, observation error ϵ_k^o, covariance matrix $\mathbf{R}_k$

Kalman's hypotheses, schematically:

- Model and observations operators $\mathbf{M}_{k,k+1}$ and $\mathbf{H}_k$ are linear ;
- Errors are unbiased, gaussian, independant and white in time.

Kalman's hypotheses

Schematically:

- Model and observations operators are linear, denoted $\mathbf{M}_{k,k+1}$ and $\mathbf{H}_k$;
- Errors are unbiased, gaussian, independant and white in time.

Kalman's hypotheses

Uncensored version:

- Initial state is gaussian: $\mathbf{x}_0^t \sim \mathcal{N}(\mathbf{x}_0^b, \mathbf{P}_0^b)$;
- The dynamical model $\mathcal{M}_k$ is linear and denoted $\mathbf{M}_{k,k+1}$;
- The model errors are unbiased and gaussian: $\eta_k \sim \mathcal{N}(0, \mathbf{Q}_k)$;
- The model errors are white in time: $< \eta_k \eta_j^T >= 0$ if $k \neq j$;
- The observation operators $\mathcal{H}_k$ are linear and denoted $\mathbf{H}_k$;
- The observation errors are unbiased and gaussian: $\epsilon_k^o \sim \mathcal{N}(0, \mathbf{R}_k)$;
- The observation errors are white in time: $< \epsilon_k^o {\epsilon_j^o}^T >= 0$ if $k \neq j$;
- Errors of different types are independent: $< \eta_k {\epsilon_j^o}^T >= 0$, $< \eta_k {\epsilon_0^b}^T >= 0$, $< \epsilon_k^o {\epsilon_0^b}^T >= 0$.

The Kalman filter equations

1. Initialization:
   $\mathbf{x}_0^f$ and $\mathbf{P}_0^f$ are given, e.g. equal to $\mathbf{x}^b$ and $\mathbf{B}$
2. Analysis step: (BLUE analysis)

$$\begin{aligned} \mathbf{K}_k &= (\mathbf{H}_k\mathbf{P}_k^f)^T[\mathbf{H}_k(\mathbf{H}_k\mathbf{P}_k^f)^T + \mathbf{R}_k]^{-1}, \\ \mathbf{x}_k^a &= \mathbf{x}_k^f + \mathbf{K}_k(\mathbf{y}_k^o - \mathbf{H}_k\mathbf{x}_k^f), \\ \mathbf{P}_k^a &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^f. \end{aligned}$$

3. Forecast step: (evolution model)

$$\begin{aligned} \mathbf{x}_{k+1}^f &= \mathbf{M}_{k,k+1}\mathbf{x}_k^a, \\ \mathbf{P}_{k+1}^f &= \mathbf{M}_{k,k+1}\mathbf{P}_k^a\mathbf{M}_{k,k+1}^T + \mathbf{Q}_k. \end{aligned}$$

Back to notations

Vectors

- $\mathbf{x}$ state vector or input parameters
- $\mathbf{x}^b$ background state (a priori information), background error $\epsilon^b = \mathbf{x}^b - \mathbf{x}^t$, covariance matrix $\mathbf{B}$
- $\mathbf{x}^a$ analyzed state (result of the assimilation process)
- $\mathbf{y}^o$ observation vector

Operators

- model operator $\mathbf{x}_k^t = \mathbf{M}_{k,k-1}(\mathbf{x}_{k-1}^t) = \mathbf{M}_{0\to k}(\mathbf{x}_0^t)$
- observation operator $\mathbf{y}^o = \mathbf{H}(\mathbf{x}^t) + \epsilon^o$, $\mathbf{y}_k^o = \mathbf{H}_k(\mathbf{x}^t) + \epsilon_k^o$, observation error ϵ_k^o, covariance matrix $\mathbf{R}_k$

Back to the BLUE

Variational approach of BLUE consists in finding $\mathbf{x}^a = \arg\min \mathcal{J}$:

