Les principaux algorithmes d'assimilation

Rappel du BLUE:

forme matricielle: $x_a = x_b + BH^T(HBH^T+R)^{-1}\,[y-H(x_b)]$

forme variationnelle: minimiser $J(x) = (x-xb)^T\, B^{-1}\, (x-x_b) + [y-H(x)]^T\, R^{-1}\, [y-H(x)]$

Les algorithmes variationnels 3D:

- 3DVar
- 3DVar en forme incrémentale et/ou préconditionnée
- 3DEnVar
- 3DEnVar racine carrée

Les algorithmes variationnels 4D:

- 4DVar (+ forme incrémentale et/ou préconditionnée)
- 4DEnVar
- 4DVar à contrainte faible

Les filtres de Kalman d'ensemble:

- EnKF
- LETKF (local ensemble transform Kalman filter)

Les curiosités: interpolation optimale, PSAS, filtres particulaires

Le 3DVar : analyse variationnelle 3D

Principe: **minimiser itérativement J(x)** = $(x-x_b)^T B^{-1} (x-x_b) + [y-H(x)]^T R^{-1} [y-H(x)]$ par récurrence:

1. on initialise l'itération: $x_0=x_b$
2. à partir de x_i on calcule x_{i+1} avec un **algorithme de minimisation** qui utilise les opérateurs:
   - J: x -> J(x) évaluation de la **fonction-coût**
   - ∇J: x -> $\nabla J(x) = 2 B^{-1} (x-x_b) - 2 H^T R^{-1} [y-H(x)]$ évaluation de son **gradient** (=sa dérivée)
3. on arrête la récurrence,
   - soit lorsque $|| \nabla J(x_i)|| < a || \nabla J(x_i)||$ avec *a* = facteur de tolérance (ex: 0.01),
   - soit au bout d'un nombre prédéfini d'itérations (ex: 100)
4. l'analyse est définie par $x_a = x_i$

NB. si H est non linéaire, H^T est la transposée de H linéarisé au voisinage de x_i

Exemple d'algorithme python de minimisation: *scipy.optimize.minimize(method='BFGS')*

Le 3DVar incrémental

idée: **variante numériquement moins chère du 3DVar:**

- On définit S, une **interpolation de x à plus basse résolution**, et S^{-1} son pseudo inverse (interpolation à pleine résolution)
- On effectue un **changement de variable**: on remplace l'argument x de J par $\mathbf{z = S(x - x_b)}$
- et on linéarise H au voisinage de l'ébauche x_b: $H(x) \sim H(x_b) + H'(x-x_b)$

alors $J(x) = (x-x_b)^T B^{-1} (x-x_b) + [y-H(x)]^T R^{-1} [y-H(x)]$

devient $\bar{J}(z) = J(x) = z^T S^{-T} B^{-1} S^{-1} z + [y-H(x_b)+H'S^{-1}z]^T R^{-1} [y-H(x_b)+H'S^{-1}z]$ où H' est linéaire

- on démarre la minimisation avec $z_0 = 0$
- à la fin (après i itérations) on revient à pleine résolution par $x_a = x_b + S^{-1} z_i$

intérêt: la **fonction $\bar{J}(z)$ est exactement quadratique** même si H est non-linéaire, et son argument est **de plus petite dimension** que J, elle est donc (généralement) plus facile à minimiser.

Le préconditionnement du 3DVar

idée: variante plus efficace du 3DVar

- on fait le changement de variable $z = B^{-1}(x - x_b)$
- on linéarise H au voisinage de l'ébauche x_b: $H(x) \sim H(x_b) + H'(x-x_b)$

alors $J(x) = (x-x_b)^T B^{-1} (x-x_b) + [y-H(x)]^T R^{-1} [y-H(x)]$

$= (x-x_b)^T B^{-1} B B^{-1} (x-x_b) + [y-H(x)]^T R^{-1} [y-H(x)]$

devient $\bar{J}(z) = z^T B z + [y-H'(x_b)+H'Bz]^T R^{-1} [y-H(x_b)+H'Bz]$ avec H' opérateur linéaire

- on démarre la minimisation avec $z_0 = 0$
- à la fin (après i itérations) on prend $x_a = x_b + B z_i$

intérêt:

- la fonction $\bar{J}(z)$ est exactement quadratique -> minimisation plus rapide
- on n'a besoin que de B, pas de B^{-1} -> plus besoin d'inverser B
- on peut combiner avec l'incrémental (cf. planche précédente)

variante: on peut préconditionner par une racine carrée de B : $B=LL^T$

Le 3DenVar "3DVar ensembliste"

idée: **améliorer B avec une prévision d'ensemble:**

- B_s une racine carrée de la matrice B climatologique ("statique")
- { xi } une prévision d'ensemble à n membres qui représente les incertitudes de l'ébauche x_b (utilise une assimilation d'ensemble) avec $\overline{x}$ sa moyenne

Alors la "matrice B du jour" $1/(n-1) \sum_i (x_i - \overline{x}) (x_i - \overline{x})^T$ est une approximation du B exact (**"flow-dependent"**)

Pb: B du jour est bruitée et n'est pas définie positive si n<dim(x) -> J n'est pas strictement convexe

Solutions :

- **"localisation"** (~lissage) des corrélations dans $\sum_i (x_i - \overline{x}) (x_i - \overline{x})^T$
- matrice **B hybride** : $B = a\, B_s + (1-a)\, 1/(n-1) \sum_i (x_i - \overline{x}) (x_i - \overline{x})^T$ où *a* est un coefficient entre 0 et 1.

