Didier Auroux

Laboratoire Jean-Alexandre Dieudonné

Université de Nice Sophia Antipolis

auroux@unice.fr

Assimilation de données

Première réunion ANR IODISSEE - IMT - 2010

Plan

1. Qu'est-ce que l'assimilation de données ?
2. Méthodes variationnelles : 4D-VAR, estimation de paramètres
3. Méthodes séquentielles : filtres de Kalman
4. Réduction d'ordre : POD, 4D-VAR réduit

Qu'est-ce que l'Assimilation de Données ?

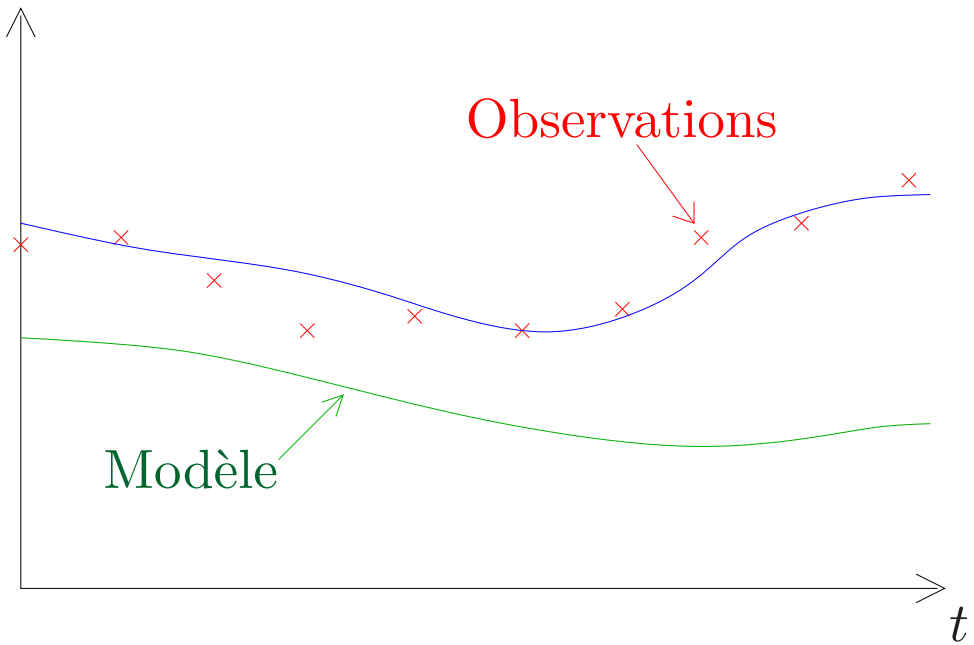
Assimilation de données

C'est l'ensemble des techniques permettant de combiner de façon optimale l'information contenue dans le modèle (équations mathématiques) et celles venant des observations (mesures physiques).

Pour les fluides géophysiques (atmosphère, océan, ...) : systèmes turbulents $\rightarrow$ forte dépendance par rapport à l'état initial.

On cherche donc à identifier l'état initial à partir des observations.

Assimilation de données



combinaison

modèle + observations

$\Downarrow$

estimation de la condition initiale

état de l'écoulement

- Équations non-linéaires
- Non reproductibilité (unicité d'une trajectoire)
- Condition initiale et conditions aux limites ? ?

Comment identifier ces conditions ?

Couplage modèles-données

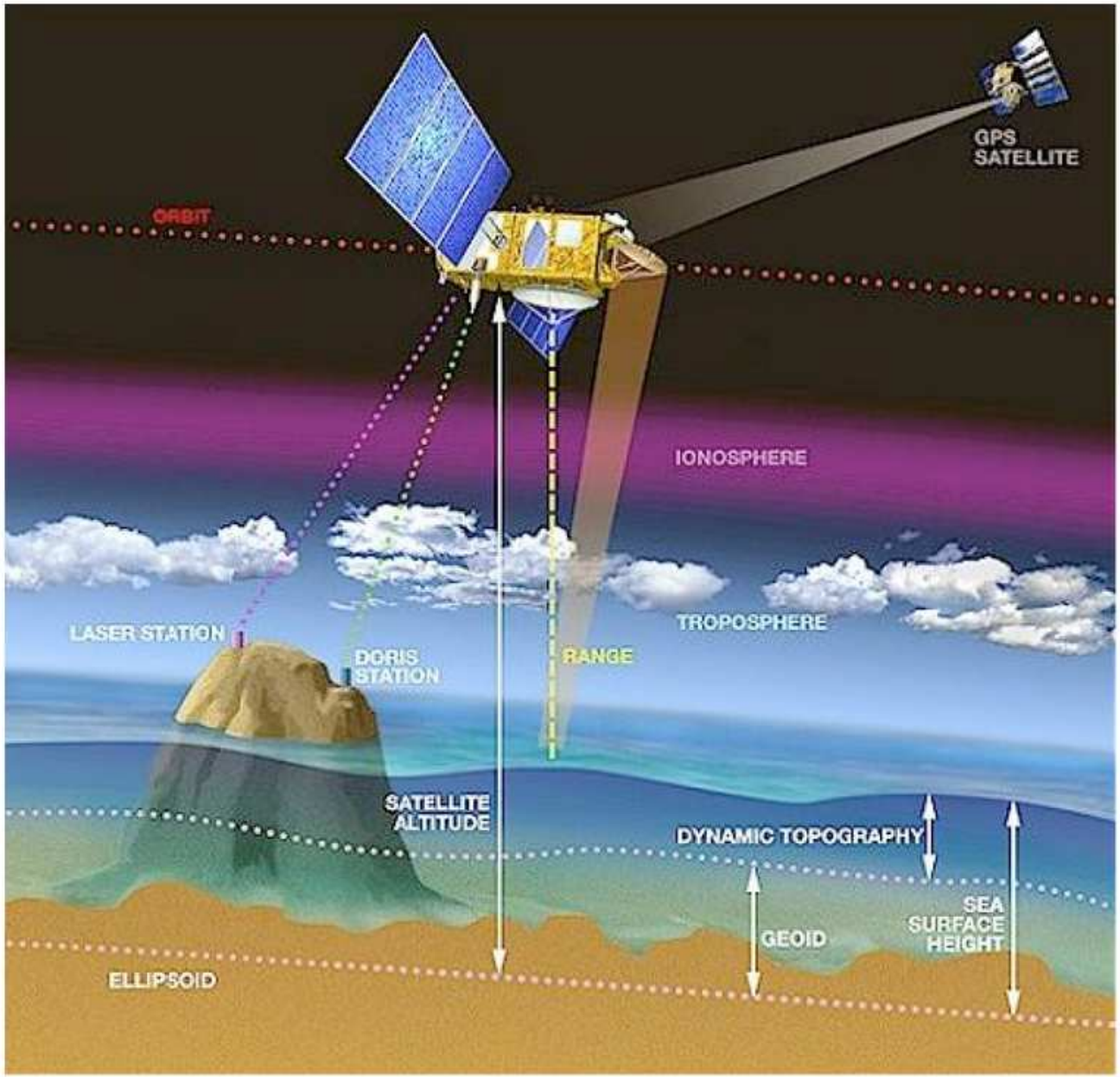
Modèles :