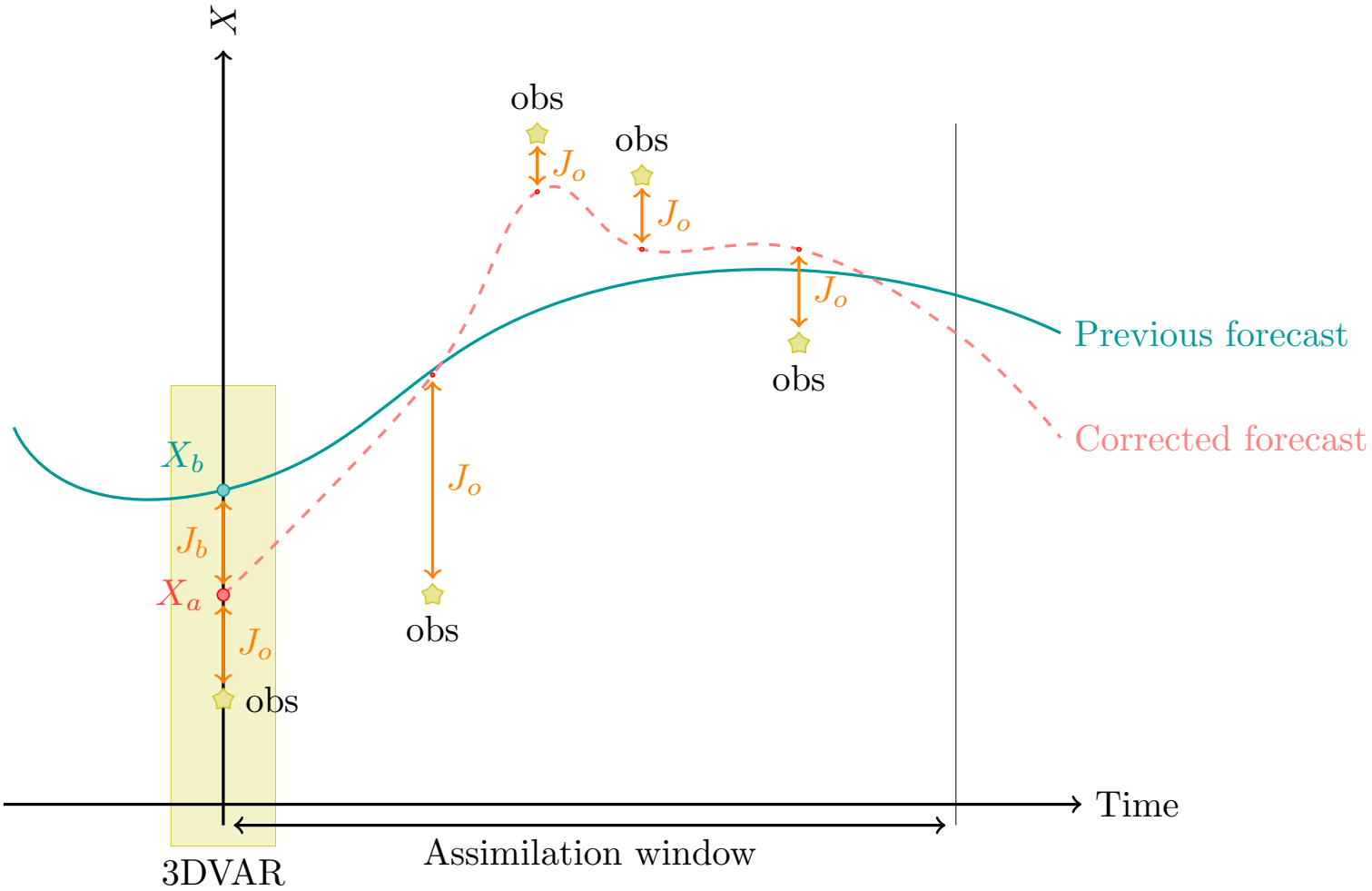
$$\begin{aligned}\mathcal{J}(\mathbf{x}) &= (\mathbf{x}-\mathbf{x}^b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}^b) + (\mathbf{y}^o-\mathbf{H}(\mathbf{x}))^T\mathbf{R}^{-1}(\mathbf{y}^o-\mathbf{H}(\mathbf{x})) \\ &= \underbrace{\frac{1}{2}\|\mathbf{x}-\mathbf{x}^b\|^2_{\mathbf{B}}}_{\mathcal{J}^b} + \underbrace{\frac{1}{2}\|\mathbf{H}(\mathbf{x})-\mathbf{y}^{\mathbf{o}}\|^2_{\mathbf{R}}}_{\mathcal{J}^o}\end{aligned}$$

