Nicolas Daget

Utilisation d'un filtre récursif 1D pour corréler les perturbations de la SST

29 mars 2006

Technical Report
TR/CMGC/06/17
GLOBC - CERFACS

Table des matières

1 Introduction **3**

2 Description théorique du filtre récursif **5**
- 2.1 Les équations de Langevin . . . 5
  - 2.1.1 Présentation . . . 5
  - 2.1.2 Discrétisation . . . 6
- 2.2 modèle auto-regressif du second ordre . . . 7
- 2.3 Les conditions limites . . . 9
  - 2.3.1 Conditions limites pour $i = 0$ . . . 10
  - 2.3.2 Conditions limites pour $i = n + 1$ . . . 11
  - 2.3.3 Résumé des trois premières conditions limites . . . 12

3 Implémentation informatique **13**
- 3.1 Implémentation du filtre récursif . . . 13
- 3.2 Utilisation du filtre récursif . . . 14
  - 3.2.1 Théorie . . . 14
  - 3.2.2 Mise en œuvre . . . 14

4 Présentation des résultats **16**

5 Conclusion **19**

A Inteface du module du filtre récursif **21**

B Code source du filtre récursif 1D **22**

C Code source de l'adjoint du filtre récursif 1D **24**

D Code source du test de l'adjoint du filtre récursif 1D **27**

Table des figures

4.1 Hoevmoller de l'année 1993 représentant les perturbations de SST pour une latitude de 56° sud. 16
4.2 Perturbation de la SST pour le 1er janvier 1993 sans corrélation 17
4.3 Perturbation de la SST pour le 1er janvier 1993 avec une corrélation de 5 jours 17
4.4 Perturbation de la SST pour le 1er janvier 1993 avec une corrélation de 7 jours 18
4.5 Perturbation de la SST pour le 1er janvier 1993 avec une corrélation de 10 jours 18

Chapitre 1

Introduction

L'utilisation de méthode d'ensembles nécessite de perturber les différents membres. Dans notre cas, chaque membre est une expérience d'assimilation de données variationnelle de l'océan. Le système utilisé est OPAVAR : une méthode d'assimilation de données variationnelle (3DVar et 4DVar) développée au CERFACS utilisant le modèle d'océan OPA[1] développé au LOCEAN[2].

Pour perturber les différents membres, les forçages en vents et la température de surface utilisés par le modèle OPA sont perturbés, ainsi que les observations utilisées directement par la méthode d'assimilation.

Les perturbations sur les observations sont effectuées pendant l'exécution du système, alors que les perturbations sur les forçages sont fabriquées séparément.

Les méthodes permettant de fabriquer des perturbations sur les forçages consistent globalement à comparer deux champs aux mêmes dates. Les différences obtenues reflètent alors les incertitudes liées à ces champs. Les méthodes utilisées sont celles développées au CEPMMT[3] pour le projet européen ENSEMBLES. Ces méthodes sont décrites par Antje Weisheimer (2005) [11].

Les forçages en vents obtenus reflètent une série temporelle et sont donc corrélés temporellement entre eux. Par contre, la méthode utilisée pour construire les perturbations de température de surface fabrique des champs temporellement indépendants. Il s'agit donc de corréler temporellement les champs de perturbations de température de surface.

Différentes méthodes permettent d'atteindre cet objectif comme par exemple une moyenne glissante. Cependant, il nous apparait plus intéressant d'appliquer une fonction de corrélation précise dont l'échelle de corrélation puisse être fixée.

Pour appliquer une fonction de corrélation, le filtre récursif est une méthode à la fois rapide et efficace. Dans cette étude, un filtre récursif sera utilisé pour simuler un modèle auto-regressif du second ordre (SOAR).

Dans la première partie, l'aspect théorique sera présenté. Ensuite, l'implémentation informatique sera décrite. Finalement, des résultats seront présentés.

Toute l'étude théorique reposent sur les papiers de Gneiting (1999) [3], Lorenc (1992) [5], Purser (1982) [7] (2003) [8], Weaver (2001) [10], Bennet (1996) [1], Hayden (1995) [4], Ngodock (2005) [6], et

[1]Océan PArallèlisé
[2]anciennement LODyC
[3]Centre Européen pour les Prévisions Météorologiques à Moyen Terme

Gao (2004) [2].

Chapitre 2

Description théorique du filtre récursif

Dans ce chapitre, l'origine du filtre récursif sera expliqué, ainsi que son application à un modèle auto-regressif du second ordre.

2.1 Les équations de Langevin

2.1.1 Présentation

Bennett et al. (1996) [1] ont montré que pour une corrélation temporelle de Markov ($e^{-\frac{|t-t'|}{\tau}}$), le produit de convolution peut-être obtenu par la résolution de deux équations de Langevin.

Considérons l'équation suivante :

$$\varphi(t) \quad = \quad \int_0^T e^{-\frac{|t-t'|}{\tau}} \lambda(t') dt' \tag{2.1}$$

Où $\lambda(T) = 0$ et où les dépendances spatiales de φ et λ sont omises pour des soucis de clarté.

La solution proposée par Bennet et al. [1] est de remarquer que φ est la solution de :

$$\left\{ \begin{array}{ll} \frac{\partial^2 \varphi}{\partial t^2} - \tau^{-2}\varphi = -2\tau^{-1}\lambda & 0 \leq t \leq T \\ \frac{\partial \varphi}{\partial t} - \tau^{-1}\varphi = 0 & \text{pour } t = 0 \\ \frac{\partial \varphi}{\partial t} + \tau^{-1}\varphi = 0 & \text{pour } t = T \end{array} \right. \tag{2.2}$$

La démonstration peut-être retrouvée dans "Inverse problem theory" de Tarantola [9] au chapitre 7 pages 572-576.

Cette équation (2.2), comme le montre Ngodock (2005) [6], peut-être décomposée en deux équations à résoudre l'une après l'autre. D'abord celle-ci :

$$\left\{ \begin{array}{ll} \frac{\partial \alpha}{\partial t} + \tau^{-1}\alpha = -2\tau^{-1}\lambda & 0 \leq t \leq T \\ \alpha(0) = 0 & \end{array} \right. \tag{2.3}$$

Ensuite, en utilisant la solution α, celle-là :

$$\left\{ \begin{array}{ll} \frac{\partial \varphi}{\partial t} - \tau^{-1}\varphi = \alpha & 0 \leq t \leq T \\ \varphi(T) = -\frac{\tau}{2}\alpha(T) & \end{array} \right. \tag{2.4}$$

En effet, en dérivant, l'équation 2.4, on obtient :

$$\frac{\partial^2\varphi}{\partial t^2} - \tau^{-1}\frac{\partial\varphi}{\partial t} = \frac{\partial\alpha}{\partial t} \tag{2.5}$$

Et en substituant avec les équations 2.3 et 2.4 :

$$\frac{\partial^2\varphi}{\partial t^2} - \tau^{-1}(\alpha + \tau^{-1}\varphi) = -\tau^{-1}\alpha - 2\tau^{-1}\lambda \tag{2.6}$$

En simplifiant, on retrouve l'équation 2.2 :

$$\frac{\partial^2\varphi}{\partial t^2} - \tau^{-2}\varphi = -2\tau^{-1}\lambda \tag{2.7}$$

Les conditions limites de 2.2 peuvent être obtenues en réinjectant les conditions limites de 2.3 et 2.4 dans 2.7.