Le 3DEnVar se minimise comme un 3DVar classique: $J(x) = (x-x_b)^T B^{-1} (x-x_b) + [y-H(x)]^T R^{-1} [y-H(x)]$

s'utilise avec incrémental + préconditionnement pour éviter d'inverser une B devenue compliquée

Le 3DenVar "racine carrée"

idée: **3DenVar défini par la racine carrée de B:**

- $L_s L_s^T = B_s$ une matrice B climatologique ("statique")
- $\{ x_i \}$ une prévision d'ensemble à n membres

Alors on définit une racine carrée de la "matrice B du jour" avec L_s et les x_i :

$L = a\, L_s + (1-a)/(n-1) [xi]$ et $B = LL^T$

Se minimise avec le changement de variable $z = L^{-1} (x - x_b)$

et alors: $J(z) = z^T z + [y-H(Lz)]^T R^{-1} [y-H(Lz)]$

à la fin on définit l'analyse par $x_a = x_b + Lz_i$

Intérêt : facile à minimiser car parfaitement préconditionné.

Le 4DVar : analyse variationnelle 4D

Principe: assimiler une **séquence d'observations y(t) :**

$J(x) = (x-x_b)^T B^{-1} (x-x_b) + \sum_t [y(t)-H_t(x(t))]^T R_t^{-1} [y_t-H_t(x(t))]$

avec $x(t) = M_t(x)$ défini par un modèle de prévision M depuis l'instant de x_b jusqu'à l'instant t.

1. on initialise l'itération: $x_0(t=0)=x_b$
2. à partir de x_i on calcule les $x_i(t) = M_t(x_i(t=0))$ par une prévision numérique avec M
3. On calcule la fonction-coût avec cette prévision J: x -> J(x)
4. On calcule le gradient de chaque terme par rapport à x à l'instant 0 :

∇J: x -> $\nabla J(x) = 2 B^{-1} (x-x_b) - 2 \sum_t M_t^T H_t^T R_t^{-1} [y_t-H_t(M_t(x))]$

5. et on minimise comme en 3DVar

<u>problème:</u> on doit calculer de nombreuses prévisions M et M^T.
<u>solution:</u> comme M est une succession d'opérateurs d'avance temporelle $M_t M_{t-1} ... M_1$,on peut factoriser les M_t^T dans le terme de droite -> coûte un M et un M^T par itération

M^T s'appelle le modèle adjoint de M (= transposée de la différentielle de M)

Le 4DenVar : 4DVar + 3DenVar

3DenVar appliqué à une séquence temporelle d'observations (comme en 4DVar),
mais on minimise par rapport à toute la séquence des $(x_0,x_1,...,x_t)$:

$$J(x0,x1,...,xt) = \sum_t (x_t - x_{bt})^T B_t^{-1}(x_t-x_{bt}) + \sum_t (y_t - H_t(x_t))^T R_t^{-1} (y_t - H_t(x_t))$$

avantage : reproduit une partie de la dynamique du 4DVar (parce que le modèle M est utilisé pour définir les $(B_0,B_1,...,B_t)$ correspondants mais évite le codage du modèle adjoint M^T.

Le 4DVar à contrainte faible

Comme le 4DVar, mais on ajoute un terme Q qui lie entre eux les xt : la séquence des analyses (x0,x1,...xt) n'est plus contrainte à être exactement une prévision de M.

$$J(x_0) = \sum_t (x_t - x_{bt})^T B_t^{-1}(x_t - x_{bt}) + \sum_t (y_t - H_t(x_t))^T R_t^{-1} (y_t - H_t(x_t)) + \sum_t (x_t - M_t(x_0))^T Q^{-1} (x_t - M_t(x_0))$$

avantage : permet de représenter plus réalistement les erreurs de modélisation dans M.

Q est une matrice de covariances d'erreurs de modélisation

(on peut même combiner 4DenVar et contrainte faible)

Les filtres de Kalman d'ensemble

On calcule un ensemble d'analyses avec chaque ébauche $\{x_{bi}\}$ fournie par une prévision d'ensemble:

$$x_{ai} = x_{bi} + BH^T (HBH^T + R)^{-1} (y - H(x_{bi}))$$

et on définit l'analyse par la moyenne de l'ensemble d'analyses $x_a = 1/b \sum_i x_{ai}$

Le LETKF
(local ensemble transform Kalman Filter)

On fait l'analyse BLUE avec un B amélioré par l'ensemble : $B = \textit{1/(n-1)} \sum_i (x_i - \bar{x})(x_i - \bar{x})^T$
Et on calcule l'analyse avec le BLUE matriciel:

$$x_a = x_b + BH^T (HBH^T + R)^{-1} (y - H(x_b))$$

mais comme B n'est pas définie positive (l'ensemble est trop petit) on calcule les champs de x_a par morceaux, sur des sous-domaines où $(HBH^T + R)$ est inversible