TP méthodes inverses

Kévin Polisano

17/12/2012

1 Présentation du problème

On considère les équations de Lorenz :

$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = \rho x - y - xz \\ \dot{z} = xy - \beta z \end{cases}$$

La condition initiale de référence est $u_0^{ref} = (-4.62, -6.61, 17.94)$, supposée inconnue. On connait seulement une ébauche pas trop éloignée $u^b = (-5, -7, 17)$ et des observations générées à partir de la vraie solution à des dates t_i : $u^{obs}(t_i) = (x(t_i), y(t_i), z(t_i))$.

Le but de ce TP est d'effectuer une assimilation de données pour retrouver la condition initiale de référence à partir de celle d'ébauche et des observations.

2 Résultats théoriques sur le modèle continu

Question 1. Soit $u(t) = (x(t), y(t), z(t))$ la solution du système différentiel correspondant à la condition initiale $u_0 = (x_0, y_0, z_0)$. La fonction coût suivante mesure la distance entre la solution u et l'observation u^{obs} sur toute la durée d'observation T :

$$\mathcal{J}(u_0) = \frac{1}{2} \int_0^T \|u(t) - u^{obs}(t)\| dt$$

remarque : j'ai pris l'initiative de multiplier par le facteur 1/2 pour ne pas avoir des facteurs 2 un peu partout dans l'expression du modèle adjoint et du gradient.

On cherche la condition initiale du système qui donne lieu à une solution qui soit la plus proche possible de ce que l'on observe (problème inverse), c'est-à-dire qu'on cherche u_0 telle que $\mathcal{J}(u_0)$ soit minimal.

Modèle linéaire tangent. On perturbe u_0 dans la direction v_0 et on note $\tilde{u}$ la trajectoire correspondante :

$$\begin{cases} \frac{d\tilde{u}}{dt} = f(\tilde{u}) \\ \tilde{u}(t=0) = u_0 + \alpha v_0 \end{cases} \qquad (2)$$

Par soustraction de (1) et (2) on obtient :

$$\begin{cases} \frac{d\tilde{u}-u}{dt} = f(\tilde{u}) - f(u) \\ \tilde{u}(t=0) - u(t=0) = \alpha v_0 \end{cases} \qquad (3)$$

Or on a :

$$f(\tilde{u}) - f(u) = \begin{pmatrix} \sigma[(\tilde{y}-y)-(\tilde{x}-x)] \\ \rho(\tilde{x}-x)-(\tilde{y}-y)-(\tilde{x}\tilde{z}-xz) \\ \tilde{x}\tilde{y}-xy-\beta(\tilde{z}-z) \end{pmatrix}$$

avec

$$\frac{\tilde{x}\tilde{y}-xy}{\alpha} = \tilde{x}\frac{\tilde{y}-y}{\alpha} + y\frac{\tilde{x}-x}{\alpha} \underset{\alpha\to 0}{\longrightarrow} x\hat{y}+y\hat{x}$$

En divisant par α et en passant à la limite on obtient :

$$\frac{d\hat{u}}{dt} = \begin{pmatrix} \sigma(\hat{y}-\hat{x}) \\ \rho\hat{x}-\hat{y}-x\hat{z}-z\hat{x} \\ x\hat{y}+y\hat{x}-\beta\hat{z} \end{pmatrix} = \begin{pmatrix} -\sigma & \sigma & 0 \\ \rho-z & -1 & -x \\ y & x & -\beta \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \frac{\partial f}{\partial u}\hat{u}$$

$$\begin{cases} \frac{d\hat{u}}{dt} = \frac{\partial f}{\partial u}\hat{u} \\ \hat{u}(t=0) = v_0 \end{cases}$$

Modèle linéaire adjoint.

$$\begin{cases} \frac{dP}{dt} + \left[\frac{\partial f}{\partial u}\right]^t P = u - u^{obs} \\ P(t=T) = 0 \end{cases}$$

Le gradient est alors donné par $(\nabla \mathcal{J})_{u_0} = -P(0)$

Question 2. On a calculé plus généralement en cours le gradient de l'application $x \in \mathbb{R}^m \mapsto \frac{1}{2}\|Ax-y\|^2$ où $A \in \mathcal{M}_{n,m}(\mathbb{R})$ et $y \in \mathbb{R}^n$ qui est $A^t(Ax-y)$. Ici avec $y = u^b$ et $A = I$ on a donc

$$(\nabla \mathcal{J})_{u_0} = u_0 - u^b - P(0)$$

remarque : l'expression du modèle tangent et adjoint reste inchangée.

Question 3. On discrétise l'évolution en notant $u_i = (x_i, y_i, z_i)$ l'état du système à la date ih où h est le pas de temps.

Le schéma numérique le plus simple est celui d'Euler :

$$u_{i+1} = u_i + f(u_i) \Leftrightarrow \begin{cases} x_{i+1} = x_i + h\sigma(y_i - x_i) \\ y_{i+1} = y_i + h(\rho x_i - y_i - x_i z_i) \\ z_{i+1} = z_i + h(x_i y_i - \beta z_i) \end{cases}$$

Ou encore écrit sous forme matricielle :

$$u_{i+1} = \begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{pmatrix} = \begin{pmatrix} 1-h\sigma & h\sigma & 0 \\ h(\rho - z_i) & 1-h & -hx_i \\ hy_i & hx_i & 1-h\beta \end{pmatrix} \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} = M_i u_i$$

remarque : on voit que $\frac{u_{i+1}-u_i}{h} = \frac{1}{h}(M_i - I)u_i = \frac{\partial f}{\partial u}u_i$. On va donc utiliser tacitement le filtre de Kalman étendu.

Comme on se ramène à une approximation linéaire il faut normalement prendre en compte une certaine erreur e_i dans le modèle linéaire, sans biais et de matrice de covariance Q_i. Les observations sont données directement donc l'opérateur d'observation H_i est l'identité, mais on prend également en compte une certaine marge d'erreur sur celles-ci ϵ_i sans biais et de matrice de covariance R_i.

L'algorithme de Kalman s'écrit donc :

Initialisation :

$$u_0^a = u_0^b$$
$$P_0^a = P^b$$

où $u_0^b = (-5, -7, 17)$ est l'état initial approché et P^b est sa matrice de covariance d'erreur.