2.1.2 Discrétisation

Pour résoudre les équations de Langevin (2.3 et 2.4) rappelées ci-dessous, il faut les discrétiser.

$$\begin{cases} \frac{\partial\alpha}{\partial t} + \tau^{-1}\alpha = -2\tau^{-1}\lambda \\ \frac{\partial\varphi}{\partial t} - \tau^{-1}\varphi = \alpha \end{cases} \tag{2.8}$$

En utilisant un schéma d'Euler, on obtient :

$$\begin{cases} \frac{\alpha_i - \alpha_{i-1}}{\Delta t} - \tau^{-1}\alpha_{i-1} = 2\tau^{-1}\lambda_i \\ \frac{\varphi_{i+1} - \varphi_i}{\Delta t} - \tau^{-1}\varphi_{i+1} = \alpha_i \end{cases} \tag{2.9}$$

Soit :

$$\begin{cases} \alpha_i = (1 - \Delta t\ \tau^{-1})\alpha_{i-1} - 2\Delta t\ \tau^{-1}\lambda_i \\ \varphi_i = (1 - \Delta t\ \tau^{-1})\varphi_{i+1} - \Delta t\ \alpha_i \end{cases} \tag{2.10}$$

En faisant les changements de variables $\tilde{\varphi} = \frac{\varphi}{2\tau}$ et $\tilde{\alpha} = -\frac{\alpha}{2}$, le système 2.10 devient :

$$\begin{cases} -2\tilde{\alpha}_i = -2(1 - \Delta t\ \tau^{-1})\tilde{\alpha}_{i-1} - 2\Delta t\ \tau^{-1}\lambda_i \\ 2\tilde{\varphi}_i = 2(1 - \Delta t\ \tau^{-1})\tilde{\varphi}_{i+1} + 2\Delta t\ \tau^{-1}\tilde{\alpha}_i \end{cases} \tag{2.11}$$

Soit :

$$\begin{cases} \tilde{\alpha}_i = (1 - \Delta t\ \tau^{-1})\tilde{\alpha}_{i-1} + \Delta t\ \tau^{-1}\lambda_i \\ \tilde{\varphi}_i = (1 - \Delta t\ \tau^{-1})\tilde{\varphi}_{i+1} + \Delta t\ \tau^{-1}\tilde{\alpha}_i \end{cases} \tag{2.12}$$

On retrouve alors la forme classique du filtre récursif et les conditions limites deviennent $\tilde{\alpha}_0 = 0$ et $\tilde{\varphi}_{n+1} = \frac{1}{2}\tilde{\alpha}_{n+1}$.

2.2 modèle auto-regressif du second ordre

À partir de maintenant, nous reprendrons la notation de Purser et McQuigg (1982) [7] , Hayden et Purser (1995) [4], Purser et al (2002) [8] et Lorenc (1992) [5] :

$$\left\{ \begin{array}{ll} q_i = (1-\alpha)p_i + \alpha q_{i-1} & \text{pour } i = 0, \cdots, n \\ s_i = (1-\alpha)q_i + \alpha s_{i+1} & \text{pour } i = n, \cdots, 1 \end{array} \right. \tag{2.13}$$

Où p_i est la valeur initiale au point de grille i, q_i est la valeur après le filtrage de $i = 1$ à n, s_i est la valeur après le passage du filtre dans chaque direction et α est le coefficient de filtrage.

Ceci est donc un filtre recursif du premier ordre qui doit être appliqué dans les deux directions pour assurer que la phase reste nulle.

Des filtres d'ordre supérieur peuvent être fabriqués en appliquant plusieurs fois le filtre de premier ordre (2.13). Dans ce cas, les conditions limites changent après chaque application du filtre. L'étude des conditions limites sera effectuée dans la section 2.3. Cependant, au premier ordre, et comme étudié dans la section 2.1, les conditions limites en $i = 1$ et $i = n$, sont :

$$\left\{ \begin{array}{l} q_1 = (1-\alpha)p_1 \\ s_n = \frac{1}{1+\alpha} q_n \end{array} \right. \tag{2.14}$$

À la différence de la section 2.1, les conditions limites sont données à l'intérieur du domaine et non à l'extérieur.

Pour analyser plus facilement le filtre, il est intéressant de considérer son inverse, qui n'est pas récursif. Après réarrangement de (2.13), on trouve :

$$p_i = s_i - \frac{\alpha}{(1-\alpha)^2}(s_{i-1} - 2s_i + s_{i+1}) \tag{2.15}$$

Si on calcule le réponse spectrale du filtre récursif, il faut définir s tel que $s = \sum s_i \delta_{t_i}$ avec δ_{t_i} le dirac au temps t_i, alors :

$$\begin{aligned} \mathfrak{F}(p_i) &= \int p_i \delta_{t_i} e^{-ikt} dt \\ &= \int s_i \delta_{t_i} e^{-ikt} dt \\ &- \frac{\alpha}{(1-\alpha)^2} \left(\int s_{i-1} \delta_{t_{i-1}} e^{-ikt} dt - 2 \int s_i \delta_{t_i} e^{-ikt} dt + \int s_{i+1} \delta_{t_{i+1}} e^{-ikt} dt \right) \end{aligned} \tag{2.16}$$

En définissant s_i comme une onde telle que $s_i = e^{ikt_i}$, on trouve alors :

$$\begin{aligned} \mathfrak{F}(p_i) &= \int e^{ikt_i} \delta_{t_i} e^{-ikt} dt \\ &- \frac{\alpha}{(1-\alpha)^2} \left(\int e^{ikt_{i-1}} \delta_{t_{i-1}} e^{-ikt} dt - 2 \int e^{ikt_i} \delta_{t_i} e^{-ikt} dt + \int e^{ikt_{i+1}} \delta_{t_{i+1}} e^{-ikt} dt \right) \end{aligned}$$

$$
\begin{aligned}
&= \int \delta_{t_i} dt - \frac{\alpha}{(1-\alpha)^2}\left(\int e^{ik(t_i-\delta t)}\delta_{t_i} e^{-ikt} dt - 2\int \delta_{t_i} dt + \int e^{ik(t_i+\delta t)}\delta_{t_i} e^{-ikt} dt\right) \\
&= \int \delta_{t_i} dt - \frac{\alpha}{(1-\alpha)^2}\left(\int e^{ik\delta t}\delta_{t_i} dt - 2\int \delta_{t_i} dt + \int e^{ik\delta t}\delta_{t_i} dt\right) \\
&= 1 - \frac{\alpha}{(1-\alpha)^2}\left(e^{ik\delta t} - 2 + e^{ik\delta t}\right) \\
&= 1 - \frac{\alpha}{(1-\alpha)^2}\left(2\cos(k\delta t) - 2\right) \\
&= 1 - \frac{2\alpha}{(1-\alpha)^2}\left(\left(1 - 2\sin^2(\frac{k\delta t}{2})\right) - 1\right) \\
&= 1 + \frac{\alpha}{(1-\alpha)^2}\left(2\sin(\frac{k\delta t}{2})\right)^2
\end{aligned}
\tag{2.17}
$$