If the problem is time-dependant, and the unknown $\mathbf{x}$ is the **initial state vector**:

$$\begin{aligned}\mathcal{J}(\mathbf{x}) &= \tfrac{1}{2}\|\mathbf{x}-\mathbf{x}^b\|^2_{\mathbf{B}} + \tfrac{1}{2}\sum_k \|\mathbf{H}_k(\mathbf{x}_k)-\mathbf{y}^o_k\|^2_{\mathbf{R}_k} \\ &= \underbrace{\frac{1}{2}\|\mathbf{x}-\mathbf{x}^b\|^2_{\mathbf{B}}}_{\mathcal{J}^b} + \underbrace{\frac{1}{2}\sum_k \|\mathbf{H}_k(\mathbf{M}_{0\to k}(\mathbf{x}))-\mathbf{y}^o_k\|^2_{\mathbf{R}_k}}_{\mathcal{J}^o}\end{aligned}$$

Variational methods principle

$$\begin{aligned}\mathcal{J}(\mathbf{x}) &= \frac{1}{2}\|\mathbf{x}-\mathbf{x}^b\|^2_{\mathbf{B}} + \frac{1}{2}\sum_k \|\mathbf{H}_k(\mathbf{x}_k)-\mathbf{y}^o_k\|^2_{\mathbf{R}_k} = \mathcal{J}^b + \mathcal{J}^o \\ &= \frac{1}{2}\|\mathbf{x}-\mathbf{x}^b\|^2_{\mathbf{B}} + \frac{1}{2}\sum_k \|\mathbf{H}_k(\mathbf{M}_{0\to k}(\mathbf{x}))-\mathbf{y}^o_k\|^2_{\mathbf{R}_k}\end{aligned}$$



Wake up here!

Fundamental remark

Once $\mathcal{J}$ is defined (i.e. once all the ingredients are chosen: control variables, error statistics, norms, observations...), the problem is entirely defined. Hence its solution.

$\Rightarrow$ **The "physical part" of data assimilation lies in the definition of $\mathcal{J}$.**

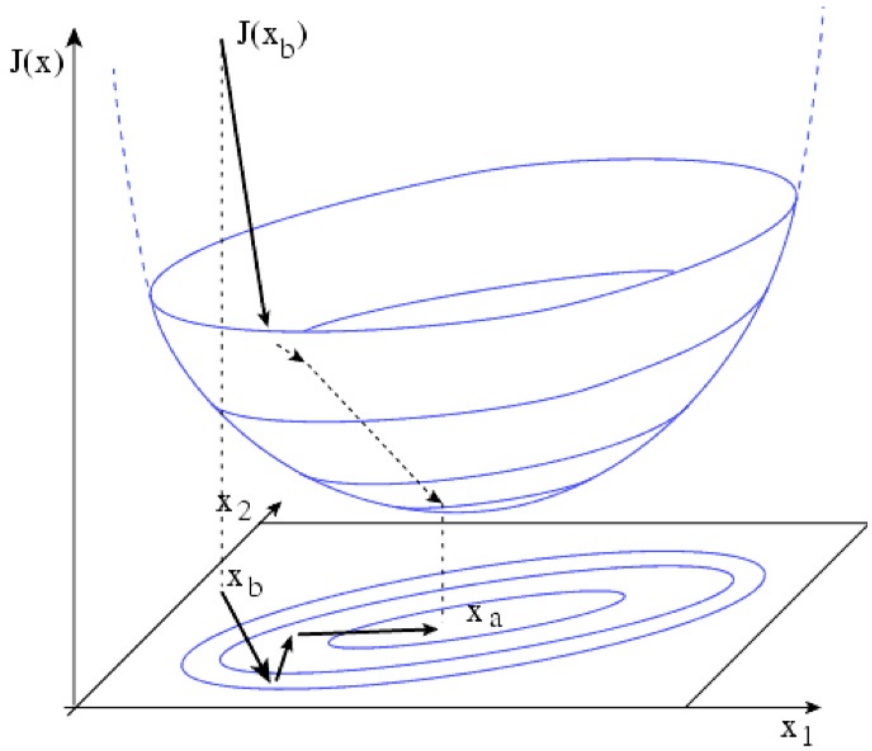
The rest of the job, i.e. minimizing $\mathcal{J}$, is "technical" work.

