Nicolas Daget

Interpolation d'une grille ORCA2 vers une grille régulière

29 mars 2006

Technical Report
TR/CMGC/06/18
GLOBC - CERFACS

Table des matières

Table des figures

Chapitre 1

Introduction

Certains outils existaient déjà pour interpoler des champs d'une grille ORCA[1] vers une grille régulière. Un package utilisant IDL[2] a, par exemple, été développé au sein du projet ENACT[3]. Il permet d'interpoler les sorties du modèle sur la grille ORCA 2 degrés vers une grille régulière.

Cependant, ce package souffre d'insuffisances :
- Il est nécessite IDL qui n'est pas présent sur toutes les plateformes ;
- Les performances, du point de vue du temps de calcul, sont décevantes ;
- Le code source est illisible, ce qui rend la compréhension de l'algorithme impossible, ce qui entraîne donc une défiance légitime ;
- L'utilisation de ce package est compliquée ;
- L'interpolation horizontal est une interpolation bilinéaire.

L'objectif de ce travail était donc de créer un nouvel outils ergonomique et efficace qui supplée les insufissances de l'ancien.

Ce rapport présentera donc d'abord l'interpolation horizontal qui utlise un schéma bicubique avec OASIS3[4]. Ensuite l'interpolation vertical utilisant un spline cubique sera présenté. Dans une dernière partie, le package sera présenté avec le système de scripts.

[1] grille tordue où les pôles sont sur les continents
[2] logiciel d'analyse de données et de visualisation développé par RSI.
[3] ENhAnced ocean data assimilation and ClimaTe prediction
[4] Coupleur développé au CERFACS et qui permet, entre autres, de faire de l'interpolation.

Chapitre 2

L'interpolation horizontale

Dans ce chapitre, différentes méthodes d'interpolation bidimensionnelle seront présentées ainsi que celle utilisée dans notre cas.

2.1 Présentation des différentes interpolations horizontales

2.1.1 Présentation

Les différentes méthodes d'interpolation seront développées dans la suite et mise en évidence avec le graphique représentant un champs sur un maillage 3x3 (figure 2.1).

2.1.2 Le plus proche voisin

La méthode du plus proche voisin consiste à attribuer à chaque point cible une combinaison des valeurs des points sources les plus proches de son antécédent par la transformation inverse. C'est une interpolation polynomiale d'ordre 0.

Le code cherche les n plus proches voisins puis calcule les poids tels que :

$$w \quad = \quad \frac{\frac{1}{(d+\varepsilon)}}{\sum_{i=1}^{n}(\frac{1}{d_i+\varepsilon})} \tag{2.1}$$

Où ε est un petit nombre qui permet d'éviter la division par zéro. n est le nombre de voisins utilisés. Ce nombre est déterminé par l'utilisateur. d correspond à la distance angulaire entre le point cible et le point source défini par :

$$d \quad = \quad cos^{-1}(\ cos\theta_c\ cos\theta_s\ (cos\phi_c\ cos\phi_s + sin\phi_c\ sin\phi_s) + sin\theta_c\ sin\theta_s\ ) \tag{2.2}$$

Avec (θ, ϕ) comme coordonnées longitudinales et latitudinales relatives à la grille source (s) et à la grille cible (c).

Cette méthode a le mérite de conserver la dynamique du champs original et de ne pas créer de niveaux intermédiaires. En revanche, elle détermine une zone d'influence exclusive pour chaque point source. Dans le cas du sous-échantillonnage, cela se traduit par une sensibilité accrue au bruit. Pour le sur-échantillonnage, la sensation de discontinuité est accentuée par la formation de larges paliers.

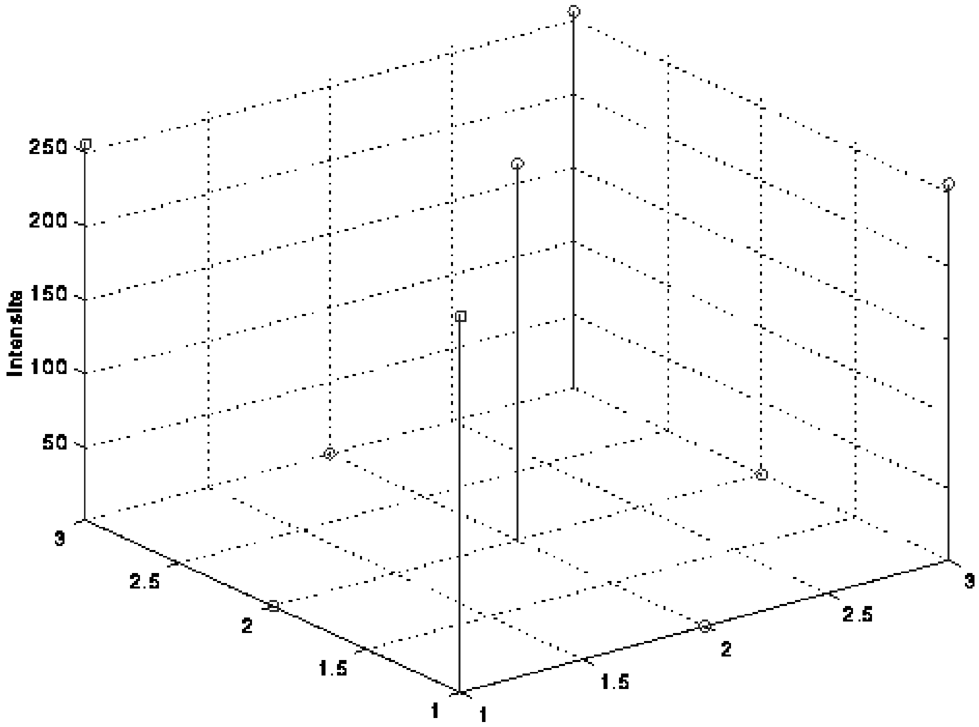

FIG. 2.1 – Représentation en élévation d'un champs sur un maillage 3x3 sur laquelle les interpolations seront illustrées.