La réponse spectrale de l'inverse du filtre est alors :

$$
S_{1pass}(k) \quad = \quad \frac{1}{1 + \frac{\alpha}{(1-\alpha)^2}\left(2\sin(\frac{k\delta t}{2})\right)^2} \tag{2.18}
$$

Et pour un filtre d'ordre N, ce facteur doit être appliqué N fois. Si $k\delta t$ et α sont petits, alors en appliquant les développement de Taylor à l'ordre un, on trouve :

$$
S_{Npass}(k) \quad \simeq \quad 1 - \frac{\alpha}{(1-\alpha)^2} N k^2 \delta t^2 \tag{2.19}
$$

Ce filtre doit alors être comparé à l'effet de la convolution d'un champs par une fonction de covariance. Dans notre cas, nous voulons utiliser une fonction auto-regressive du second ordre (SOAR) car elle est plus énergétique que la classique fonction de covariance de Gauss.

$$
b(t) \quad = \quad \varepsilon_b^2\left(1 + \frac{|t|}{\tau}\right)e^{-\frac{|t|}{\tau}} \tag{2.20}
$$

Il faut alors calculer la transformée de Fourier de la fonction SOAR :

$$
\begin{aligned}
S_{SOAR}(k) &= \mathfrak{F}(b(t)) = \varepsilon^2 \int_{-\infty}^{+\infty} b(x) e^{-ikt} dt \\
&= \varepsilon^2 \int_{-\infty}^{+\infty} \left(e^{-\frac{|t|}{\tau} - ikt} + \frac{|t|}{\tau} e^{-\frac{|t|}{\tau} - ikt}\right) dt \\
&= \varepsilon^2 \left[\int_{-\infty}^{0} e^{\frac{t}{\tau} - ikt} dt + \int_{0}^{+\infty} e^{-\frac{t}{\tau} - ikt} dt + \int_{-\infty}^{0} \frac{-t}{\tau} e^{\frac{t}{\tau} - ikt} dt + \int_{0}^{+\infty} \frac{t}{\tau} e^{-\frac{t}{\tau} - ikt} dt\right] \\
&= \varepsilon^2 \left[\int_{0}^{+\infty} \left(e^{-\frac{t}{\tau} - ikt} + e^{-\frac{t}{\tau} + ikt}\right) dt + \int_{0}^{+\infty} \left(\frac{t}{\tau} e^{-\frac{t}{\tau} - ikt} + \frac{t}{\tau} e^{-\frac{t}{\tau} + ikt}\right) dt\right]
\end{aligned}
$$

$$= \quad \varepsilon^2 \left[\int_0^{+\infty} \left(e^{-t(\frac{1}{\tau}+ik)} + e^{-t(\frac{1}{\tau}-ik)} \right) dt + \frac{1}{\tau} \int_0^{+\infty} \left(te^{-t(\frac{1}{\tau}+ik)} + te^{-t(\frac{1}{\tau}-ik)} \right) dt \right] \quad (2.21)$$

On reconnait les intégrales de la forme $\int_0^{+\infty} x^n e^{-ax} dx = \frac{\Gamma(n+1)}{a^{n+1}}$. L'équation 2.21 peut alors se résoudre :

$$\begin{aligned} S_{SOAR}(k) &= \varepsilon^2 \left[\frac{\tau}{1+i\tau k} + \frac{\tau}{1-i\tau k} + \frac{\tau}{(1+i\tau k)^2} + \frac{\tau}{(1-i\tau k)^2} \right] \\ &= \tau\varepsilon^2 \left[\frac{2}{1+\tau^2 k^2} + \frac{2-2\tau^2 k^2}{(1+\tau^2 k^2)^2} \right] \\ &= \varepsilon_b^2 \frac{4\tau}{(1+k^2\tau^2)^2} \end{aligned} \quad (2.22)$$

En appliquant les développement de Taylor à l'ordre un, on obtient alors :

$$S_{SOAR}(k) \quad \simeq \quad \varepsilon_b^2 4\tau(1 - 2k^2\tau^2) \quad (2.23)$$

En supposant que $k\delta t$ est petit, il est possible d'obtenir une fonction SOAR avec le filtre récursif, car (2.23) est similaire à (2.19) pour $N = 2$.

Il est alors possible de définir un facteur d'échelle β tel que :

$$\beta \quad = \quad \varepsilon_b^2 4\tau \quad (2.24)$$

Il faut alors définir α tel que :

$$\frac{\alpha}{(1-\alpha)^2} N k^2 \delta t^2 \quad = \quad 2k^2\tau^2 \quad (2.25)$$

En posant $E = \frac{N\delta t^2}{4\tau}$, on obtient le polynôme du deuxième ordre :

$$\alpha^2 - 2(1+E)\alpha + 1 \quad = \quad 0 \quad (2.26)$$

En le résolvant, on trouve α :

$$\alpha \quad = \quad 1 + E - \sqrt{E(E+2)} \quad (2.27)$$

Si on calcule la l'échelle de corrélation du filtre récursif en appliquant la formule $L_b = \left(\frac{\partial^2 b}{\partial t^2}\Big|_{t=0} \right)^{-\frac{1}{2}}$, on trouve que $L_b = \tau$.

Comme N est fixé à 2 et que δt est fixé par la grille, il apparaît que α ne dépend que de l'échelle de corrélation τ.

2.3 Les conditions limites

Comme il faut appliquer deux fois le filtre récursif pour obtenir une fonction auto-regressive du second ordre, il est nécessaire de calculer les conditions limites pour chaque application du filtre. En se référant aux travaux de Purser et al (2002) [8] et de Hayden et Purser (1995) [4], les conditions limites peuvent être calculées de la manière suivante.

En écrivant le filtre récursif (2.13) sous forme matricielle, on a :

$$\left\{ \begin{array}{l} Aq = p \\ Bs = q \end{array} \right. \tag{2.28}$$

avec

$$A = \begin{pmatrix} \frac{1}{1-\alpha} & 0 & \cdots \\ \frac{-\alpha}{1-\alpha} & \ddots & 0 \\ 0 & \ddots & \frac{1}{1-\alpha} \\ \vdots & 0 & \frac{-\alpha}{1-\alpha} \end{pmatrix} \qquad B = \begin{pmatrix} \frac{1}{1-\alpha} & \frac{-\alpha}{1-\alpha} & 0 & \cdots \\ 0 & \ddots & \ddots & \cdots \\ \vdots & 0 & \ddots & \frac{-\alpha}{1-\alpha} \\ \vdots & \vdots & 0 & \frac{1}{1-\alpha} \end{pmatrix}$$

Ce système peut donc être résolu en inversant les matrices :

$$\left\{ \begin{array}{l} q = A^{-1}p \\ s = B^{-1}q \end{array} \right. \tag{2.29}$$

Ainsi :

$$s = B^{-1}A^{-1}p \tag{2.30}$$

Sur une grille infinie, s est symétrique et on a :

$$s = B^{-1}A^{-1}p \equiv A^{-1}B^{-1}p \tag{2.31}$$

Si on travaille sur un domaine fini tel que $i \in [1, n]$, alors les éléments à l'extérieur de cet intervalle sont nuls à l'origine et ne sont modifiés que par B pour $i < 1$ et par A pour $i > n$.