- Variables du modèle : vitesse du fluide, pression, température, humidité (atmosphère) ou salinité (océan)
- Équations d'état : conservation de la masse, de la quantité de mouvement, conservation d'énergie, lois de comportement

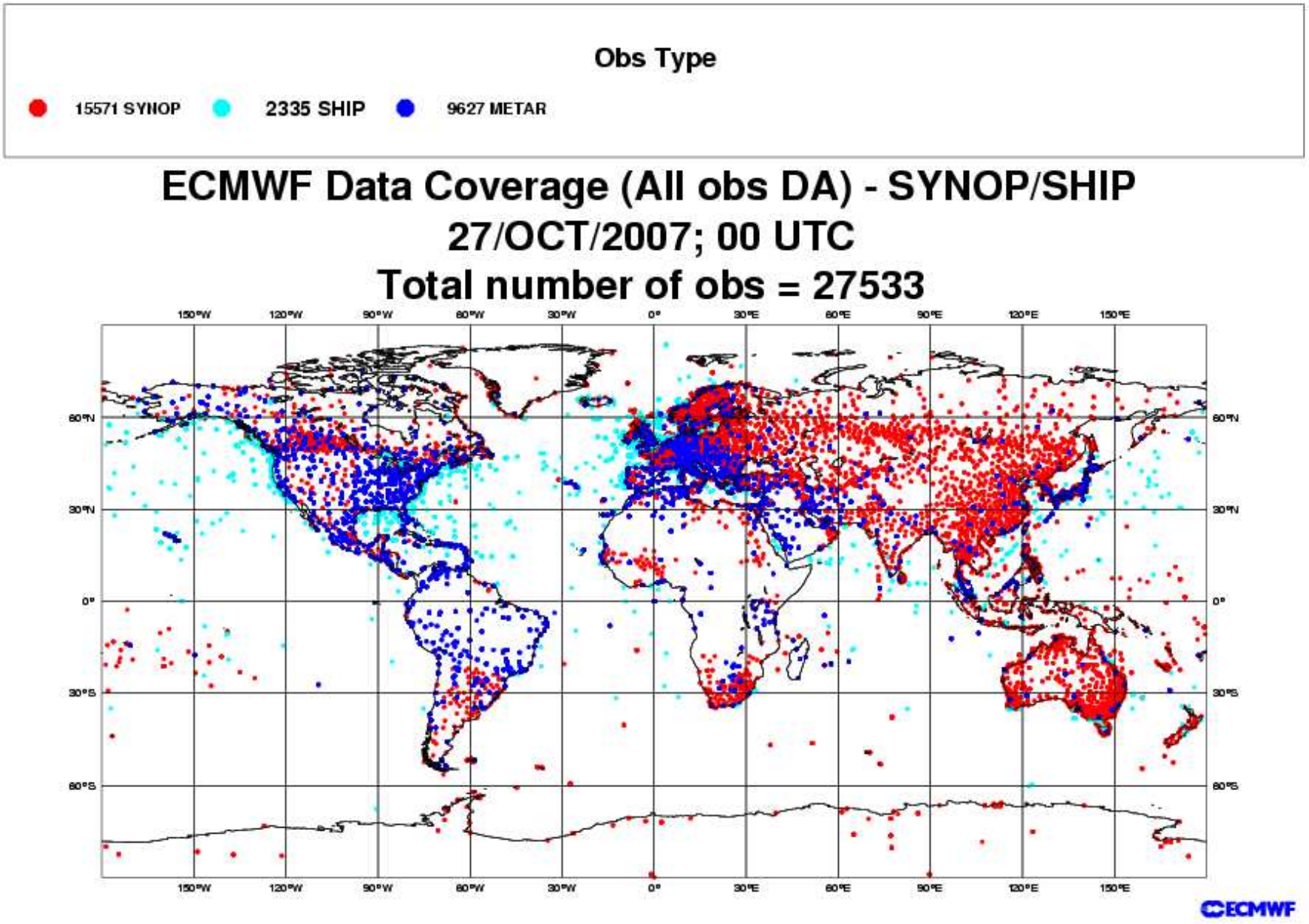
Données :

- Observations terrestres : mesures de direction et vitesse du vent, température, humidité, pression
- Radiosondes : profils verticaux de température, pression, ...
- Ballons pilotes : mesures de vent
- Satellites : profils de température, vitesse du vent (nuages)