2.1.3 L'interpolation bilinéaire

L'interpolation bilinéaire est une méthode se basant sur des polynômes d'ordre 1. Elle consiste à attribuer à chaque point cible une combinaison linéaire des quatre points sources les plus proches de son antécédent par la transformation inverse.

L'interpolation bilinéaire utilisée est un schéma itératif qui implique de mapper une cellule quadrilatère sur une cellule carrée de coordonnées (0,0), (1,0), (0,1) et (1,1). Les coordonnées longitudinales et latitudinales du point 1 sont $(\theta(i,j), \phi(i,j))$, celles du point 2 $(\theta(i+1,j), \phi(i+1,j))$, etc. Définissons α et β les coordonnées locales continues telles que les coordonnées du point 1 sont (0,0), du point 2 (1,0), du point 3 (1,1) et du point 4 (0,1). Si le point P est dans la cellule formée par les quatre points précédents, la fonction f au point P peut être approximée par :

$$\begin{aligned} f_P &= (1-\alpha)(1-\beta)\ f(i,j) + \alpha(1-\beta)\ f(i+1,j) + \alpha\beta\ f(i+1,j+1) + (1-\alpha)\beta\ f(i,j+1) \\ &= w_1\ f(i,j) + w_2\ f(i+1,j) + w_3\ f(i+1,j+1) + w_4\ f(i,j+1) \end{aligned} \tag{2.3}$$

Déterminer les poids revient donc à calculer α et β au point P. Les coordonnées longitudinales et latitudinales (θ, ϕ) du point P sont connues et peuvent être approximées par :

$$\begin{aligned} \theta &= (1-\alpha)(1-\beta)\theta_1 + \alpha(1-\beta)\theta_2 + \alpha\beta\theta_3 + (1-\alpha)\beta\theta_4 \\ \phi &= (1-\alpha)(1-\beta)\phi_1 + \alpha(1-\beta)\phi_2 + \alpha\beta\phi_3 + (1-\alpha)\beta\phi_4 \end{aligned} \tag{2.4}$$

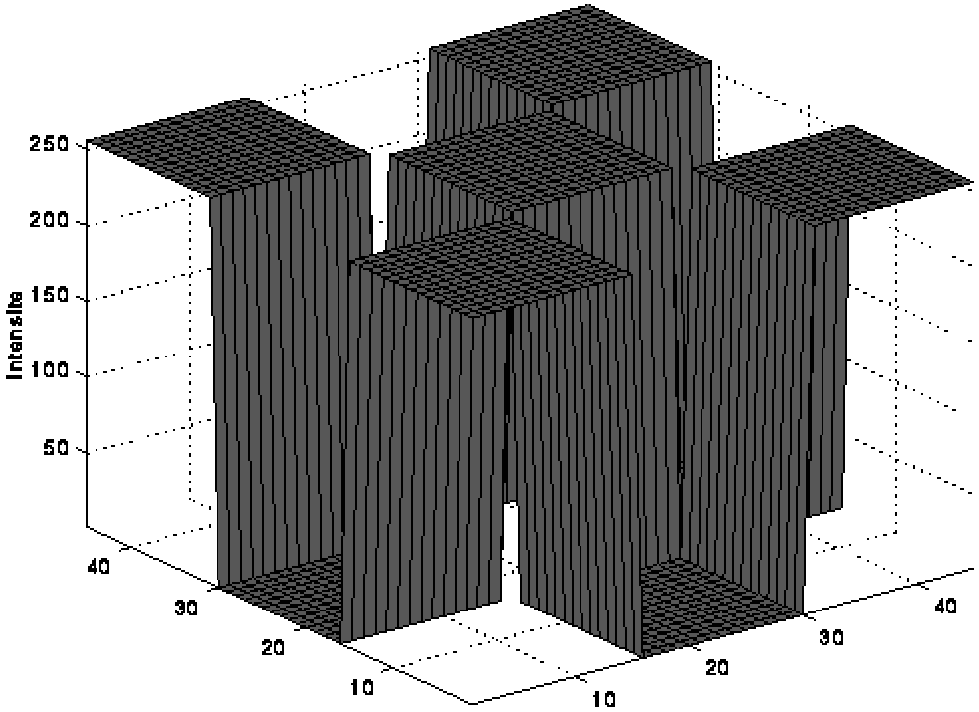

FIG. 2.2 – Représentation en élévation du champs de la figure 2.1 reéchantillonnée à 45x45 par la méthode du plus proche voisin ($n = 1$).