Etape i :

$$\begin{aligned} u_{i+1}^f &= M_i u_i^a && (KF1) \\ P_{i+1}^f &= M_i P_i^a M_i^T + Q_i && (KF2) \end{aligned}$$

$$\begin{aligned} K_{i+1} &= P_{i+1}^f [P_{i+1}^f + R_{i+1}]^{-1} && (KF3) \\ u_{i+1}^a &= u_{i+1}^f + K_{i+1}[u_i^{obs} - u_{i+1}^f] && (KF4) \\ P_{i+1}^a &= P_{i+1}^f - K_{i+1} P_{i+1}^f && (KF5) \end{aligned}$$

Question 4. Rappelons rapidement le lien entre le filtrage de Kalman et la minimisation de la fonctionnelle $\mathcal{J}$ discrétisée suivante :

$$\mathcal{J}(u_0) = (u_0 - u_0^b)^t (u_0 - u_0^b) + \sum_{i=0}^{n} (u_i^{obs} - u_i)^t (u_i^{obs} - u_i)$$

L'idée est de découper la fonctionnelle en deux parties $\mathcal{J}(u_0) = \mathcal{J}_{0,m}(u_0) + \mathcal{J}_{m,n}(u_0)$. On se rend alors compte que minimiser la fonctionnelle globale sur $[0, t_n]$ revient à minimiser d'abord $\mathcal{J}_{0,m}(u_0)$ sur $[0, t_m]$ donnant un résultat u_0^a qu'on utilise ensuite comme ébauche dans la minimisation de $\mathcal{J}_{m,n}(u_0)$. C'est ce que l'on appelle communément le transfert d'optimalité.

On pousse alors ce raisonnement à l'extrême en subdivisant $[0, t_n]$ en segments élémentaires $[t_i, t_{i+1}]$, et on effectue les minimisations des $\mathcal{J}_{i,i+1}(u_0)$ en cascades sur chacun d'eux en transférant l'optimalité. Or on a vu que minimiser $\mathcal{J}_{i,i+1}(u_0)$ (une 3d-var donc) est équivalent à faire une BLUE. Autrement dit on fait une BLUE à chaque pas de temps, ce qui est précisément ce que fait l'algorithme de Kalman !

remarque : cette démarche n'est toutefois pas complètement équivalente à l'optimisation de la fonctionnelle sur $[0, t_n]$ qui est quant à elle globale. Les deux méthodes donneront seulement lieu à la même estimation de u au temps final t_n.

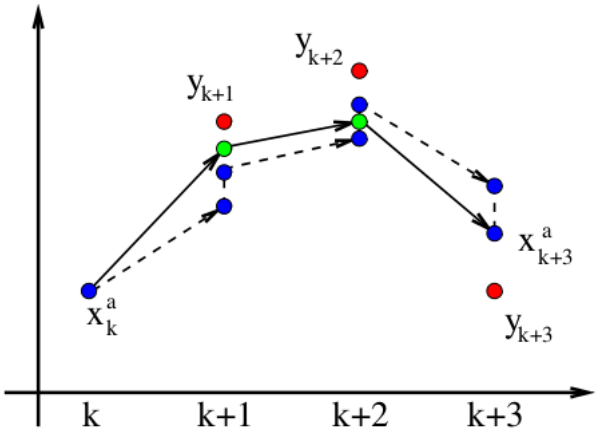

Figure 1: Comparaison assimilation par filtre Kalman et 4d-var (vert)

3 Modèles discrets

Question 5. Le fichier matlab `euler.m` contient 3 méthodes de résolution numérique des équations de Lorenz. La première utilise une primitive de Matlab, à savoir `ode45` qui permet de résoudre de tel système différentiel via un schéma de Runge-Kutta d'ordre 4 ou 5 à pas adaptatif. On obtient bien ci-dessous les même trajectoires que celles figurant dans le sujet.

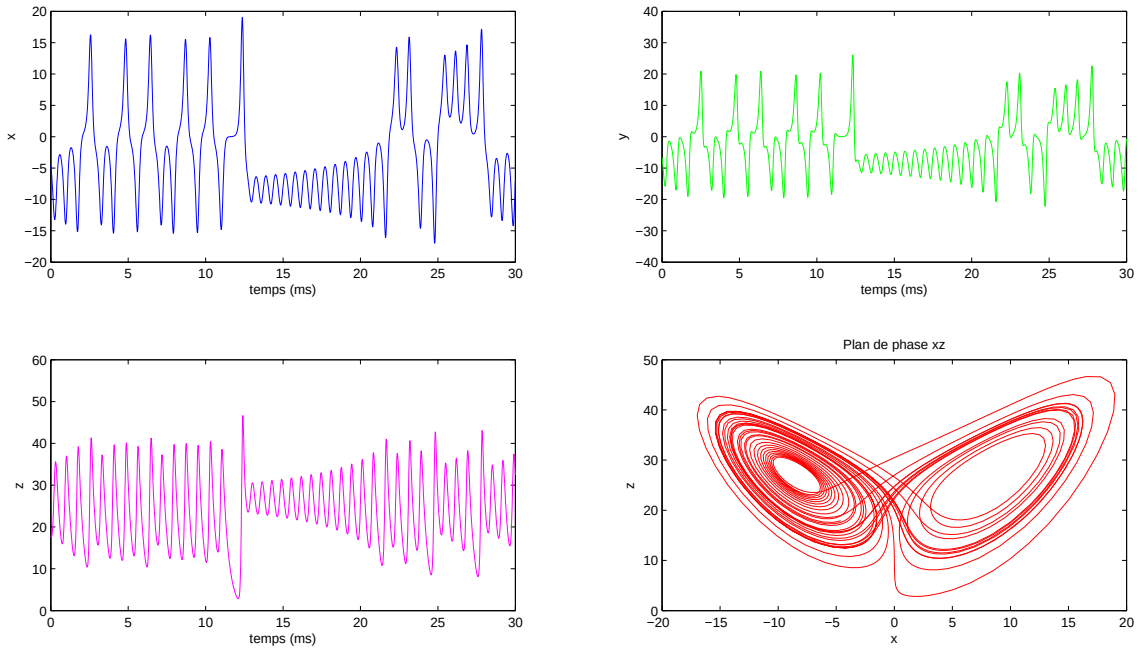

Figure 2: Résolution du système différentiel via ode45

Il était possible de répondre directement à toutes les questions du TP avec cette primitive, en particulier pour le point délicat du calcul du gradient, on pouvait le court-circuiter en résolvant l'équation du modèle adjoint sur P et en prenant $-P(0)$. Seulement, en pratique, on ne discrétise pas le modèle continu de l'adjoint pour calculer le gradient. Plutôt, on part d'un modèle discret caractérisé par un code numérique, et on en détermine un code adjoint (discret lui aussi, via des méthodes de différentiation automatique pour les cas complexes). J'ai décidé de répondre aux questions du TP sous cet angle, afin de bien comprendre comment tout cela fonctionne. C'est pourquoi par la suite je n'aurai plus recours à `ode45`, j'ai implémenté dans `euler.m` les méthodes d'Euler et de Runge-Kutta du second ordre avec un pas de discrétisation h :

$$u_{i+1} = u_i + hf(u_i + \frac{h}{2}f(u_i))$$

Les avantages de cette dernière par rapport à la méthode d'Euler est qu'elle donne une estimation de la dérivée à un ordre supérieur, et surtout qu'elle est stable, tandis que dans le cas d'Euler il faut prendre un pas de temps h excessivement petit pour ne pas que la solution n'explose. C'est toujours le risque avec des méthodes explicites. Dans les simulations numériques on prendra $h = 0.05$ pour RK2 et $h = 0.01$ pour Euler.

remarque : Dans le souci de rester cohérent dans la progression de mon travail, je traiterai la question 9 avant les questions 7 et 8, qui sont les briques de base de la 4d-var faisant l'objet de

la question 10. De même la question 11 portant sur la fréquence d'observation sera traitée et illustrée dans les questions 9 et 10.