Descent methods

Descent methods to minimize a function require knowledge of (an estimate of) its gradient.

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

(k iteration number)



$$\text{with } \mathbf{d}_k = \begin{cases} -\nabla \mathcal{J}(\mathbf{x}_k) & \text{gradient method} \\ -\left[\text{Hess}(\mathcal{J})(\mathbf{x}_k)\right]^{-1} \nabla \mathcal{J}(\mathbf{x}_k) & \text{(quasi-)Newton method} \\ -\nabla \mathcal{J}(\mathbf{x}_k) + \frac{\|\nabla \mathcal{J}(\mathbf{x}_k)\|^2}{\|\nabla \mathcal{J}(\mathbf{x}_{k-1})\|^2} d_{k-1} & \text{conjugate gradient} \\ \ldots & \ldots \end{cases}$$

Gradient computation

The computation of $\nabla \mathcal{J}(\mathbf{x})$ may be difficult if the dependency of $\mathcal{J}$ with regard to the control variable $\mathbf{x}$ is not direct.

Example: parameter estimation

- $\mathbf{u}(t)$ solution of a differential equation:
  $-K_1\mathbf{u}''(t) + K_2\mathbf{u}'(t) = f(t), t \in]0,1[\quad ; \quad \mathbf{u}(0) = \mathbf{u}(1) = 0$
- $\mathbf{x} = K = (K_1, K_2)$ parameters of this equation
- $\mathbf{u}^{\text{obs}}(t)$ an observation of $\mathbf{u}(t)$
- $\mathcal{J}(K) = \frac{1}{2} \|\mathbf{u} - \mathbf{u}^{\text{obs}}\|^2$

$$\longrightarrow \quad \nabla \mathcal{J}(K).k = < \hat{\mathbf{u}}[K](k), \mathbf{u} - \mathbf{u}^{\text{obs}} > \qquad \text{with } \hat{\mathbf{u}}[K](k) = \lim_{\alpha \to 0} \frac{\mathbf{u}_{K+\alpha k} - \mathbf{u}_K}{\alpha}$$

The computation of $\nabla \mathcal{J}(\mathbf{x})$ may be difficult if the dependency of $\mathcal{J}$ with regard to the control variable $\mathbf{x}$ is not direct.
It is often difficult (or even impossible) to obtain the gradient through the computation of growth rates.

Example: initial state estimation

$$\begin{cases} \dfrac{d\mathbf{u}(t))}{dt} = M(\mathbf{u}(t)) \quad t \in [0, T] \\ \mathbf{u}(t=0) = \mathbf{x} \end{cases} \qquad \text{with } \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}$$

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2} \int_0^T \|\mathbf{u}(t) - \mathbf{u}^{\text{obs}}(t)\|^2 \qquad \longrightarrow \text{ requires one model run}$$

$$\nabla \mathcal{J}(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial \mathcal{J}}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \dfrac{\partial \mathcal{J}}{\partial x_N}(\mathbf{x}) \end{pmatrix} \simeq \begin{pmatrix} [\mathcal{J}(\mathbf{x} + \alpha\, \mathbf{e}_1) - \mathcal{J}(\mathbf{x})]\,/\alpha \\ \vdots \\ [\mathcal{J}(\mathbf{x} + \alpha\, \mathbf{e}_N) - \mathcal{J}(\mathbf{x})]\,/\alpha \end{pmatrix}$$

$\longrightarrow N + 1$ model runs

Gradient computation

In actual applications like meteorology / oceanography,
$N = [\mathbf{x}] = \mathcal{O}(10^6 - 10^8)$ $\longrightarrow$ this method cannot be used.

Adjoint method

The adjoint method provides a very efficient way to compute $\nabla \mathcal{J}$, with only one run of the **adjoint model** (computationally 4-10 times the cost of the direct model).

Attention!

On the contrary, do not forget that, if the size of the control variable is very small (< 10), $\nabla \mathcal{J}$ can be easily estimated by the computation of growth rates.