Comme ce système (2.4) est non linéaire en α et β, il faut linéariser et itérer autour de la solution. En différenciant (2.4), le système suivant (2.5) est obtenu.

$$\begin{bmatrix} \delta\theta \\ \delta\phi \end{bmatrix} = A \begin{bmatrix} \delta\alpha \\ \delta\beta \end{bmatrix} \tag{2.5}$$

avec

$$A = \begin{bmatrix} (\theta_2 - \theta_1) + (\theta_1 - \theta_4 + \theta_3 - \theta_2)\beta & (\theta_4 - \theta_1) + (\theta_1 - \theta_4 + \theta_3 - \theta_2)\alpha \\ (\phi_2 - \phi_1) + (\phi_1 - \phi_4 + \phi_3 - \phi_2)\beta & (\phi_4 - \phi_1) + (\phi_1 - \phi_4 + \phi_3 - \phi_2)\alpha \end{bmatrix} \tag{2.6}$$

En inversant le sytème,

$$\delta\alpha = \begin{vmatrix} \delta\theta & (\theta_4 - \theta_1) + (\theta_1 - \theta_4 + \theta_3 - \theta_2)\alpha \\ \delta\phi & (\phi_4 - \phi_1) + (\phi_1 - \phi_4 + \phi_3 - \phi_2)\alpha \end{vmatrix} \div det(A) \tag{2.7}$$

et

$$\delta\beta = \begin{vmatrix} (\theta_2 - \theta_1) + (\theta_1 - \theta_4 + \theta_3 - \theta_2)\beta & \delta\theta \\ (\phi_2 - \phi_1) + (\phi_1 - \phi_4 + \phi_3 - \phi_2)\beta & \delta\phi \end{vmatrix} \div det(A) \tag{2.8}$$

En démarrant avec une condition initiale pour α et β (par exemple $\alpha = \beta = 0$), les équations (2.7) et (2.8) peuvent être résoluent par itérations jusqu'à ce que $\delta\alpha$ et $\delta\beta$ soient suffisament petits. Les poids peuvent alors être calculés à partir de l'équation (2.3). À noter que pour une grille simple en longitude-latitude, le système converge à la première itération.

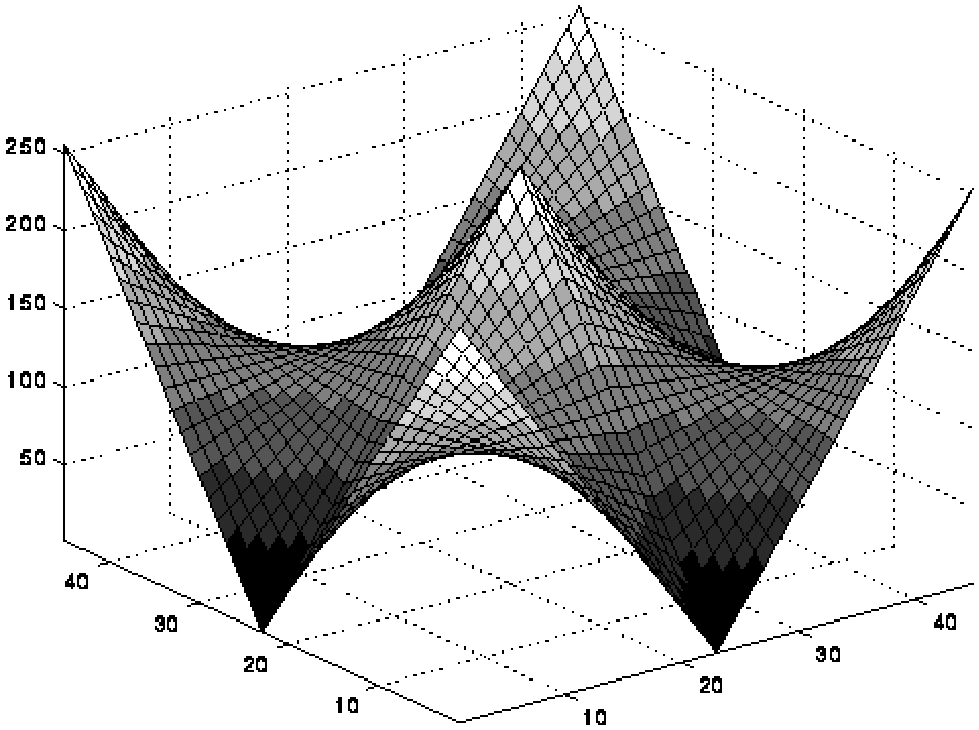

FIG. 2.3 – Représentation en élévation du champs de la figure 2.1 reéchantillonnée à 45x45 par la méthode du bilinéaire.

La mise en oeuvre est bien entendue un peu plus lourde que pour le plus proche voisin, mais il ne se crée plus de paliers lors du sur-échantillonnage. Le passage d'un niveau à un autre est dans ce cas linéaire, mais les discontinuités d'ordre 1 (gradient) sont conservées sous la forme d'arêtes. En contrepartie, des niveaux intermédiaires sont ajoutés dans la dynamique. Néanmoins, dans le cas du sous-échantillonnage, la prise en compte des quatre voisins rend la méthode moins sensible au bruit qu'avec le plus proche voisin.

2.1.4 L'interpolation bicubique

Pour une interpolation polynômiale d'ordre encore supérieur, on trouve la méthode dite bicarrée. Elle impose cependant des conditions dissymétriques dans les champs qui lui font préférer un ordre trois : l'interpolation bicubique.