Question 9. L'algorithme de Kalman a été décrit à la question 3, en ayant discrétisé le modèle via un schéma d'Euler, qui nous a conduit à une matrice M_i régissant l'évolution temporelle du système ($u_{i+1} = M_i u_i$). Pour des raisons de stabilité j'ai finalement opté pour un schéma de Runge-Kutta. Il est donc nécessaire de recalculer la matrice M_i, c'est-à-dire de connaître explicitement la relation reliant u_i à u_{i+1}. Car dans le fichier `euler.m` $u_{i+1} = u_i + hf(u_i + \frac{h}{2}f(u_i))$ était calculé à partir de variables auxiliaires k_i (voir code). Afin de ne pas noyer le correcteur dans les calculs je me suis limité à du Runge-Kutta d'ordre 2, et pour ne pas me noyer moi-même (après avoir survécu à la fin du monde s'eut été dommage) j'ai contrôlé les expressions avec Maple (voir feuille Lorenz.mv). On obtient les 9 coefficients assez compliqués de la matrice M_l (cf. code `question9.m`). L'algorithme requiert aussi la connaissance des matrices de covariance d'erreur P^b et R_i que nous prenons toutes égales à l'identité (puisque dans la fonction coût on donne le même poids à toutes les variables) et nous supposons que le modèle est parfait $Q_i = 0$.

Toutes les simulations numériques ont été réalisées avec les paramètres de Lorenz suivants :

$$\begin{cases} \sigma = 10 \\ \rho = 28 \\ \beta = \frac{8}{3} \end{cases}$$

Cas où les observations sont parfaites

Ayant au départ oublié d'initialiser toutes les matrices R_i à l'identité, j'ai obtenu le résultat du filtrage ci-dessous. Dans toute la suite la courbe bleue représente la vraie solution, la noire l'ébauche et la rouge l'analyse. Les observations sont repérées par les points verts. En absence de bruit celles-ci se situent exactement sur la courbe bleue.

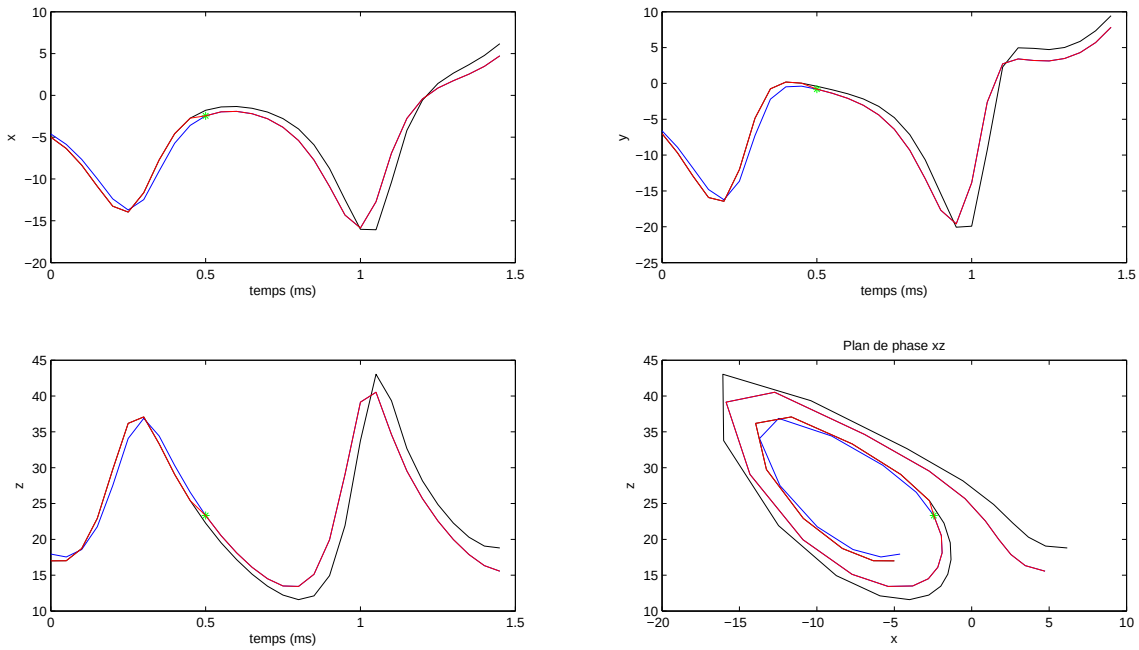

Figure 3: Filtrage de Kalman sur les premières secondes avec des observations parfaites $R_i = 0$

On constate alors que dès la première observation la courbe d'analyse rejoint ce point puis se confond avec la courbe bleue. En effet comme toutes les R_i sont nulles, les variances valent 0 donc les observations sont parfaites et il s'en suit que $K_{i+1} = P_{i+1}^f[P_{i+1}^f + R_i]^{-1} = I$ puis que $u_{i+1}^a = u_{i+1}^f + I[u_{i+1}^{obs} - u_f^{i+1}] = u_{i+1}^{obs}$ et en absence de bruit $u_{i+1}^{obs} = u_{i+1}$. Enfin comme le système est déterministe, les deux courbes se confondent sur tout le reste de la simulation puisqu'elles coincident au point d'observation.

Digression : on voit que bien que les conditions initiales de la courbe d'ébauche et de la vraie solution soient proches, les trajectoires correspondantes se différentient dès les premières secondes de simulation, si on les représente sur la durée totale de l'expérience à savoir 30 s alors il apparait nettement que leur comportement est complètement différent. Ceci est du au comportement chaotique de l'attracteur de Lorenz, même pour une variation infime de 10^{-4} sur la condition initiale, les courbes résultantes peuvent adopter des trajectoires très distinctes. Historiquement c'est d'ailleurs de cette manière que Lorenz a découvert ce phénomène de chaos, en enregistrant d'un jour à l'autre les résultats fournis par son ordinateur avec seulement 4 chiffres significatifs après la virgule. Ceci explique au passage pourquoi on obtient des résultats différents selon la méthode de discrétisation choisie (ode45, Euler ou RK2).