Les conditions limites sont donc définie par :

$$B^n q_0^n = p_0^0 = 0 \tag{2.32}$$

$$A^n s_{n+1}^n = p_{n+1}^0 = 0 \tag{2.33}$$

Où l'indice sur p, q et s indique le nombre d'applications du filtre.

Il faut ensuite compléter par le nombre d'équations nécessaires pour fermer le système.

2.3.1 Conditions limites pour $i = 0$

Quand le filtre B n'a jamais été appliqué, on a le système suivant :

$$\left\{ \begin{array}{l} B^0 q_0^0 = q_0^0 = 0 \\ q_1^0 = (1-\alpha) p_1^0 + \alpha q_0^0 \end{array} \right. \tag{2.34}$$

Soit :

$$q_1^0 = (1-\alpha) p_1^0 \tag{2.35}$$

Quand le filtre B a été appliqué une seule fois, on a le système suivant :

$$\left\{ \begin{array}{l} B^1 q_0^1 = q_0^1 - \alpha q_1^1 = 0 \\ q_1^1 = (1-\alpha) p_1^1 + \alpha q_0^1 \end{array} \right. \tag{2.36}$$

Soit :

$$\begin{aligned} q_1^1 &= \frac{1-\alpha}{1-\alpha^2} p_1^1 \\ q_1^1 &= \frac{1}{1+\alpha} p_1^1 \end{aligned} \tag{2.37}$$

Quand le filtre B a été appliqué une deuxième fois, on a le système suivant :

$$\left\{ \begin{array}{l} B^2 q_0^2 = q_0^2 - 2\alpha q_1^2 + \alpha^2 q_0^2 = 0 \\ q_1^2 = (1-\alpha) p_1^2 + \alpha q_0^2 \\ q_2^2 = (1-\alpha) p_2^2 + \alpha q_1^2 \end{array} \right. \tag{2.38}$$

Soit :

$$q_1^2 = \frac{1-\alpha}{(1+\alpha^2)^2} (p_1^2 - \alpha^3 p_2^2) \tag{2.39}$$

Et ainsi de suite pour chaque application du filtre B.

2.3.2 Conditions limites pour $i = n + 1$

Quand le filtre A a été appliqué une seule fois, on a le système suivant :

$$\left\{ \begin{array}{l} A^1 s_{n+1}^1 = s_{n+1}^1 - \alpha s_{n+1}^1 = 0 \\ s_n^1 = (1-\alpha) q_n^1 + \alpha q_{n+1}^1 \end{array} \right. \tag{2.40}$$

Soit :

$$\begin{aligned} s_n^1 &= \frac{1-\alpha}{1-\alpha^2} q_n^1 \\ s_n^1 &= \frac{1}{1+\alpha} q_n^1 \end{aligned} \tag{2.41}$$

Quand le filtre A a été appliqué une deuxième fois, on a le système suivant :

$$\left\{\begin{array}{l} A^2 s^2_{n+1} = \alpha^2 s^2_{n-1} - 2\alpha s^2_n + s^2_{n+1} = 0 \\ s^2_n = (1-\alpha) q^2_n + \alpha s^2_{n+1} \\ s^2_{n-1} = (1-\alpha) q^2_{n-1} + \alpha s^2_n \end{array}\right. \tag{2.42}$$

Soit :

$$s^2_n = \frac{1-\alpha}{(1+\alpha^2)^2}\big(q^2_n - \alpha^3 q^2_{n-1}\big) \tag{2.43}$$

Quand le filtre A a été appliqué une troisième fois, on a le système suivant :

$$\left\{\begin{array}{l} A^3 s^3_{n+1} = -\alpha^3 s^3_{n-2} + 3\alpha^2 s^3_{n-1} - 3\alpha s^3_n + s^3_{n+1} = 0 \\ s^3_n = (1-\alpha) q^3_n + \alpha s^3_{n+1} \\ s^3_{n-1} = (1-\alpha) q^3_{n-1} + \alpha s^3_n \\ s^3_{n-2} = (1-\alpha) q^3_{n-2} + \alpha s^3_{n-1} \end{array}\right. \tag{2.44}$$

Soit :

$$s^2_n = \frac{1-\alpha}{(1+\alpha^2)^3}\big(-q^3_n + (3-\alpha^2)\alpha^3 q^3_{n-1} - \alpha^4 q^3_{n-2}\big) \tag{2.45}$$

Et ainsi de suite pour chaque application du filtre A.

2.3.3 Résumé des trois premières conditions limites

On peut donc calculer les conditions limites pour N applications du filtre récursif. Cependant, dans la pratique les conditions limites changent pour les trois premières applications du filtre. Ensuite, les conditions limites appliquées restent inchangées.

$$\begin{array}{l} \left\{\begin{array}{l} q^0_1 = (1-\alpha) p^0_1 \\ s^1_n = \frac{1}{1+\alpha} q^1_n \end{array}\right. \\ \left\{\begin{array}{l} q^1_1 = \frac{1}{1+\alpha} p^1_1 \\ s^2_n = \frac{1-\alpha}{(1-\alpha^2)^2}\big(q^2_n - \alpha^3 q^2_{n-1}\big) \end{array}\right. \\ \left\{\begin{array}{l} q^2_1 = \frac{1-\alpha}{(1-\alpha^2)^2}\big(p^2_1 - \alpha^3 p^2_2\big) \\ s^3_n = \frac{1-\alpha}{(1-\alpha^2)^3}\big(-q^3_n + (3-\alpha^2)\alpha^3 q^3_{n-1} - \alpha^4 q^3_{n-2}\big) \end{array}\right. \\ \left\{\begin{array}{l} q^k_1 = \frac{1-\alpha}{(1-\alpha^2)^2}\big(p^k_1 - \alpha^3 p^k_2\big) \text{ pour } k > 2 \\ s^k_n = \frac{1-\alpha}{(1-\alpha^2)^3}\big(-q^k_n + (3-\alpha^2)\alpha^3 q^k_{n-1} - \alpha^4 q^k_{n-2}\big) \text{ pour } k > 3 \end{array}\right. \end{array} \tag{2.46}$$

Chapitre 3

Implémentation informatique

3.1 Implémentation du filtre récursif

Pour utiliser le filtre récursif, un module fortran a été développé. (voir les annexes A, B, C et D). Il permet d'appliquer le filtre à des tableaux de une, deux ou trois dimensions contenant des réels en simple ou double précision. Quelque soit la dimension du tableau, le filtre n'est appliqué que selon sa dernière dimension. Ce n'est donc toujours qu'un filtre récursif monodimensionnel. Dans tout les cas, pour l'utiliser, il suffit de faire :

```
...
USE filters
...
CALL recursive_filter(A,alpha,n)
...
```

Où A est le tableau avec les valeurs non filtrées en entrée et avec les valeurs filtrées en sortie, alpha la valeur du paramètre α et n le nombre d'application du filtre.