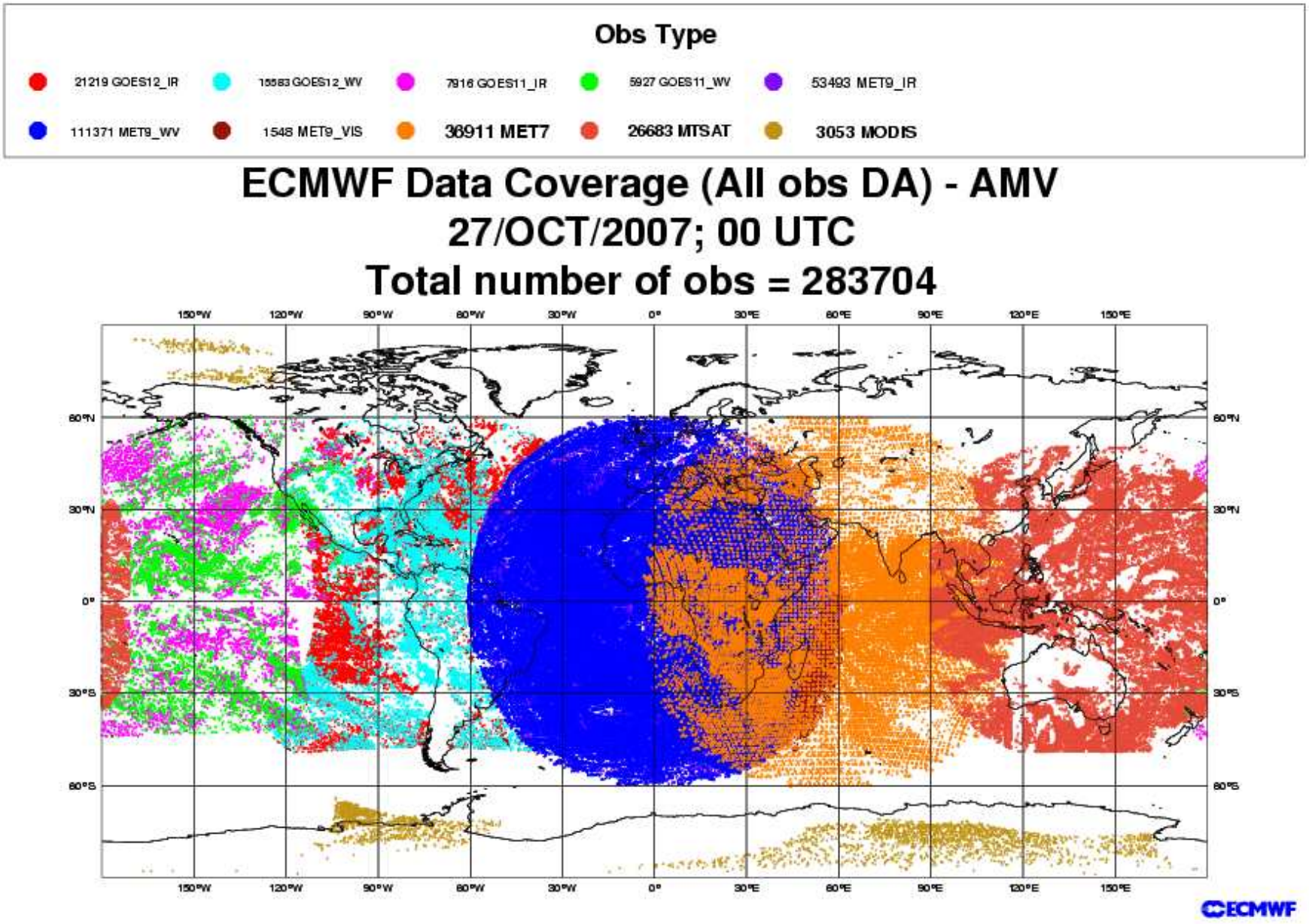
Opérateur d'observation



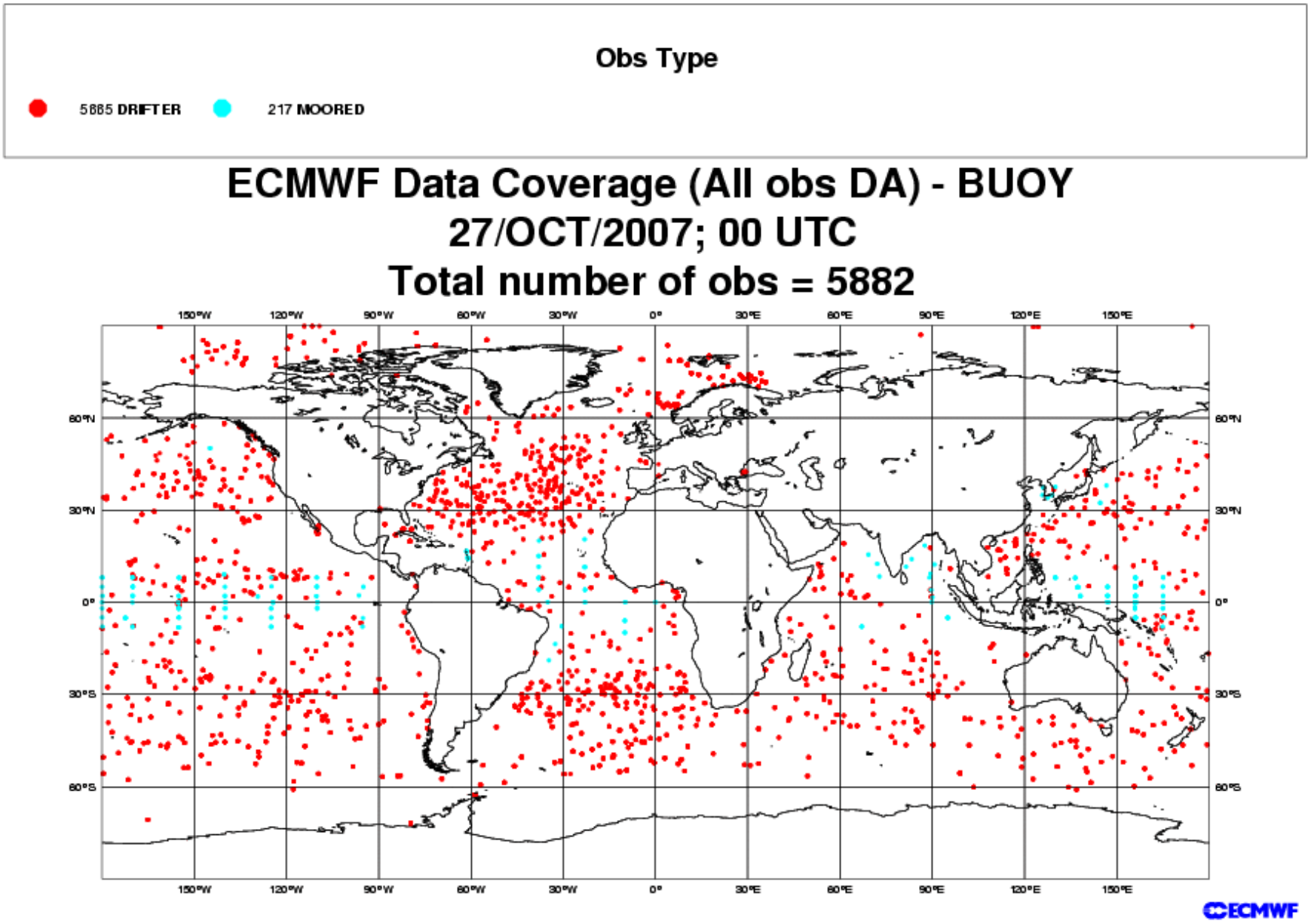
Observations 1



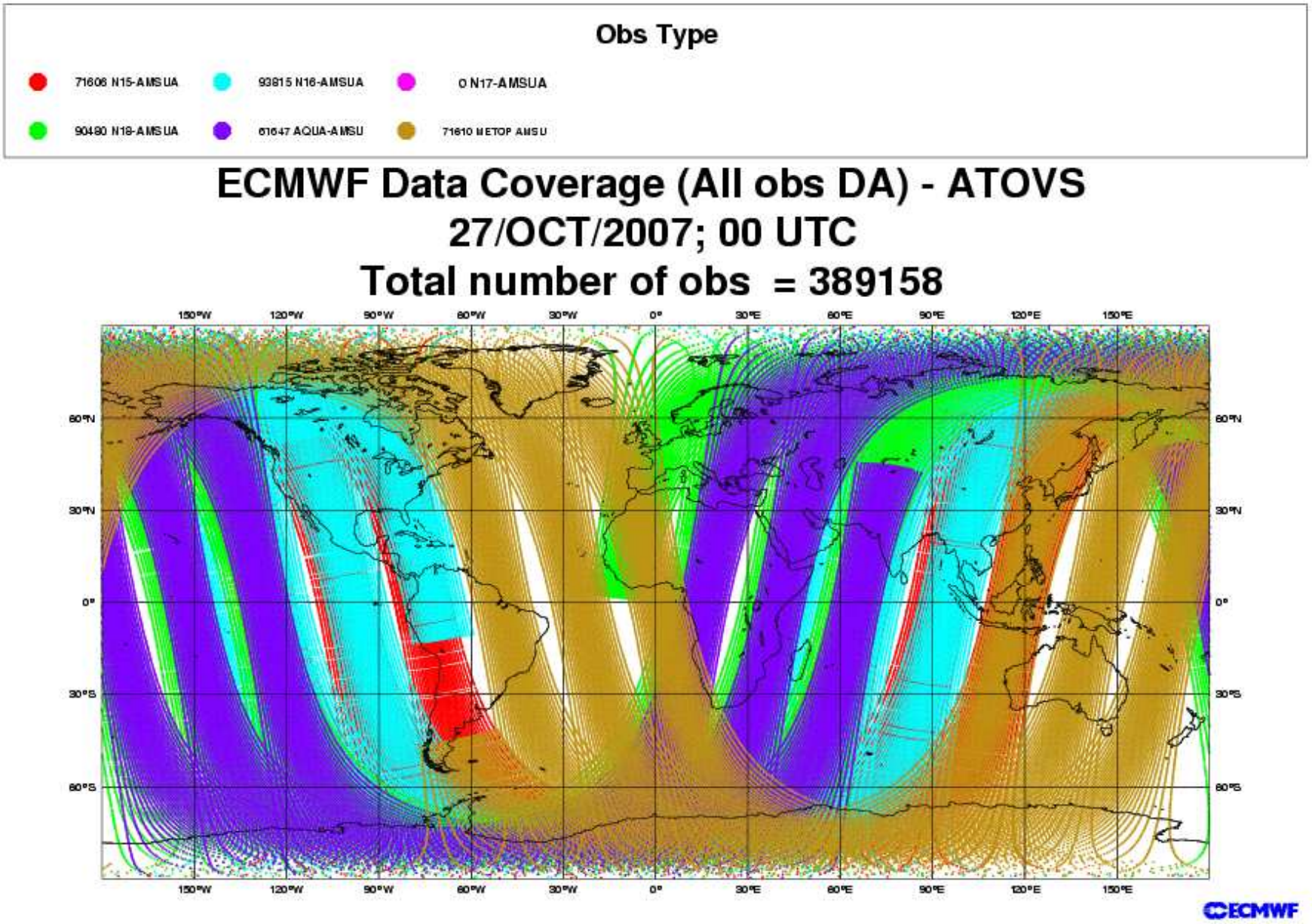
Observations 2



Observations 3



Observations 4



Méthodes d'assimilation de données

Méthodes séquentielles :

- interpolation optimale (mise en œuvre très facile, mais physiquement inconsistant)
- filtre de Kalman (théorie de l'estimation statistique optimale, utilise des matrices de covariance de grande dimension)
- filtre SEEK (filtre réduit)

Méthodes variationnelles :

- 3D-VAR (recherche de la meilleure estimation de l'état à un instant donné)
- 4D-VAR (ajout de la dimension temporelle, théorie du contrôle optimal)

Méthodes Variationnelles

Notations

On considère un système physique dont l'évolution est régie par un système d'équations différentielles de la forme

$$\begin{cases} \dfrac{dx}{dt} = F(x), \\ x(0) = x_0 \end{cases}$$

$x(t) \in \mathbb{R}^n$ est le vecteur d'état, $x_0 \in \mathbb{R}^n$ est la condition initiale.