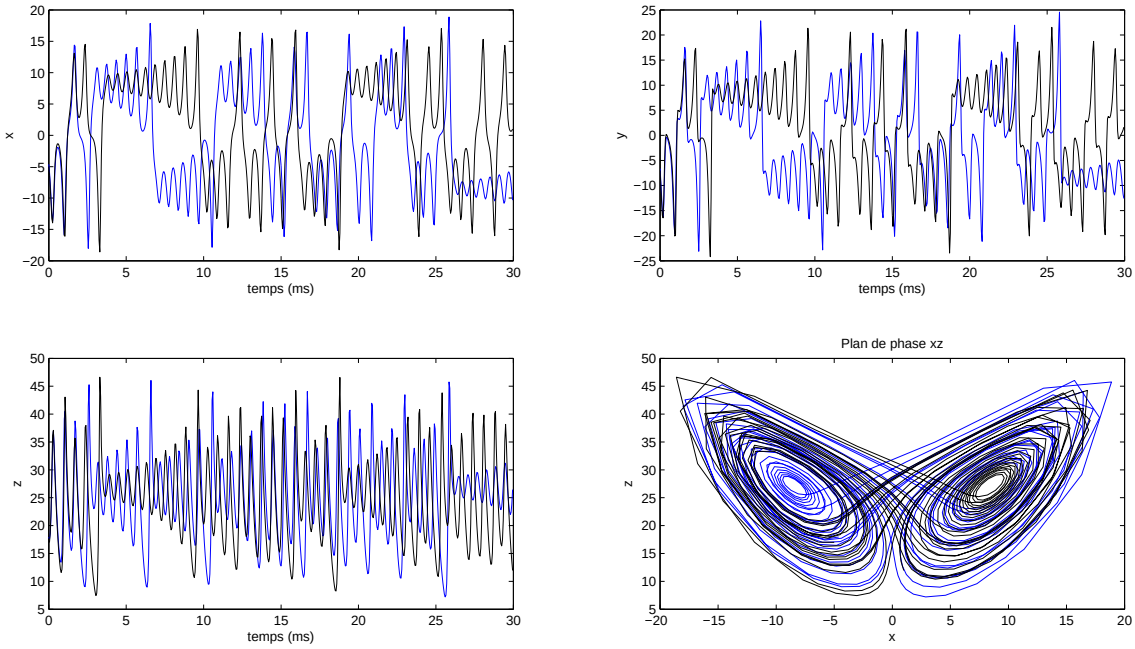

Figure 4: Illustration de la sensibilité à la condition initiale : $u_{ref} = (-4.62, -6.61, 17.94)$ versus $u_b = (-5, -7, 17)$

Cas où les observations sont bruitées

On ajoute aux observations un bruit gaussien de variance donnée `var` via la fonction `randn` de Matlab. Les matrices R_i ne sont alors plus nulles mais diagonales avec les variances du bruit sur la diagonale.

```
var = 1;
```

```
std = sqrt(var);
bruitx = std*randn(nobs+1,1);
bruity = std*randn(nobs+1,1);
bruitz = std*randn(nobs+1,1);

% Le vecteur d'observation est nul sauf aux temps d'observations tobs
j = 1;
for i = tobs
    xob(i) = x(i) + bruitx(j);
    yob(i) = y(i) + bruity(j);
    zob(i) = z(i) + bruitz(j);
    j = j+1;
end

% Matrices d'erreurs d'observations
R = zeros(3,3,tfinal);
for i = 1 : tfinal
    R(:,:,i) = var*eye(3);
end
```

Si on reprends la simulation précédente mais avec des observations bruitées, on obtient le résultat suivant.

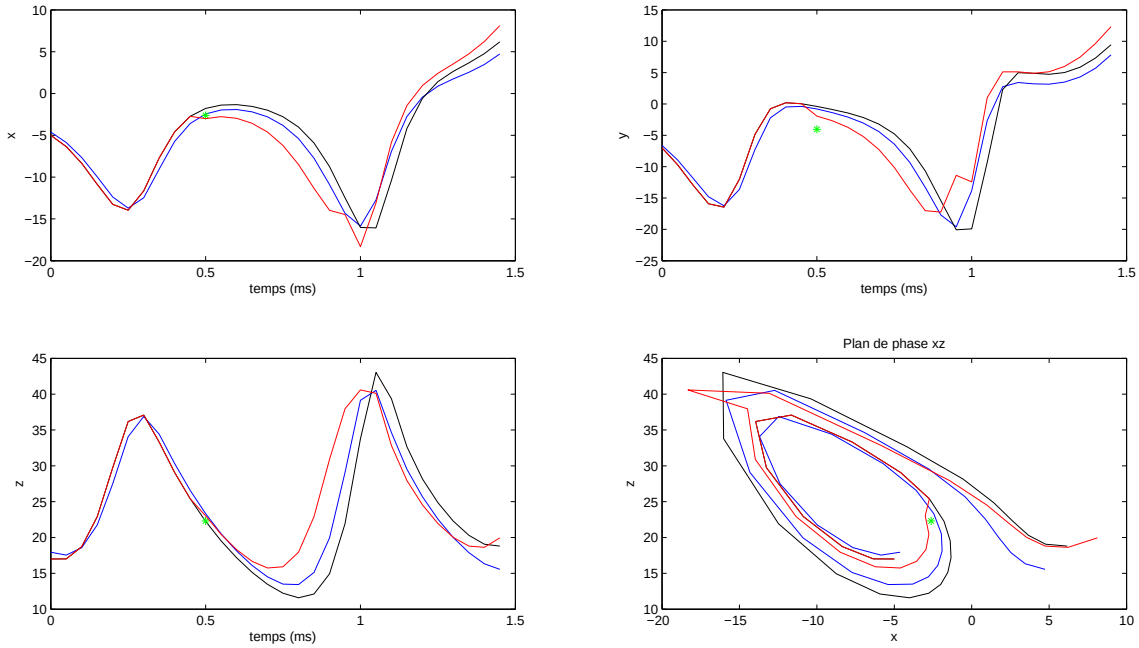

Figure 5: Filtrage de Kalman sur les premières secondes avec observations bruitées ($var = 1$)

Cette fois on remarque que la courbe analyse ne passe plus exactement par le point d'observation mais s'en rapproche toutefois en se démarquant de la courbe d'ébauche noire. On reconnait bien là le fonctionnement de l'algorithme de Kalman effectuant un BLUE à chaque itération :

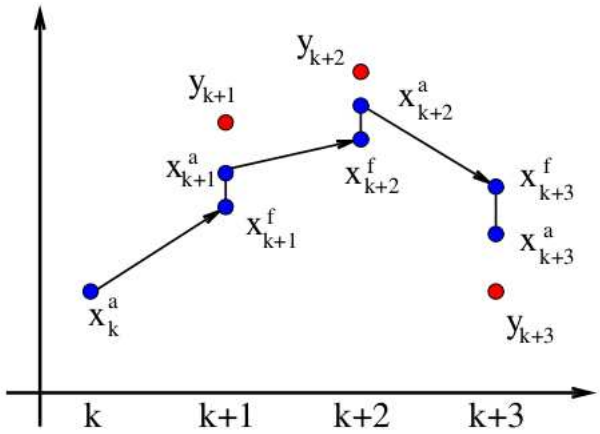

Figure 6: Principe de fonctionnement du filtrage de Kalman

Si on procède à une assimilation de 15 secondes (avec des observations équiréparties : une toute les $10h = 0.5$ s) sur une durée totale d'expérience de 30 secondes, on obtient le résultat ci-dessous. On voit bien que comparé à la courbe d'ébauche noire, la courbe d'analyse elle, a tendance à suivre la courbe bleue grâce aux points de contrôles que sont les observations. Bien sûr les trajectoires sont quand même bien distinctes car l'attracteur de Lorenz est hautement instable comme souligné précédemment.