Time-independant problems: 3D-Var

Cost function:

$$\mathcal{J}(\mathbf{x}_0) = (\mathbf{x}_0 - \mathbf{x}^b)^T \mathbf{B}^{-1} (\mathbf{x}_0 - \mathbf{x}^b) + (\mathbf{y} - H(\mathbf{x}_0))^T \mathbf{R}^{-1} (\mathbf{y} - H(\mathbf{x}_0))$$

Gradient:

$$\nabla \mathcal{J}(\mathbf{x}) = 2\mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}^b) - 2\mathbf{H}^T \mathbf{R}^{-1} (\mathbf{y} - H(\mathbf{x}))$$

Iterative minimization algorithm

- Initialisation : $\mathbf{x}_0 = \mathbf{x}^b$, $n = 0$
- While $\|\nabla \mathcal{J}\| > \varepsilon$ or $n \leq n_{\max}$, do :

1. Compute $\mathcal{J}$
2. Compute $\nabla \mathcal{J}$
3. Descent and update of $\mathbf{x}_0$
4. $n = n + 1$

Time-dependant problems: 4D-Var

Cost function:

$$\mathcal{J}(\mathbf{x}_0) = \|\mathbf{x}_0 - \mathbf{x}_0^b\|^2_{\mathbf{B}} + \sum_{i=0}^{n} \|\mathbf{y}_i^o - H_i(M_i(M_{i-1}(\dots M_1(\mathbf{x}_0)))\|^2_{\mathbf{R}}$$

Gradient involves the adjoint model:

$$-\frac{1}{2}\nabla\mathcal{J}^o(\mathbf{x}) = \sum_{i=0}^{n} \mathbf{M}_1^T \dots \mathbf{M}_{i-1}^T \mathbf{M}_i^T \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{d}_i$$

Innovation vector $\mathbf{d}_i$:

$$\mathbf{d}_i = y_i^o - H_i(M_i(M_{i-1}(\dots M_1(\mathbf{x}))))$$

4D-Var algorithm

- Initialization : $\mathbf{x} = \mathbf{x}^0$, $n = 0$
- While $\|\nabla \mathcal{J}\| > \varepsilon$ or $n \leq n_{\max}$, do :

1. Compute $\mathcal{J}$ thanks to the direct model M and the observation operator H
2. Store innovation vectors $\mathbf{d}_i$
3. Compute $\nabla \mathcal{J}$ thanks to the backward integration of the adjoint model $\mathbf{M}^T$ and the adjoint of the observation operator $\mathbf{H}^T$
4. Descent and update of $\mathbf{x}$
5. $n = n + 1$

Equivalence 4D-Var – Kalman filter

Under the Kalman Filter hypotheses, 4D-Var and Kalman filter algorithms are equivalent.

More precisely: starting with the same background, with equal covariance matrices at the beginning of the time-window, and assimilating the same observations, the same result is reached at the end of the time-window.

These algorithms are both optimal in a least squares and minimal variance point of view.

Linear assumption

Validity of BLUE analysis

Kalman Filter and 4D-Var are valid (and produce equivalent results) if model operator **M** and observation operator **H** are linear.

In presence of nonlinearities:

1. KF and 4DV not equivalent anymore,
2. optimality of analysis is lost.

Tangent linear hypothesis

In case of weak non linearities: we can hope the linear analysis still gives some information...

Tangent linear hypothesis

$$\mathcal{M}_{0\to i}(\mathbf{x}_0) - \mathcal{M}_{0\to i}(\mathbf{x}_0^b) \simeq \mathbf{M}_{0\to i}(\mathbf{x}_0 - \mathbf{x}^b)$$
$$\mathcal{H}_i(\mathbf{x}_i) - \mathcal{H}_i(\mathbf{x}_i^b) \simeq \mathbf{H}_i(\mathbf{x}_i - \mathbf{x}_i^b)$$

- Stochastic $\longrightarrow$ extended Kalman Filter
- Variational $\longrightarrow$ incremental 3D- and 4D-Var

Extended Kalman Filter

Non linear operators $\mathcal{M}$ and $\mathcal{H}$ are replaced with their tangent operators $\mathbf{M}$ and $\mathbf{H}$ in the algo, *except in innovation and state forecast steps*:

1. Initialization:
   $\mathbf{x}_0^f$ and $\mathbf{P}_0^f$ are given, e.g. equal to $\mathbf{x}^b$ and $\mathbf{B}$
2. Analysis step:

$$\begin{aligned} \mathbf{K}_k &= (\mathbf{H}_k\mathbf{P}_k^f)^T[\mathbf{H}_k(\mathbf{H}_k\mathbf{P}_k^f)^T + \mathbf{R}_k]^{-1}, \\ \mathbf{x}_k^a &= \mathbf{x}_k^f + \mathbf{K}_k(\mathbf{y}_k^o - \mathcal{H}_k\mathbf{x}_k^f), \\ \mathbf{P}_k^a &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^f. \end{aligned}$$

