Compléments algorithmiques:

- *minimisation dans l'analyse variationnelle*
- *assimilation en temps réel*

Minimisation en analyse variationnelle:

- comment calculer le gradient de J
- comment trouver le minimum de J
- le cas du 4D-Var

rappel mathématique: dérivée vectorielle, gradient

car la plupart des minimiseurs utilisent la dérivée de J

Soit F une application vectorielle $R^n \rightarrow R^p$:

- la dérivée (ou **différentielle**) de F est l'opérateur linéaire F' tel que

$$\forall (x, h) \quad F(x+h) = F(x) + F'(x)\, h + O(||h||^2)$$

- F'(x) est une matrice à p lignes et n colonnes
- les coefficients F_{ij} de la matrice F' sont les dérivées partielles $\partial F_i / \partial x_j$ de F. Ils mesurent la sensibilité de la i-ième sortie de F par rapport à sa j-ième entrée, au voisinage de x.

Cas de la fonction-coût J de l'analyse variationnelle:

- J(x) est un scalaire, J' est un vecteur (identifié à une forme linéaire) ∇J appelé **gradient de J.**
- la dérivée de $\mathbf{J(x) = (x - x_b)^T B^{-1} (x - x_b) + (y - H[x])^T R^{-1} (y - H[x])}$

est le *vecteur* $\mathbf{\nabla J(x) = 2\, B^{-1} (x - x_b) \; - \; 2\, H'^T R^{-1} (y - H[x])}$

calcul de la dérivée de J par code adjoint

•la dérivée de $\mathbf{J(x) = (x - x_b)^T B^{-1} (x - x_b) + (y - H[x])^T R^{-1} (y - H[x])}$

est le *gradient* $\mathbf{\nabla J(x) = 2 B^{-1} (x - x_b) - 2 H'^T R^{-1} (y - H[x])}$

Si x ou y sont grands, et H est complexe, on évite de calculer la matrice .

H est une composée d'applications (une suite de lignes de code)

H1 o H2 o ... o Hm

H' est sa différentielle = produit des différentielles:

H'1 H'2 ... H'm

H'^T est sa transposée = produit des transposées:

$H'm^T$ ... $H'2^T$ $H'1^T$

c'est le **code adjoint de H** = suite inverse des transposées des lignes de code (faisable par différentiation automatique)

ex: a = b**2
c = 3 * b

devient:

```
db = 0                  # injection canonique = adjoint du end-of-scope de b
db = db + 3 * dc
db = db + 2 * b * da    # b est la valeur de b, db sa perturbation
```

préconditionnement de la minimisation

- on veut minimiser $\mathbf{J(x) = (x - x_b)^T B^{-1} (x - x_b) + (y - H[x])^T R^{-1} (y - H[x])}$

la dérivée seconde est $\mathbf{J'' = 2B^{-1} + 2H^T R^{-1} H}$

la vitesse de minimisation est améliorée si J est à peu près diagonale (minimiser le ratio entre +grande et +petite valeur propre)

Exemple: préconditionnement par changement de variable:

si $B^{-1} = L^{-T} L$ avec L inversible: on pose $z = L(x - x_b)$

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

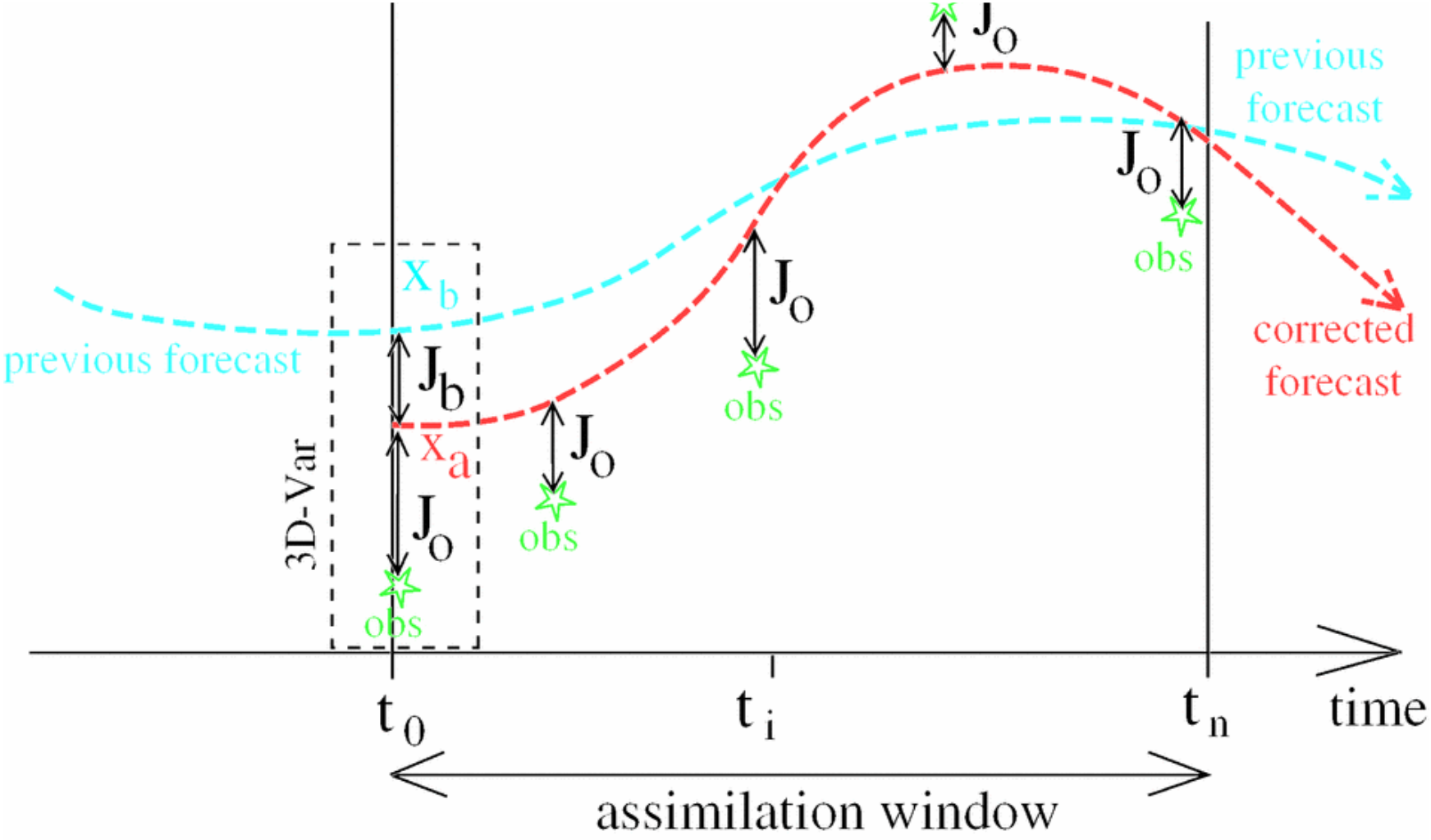
et $G'' = I + ...$

si le nombre d'obs est << dim x, cela permet de gagner plusieurs ordres de grandeur sur la vitesse de minimisation

Assimilation 4DVar: généralisation du BLUE à un historique d'observations, à l'aide d'un modèle de prévision M

$$J(x) = (x - x_b)^T B^{-1} (x - x_b) + \Sigma_i \, (y_i - H_i[\mathbf{M_i}(x)])^T \; R_i^{-1} \, (y_i - H_i[\mathbf{M_i}(x)])$$

La variable de minimisation est l'état x du modèle au début de la fenêtre.