Par exemple, pour faire un filtre SOAR, il faut faire :

```
...
USE filters
...
E = 2. * dt**2 / ( 4 * z_tau**2)
alpha = 1 + E - sqrt( E * ( E + 2 ) )
CALL recursive_filter(A,alpha,1)
CALL recursive_filter(A,alpha,2)
...
```

Pour l'implémentation du filtre récursif, seules les trois premières conditions limites ont été codées. Pour des filtres d'ordre supérieur à trois, les conditions limites appliquées sont celles du filtre récursif d'ordre trois (cf. 2.3).

3.2 Utilisation du filtre récursif

3.2.1 Théorie

Dans notre application, les perturbations de la SST doivent être corrélées et la fonction de corrélation désirée est une fonction auto-régréssive du second ordre (SOAR). Il ne faut donc appliquer le filtre qu'une seule fois. En effet, comme le montre Weaver et Courtier (2001) [10], la matrice d'auto-corrélation C peut être écrite :

$$\begin{aligned} C &= \Lambda R^{1/2} W^{-1} R^{T/2} \Lambda \\ &= \left(\Lambda R^{1/2} W^{-1/2}\right)\left(\Lambda R^{1/2} W^{-1/2}\right)^T \\ &= C^{1/2} C^{T/2} \end{aligned} \tag{3.1}$$

Avec Λ la matrice de normalisation, R l'opérateur de corrélation et W la matrice des facteurs d'échelle.

Si on définit une perturbation gaussienne δv ($E[\delta v] = 0$ et $E[\delta v \delta v^T] = I$), alors en la filtrant, on obtient δx tel que :

$$\begin{aligned} \delta x &= \Lambda R^{1/2} W^{-1/2} \delta v \\ &= C^{1/2} \delta v \end{aligned} \tag{3.2}$$

Et si on calcule l'auto-corrélation de δx, on trouve :

$$\begin{aligned} E[\delta x \delta x^T] &= C^{1/2} E[\delta v \delta v^T] C^{T/2} \\ &= C^{1/2} C^{T/2} \\ &= C \end{aligned} \tag{3.3}$$

On voit alors qu'il suffit d'appliquer la racine de la fonction de corrélation pour obtenir l'auto-corrélation recherchée de la variable filtrée.

3.2.2 Mise en œuvre

Le programme permettant de filtrer les perturbations de la SST effectue donc les opérations suivantes :

- lecture des paramètres, avec entres autres l'année initiale, le nombre d'enregistrements temporels, la valeur de τ, le nombre de variables à traiter et le nombre de fichiers annuels ;
- Lecture du mask et des facteurs d'échelle ;
- Pour chaque fichier, lecture des données ;
- Si demandé, test de l'adjoint du filtre récursif :
- Si demandé, comparaison du filtre récursif avec la valeur analytique de la fonction SOAR ;
- Calcul de la moyenne et de l'écart-type du champs initial pour chaque date ;
- Recentrage du champs initial pour avoir une moyenne nulle à chaque date ;
- Filtrage du champs initial ;
- Calcul de la moyenne et de l'écart-type du champs filtré pour chaque date ;
- Filtrage du rapport des variabilités ($\sigma_{ini}/\sigma_{filtered}$) ;
- Multiplication du champs filtré par le rapport des variabilités filtrées ;
- Ecrétage du champs obtenu aux valeurs limites du champs initial ;

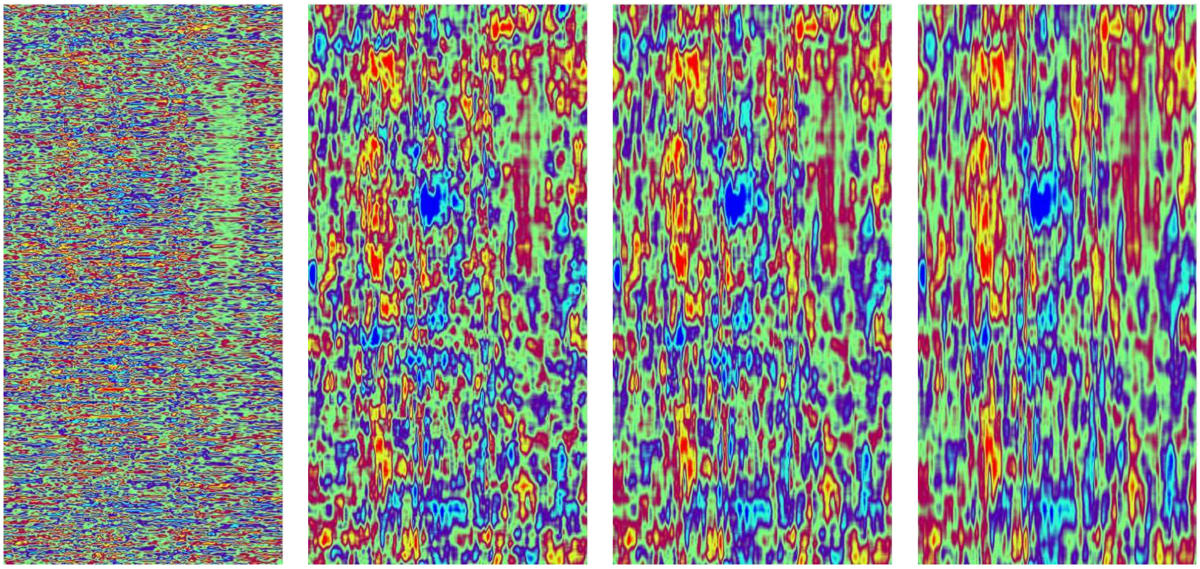
- Calcul de la moyenne du champs initial pour chaque date ;
- Recentrage du champs obtenu pour avoir une moyenne nulle à chaque date ;
- Calcul de la moyenne et de l'écart-type du champs final pour chaque date ;
- Ecriture du champs filtré dans des fichiers annuels.

Le champs obtenu a donc l'auto-corrélation temporelle voulue, ainsi que la même variabilité que le champs initial.

Chapitre 4

Présentation des résultats

Les hoevmoller de la figure 4.1 permettent de bien voir l'impact du filtre récursif, et donc de la fonction de corrélation appliquée. Il apparaît ainsi que le champs initial est totalement décorrélé. Les figures 4.1(b), 4.1(c) et 4.1(d) montrent l'impact de la fonction de corrélation SOAR pour différentes échelles de corrélation. Il reste ainsi à déterminer l' échelle de corrélation la plus adaptée.



(a) non corrélé (b) corrélation à 5 jours (c) corrélation à 7 jours (d) corrélation à 10 jours

FIG. 4.1 – Hoevmoller de l'année 1993 représentant les perturbations de SST pour une latitude de 56° sud.

Les figures 4.2, 4.3, 4.4 et 4.5 montrent le champs de la SST perturbée à une date donnée (le 1er janvier 1993). On peut ainsi constater les modifications apportées à la structure du champs par la fonction de corrélation. On voit ainsi que les structures les plus marquées apparaîssent dans tout les cas. Par contre, pour les structures les moins intenses, la fonction de corrélation a tendance à les modifier complètement. Ce qui est, somme toute, normal.