3. Forecast step:

$$\begin{aligned} \mathbf{x}_{k+1}^f &= \mathcal{M}_{k,k+1}\mathbf{x}_k^a, \\ \mathbf{P}_{k+1}^f &= \mathbf{M}_{k,k+1}\mathbf{P}_k^a\mathbf{M}_{k,k+1}^T + \mathbf{Q}_k. \end{aligned}$$

Incremental 4D-Var

4D-Var cost function:

$$\mathcal{J}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^b\|^2_{\mathbf{B}} + \sum_{k=0}^{n} \|\mathbf{y}^o_k - \mathcal{H}_k(\mathcal{M}_k(\mathcal{M}_{k-1}(\ldots \mathcal{M}_1(\mathbf{x})))\|^2_{\mathbf{R}}$$

Cost function linearized around $\mathbf{x}^b$, with increment $\delta\mathbf{x}$: $\mathbf{x} = \mathbf{x}^b + \delta\mathbf{x}$

Incremental innovation vector $\mathbf{d}_k$:

$$\mathbf{d}_k = \mathbf{y}^o_k - \mathcal{H}_k(\mathcal{M}_k(\mathcal{M}_{k-1}(\ldots \mathcal{M}_1(\mathbf{x}^b))))$$

Incremental cost function (quadratic):

$$\mathbf{J}(\delta\mathbf{x}) = \|\delta\mathbf{x}\|^2_{\mathbf{B}} + \sum_{k=0}^{n} \|\mathbf{d}_k - \mathbf{H}_k(\mathbf{M}_k(\mathbf{M}_{k-1}(\ldots \mathbf{M}_1(\delta\mathbf{x})))\|^2_{\mathbf{R}}$$

Incremental 4D-Var

– Initialization : $\mathbf{x}_0^r = \mathbf{x}_0^b$

START OUTER LOOP

- Non linear model integration: $\mathbf{x}_k^r = \mathcal{M}_{0\to k}[\mathbf{x}^r]$
- Innovation vector computation: $d_k = \mathbf{y}_k^o - \mathcal{H}_k(\mathbf{x}_k^r)$

  START INNER LOOP

  - Computation of $\mathbf{J}$, using $\mathbf{M}$ and $\mathbf{H}$ linearized operators around $\mathbf{x}^r$
  - Computation of $\nabla\mathbf{J}$, using adjoint operators $\mathbf{M}^T$ and $\mathbf{H}^T$
  - Minimization via a descent method