Elle consiste à attribuer à chaque point cible une combinaison polynômiale cubique des douze (parfois seize) points sources les plus proches de son antécédent par la transformation inverse.

Comme en bilinéaire, on tient non seulement directement compte de l'intensité des quatre points les plus proches, mais également du gradient.

Les détails mathématiques de cette interpolation sont expliqués dans le cas monodimensionnel dans la section 3.1 consacrée au spline cubique.

En définissant α et β comme dans la section précédente (section 2.1.3), la fonction f au point P peut être approximée par :

$$
\begin{aligned}
f_P &= (1-\beta^2(3-2\beta))(1-\alpha^2(3-2\alpha))\ f(i,j) \\
&+ (1-\beta^2(3-2\beta))\alpha^2(3-2\alpha)\ f(i+1,j) \\
&+ \beta^2(3-2\beta)\alpha^2(3-2\alpha)\ f(i+1,j+1) \\
&+ \beta^2(3-2\beta)(1-\alpha^2(3-2\alpha))\ f(i,j+1) \\
&+ (1-\beta^2(3-2\beta))\alpha(\alpha-1)^2\ \frac{\partial f}{\partial i}(i,j) \\
&+ (1-\beta^2(3-2\beta))\alpha^2(\alpha-1)\ \frac{\partial f}{\partial i}(i+1,j) \\
&+ \beta^2(3-2\beta)\alpha^2(\alpha-1)\ \frac{\partial f}{\partial i}(i+1,j+1) \\
&+ \beta^2(3-2\beta)\alpha(\alpha-1)^2\ \frac{\partial f}{\partial i}(i,j+1) \\
&+ \beta(\beta-1)^2(1-\alpha^2(3-2\alpha))\ \frac{\partial f}{\partial j}(i,j) \\
&+ \beta(\beta-1)^2\alpha^2(3-2\alpha)\ \frac{\partial f}{\partial j}(i+1,j) \\
&+ \beta^2(\beta-1)\alpha^2(3-2\alpha)\ \frac{\partial f}{\partial j}(i+1,j+1) \\
&+ \beta^2(\beta-1)(1-\alpha^2(3-2\alpha))\ \frac{\partial f}{\partial j}(i,j+1) \\
&+ \alpha(\alpha-1)^2\beta(\beta-1)^2\ \frac{\partial^2 f}{\partial i\partial j}(i,j) \\
&+ \alpha^2(\alpha-1)\beta(\beta-1)^2\ \frac{\partial^2 f}{\partial i\partial j}(i+1,j) \\
&+ \alpha^2(\alpha-1)\beta^2(\beta-1)\ \frac{\partial^2 f}{\partial i\partial j}(i+1,j+1) \\
&+ \alpha(\alpha-1)^2\beta^2(\beta-1)\ \frac{\partial^2 f}{\partial i\partial j}(i,j+1) \qquad (2.9)
\end{aligned}
$$

De la même manière que pour l'interpolation bilinéaire, en différanciant et en itérant autour de la solution, le système peut être approximé. À la différence notable qu'il faut connaître les dérivées secondes de la fonction f. Ce qui est expliqué dans le cas monodimensionnel du spline cubique dans la section 3.1.

Avec l'interpolaiton bicubique, on étend les propriétés intéressantes de l'interpolation bilinéaire en y ajoutant la continuité du gradient. La prise en compte des tangentes permet en effet de conserver une grande régularité d'un bloc de quatre points à un autre bloc contigu. Cependant, non seulement des niveaux intermédiaires peuvent être créés comme en bilinéaire, mais encore, la dynamique du champs est augmentée : ce phénomène est justement dû à la prise en compte des tangentes. Elle donne une certaine inertie dans la variation des niveaux, d'où un dépassement de la dynamique originale aux valeurs limites.