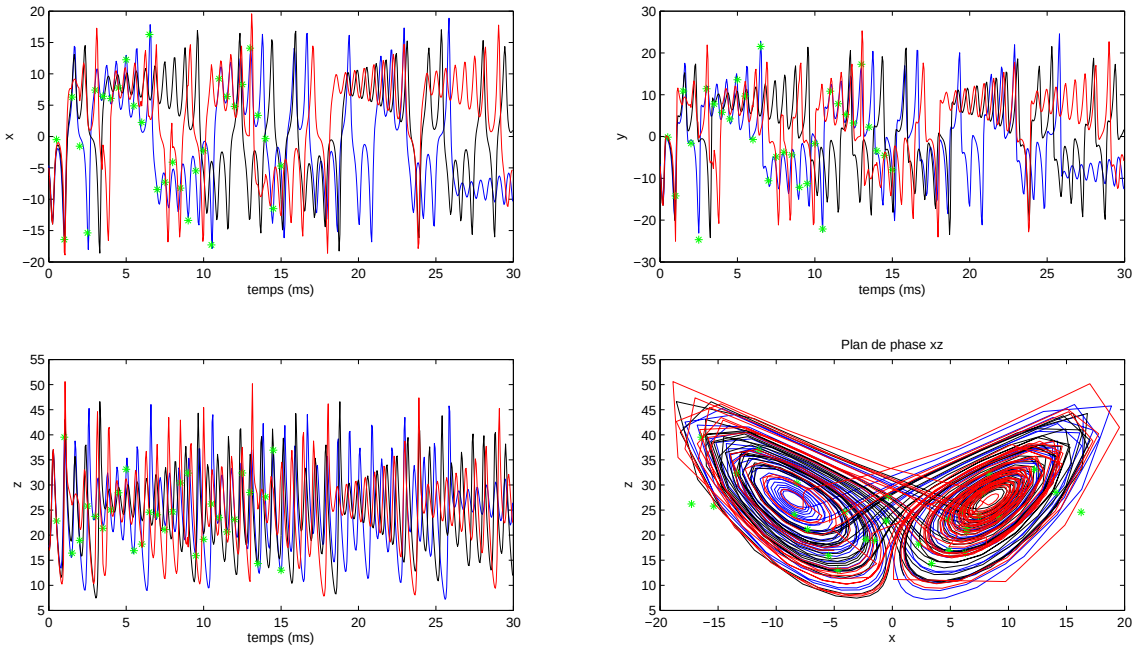

Figure 7: Assimilation de 15 s sur une durée totale de 30 s

Ci-après on observe l'influence de la fréquence d'observations sur le résultat obtenu (question 11). Si on double celle-ci (via le paramètre `ob_step` qu'on passe à 5) alors on voit que l'analyse est meilleure, plus il y a de points de contrôles et plus la trajectoire est contrainte à suivre la courbe bleue. En revanche on constate très nettement dans les deux cas qu'une fois la période d'assimilation terminée la trajectoire de l'analyse repart dans les choux puisqu'elle n'est plus soumise aux observations.

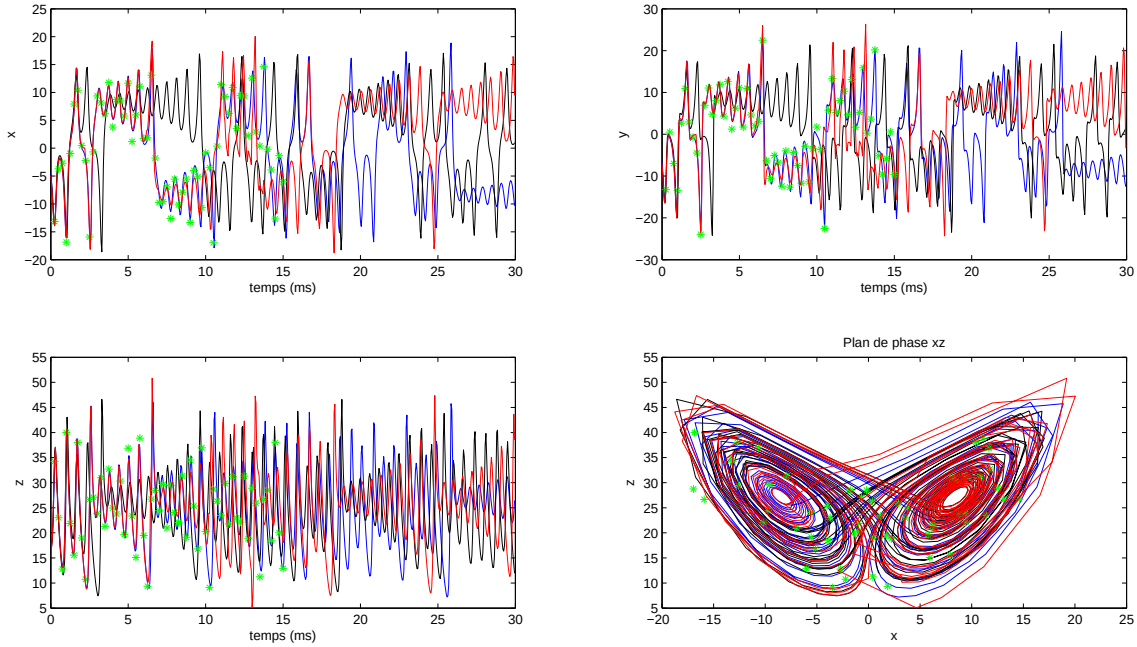

Figure 8: Influence de la fréquence d'observation (doublée)

Encore mieux si on dispose des observations à chaque pas de temps ($ob\_step = 1$) alors l'approximation est très satisfaisante :