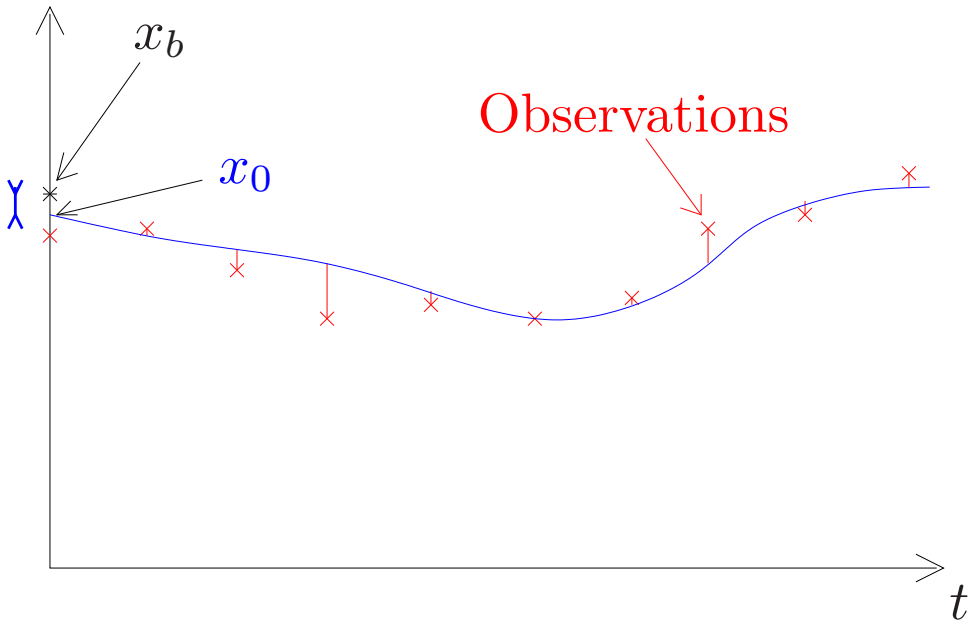
$x_{obs}(t) \in \mathbb{R}^p$ désigne les observations du système disponibles à l'instant t.

H désigne l'opérateur d'observation de $\mathbb{R}^n$ dans $\mathbb{R}^p$ qui permet de relier le vecteur d'état x et les observations x_{obs}.

x_b désigne l'ébauche de l'état du système.

4D-VAR

Généralisation du 3D-VAR en ajoutant la dimension temporelle.



$$\begin{cases} \dfrac{dx}{dt} = F(x), \\ \\ x(0) = x_0, \end{cases}$$

$(t_i)_{0 \leq i \leq n}$, les instants où des observations $x_{obs}(t_i)$ sont disponibles, H_i les opérateurs d'observation et R_i leur matrice de covariance d'erreur.

$$\begin{aligned} J(x_0) &= \frac{1}{2}(x_0 - x_b)^T B^{-1}(x_0 - x_b) \\ &+ \frac{1}{2}\sum_{i=0}^{n} \left(x_{obs}(t_i) - H_i(x(t_i))\right)^T R_i^{-1} \left(x_{obs}(t_i) - H_i(x(t_i))\right) \end{aligned}$$

4D-VAR

Difficultés :

La dépendance de J par rapport au contrôle x_0 est désormais implicite ($x(t_i)$).

Comment dépendent les termes $x(t_i)$ en fonction de la condition initiale x_0 ?

Comment calculer le gradient de J ?

Méthode des différences finies : généralement exclue pour des raisons de dimension et coût de calcul (le calcul de J en un point nécessite déjà la résolution du modèle).

Formulation lagrangienne

On réécrit le problème de minimisation de la fonction coût J sous la contrainte que x soit une solution de l'équation du modèle, en introduisant un multiplicateur de Lagrange p :

$$\mathcal{L}(x_0, x, p) = J(x_0) + \int_0^T \left\langle p, \frac{dx}{dt} - F(x) \right\rangle dt$$

Dans un cadre linéaire (modèle linéaire) : Si (x_0^*, x^*, p^*) est un point-selle de $\mathcal{L}$, alors x_0^* est solution du problème de minimisation de départ.

Point-selle (x_0^*, x^*, p^*) :

$$\mathcal{L}(x_0^*, x^*, p) \leq \mathcal{L}(x_0^*, x^*, p^*) \leq \mathcal{L}(x_0, x, p^*), \quad \forall x_0, x, p.$$

Système d'optimalité

Modèle direct : $\begin{cases} \dfrac{dx}{dt} = F(x) \\ x(0) = x_0 \end{cases}$

Modèle adjoint : $\begin{cases} -\dfrac{dp}{dt} = \left[\dfrac{\partial F}{\partial x}\right]^T p - \displaystyle\sum_{i=0}^{n} H_i^T R_i^{-1}(H_i(x(t_i)) - x_{obs}(t_i)) \\ p(T) = 0 \end{cases}$