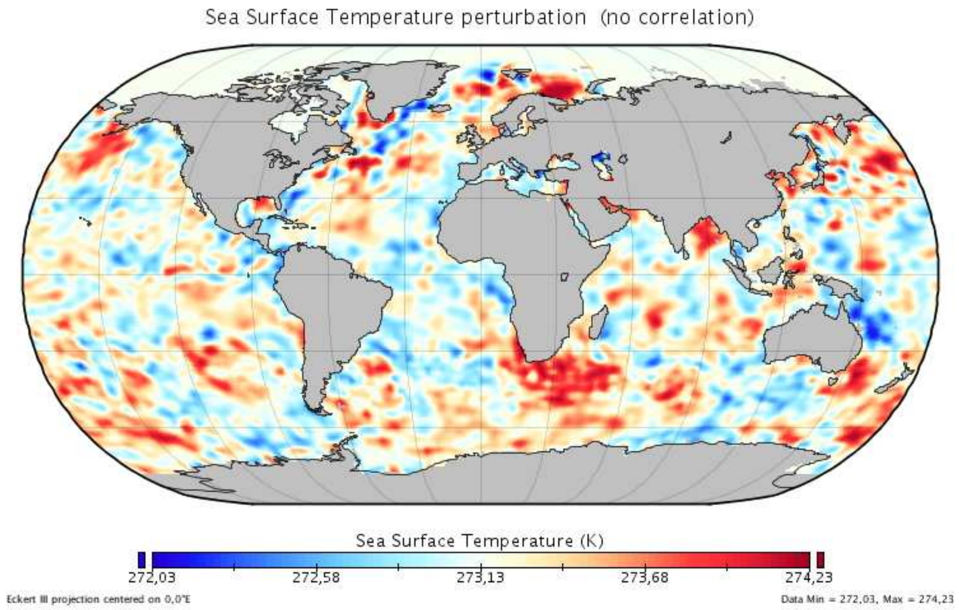


FIG. 4.2 – Perturbation de la SST pour le 1er janvier 1993 sans corrélation

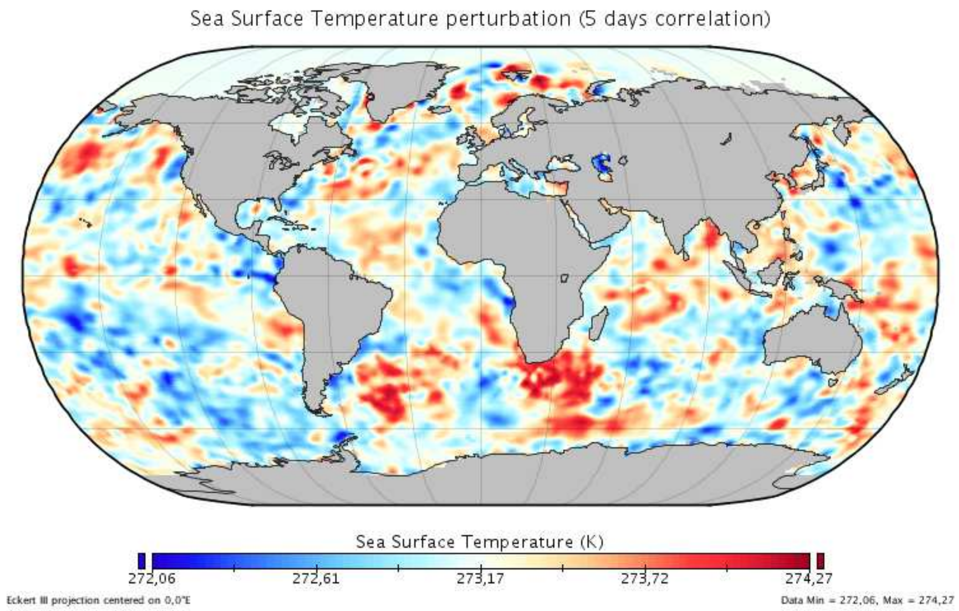


FIG. 4.3 – Perturbation de la SST pour le 1er janvier 1993 avec une corrélation de 5 jours

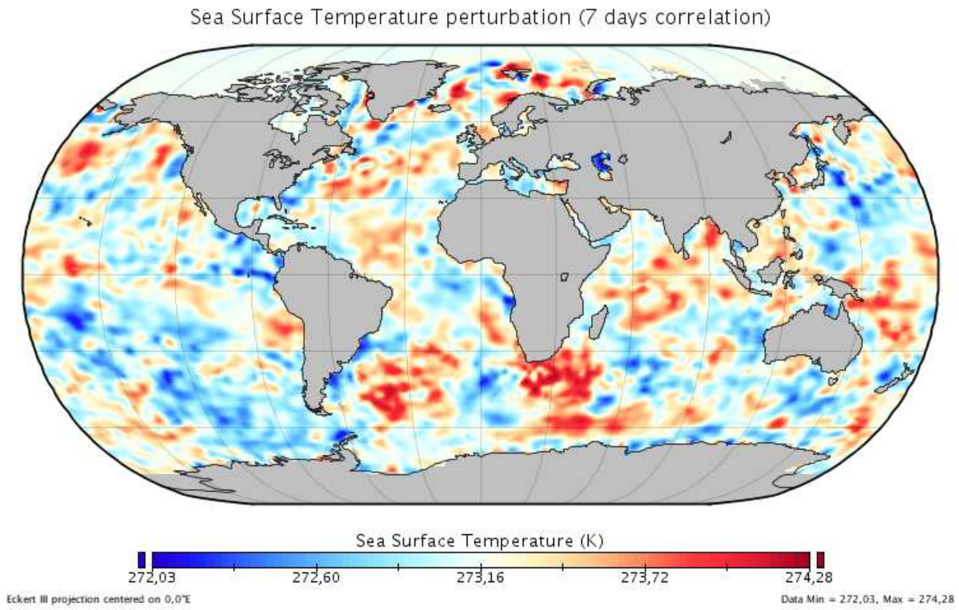


FIG. 4.4 – Perturbation de la SST pour le 1er janvier 1993 avec une corrélation de 7 jours