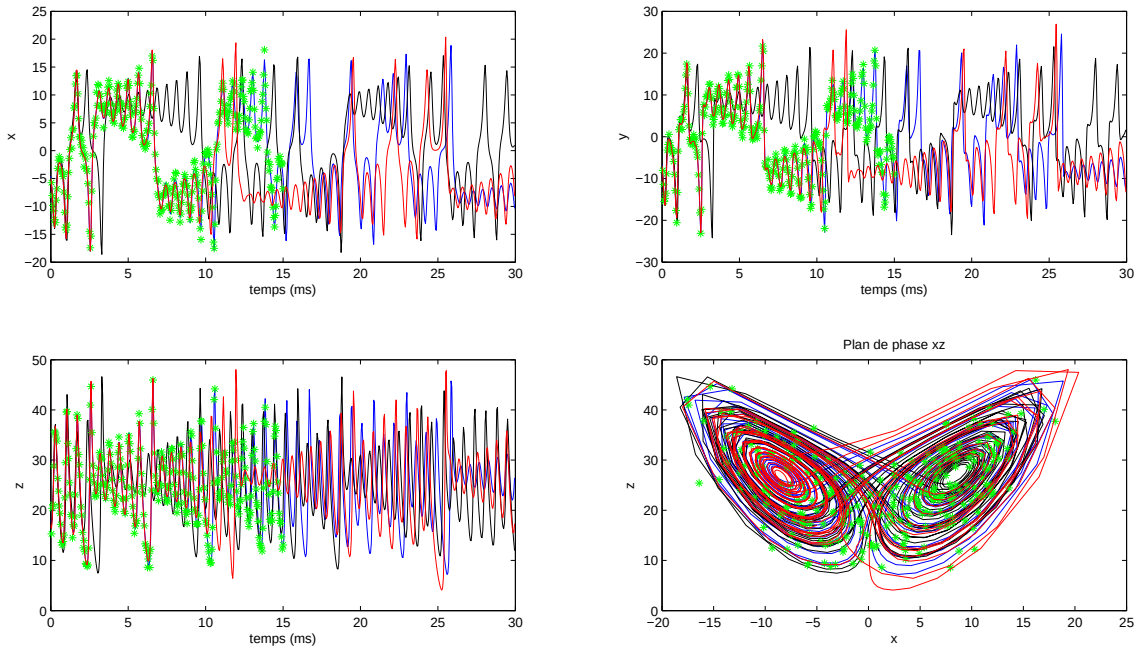

Figure 9: Filtrage de Kalman avec observation à chaque instant

Question 7. On rappelle que le code numérique de résolution des équations de Lorenz figure dans le fichier `euler.m`, et qu'il peut se résumer aux matrices M_i telles que $u_{i+1} = M_i u_i$ qui sont calculées dans le filtrage de Kalman par exemple (variable `M1`). Ainsi le code adjoint se résume à la connaissance des matrices M_i^T qu'on va utiliser pour calculer le gradient (cf. le code de la fonction `cost.m`). En effet le gradient peut se calculer "à rebours" - d'ailleurs dans le modèle adjoint continu cela se manifeste par la condition initiale $P(t = T) = 0$ - via la procédure suivante :

On calcule les vecteurs d'innovations $d_i = u_i - u_i^{obs}$. On part de $\tilde{u}_{fin} = d_{fin}$ puis on calcule les variables adjointes en remontant

$$\tilde{u}_i = d_i + M_i^T \tilde{u}_{i+1}$$

Dans le code j'ai noté $(m_{ij})_{1 \leqslant i,j \leqslant 3}$ les coefficients de la matrice M_i^T, ainsi la variable auxiliaire $\tilde{u}_1 = M_i^t \tilde{u}_{i+1}$ est donnée par :

```
x1_tilde = m11*x_tilde(i+1)+m12*y_tilde(i+1)+m13*z_tilde(i+1);
y1_tilde = m21*x_tilde(i+1)+m22*y_tilde(i+1)+m23*z_tilde(i+1);
z1_tilde = m31*x_tilde(i+1)+m32*y_tilde(i+1)+m33*z_tilde(i+1);
```

et les variables adjointes par :

```
x_tilde(i) = x1_tilde + (x(i)-xob(i))*D(i);
y_tilde(i) = y1_tilde + (y(i)-yob(i))*D(i);
z_tilde(i) = z1_tilde + (z(i)-zob(i))*D(i);
```

Le gradient est alors obtenu par `nablaJ = [x_tilde(1),y_tilde(1),z_tilde(1)];`

La fonction `cost.m` renvoie donc à partir d'une trajectoire et des observations, le résultat de la fonction coût (écart aux observations) et son gradient.

Question 8. Afin de valider le modèle adjoint, et donc de surcroît le gradient dont nous aurons besoin pour l'algorithme de 4d-var, j'ai soumis la fonction `cost.m` aux tests de gradients regroupés dans la fonction `question8.m`.

Test du premier ordre

Consiste à vérifier que les quantités $\tau(\alpha, e_i) = \frac{J(u+\alpha e_i) - J(u)}{\alpha}$ et $\delta(e_i) = < \nabla J, e_i >$ sont proches, i.e vérifier que l'erreur relative $\epsilon(\alpha, e_i) = \frac{|\tau(\alpha, e_i) - \delta(e_i)|}{|\delta(e_i)|}$ tend bien vers 0 quand α tend vers 0 pour différentes directions e_i.

Dans le code j'ai pris comme suggéré la direction $e_i = d = u^b - u_0^{ref}$ et $\alpha = 10^{-k}$ pour $nscale$ valeurs de k. Le vecteur initial u est choisit aléatoirement puis normalisé. Le code étant suffisamment commenté, mieux vaut un dessin plutôt qu'un long discours :