Minimisation du 4DVar

fonction-coût du 4DVar à minimiser:

$$x \rightarrow \mathbf{J(x) = (x - x_b)^T B^{-1} (x - x_b) + \sum_k (y_k - H_k \circ M_k[x])^T R_k^{-1} (y_k - H_k \circ M_k[x])}$$

avec le modèle de prévision M_k: $x=x(t_0) \rightarrow x(t_k)=M_k(x)$

le gradient devient $\mathbf{2\, B^{-1} (x - x_b) \; - \; 2 \sum_k M_k^T H_k^T R_k^{-1} (y_k - H_k \circ M_k[x])}$

or M est une succession de pas de temps: $M_k = N_k\, N_{k-1}\, N_{k-2}\, \ldots\, N_1$

en factorisant on organise le calcul du gradient de droite ainsi:

1) pour chaque xi, intégrer et stocker la '**trajectoire**' (M1(x) , M2(x),..., Mk(x))

2) calculer les écarts obs-trajectoire: $d_k = (y_k - H_k \circ M_k[x])$

3) les multiplier par $H_k^T R_k^{-1} d_k = e_k$

4) intégrer et stocker le '**modèle adjoint forcé**': $z_k=0$, $z_{k-1} = Mk\text{-}1\, z_k + e_k$

Chaque itération de la minimisation coûte 1 intégration du modèle de prévision + 1 intégration adjointe

e modèle est parallélisable, mais pas la minimisation

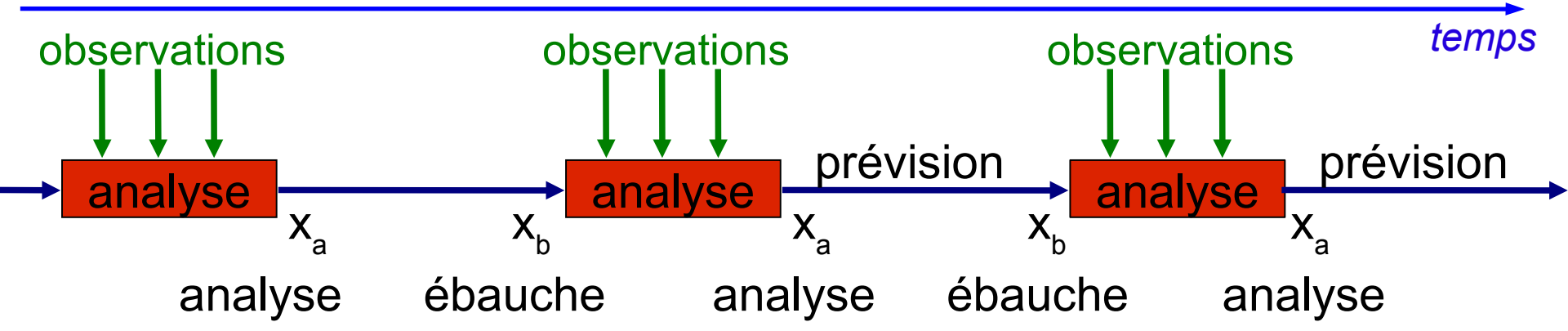
Assimilation en temps réel:

- notion de cutoff des observations
- production de prévisions en temps réel

Assimilations de données temps réel

:
- pour faire des prévisions
- pour le suivi climatique

Problèmes:
- les observations arrivent plus ou moins tard
- des défaillances peuvent rompre la chaîne d'assimilation