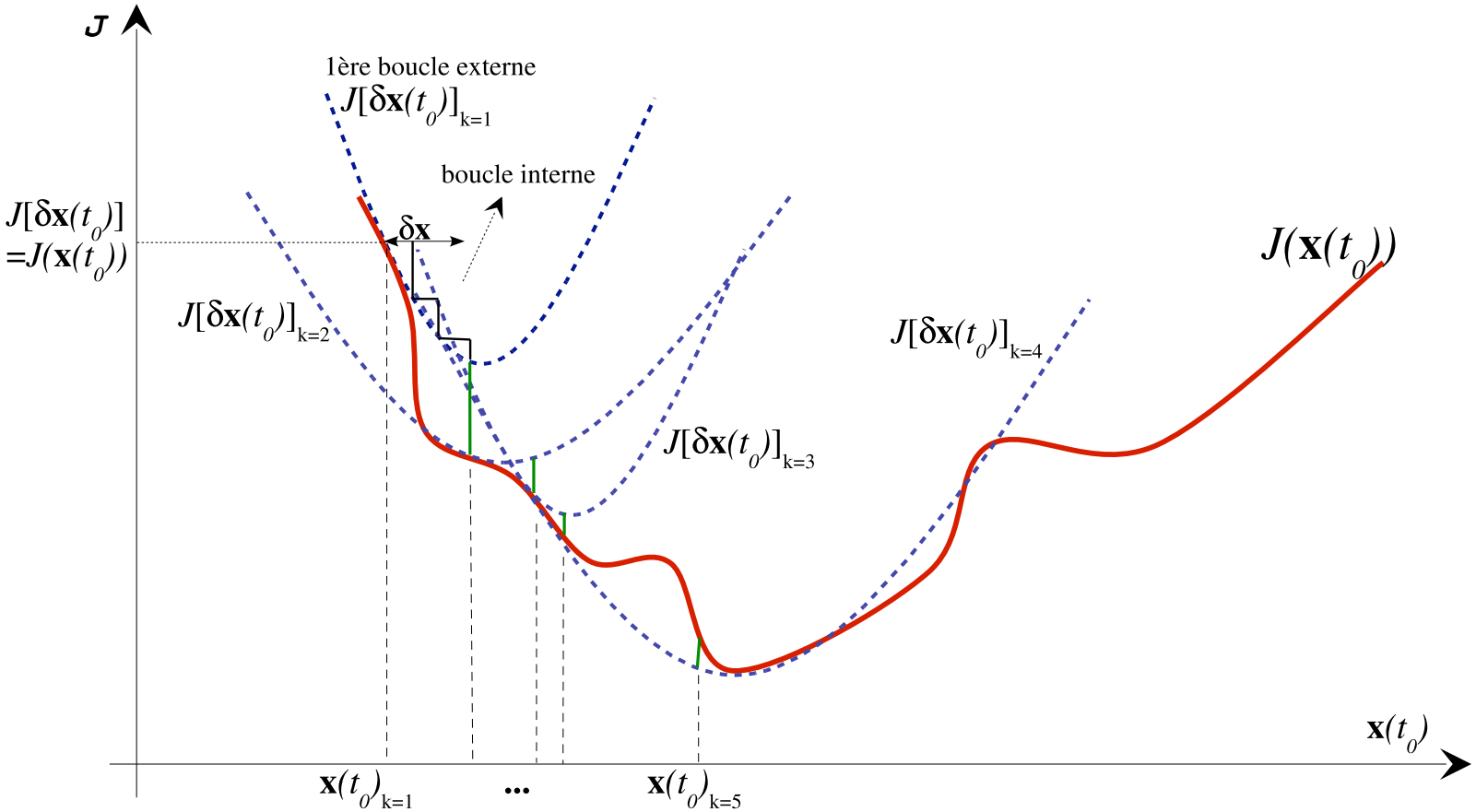
  END OF INNER LOOP

- Analysis increment update $\delta\mathbf{x}_0^a = \delta\mathbf{x}_0$
- Reference state update $\mathbf{x}_0^r = \mathbf{x}_0^r + \delta\mathbf{x}_0^a$

END OF OUTER LOOP

– Compute final analysis: $\mathbf{x}_0^a = \mathbf{x}_0^r$, $\mathbf{x}_k^a = \mathbf{M}_{0,k}[\mathbf{x}_0^a]$.

Incremental 4D-Var



(from YAO user's guide)

Matrix size problems

Vector and matrix sizes

$$\begin{array}{l} \text{size of } \mathbf{x}\text{: } n \\ \text{size of } \mathbf{y}^o\text{: } m \end{array} \quad \Longrightarrow \quad \begin{array}{l} \text{size of } \mathbf{B}\text{: } n \times n \\ \text{size of } \mathbf{H}\text{: } m \times n \end{array}$$

For some applications, n and m are large (10^6 to 10^8) $\Rightarrow$ impossible to store/compute/multiply/inverse data assimilation matrices $\mathbf{B}$ and $\mathbf{K}$!

Possible solutions to model covariance matrices:

- Rank reduction methods: e.g. replace $\mathbf{B}$ by $\mathbf{S}\mathbf{S}^T$ where $\mathbf{S}$ is smaller ($n \times r$, with $r \ll n$) ;
- Ensemble modelling, using Monte-Carlo method.

Rank reduction

Square root decomposition

A symetric positive definite matrix $\mathbf{B}$ can be decomposed into $\mathbf{S}\mathbf{S}^T$, where $\mathbf{S}$ is a $n \times n$ matrix

As before, if n is large, $\mathbf{S}$ cannot be computed/stored.

Rank reduction consists in

1. choosing only a small number r of *significative* columns, to get matrix $\mathbf{S}_r$ with size $n \times r$,
2. setting $\mathbf{B}_r = \mathbf{S}_r \mathbf{S}_r^T$ and hope for $\mathbf{B}_r \simeq \mathbf{B}$.