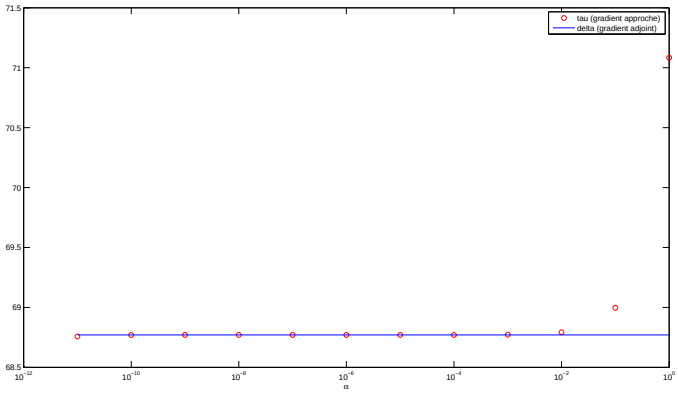

Figure 10: Test du gradient du premier ordre

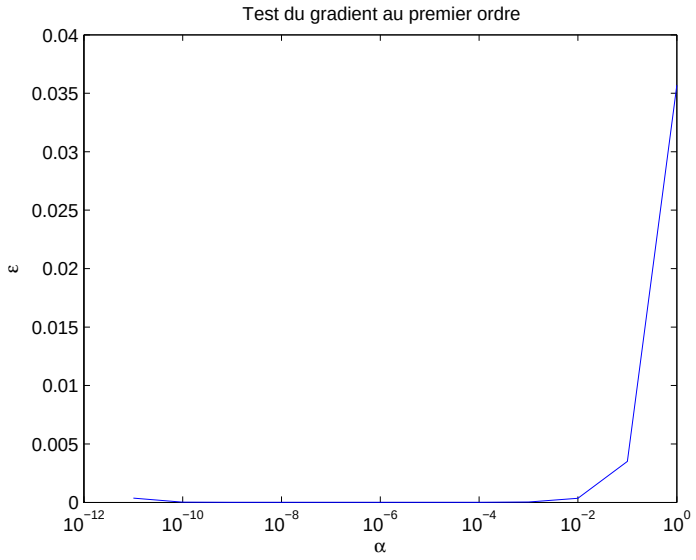

Figure 11: Tracé de l'erreur relative $\epsilon(\alpha, d)$

Test du second ordre

Consiste à vérifier que la quantité $r(\alpha, e_i) = \frac{\tau(\alpha, e_i) - \delta(e_i)}{\alpha}$ tend vers une constante $C(e_i)$ quand $\alpha \to 0$. Malheureusement les erreurs numériques bien que petites font que le test du second ordre échoue. On se contentera donc d'une approximation au premier ordre.

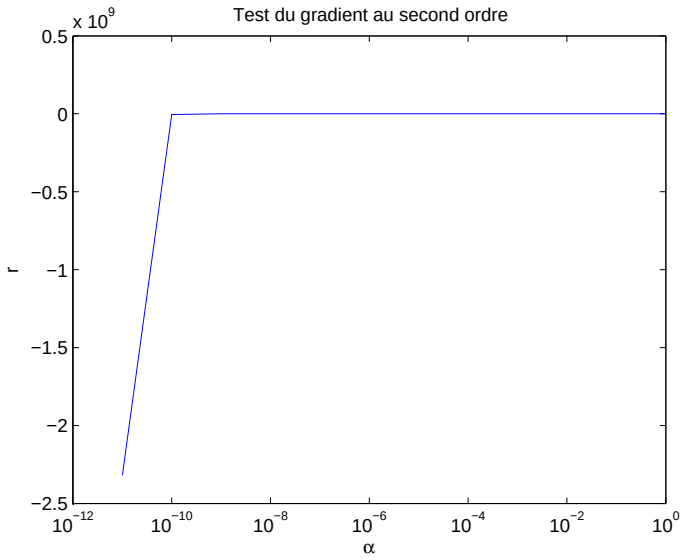

Figure 12: Test du gradient du second ordre

Question 10. Maintenant que l'on s'est assuré que le calcul du gradient était satisfaisant, nous allons pouvoir l'appliquer à l'algorithme de 4d-Var. L'idée on le rappelle est de trouver la condition initiale optimale $u_0^\star$ qui minimise l'écart aux observations, c'est-à-dire la fonction coût $\mathcal{J}(u_0)$. Afin de procéder à cette minimisation j'ai opté pour une descente de gradient :

$$u_0 \leftarrow u_0 - \alpha_k \nabla J(u_0)$$

Le gradient $\nabla J(u_0)$ est donné par la fonction `cost.m` qui requiert la trajectoire ayant pour condition initiale u_0 (donnée par `euler.m`) et des observations. Etant donné que les observations sont générées à partir de la trajectoire ayant pour condition initiale u_{ref} nous devrions donc en théorie trouver $u_0^\star = u_{ref}$ mais dans la pratique nous pouvons faire qu'un nombre fini d'itérations donc nous arrêtons l'algorithme dès que la norme du gradient est inférieure à une certaine tolérance $\|\nabla J(u_0)\| < \epsilon$, ou que le gradient ne bouge plus de manière significative $|\|\nabla J(u_0^k)\| - \|\nabla J(u_0^{k-1})\|| < \epsilon$, ou encore que le nombre maximal d'itération `max_iter` est atteint. Dans le code par défaut j'ai choisi $max\_iter = 30$ et $\epsilon = 10^{-3}$, enfin u_0 est initialisée à $u^b = (-5, -7, 17)$.

En prenant une fréquence d'observation de 2 (une observation un instant sur deux) le résultat obtenu est satisfaisant :

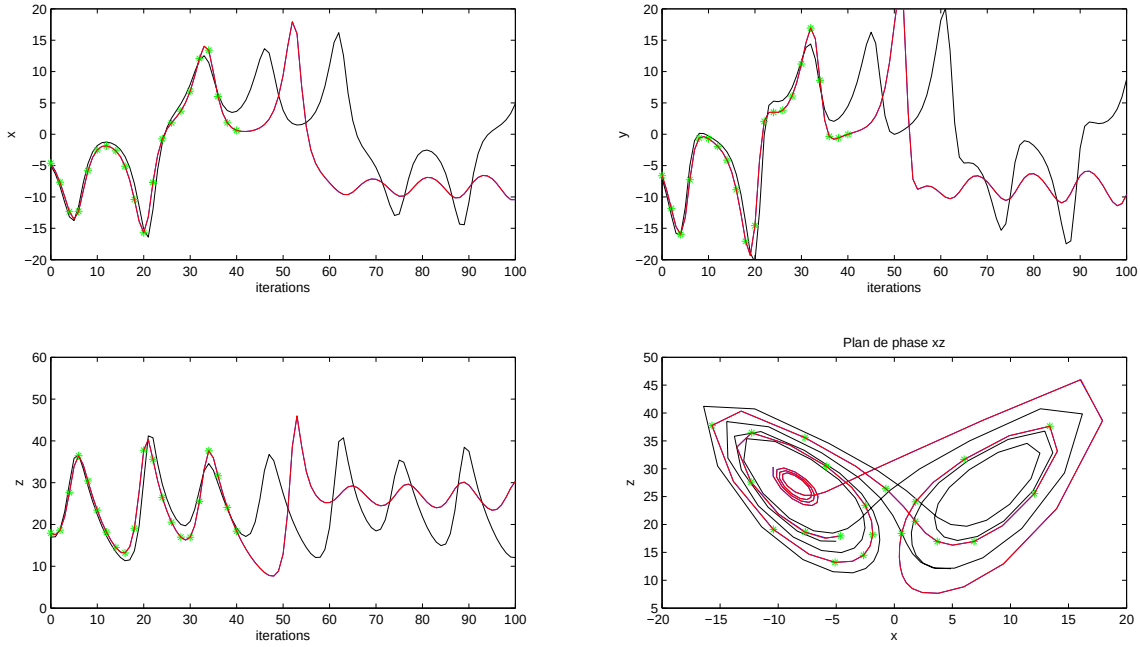

Figure 13: Résultat obtenu pour $ob\_step = 2$

De plus en traçant l'évolution du coût et du gradient on constate bien une décroissance puis une stagnation, le gradient restant inchangé à 10^{-3} près, l'algorithme converge en 22 itérations.