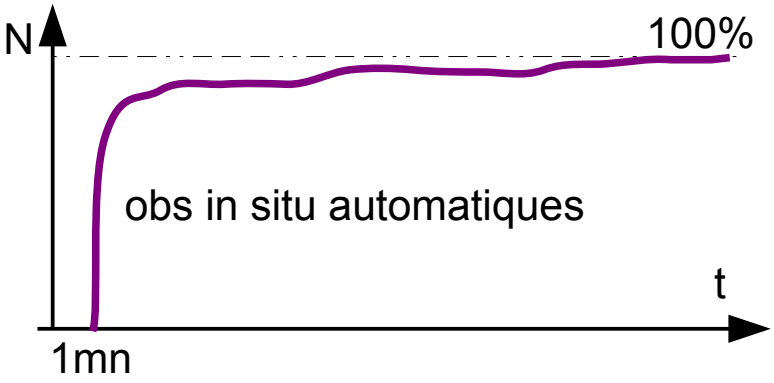
Temps réel: notion de 'cutoff'

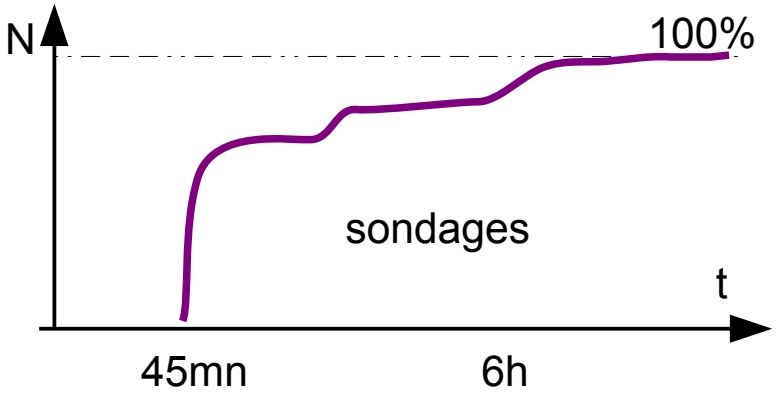
Plus on attend, et...
- ...plus il y a d'obs à assimiler -> meilleures analyses -> meilleures ébauches
- ...moins l'analyse est utile, parce que les prévisions seront disponibles plus/trop tard pour les usagers.

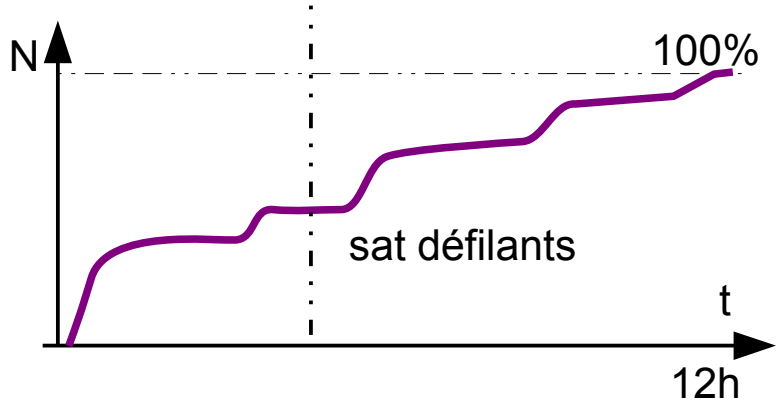
Chaque type d'obs présente des ***latences***: différence entre instant de mesure, et instant de disponibilité dans le système d'analyse:
- réseau de surface in situ automatisé: <1mn
- sondages: 45mn (temps d'ascendance)
- radar: 5 à 15mn (période de balayage)
- satellite géostationnaire: quelques mn (segment sol) + temps de balayage de l'imageur
- satellite défilant: de quelques mn à plusieurs heures en fonction de l'orbitographie et des stations de réception au sol

Temps réel: notion de 'cutoff'

diagrammes typiques: Nobs=f($t_{latence}$)







'cutoff': instant à partir duquel on cesse d'attendre les observations, et on commence

à calculer l'analyse.

Centre de prévision temps réel

On fait tourner 2 chaînes couplées:

- une à cutoff long (~5h), de haute qualité mais tardive
- une à cutoff court (~10mn), non cyclée, pour la production à flux tendus, avec uniquement les obs les plus fraîches