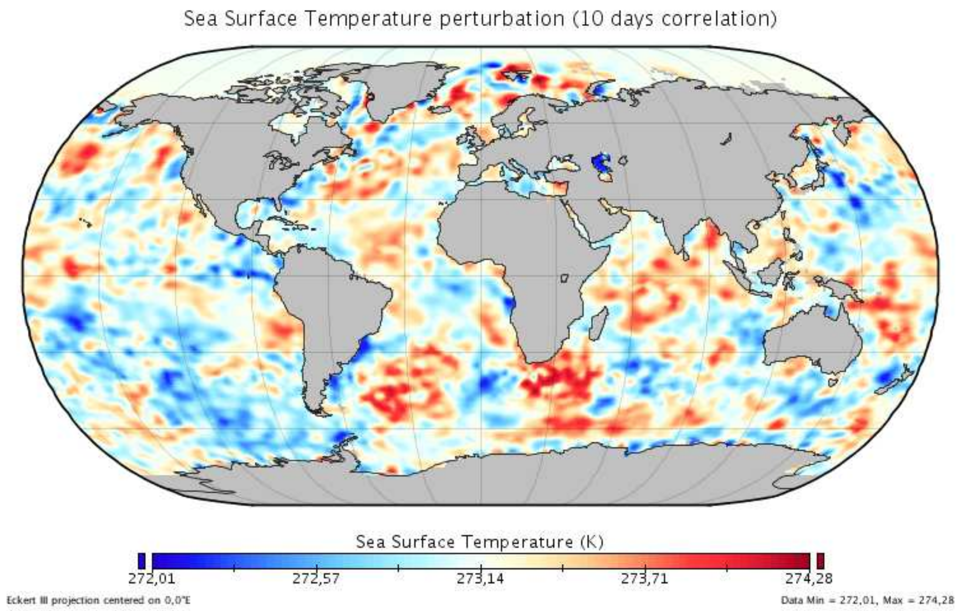


FIG. 4.5 – Perturbation de la SST pour le 1er janvier 1993 avec une corrélation de 10 jours

Chapitre 5

Conclusion

Le filtre récursif s'avère être un outils très puissant permettant de modéliser de nombreuses fonctions de corrélation. Sa mise en oeuvre est très simple pour le cas mono-dimensionnel et ses performance en temps de calcul sont impressionnantes.

Un module fortran a été développé. Il comprend le filtre récursif pour des tableaux d'une, deux ou trois dimensions de réels en simple ou double précision, les filtres adjoints correspondants et les tests des ces adjoints. De plus, les sources sont maintenant dans l'arborescence CVS d'OPAVAR.

En ce qui concerne l'utilisation du filtre pour corréler les perturbations de la SST journalière, l'outils développé permet de traiter la période 1962-2002 d'un seul tenant. L'échelle de corrélation peut être adaptée, mais pour notre application, elle a été définie à 7 jours.

Les champs obtenus sont maintenant utilisés dans les méthodes d'ensembles d'OPAVAR.

Bibliographie

[1] A.F. Bennet, B.S. Chua, and L.M. Leslie. Generalized inversion of a global numerical weather prediction model. *Meteorology and Atmospherics Physics*, 60 :165–178, 1996.

[2] J. Gao, M. Xue, K. Brewster, and K.K. Droegemeier. A three-dimensional variational data analysis method with recursive filter for doppler radars. *Quaterly Journal of the Royal Meteorological Society*, 127 :1815–1846, 2001.

[3] T. Gneiting. Correlation functions for atmospheric data analysis. *Quaterly Journal of the Royal Meteorological Society*, 125 :2449–2464, 1999.

[4] C.M. Hayden and R.J. Purser. Recursive filter objective analysis of meteorological fields, applications to nesdis operational processing. *Journal of Applied Meteorology*, 34 :3–15, 1995.

[5] A. Lorenc. Iterative analysis using covariance functions and filters. *Quaterly Journal of the Royal Meteorological Society*, 118 :569–591, 1992.

[6] H.E. Ngodock. Efficient implementation of covariance multiplication for data assimilation with the representer method. *Ocean Modelling*, 8 :237–251, 2005.

[7] R.J. Purser and R. McQuigg. A successive correction analyses scheme using recursive numerical filter. Technical Note 154, Met Office, Bracknell, 1982.

[8] R.J. Purser, Wan-Shu Wu, D.F. Parrish, and N.M. Roberts. Numerical aspect of the application of recursive filters to variational statistical analysis. part i : Spatially homogeneous and isotropic gaussian covariances. *Monthly Weather Review*, 13 :1524–1535, 2003.

[9] A. Tarantola. *Inverse problem theory - methods for data fitting and model parameter estimation*. Elsevier, 1987.

[10] A. Weaver and P. Courtier. Correlation mmodeling on the sphere using a generalized diffusion equation. *Quaterly Journal of the Royal Meteorological Society*, 127 :1815–1846, 2001.

[11] A. Weisheimer. Sst and windstress perturbations for seasonal and annual simulations. Ensembles rt1 documentation, ECMWF, 2005.

Annexe A

Inteface du module du filtre récursif

```
MODULE filters
!
!----------------------------------------------------------------------
!
!                       MODULE filters
!
! Purpose : apply filter
!
! Why     : to simplify filters use
!
! Version : N. Daget      2005-01      Original
!
!----------------------------------------------------------------------
!
PRIVATE
PUBLIC :: recursive_filter, recursive_filter_adj, test_recursive_filter_adj
!
INTERFACE recursive_filter
  MODULE PROCEDURE recursive_filter_1d_r, recursive_filter_1d_d, &
                   recursive_filter_2d_r, recursive_filter_2d_d, &
                   recursive_filter_3d_r, recursive_filter_3d_d
END INTERFACE
!
INTERFACE recursive_filter_adj
  MODULE PROCEDURE recursive_filter_adj_1d_r, recursive_filter_adj_1d_d, &
                   recursive_filter_adj_2d_r, recursive_filter_adj_2d_d, &
                   recursive_filter_adj_3d_r, recursive_filter_adj_3d_d
END INTERFACE
!
INTERFACE test_recursive_filter_adj
  MODULE PROCEDURE test_recursive_filter_adj_1d_r, test_recursive_filter_adj_1d_d, &
                   test_recursive_filter_adj_2d_r, test_recursive_filter_adj_2d_d, &
                   test_recursive_filter_adj_3d_r, test_recursive_filter_adj_3d_d
END INTERFACE
!
```