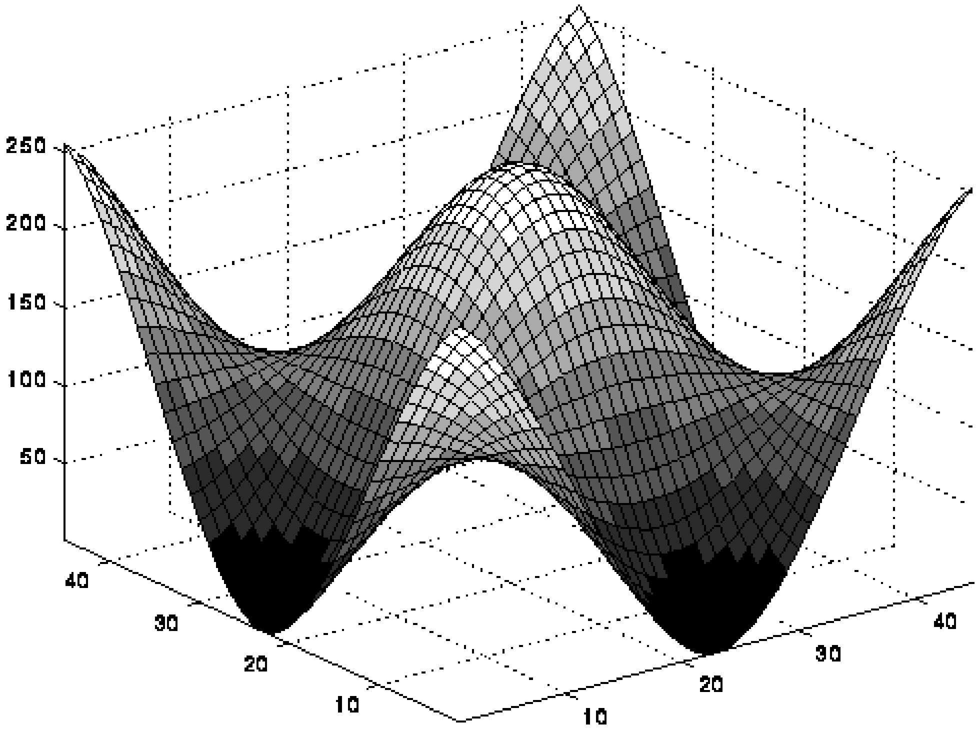

FIG. 2.4 – Représentation en élévation du champs de la figure 2.1 reéchantillonnée à 45x45 par la méthode du bicubique.

2.2 Interpolation horizontale utilisée

Dans notre cas, OASIS est utilisé pour faire l'interpolation horizontale. OASIS utilise la librairie SCRIP[1] et l'interpolation choisie est l'interpolation bicubique. Dans le cas où ils manquent des points sources pour l'interpolation, comme près des côtes, c'est l'interpolation des plus proches voisins qui est utilisé.

[1]Spherical Coordinate Remapping and Interpolation Package développé par Los Alamos National Laboratory

Chapitre 3

L'interpolation verticale

Différentes intepolations monodimensionnelles sont disponibles telles que le plus proche voisin (cf. section 2.1.2), l'interpolation linéaire (cf. section 2.1.3) ou le spline cubique. Le plus proche voisin et l'interpolation linéaire sont décrits dans le cas bidimensionnel dans les sections 2.1.2 et 2.1.3. Dans cette partie, nous nous attacherons à expliquer le cas du spline cubique et à présenter les interpolations utilisées.

3.1 Le spline cubique

Il est possible de fabriquer des interpolateurs polynômiaux qui passent par une série de points ((x_0, y_0), (x_1, y_1), ..., (x_n, y_n)). Cependant, ce type d'interpolateur peut parfois beaucoup osciller.

Pour fabriquer un meilleur interpolateur, il faut permettre la courbure entre les points. Pour cela, les dérivées premières et secondes doivent être continues sur l'intervalle $[x_0, x_n]$. L'interpolateur qui lie les points (x_k, y_k) et (x_{k+1}, y_{k+1}) est de forme polynômial du troisième ordre :

$$s_k(x) = s_{k,0} + s_{k,1}(x - x_k) + s_{k,2}(x - x_k)^2 + s_{k,3}(x - x_k)^3 \qquad x \in [x_k, x_{k+1}] \tag{3.1}$$

Et est soumis à ces contraintes :

$$\begin{aligned} s_k(x) &= y_k && (3.2)\\ s_k(x_{k+1}) &= s_{k+1}(x_{k+1}) && (3.3)\\ s'_k(x_{k+1}) &= s'_{k+1}(x_{k+1}) && (3.4)\\ s''_k(x_{k+1}) &= s''_{k+1}(x_{k+1}) && (3.5) \end{aligned}$$

L'équation 3.2 exprime l'interpolation de (x_k, y_k), l'équation 3.3 exprime la continuité de l'interpolateur, l'équation 3.4 celle des dérivées premières et l'équation 3.5 celle des dérivées secondes.

A noter que pour chaque spline $s_k(x)$ (à l'exception du dernier), il y a quatre équations et quatre inconnues (les $s_{k,i}$). Pour le dernier spline, il n'y a pas les deux dernières conditions. Il faut donc finir avec deux inconnues de plus que d'équations. Pour fermer le système, il faudra imposer des conditions aux derniers points.

3.1.1 Construction du spline cubique

Définissons les trois quantitées suivantes :

$$\begin{aligned} h_k &= x_{k+1} - x_k & (3.6)\\ d_k &= \frac{y_{k+1} - y_k}{x_{k+1} - x_k} & (3.7)\\ m_k &= s''(x_k) & (3.8) \end{aligned}$$

Où h_k est l'écart entre x_k et x_{k+1}, d_k est la pente entre (x_k, y_k) et (x_{k+1}, y_{k+1}) et m_k est la dérivée seconde de l'interpolateur en x_k.

Il est nécessaire que $s(x)$ soit cubique par morceaux. Ce qui implique que $s'(x)$ soit quadratique par morceaux et que $s''(x)$ soit linéaire par morceaux.
Effectuons l'interpolation linéaire (Lagrange) de la dérivée seconde par morceaux :

$$s_k''(x) = s''(x_k)\frac{x - x_{k+1}}{x_k - x_{k+1}} + s''(x_{k+1})\frac{x - x_k}{x_{k+1} - x_k} \qquad (3.9)$$

En substituant par m_k et h_k, l'équation 3.9 devient :

$$s_k''(x) = \frac{m_k}{h_k}(x_{k+1} - x) + \frac{m_{k+1}}{h_k}(x - x_k) \qquad x \in [x_k, x_{k+1}] \textit{ et } k = 0, 1, \cdots, n-1 \qquad (3.10)$$