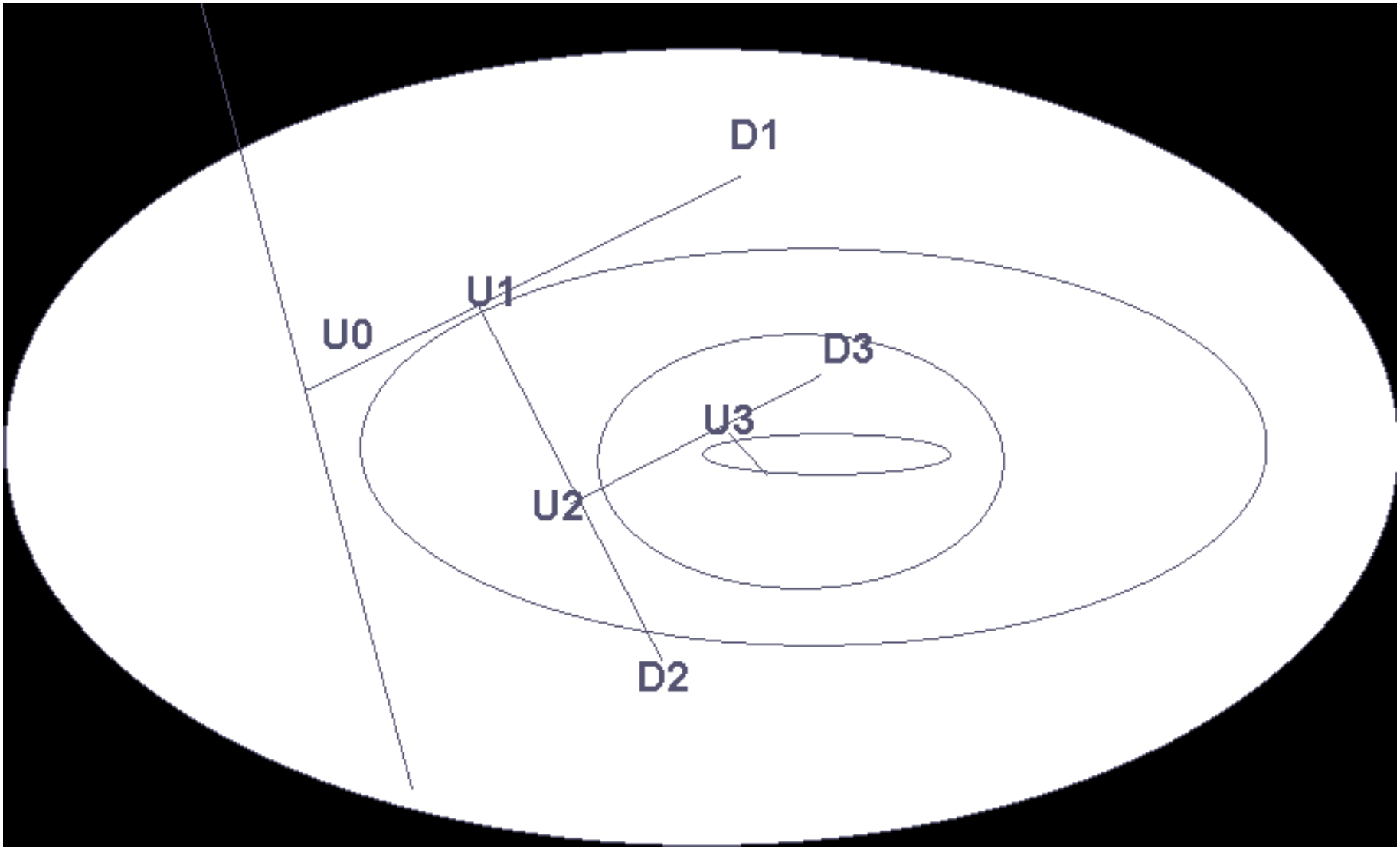
À l'optimum, le gradient de J est nul : $x_0 = x_b + Bp(0)$.

Algorithme 4D-VAR

$$\begin{cases} x_{n-1}(0) & \overset{\text{modèle direct}}{\longrightarrow} & x_{n-1}(t) & \overset{\text{modèle adjoint}}{\longrightarrow} & p_{n-1}(t) \\ & & & & \downarrow \\ x_n(0) & \overset{\text{algorithme de descente}}{\longleftarrow} & & \nabla J_{n-1} = & \nabla J(x_{n-1}(0)) \end{cases}$$

Chaque itération dans l'algorithme de minimisation de J ne coûte *que* 2 résolutions de systèmes d'EDP dans $\mathbb{R}^n$, l'une directe (modèle direct), l'autre rétrograde (modèle adjoint).

Minimisation



Difficultés de mise en œuvre

Discrétisation et contrôle :

$$\begin{array}{ccc} \text{Modèle direct continu} & \longrightarrow & \text{Modèle direct discret} \\ \downarrow & & \downarrow \\ \text{Adjoint continu} & \overset{FAUX}{\longrightarrow} & \text{Adjoint du modèle discret} \\ \downarrow & & \downarrow \\ \text{Système d'optimalité} & & \text{Système d'optimalité discret} \end{array}$$

Le système d'optimalité discret résout le problème direct discret, et le gradient par l'adjoint est bien le gradient discret.

Dérivation du code adjoint :

- Code direct compliqué → dérivation automatique
- Linéaire tangent : dérivé du code direct en mode direct
- Adjoint : dérivé du code direct en mode adjoint

Minimisation de la fonction coût

But : minimiser la fonction coût J, si possible en ne connaissant que son gradient ∇J, éventuellement la hessienne $\nabla^2 J$ (cf adjoint 2nd ordre).

Problèmes :

- la minimisation s'effectue sur un espace de très grande dimension ($\sim 10^9$) ;
- une évaluation de la fonction coût et de son gradient coûte cher ($\sim$ quelques minutes) ;
- le modèle est non linéaire, donc la fonctionnelle n'est pas convexe/quadratique ($\rightsquigarrow$ très nombreux minima locaux).

Solution : Méthode de type quasi-Newton : L-BFGS.
(une parmi d'autres...)

4D-VAR incrémental

On travaille désormais sur l'incrément $\delta x_0 = x_0 - x_b$. La fonctionnelle incrémentale est :

$$J(\delta x_0) = \frac{1}{2}\delta x_0^T B^{-1}\delta x_0 + \frac{1}{2}\sum_{i=0}^{n}[d_i - H_i'(x_b)\delta x(t_i)]^T R_i^{-1}[d_i - H_i'(x_b)\delta x(t_i)]$$

où $d_i = x_{obs}(t_i) - H_i(x_b(t_i))$ est le vecteur d'innovation correspondant à la $i^{\text{ème}}$ observation et $\delta x(t_i)$ est la solution du système linéaire tangent à l'instant t_i.
On linéarise H_i au voisinage de l'ébauche.
Une fois le minimum de J trouvé, on l'ajoute à l'ébauche pour reconstituer l'état initial x_0.

Intérêt : On se limite à quelques modes de grande variabilité et on résout la minimisation de la fonctionnelle incrémentale sur un espace de petite dimension (~ 10).
Méthode actuellement opérationnelle à Météo France.