Annexe B

Code source du filtre récursif 1D

```
!
!-----------------------------------------------------------------------
!
! R E C U R S I V E   F I L T E R
!
! from Iterative analysis using covariance functions and filters by
!      Andrew Lorenc (Q. J. R. Meteorol. Soc. (1992), 118, pp. 569-591)
!      Appendix A pp 587-589
!
!-----------------------------------------------------------------------
!
   SUBROUTINE recursive_filter_1d_r(A, alpha, pass)
!
! 1D recursive filter for reals
!
! A is the initial value
! B is the value after filtering for i = 1 to n
! C (in our case A) is the value after one pass of the filter in each
!   direction
!
! alpha is the filter coefficient
! N is the number of time step
!
      IMPLICIT NONE
!
      REAL, DIMENSION(:), INTENT(inout) :: A
      REAL, INTENT(in)                  :: alpha
      INTEGER, INTENT(in)               :: pass
!
      INTEGER                           :: N, i
      REAL, DIMENSION(:), ALLOCATABLE   :: B, C
!
! Look for time dimension
!
      N = SIZE(A,1)
```

```
!
! Allocate array
!
      ALLOCATE (B(N))
      ALLOCATE (C(N))
!
!
! Forward Langevin
!
      B(:) = 0.0
      SELECT CASE (pass)
      CASE (1)
         B(1) = ( 1 - alpha ) * A(1)
      CASE (2)
         B(1) = 1 / ( 1 + alpha ) * A(1)
         B(1) = ( 1 - alpha ) / ( 1 - alpha**2 ) *  A(1)
      CASE (3)
         B(1) = ( 1 - alpha ) / ( 1 - alpha**2 )**2 * ( A(1) - alpha**3 * A(2) )
      CASE default
         B(1) = ( 1 - alpha ) / ( 1 - alpha**2 )**2 * ( A(1) - alpha**3 * A(2) )
      END SELECT
      DO i = 2, N
         B(i) = alpha * B(i-1) + ( 1 - alpha ) * A(i)
      ENDDO
!
! Backward langevin
!
      SELECT CASE (pass)
      CASE (1)
         C(N) = 1 / ( 1 + alpha ) * B(N)
      CASE (2)
         C(N) = ( 1 - alpha ) / ( 1 - alpha**2 )**2 * ( B(N) - alpha**3 * B(N-1) )
      CASE (3)
         C(N) = ( 1 - alpha ) / ( 1 - alpha**2 )**3 * &
                ( -1 * B(N) + ( 3 - alpha**2 ) * alpha**3 * B(N-1) - alpha**4 * B(N-2) )
      CASE default
         C(N) = ( 1 - alpha ) / ( 1 - alpha**2 )**3 * &
                ( -1 * B(N) + ( 3 - alpha**2 ) * alpha**3 * B(N-1) - alpha**4 * B(N-2) )
      END SELECT
      DO i = N-1, 1, -1
         C(i) = alpha * C(i+1) + ( 1 - alpha ) * B(i)
      ENDDO
!
      A(1:N) = C(1:N)
!
!
      DEALLOCATE (B)
      DEALLOCATE (C)
!
   END SUBROUTINE recursive_filter_1d_r
```

Annexe C

Code source de l'adjoint du filtre récursif 1D

```
!
!----------------------------------------------------------------------
!
! A D J O I N T   R E C U R S I V E   F I L T E R
!
!----------------------------------------------------------------------
!
   SUBROUTINE recursive_filter_adj_1d_r(A, alpha, pass)
!
! 1D adjoint recursive filter for reals
!
! A is the initial value
! B is the value after filtering for i = 1 to n
! C (in our case A) is the value after one pass of the filter in each
!   direction
!
! alpha is the filter coefficient
! N is the number of time step
!
      IMPLICIT NONE
!
      REAL, DIMENSION(:), INTENT(inout) :: A
      REAL, INTENT(in)                  :: alpha
      INTEGER, INTENT(in)               :: pass
!
      INTEGER                           :: N, i
      REAL, DIMENSION(:), ALLOCATABLE   :: B, C
!
! Look for time dimension
!
      N = SIZE(A,1)
!
```

```
! Allocate array
!
      ALLOCATE (B(N))
      ALLOCATE (C(N))
!
      B(:) = 0.
      C(:) = 0.
!
! Backward langevin
!
      C(1:N) = C(1:N) + A(1:N)
      A(1:N) = 0.
      DO i = 1, N - 1
        B(i) = B(i) + ( 1 - alpha ) * C(i)
        C(i+1) = C(i+1) + alpha * C(i)
        C(i) = 0.
      ENDDO
      SELECT CASE (pass)
      CASE (1)
         B(N) = B(N) + 1 / ( 1 + alpha ) * C(N)
      CASE (2)
         B(N)   = B(N)   + ( 1 - alpha ) / ( 1 - alpha**2 )**2 * C(N)
         B(N-1) = B(N-1) - alpha**3 * ( 1 - alpha ) / ( 1 - alpha**2 )**2 * C(N)
      CASE (3)
         B(N)   = B(N)   - ( 1 - alpha ) / ( 1 - alpha**2 )**3 * C(N)
         B(N-1) = B(N-1) + alpha**3 * ( 3 - alpha**2 ) * ( 1 - alpha ) / ( 1 - alpha**2 )**3 * C(N)
         B(N-2) = B(N-2) - alpha**4 * ( 1 - alpha ) / ( 1 - alpha**2 )**3 * C(N)
      CASE default
         B(N)   = B(N)   - ( 1 - alpha ) / ( 1 - alpha**2 )**3 * C(N)
         B(N-1) = B(N-1) + alpha**3 * ( 3 - alpha**2 ) * ( 1 - alpha ) / ( 1 - alpha**2 )**3 * C(N)
         B(N-2) = B(N-2) - alpha**4 * ( 1 - alpha ) / ( 1 - alpha**2 )**3 * C(N)
      END SELECT
!
! Forward Langevin
!
      DO i = N, 2, -1
        A(i) = A(i) + ( 1 - alpha ) * B(i)
        B(i-1) = B(i-1) + alpha * B(i)
        B(i) = 0.
      ENDDO
      SELECT CASE (pass)
      CASE (1)
         A(1) = A(1) + ( 1 - alpha ) * B(1)
      CASE (2)
         A(1) = A(1) + 1 / ( 1 + alpha ) * B(1)
      CASE (3)
         A(1) = A(1) + ( 1 - alpha ) / ( 1 - alpha**2 )**2 * B(1)
         A(2) = A(2) - alpha**3 * ( 1 - alpha ) / ( 1 - alpha**2 )**2 * B(1)
      CASE default
         A(1) = A(1) + ( 1 - alpha ) / ( 1 - alpha**2 )**2 * B(1)
```

```
        A(2) = A(2) - alpha**3 * ( 1 - alpha ) / ( 1 - alpha**2 )**2 * B(1)
      END SELECT
!
      DEALLOCATE (B)
      DEALLOCATE (C)
!
   END SUBROUTINE recursive_filter_adj_1d_r
```

Annexe D

Code source du test de l'adjoint du filtre récursif 1D

```
!
!---------------------------------------------------------------------
!
! T E S T   A D J O I N T   R E C U R S I V E   F I L T E R
!
!---------------------------------------------------------------------
!
  SUBROUTINE test_recursive_filter_adj_1d_r(A, alpha)
!
! test of 1D adjoint recursive filter for reals
!
     IMPLICIT NONE
!
     REAL, DIMENSION(:), INTENT(in)  :: A
     REAL, INTENT(in)                :: alpha
!
     INTEGER                         :: N
     REAL                                  :: zpdx, zpdy
     REAL, DIMENSION(:), ALLOCATABLE :: B, C
!
! Look for dimensions
!
     N    = SIZE(A,1)
!
! Allocate arrays
!
     ALLOCATE( B(N) )
     ALLOCATE( C(N) )
!
! Compute R(1/2).dx
!
     B(:) = A(:)
```

```
      CALL recursive_filter(B,alpha,1)
      zpdy = SUM(B*B)
!
! Compute R(1/2)T.R(1/2).dx
!
      C(:) = B(:)
      CALL recursive_filter_adj(C,alpha,1)
      zpdx = SUM(A*C)
!
! Test if (R(1/2).dx)T.W.R(1/2).dx = dxT.(R(1/2)T.W.R(1/2).dx
! with W = I
!
      WRITE(*,*)
      WRITE(*,*)'Test recursive_filter_adj'
      WRITE(*,*)"Difference = ",ABS(zpdx - zpdy)/ABS(zpdx)
      WRITE(*,*)
!
   END SUBROUTINE test_recursive_filter_adj_1d_r
```