En intégrant, on obtient l'équation de la dérivée première et de l'interpolateur lui-même :

$$\begin{aligned} s_k'(x) &= -\frac{m_k}{2h_k}(x_{k+1} - x)^2 + \frac{m_{k+1}}{2h_k}(x - x_k)^2 - p_k + q_k & (3.11)\\ s_k(x) &= \frac{m_k}{6h_k}(x_{k+1} - x)^3 + \frac{m_{k+1}}{6h_k}(x - x_k)^3 + p_k(x_{k+1} - x) + q_k(x - x_k) & (3.12) \end{aligned}$$

Maintenant, évaluons l'interpolateur en $x = x_k$ et en $x = x_{k+1}$ sachant que $s_k(x_k) = y_k$ et $s_k(x_{k+1}) = y_{k+1}$:

$$s_k(x_k) = y_k = \frac{m_k}{6h_k}(x_{k+1} - x_k)^3 + p_k(x_{k+1} - x) \quad \Rightarrow \quad p_k = \frac{y_k}{h_k} - \frac{m_k h_k}{6} \qquad (3.13)$$

$$s_k(x_{k+1}) = y_{k+1} = \frac{m_{k+1}}{6h_k}(x_{k+1} - x_k)^3 + q_k(x_{k+1} - x) \quad \Rightarrow \quad q_k = \frac{y_{k+1}}{h_k} - \frac{m_{k+1} h_k}{6} \qquad (3.14)$$

Donc :

$$\begin{aligned} s_k(x) &= \frac{m_k}{6h_k}(x_{k+1} - x)^3 + \frac{m_{k+1}}{6h_k}(x - x_k)^3\\ &+ \left(\frac{y_k}{h_k} - \frac{m_k h_k}{6}\right)(x_{k+1} - x) + \left(\frac{y_{k+1}}{h_k} - \frac{m_{k+1} h_k}{6}\right)(x - x_k) \qquad (3.15) \end{aligned}$$

Les seules inconnues qui restent dans les équations sont les m_k. La seule contrainte non utilisée est la continuité des dérivés premières (3.4). On peut déterminer les m_k en utilisant les dérivés premières :

$$s'_k(x) = -\frac{m_k}{2h_k}(x_{k+1}-x)^2 + \frac{m_{k+1}}{2h_k}(x-x_k)^2 - (\frac{y_k}{h_k} - \frac{m_k h_k}{6}) + (\frac{y_{k+1}}{h_k} - \frac{m_{k+1}h_k}{6}) \tag{3.16}$$

Évaluons $s'_k(x_k)$ et $s'_{k-1}(x_k)$:

$$\begin{aligned} s'_k(x_k) &= -\frac{m_k h_k}{2} - \Big(\frac{y_k}{h_k} - \frac{m_k h_k}{6}\Big) + \Big(\frac{y_{k+1}}{h_k} - \frac{m_{k+1}h_k}{6}\Big) \\ &= -\frac{1}{3}m_k h_k - \frac{1}{6}m_{k+1}h_k + d_k \qquad avec \qquad d_k = \frac{y_{k+1}-y_k}{h_k} \qquad (3.17) \\ s'_{k-1}(x_k) &= \frac{m_k h_{k-1}}{2} - \frac{y_{k-1}}{h_{k-1}} + \frac{m_{k-1}h_{k-1}}{6} + \frac{y_k}{h_{k-1}} - \frac{m_k h_{k-1}}{6} \\ &= \frac{1}{3}m_k h_{k-1} + \frac{1}{6}m_{k-1}h_{k-1} + d_{k-1} \qquad (3.18) \end{aligned}$$

Ensuite, en éxigeant la continuité des dérivées premières, $s'_k(x_k) = s'_{k-1}(x_k)$:

$$-\frac{1}{3}m_k h_k - \frac{1}{6}m_{k+1}h_k + d_k = \frac{1}{3}m_k h_{k-1} + \frac{1}{6}m_{k-1}h_{k-1} + d_{k-1} \tag{3.19}$$

Et en simplifiant :

$$h_{k-1}m_{k-1} + 2(h_{k-1}+h_k)m_k + h_k m_{k+1} = 6(d_k - d_{k-1}) \tag{3.20}$$

Ce qui donne un système linéaire avec $(n-1)$ équations qui peut être résolu pour les m_k $(n+1$ inconnues). En spécifiant les conditions limites, les coefficients du spline peuvent alors être déterminés.

En divisant par $h_{k-1}+h_k$ et en définissant $\lambda_k = \frac{h_k}{h_{k-1}+h_k}$ et $\rho_k = 1-\lambda_k$, on obtient :

$$\rho_k m_{k-1} + 2m_k + \lambda_k m_{k+1} = 6\frac{d_k - d_{k-1}}{h_{k-1}+h_k} \tag{3.21}$$

Notons qu'il y a $(n+1)$ inconnues $(m_k, k = 0, 1, \cdots, n)$ pour $(n-1)$ équations. Il faut donc spécifier m_0 et m_n.

On a alors un système linéaire $\mathbf{Mm} = \mathbf{b}$, tel que :

$$\begin{pmatrix} 2 & \lambda_1 & 0 & 0 & \cdots & 0 \\ \rho_2 & 2 & \lambda_2 & 0 & \cdots & 0 \\ 0 & \rho_3 & 2 & \lambda_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \rho_{n-2} & 2 & \lambda_{n-2} \\ 0 & \cdots & 0 & 0 & \rho_{n-1} & 2 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ \vdots \\ m_{n-3} \\ m_{n-2} \\ m_{n-1} \end{pmatrix} = \begin{pmatrix} 6\frac{d_1-d_0}{h_0+h_1} - \rho_1 m_0 \\ 6\frac{d_2-d_1}{h_1+h_2} \\ \vdots \\ \vdots \\ 6\frac{d_{n-2}-d_{n-3}}{h_{n-3}+h_{n-2}} \\ 6\frac{d_{n-1}-d_{n-2}}{h_{n-2}+h_{n-1}} - \lambda_{n-1}m_n \end{pmatrix}$$

La matrice $\mathbf{M}$ est appelée la matrice de poids.