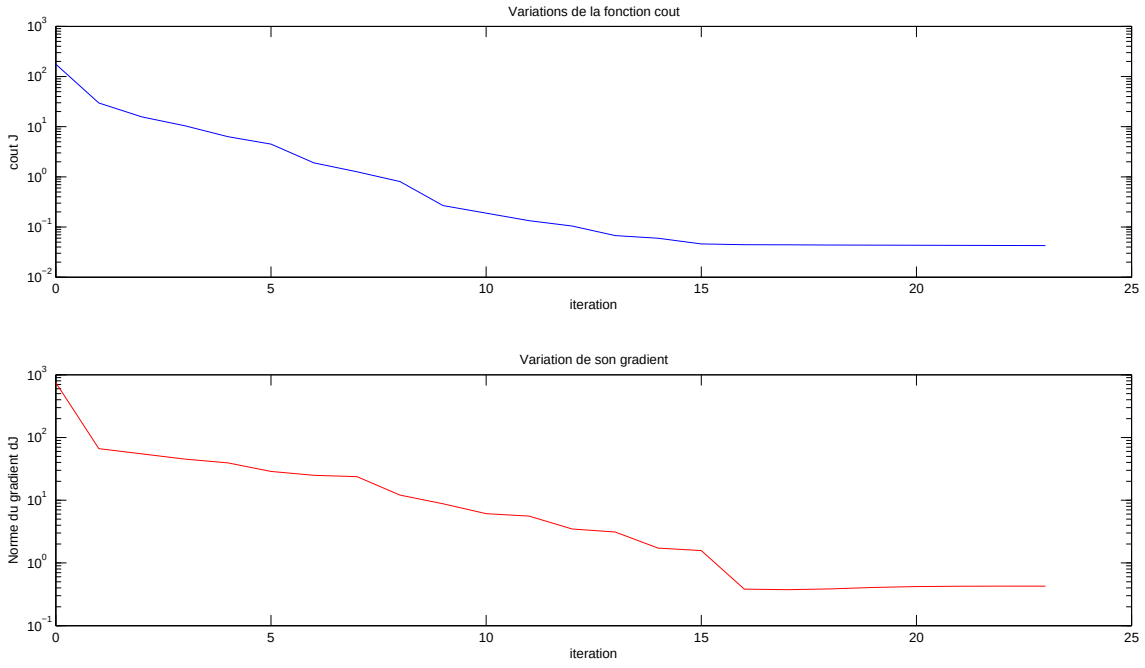

Figure 14: Evolution du coût et du gradient

La condition optimale résultante est $u_0^\star = (-4.83, -6.44, 17.87)$, de coût $\mathcal{J}(u_0^\star) = 4,27.10^{-2}$ et de gradient $\nabla\mathcal{J}(u_0^\star) = 4,26.10^{-1}$ qui donne une bien meilleure approximation que l'ébauche u^b, du

moins sur les 100 premières itérations (5 premières secondes). En effet si on prolonge la simulation (sans nouvelles observations) la courbe analyse finit fatalement par se détacher de la vraie solution, vers la 200 ème itération (10 secondes).

Même avec relativement peu d'observations le résultat de l'approximation est plutôt bon comme on peut le voir ci-dessous, même si l'estimation de la vraie condition initiale est plus éloignée $u_0^\star = (-5.15, -6.90, 17.65)$.

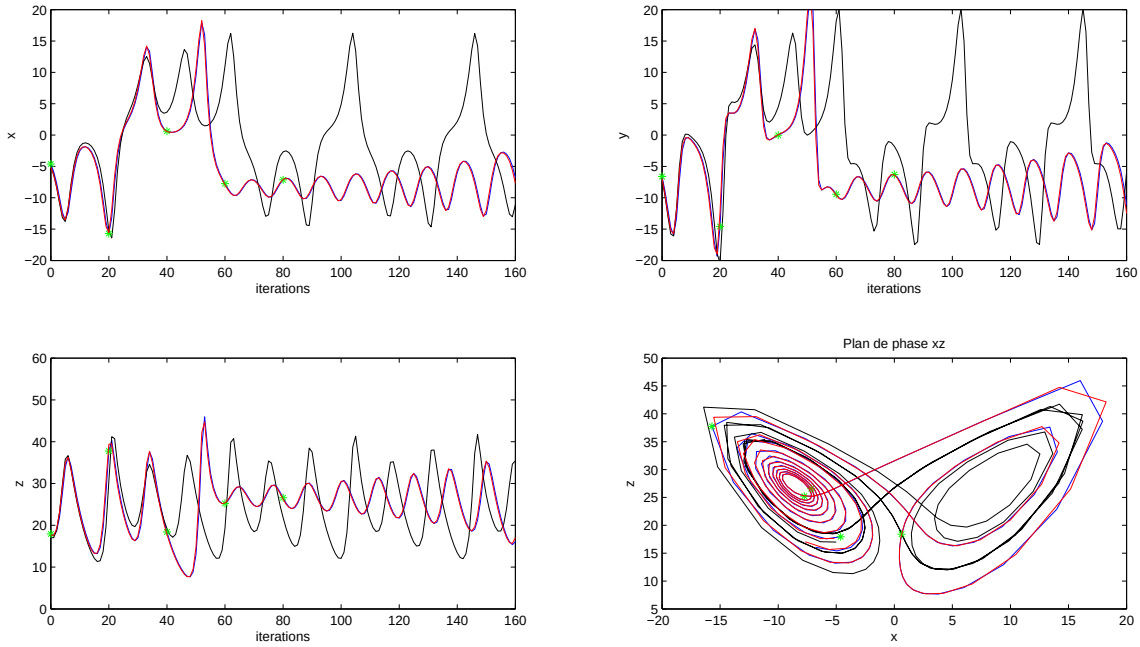

Figure 15: Résultat avec peu d'observations

remarque : Toutes les figures présentes dans le rapport peuvent facilement être reproduites en changeant comme précisé la valeur des paramètres en en-tête de chaque fichier Matlab.

4 Conclusion

L'implémentation du filtrage de Kalman et de l'algorithme de 4d-Var fut quand même assez laborieuse dans la mesure où je n'ai utilisé ni ode45 ni fminunc pour calculer les trajectoires, les gradients et procéder à la minimisation. Par conséquent j'ai été souvent exposé à des problèmes d'ordre numérique, par exemple d'instabilité lorsque j'utilisais initialement un schéma d'Euler simple. Beaucoup de temps passé donc à obtenir un code fonctionnel, mais cela m'a permis d'éclaircir certains points du cours et de me rendre compte une fois de plus qu'entre la théorie et la pratique il y a tout un lot de détails qui ont leur importance et qu'il ne faut pas négliger, même sur un modèle qui reste ici somme toute relativement simple. Je mesure maintenant davantage les difficultés que l'on peut rencontrer dans l'assimilation de données appliquée à la météorologie comprenant un très grand nombre de paramètres à estimer.