Estimation de paramètres

On suppose dans cette partie que le modèle dépend de certains paramètres u, a priori inconnus.

L'équation du modèle est désormais la suivante :

$$\begin{cases} \dfrac{dx}{dt} = F(x, u), \\ x(0) = x_0. \end{cases}$$

On souhaite désormais identifier le jeu de paramètres u (en plus de la condition initiale x_0) qui minimise l'écart aux données :

$$\begin{aligned} J(x_0, u) &= \frac{1}{2}(x_0 - x_b)^T B^{-1}(x_0 - x_b) + \frac{1}{2}(u - u_b)^T Q^{-1}(u - u_b) \\ &+ \frac{1}{2}\sum_{i=0}^{n} \left(x_{obs}(t_i) - H_i(x(t_i))\right)^T R_i^{-1} \left(x_{obs}(t_i) - H_i(x(t_i))\right) \end{aligned}$$

Remarque : on peut supposer que x_0 est donné (ou connu) et ne chercher le minimum de J que par rapport à u.

Estimation de paramètres

L'algorithme 4D-VAR fonctionne de la même façon, en ajoutant u comme varible de contrôle (en plus de x_0) :

$$\mathcal{L}\left(x_0, u; x, p\right) = J(x_0, u) + \int_0^T \left\langle p, \frac{dx}{dt} - F(x, u) \right\rangle dt$$

Cela donne toujours en une seule résolution du modèle adjoint le gradient de la fonction coût par rapport à la condition initiale et par rapport aux paramètres.

À l'optimum :

$$x_0^* = x_b + B\, p(0) \qquad u^* = u_b + Q \left[\frac{\partial F}{\partial u}(x, u)\right]^T p \quad \text{(à chaque instant)}$$

Différentiation automatique

Généralement, on travaille localement sur le code direct, ligne par ligne si possible. Les règles sont :

- L'adjoint d'une séquence d'instructions est la séquence inverse des instructions adjointes.
- L'entrée d'une routine devient la sortie de la routine adjointe et réciproquement.
- Le codage de l'adjoint se fait au niveau le plus bas des routines, sur le plus petit bloc d'instructions qui décrive un opérateur linéaire avec des cœfficients connus. L'adjoint est alors la transposée de cet opérateur.
- Chaque variable modifiée fait partie des entrées du code adjoint, sauf la toute première fois qu'elle est définie.
- Chaque variable d'entrée fait partie des sorties du code adjoint sauf si c'est la toute dernière fois qu'elle est utilisée dans le code direct.
- La commande adjointe de la fin d'utilisation d'une variable est sa définition à 0.

Différentiation automatique

Difficultés réelles :

- Non dérivabilité d'une fonction : soit on simplifie le code direct pour le rendre dérivable, soit on donne une valeur arbitraire à la dérivée.

Exemple : une fonction d'interpolation sur une grille est dérivable partout sauf sur la grille. Malgré tout, il est facile d'attribuer une valeur à la dérivée.

- Non linéarités intrinsèques du code direct : les variables directes sont requises dans les codes tangent et adjoint. Il existe plusieurs solutions pour obtenir ces valeurs :

1. stocker toutes les valeurs en mémoire ;
2. écrire toutes les valeurs sur le disque ;
3. recalculer les valeurs au fur et à mesure ;
4. écrire une partie des valeurs sur le disque et recalculer les valeurs intermédiaires au fur et à mesure ;
5. idem en interpolant les valeurs intermédiaires non stockées.

Logiciels

Il existe différents logiciels de différentiation automatique :
(`http : //www.autodiff.org`)

- ADIFOR/ADIC (pour Fortran 77 et C/C++, développé par Rice Univ. & Argonne National Lab., USA)
- ADMAT (pour Matlab, développé par Waterloo Univ., Canada)
- TAF/TAC++, anciennement TAMC (pour Fortran 77/95 et C/C++, développé par FastOpt, Allemagne)
- TAPENADE (pour Fortran 77/90, développé par l'INRIA Sophia-Antipolis, France)
- YAO (pour C/C++, développé par le LOCEAN, France)

Méthodes Séquentielles

Interpolation optimale

Principe : Chercher une combinaison linéaire optimale entre les observations et les états du système aux mêmes instants.

L'estimateur qui réalise le minimum de la variance de l'erreur d'estimation est le BLUE, Best Linear Unbiaised Estimator.

Soit x_b une estimation de l'état du système x avant assimilation, et x_a l'état analysé. Soit x_{obs} le vecteur des observations, et Hx_b l'état correspondant du système.
$d = x_{obs} - Hx_b$ est le vecteur d'innovation.

On cherche une estimation sous la forme

$$x_a = x_b + K(x_{obs} - Hx_b)$$

où K minimise la variance de l'erreur d'estimation.

Interpolation optimale

On note :

$e_b = x - x_b$ l'erreur d'ébauche,

$e_o = x_{obs} - Hx$ l'erreur d'observation, et

$e_a = x - x_a$ l'erreur d'analyse.

On a alors

$$e_a = (I - KH)e_b - Ke_o.$$

La matrice de covariance de l'erreur d'analyse est (par définition) :

$$P_a = \overline{e_a e_a^T},$$

où $\overline{\dots} = E(\dots)$ représente l'espérance mathématique (ou moyenne).