Maintenant, reprenons l'équation 3.1 rappelée ci-dessous :

$$s_k(x) = s_{k,0} + s_{k,1}(x-x_k) + s_{k,2}(x-x_k)^2 + s_{k,3}(x-x_k)^3 \qquad x \in [x_k, x_{k+1}]$$

Une fois que le système est résolu pour les m_k, le spline peut être construit avec les coefficients suivants :

$$s_{k,0} = y_k \tag{3.22}$$

$$s_{k,1} = d_k - \frac{h_k}{6}(2m_k + m_{k+1}) \tag{3.23}$$

$$s_{k,2} = \frac{m_k}{2} \tag{3.24}$$

$$s_{k,3} = \frac{m_{k+1} - m_k}{6h_k} \tag{3.25}$$

3.1.2 Les conditions limites du spline cubique

Il y a plusieurs possibiliés pour déterminer les conditions limites m_0 et m_n :

- Calculées : les dérivées sont calculées par l'utilisateur en définissant une fonction f tel que $s'(x) \equiv f'(x)$;
- Spline naturel : les dérivées secondes sont nulles aux frontières ;
- not-a-knot : les dérivées troisièmes sont nulles aux frontières.

Les conditions limites "not-a-knot" donnent les plus petites erreurs quand les conditions limites sont inconnues. Avec comme conséquence que les deux premiers et les deux derniers coefficients du spline cubique sont alors identiques ($s_{0,3} = s_{1,3}$ et $s_{n-2,3} = s_{n-1,3}$).

3.2 L'interpolation verticale utilisée

L'interpolation verticale est donc, dans le cas général, un spline cubique. Les routines utilisées sont issues du NUMERICAL RECIPES IN FORTRAN 77[1] avec une modification dans la recherche des points. En effet, les grilles verticales utilisées ne varient pas et sont monotones. Il est donc facile de calculer, au préalable, les points sources entourant le point cible.

Les conditions limites sont des splines naturels. Les dérivées secondes sont donc nulles.

La valeur des points cibles en surface est fixée à celle des points sources du premier niveaux.

Le spline est utilisé pour extrapoler les points en profondeur si nécessaire.

[1]NUMERICAL RECIPES IN FORTRAN 77 : THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X) by Cambridge University Press pages 109 et 110

Chapitre 4

Présentation de l'outils d'interpolation

4.1 Utilisation

Cet outils d'interpolation permet d'interpoler des champs d'une grille ORCA 2 degrés vers une grille régulière telle que définie dans le projet européen ENACT.

Les longitudes de la grille cible varient donc de 0 à 359 degrés par incrément de 1 degré.
Les latitudes de la grille cible varient de -89 à 89 degrés par incrément de 1 degré.
La résolution verticale est la même que celle de Levitus, avec 33 niveaux aux profondeurs suivantes : 0, 10, 20, 30, 50, 75, 100, 125, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1750, 2000, 2500, 3000, 3500, 4000, 4500, 5000 et 5500 mètres.

Cet outils permet d'interpoler des champs bi- ou tri-dimensionnels.
Il fabrique un fichier NetCDF respectant la norme CF-1.0.

Il s'utilise en exécutant la commande suivante :

```
interp.ksh -i input file -o output file -v variable -m mask -p code
```

4.2 Description des options

Quand la commande "interp.ksh" est tapée sans option, les arguments possibles sont affichés de la manière suivante :

```
Usage:
interp.ksh
         -v variable name
         -i input filename (netcdf file on orca grid)
         -o output filename (netcdf file on ENACT grid)
Optional :
         -m mask of the variable (Default tmask)
         -p code of the variable (cf ENACT conventions)
```

Il faut donc nécessairement spécifier le nom du fichier d'entrée sur la grille ORCA2, le nom du fichier de sortie et le nom de la variable à interpoler.

Deux options sont optionnelles.

L'option -m permet de spécifier le mask à utiliser pour la grille ORCA2. En effet, les longitudes, latitudes et donc le mask changent si la variable est sur la grille "T", "U" ou "V". Si la variable est sur la grille "W", les longitudes, latitudes et le mask sont les mêmes que pour la grille "T". Par contre, la grille verticale change. Ainsi les différents masks disponibles sont "tmask", "umask", "vmask" et "wmask". Si cette option n'est pas spécifiée, la variable est supposée être sur la grille "T".

L'option -p permet de donner le code de la variable tel qu'il est défini dans la nomenclature du projet européen ENACT. En spécifiant cette option, la variable sera renommée et son nom long et son unité seront spécifiés.

Quelques codes pour les variables les plus courantes :
- température : 129 ;
- salinité : 130 ;
- vitesse zonale : 131 ;
- vitesse méridienne : 132 ;
- vitesse verticale : 133 ;
- hauteur de mer : 145 ;
- température de surface : 159.