Methods to compute columns of $\mathbf{S}_r$: empirical orthogonal functions, principal component analysis, proper orthogonal decomposition, singular value decomposition, etc.

$\longrightarrow$ SEEK filter, reduced-rank (incremental) 4D-Var

Ensemble modelling

Monte-Carlo estimation

If $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_N$ are N realisations of $\mathbf{x}$, then an estimator of $\mathrm{E}(\mathbf{x})$, based on the law of large numbers, is given by

$$\hat{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

Also, if $\mathbf{x}_1$, x_2, ..., $\mathbf{x}_N$ are N *well-chosen* states of a physical system, the background error covariance matrix $\mathbf{B}$ can be estimated by

$$\mathbf{B} \simeq \hat{\mathbf{B}} = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}_i - \hat{\mathbf{x}})(\mathbf{x}_i - \hat{\mathbf{x}})^T$$

$\longrightarrow$ ensemble Kalman filter, 4D-Var with ensemble modelling of covariances.

Back to 4D-Var

Initial state estimation $\mathbf{x} = \mathbf{x}_0$, cost function:

$$\mathcal{J}(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^b\|^2_{\mathbf{B}} + \sum_{i=0}^{n} \|\mathbf{y}_i^o - H_i(M_i(M_{i-1}(\ldots M_1(\mathbf{x})))\|^2_{\mathbf{R}}$$

Gradient involves the adjoint model:

$$-\frac{1}{2}\nabla\mathcal{J}^o(\mathbf{x}) = \sum_{i=0}^{n} \mathbf{M}_1^T \ldots \mathbf{M}_{i-1}^T \mathbf{M}_i^T \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{d}_i$$

Innovation vector $\mathbf{d}_i$:

$$\mathbf{d}_i = y_i^o - H_i(M_i(M_{i-1}(\ldots M_1(\mathbf{x}))))$$

Gradient computation

Gradient of $\mathcal{J}^o$ factorization:

$$
\begin{aligned}
-\tfrac{1}{2}\nabla\mathcal{J}^o(\mathbf{x}) &= \sum_{i=0}^{n} \mathbf{M}_1^T \ldots \mathbf{M}_{i-1}^T \mathbf{M}_i^T \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{d}_i \\
&= \mathbf{H}_0^T \mathbf{R}_0^{-1} \mathbf{d}_0 + \mathbf{M}_1^T \mathbf{H}_1^T \mathbf{R}_1^{-1} \mathbf{d}_1 + \mathbf{M}_1^T \mathbf{M}_2^T \mathbf{H}_2^T \mathbf{R}_2^{-1} \mathbf{d}_2 + \ldots + \\
&\quad \mathbf{M}_1^T \ldots \mathbf{M}_{n-1}^T \mathbf{M}_n^T \mathbf{H}_n^T \mathbf{R}_n^{-1} \mathbf{d}_n \\
&= \mathbf{H}_0^T \mathbf{R}_0^{-1} \mathbf{d}_0 + \mathbf{M}_1^T \left[\mathbf{H}_1^T \mathbf{R}_1^{-1} \mathbf{d}_1 + \mathbf{M}_2^T [\mathbf{H}_2^T \mathbf{R}_2^{-1} \mathbf{d}_2 + \ldots + \right. \\
&\quad \left. \mathbf{M}_n^T \mathbf{H}_n^T \mathbf{R}_n^{-1} \mathbf{d}_n] \right]
\end{aligned}
$$

Adjoint model

$$
\left\{
\begin{array}{rcl}
\mathbf{x}_k^* &=& \mathbf{M}_{k+1}^T \mathbf{x}_{k+1}^* + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{d}_k, \ k = n, 0 \\
\mathbf{x}_n^* &=& \mathbf{H}_n^T \mathbf{R}_n^{-1} \mathbf{d}_n
\end{array}
\right. \quad \Rightarrow \quad \nabla\mathcal{J}^o = -2\mathbf{x}_0^*
$$

Automatic differentiation

Adjoint code construction can be very technical work:

- time-dependance and non-linearities
- non-differentiabilities, thresholds
- iterative solvers

Community portal for automatic differentiation: `http://www.autodiff.org`

Our favorite, with advanced features and tailored for all kind of applications, even large-scale time dependant:
Tapenade `http://www-sop.inria.fr/tropics`