Filtre de Kalman

Extension temporelle de l'interpolation optimale :

On note x_n^a l'état analysé à l'instant t_n, et M_n la résolvante du système permettant de passer de l'instant t_n à l'instant t_{n+1}.
L'ébauche de l'état du système à l'instant t_{n+1} est (**phase de prédiction**) :

$$x_{n+1}^f = M_n x_n^a.$$

On note P_n^f la matrice de covariance de l'erreur d'ébauche $x_n^f - x_n$ et P_n^a la matrice de covariance de l'erreur d'analyse $x_n^a - x_n$.

$$\begin{aligned} P_{n+1}^f &= E\left((x_{n+1}^f - x_{n+1})(x_{n+1}^f - x_{n+1})^T\right) \\ &= E\left((M_n x_n^a - M_n x_n)(M_n x_n^a - M_n x_n)^T\right) \end{aligned}$$

et donc

$$P_{n+1}^f = M_n P_n^a M_n^T.$$

Filtre de Kalman

À partir de l'ébauche x^f_{n+1} et du vecteur d'innovation $x_{obs}(t_{n+1}) - Hx^f_{n+1}$, on construit l'état analysé x^a_{n+1} de façon analogue à l'interpolation optimale (**phase de correction**) :

$$x^a_{n+1} = x^f_{n+1} + K_{n+1}(x_{obs}(t_{n+1}) - Hx^f_{n+1})$$

avec K_{n+1} choisi pour minimiser la variance de l'erreur d'analyse :

$$K_{n+1} = P^f_{n+1}H^T \left(HP^f_{n+1}H^T + R\right)^{-1}.$$

On en déduit la nouvelle matrice de covariance d'analyse :

$$P^a_{n+1} = [I - K_{n+1}H]P^f_{n+1}.$$

Équivalence avec le 4D-VAR

En supposant que la matrice de covariance d'erreur d'ébauche est la même à l'instant initial, alors l'état analysé à l'instant final x_N^a produit par le filtre de Kalman coïncide avec l'état final de la trajectoire optimale produite par le 4D-VAR.

Algorithme KF

Filtre de Kalman :

- Initialisation :

$$x_0^f \text{ et } P_0^f \text{ donnés}$$

- Analyse :

$$\begin{aligned} K_n &= P_n^f H^T [H P_n^f H^T + R]^{-1} \\ x_n^a &= x_n^f + K_n(x_{obs\,n} - H x_n^f) \\ P_n^a &= [I - K_n H] P_n^f \end{aligned}$$

- Prévision :

$$\begin{aligned} x_{n+1}^f &= M_{n;n+1} x_n^a \\ P_{n+1}^f &= M_{n;n+1} P_n^a M_{n;n+1}^T \end{aligned}$$

Problèmes numériques

- Les erreurs sont généralement mal connues $\rightsquigarrow$ matrices de covariance d'erreur P et R mal identifiées $\rightsquigarrow$ filtre sous-optimal.

- Dimension de ces matrices : $n \times n$ où n est la dimension spatiale du problème ($10^7 - 10^8$ pour les océans, $10^9 - 10^{10}$ pour l'atmosphère).

- Coût de calcul pour faire évoluer ces matrices à chaque pas de temps.

- Non-linéarités, et linéarisations locales.

Réduction d'Ordre

Réduction d'ordre

Principe : réduire la dimension du problème, afin d'accélérer et simplifier la mise en œuvre effective des algorithmes, comme par exemple le 4D-VAR.

L'idée consiste à réduire significativement la dimension de l'espace dans lequel on cherche la solution (ou la condition initiale, ou les paramètres).

Méthode classiquement utilisée : POD, Proper Orthogonal Decomposition (décomposition en modes propres orthogonaux), équivalent à une analyse en composantes principales.

Filtre SEEK

Filtre SEEK (Singular Evolutive Extended Kalman) :
prend en compte à la fois un système non linéaire et une réduction d'ordre (i.e. réduction de la dimension du problème).
La propagation de l'erreur de filtrage est donnée par

$$x^a_{n+1} - x(t_{n+1}) = (I - K_{n+1}H)M_n(x^a_n - x(t_n))$$

$$-K_{n+1}(x_{obs}(t_{n+1}) - Hx^f_{n+1})$$

et la stabilité du filtre dépend essentiellement de

$$(I - K_{n+1}H)M_n.$$

Toutes les valeurs propres de cet opérateur doivent être de module strictement inférieur à 1, pour que l'erreur décroisse au cours du temps.

L'idée du filtre SEEK est de considérer des approximations de faible rang des matrices de covariance P^a_n, tout en préservant la stabilité du filtre.

Algorithme SEEK

- Initialisation :

$$x_0^f \text{ et } P_0^f \text{ donnés (rang} = r\text{, ou sinon on approche cette matrice)}$$

- Analyse :

$$\begin{aligned}
P_n^f &= S_n^f S_n^{fT} \\
K_n &= S_n^f [I_r + (HS_n^f)^T R^{-1} (HS_n^f)]^{-1} (HS_n^f)^T R^{-1} \\
x_n^a &= x_n^f + K_n [x_{obs\,n} - H(x_n^f)] \\
P_n^a &= S_n^f [I_r + (HS_n^f)^T R^{-1} (HS_n^f)]^{-1} S_n^{fT}
\end{aligned}$$

- Prévision :

$$\begin{aligned}
x_{n+1}^f &= M_{n;n+1}(x_n^a) \\
[S_{n+1}^f]_j &= M(x_n^a + [S_n^a]_j) - M(x_n^a), \quad 1 \le j \le r \\
P_{n+1}^f &= S_{n+1}^f (S_{n+1}^f)^T
\end{aligned}$$

4D-VAR réduit

La condition initiale est cherchée sous la forme suivante :

$$x_0 = x_b + \sum_{k=1}^{r} \alpha_k(0)\phi_k(0),$$

donc la fonction coût peut se réécrire sous la forme :

$$\tilde{J}(\alpha_1(0), \ldots, \alpha_r(0)) = J(x_0) = \text{même expression que dans le 4D-VAR.}$$

De plus, le modèle peut également être réduit, en projetant l'équation du modèle suivant les différents modes propres $\phi_k(t)$, de sorte qu'à chaque instant, la trajectoire $x(t)$ du modèle réduit peut se décomposer sous la forme

$$x(t) = x_b(t) + \sum_{k=1}^{r} \alpha_k(t)\phi_k(t).$$

4D-VAR réduit

La résolution du modèle réduit est fortement accélérée, puisqu'on peut le voir comme un système d'équations en dimension r.

La minimisation de la fonction coût est également accélérée, puisqu'elle a lieu dans un espace de dimension r.

Par contre, il faut tenir compte du coût de calcul nécessaire à l'obtention des modes POD, à savoir essentiellement le coût de calculer les r trajectoires perturbées du modèle.