4.3 Limitations

Cet outils ne traite que les fichiers NetCDF en simple ou double précision.

Il ne permet d'interpoler qu'une variable à la fois et crée un nouveau fichier. Si le fichier existe déjà, il est écrasé.

Cet outils peut fonctionner sur toutes les machines disposant du Korn Shell, d'un compilateur Fortran 90 et d'une version d'OASIS.

Il ne permet d'interpoler que des champs provenant de la grille ORCA2 vers la grille régulière décrite plutôt. Il nécessite en effet en entrée la grille source et cible, ainsi que celle intermédiaire qui a les mêmes longitudes et latitudes que la grille cible et les mêmes niveaux verticaux que la grille source. Il nécessite aussi les masks pour chaque niveaux de la grille source, cible et intermédiaire.

Il est donc possible d'adapter cet outils à condition de fabriquer ces fichiers d'entrées et surtout ceux concernant la grille intermédiaire qui ne sont pas toujours disponibles.

4.4 Exemple d'en-tête NetCDF obtenu

Voici l'en-tête d'un fichier contenant la vitesse zonale :

```
netcdf U {
dimensions:
        longitude = 360 ;
        latitude = 179 ;
        depth = 33 ;
variables:
        float sea_water_x_velocity(depth, latitude, longitude) ;
                sea_water_x_velocity:valid_min = -2.097091f ;
                sea_water_x_velocity:valid_max = 1.307023f ;
                sea_water_x_velocity:data_type = "float" ;
                sea_water_x_velocity:standard_name = "sea_water_x_velocity" ;
```

```
                sea_water_x_velocity:long_name = "Zonal Current" ;
                sea_water_x_velocity:units = "m.s-1" ;
                sea_water_x_velocity:axis = "XYZ" ;
                sea_water_x_velocity:_FillValue = 9.999e+33f ;
        float longitude(longitude) ;
                longitude:data_type = "float" ;
                longitude:standard_name = "longitude" ;
                longitude:long_name = "longitude" ;
                longitude:units = "degrees_east" ;
                longitude:axis = "Y" ;
                longitude:positive = "east" ;
                longitude:valid_min = 0.f ;
                longitude:valid_max = 359.f ;
                longitude:missing_value = 9.e+19f ;
        float latitude(latitude) ;
                latitude:data_type = "float" ;
                latitude:standard_name = "latitude" ;
                latitude:long_name = "latitude" ;
                latitude:units = "degrees_north" ;
                latitude:axis = "X" ;
                latitude:positive = "north" ;
                latitude:valid_min = -89.f ;
                latitude:valid_max = 90.f ;
                latitude:missing_value = 9.e+19f ;
        float depth(depth) ;
                depth:data_type = "float" ;
                depth:standard_name = "depth" ;
                depth:long_name = "depth" ;
                depth:units = "m" ;
                depth:axis = "Z" ;
                depth:positive = "down" ;
                depth:valid_min = 0.f ;
                depth:valid_max = 5500.f ;
                depth:missing_value = 9.e+19f ;

// global attributes:
                :title = "ENACT" ;
                :conventions = "CF-1.0" ;
                :institution = "CERFACS" ;
                :model = "OPA" ;
                :method = "3dvar" ;
                :method_version = "OPA8-2_VAR2-2" ;
                :references = "http://www.ecmwf.int/research/EU_projects/ENACT" ;
}
```

4.5 Exemples de champs interpolés

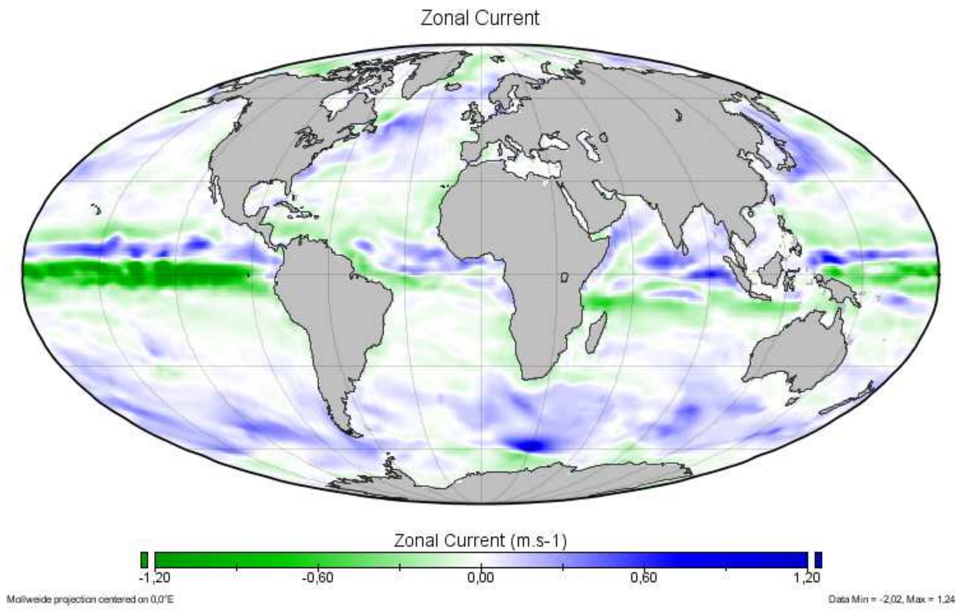

FIG. 4.1 – Vitesse zonale

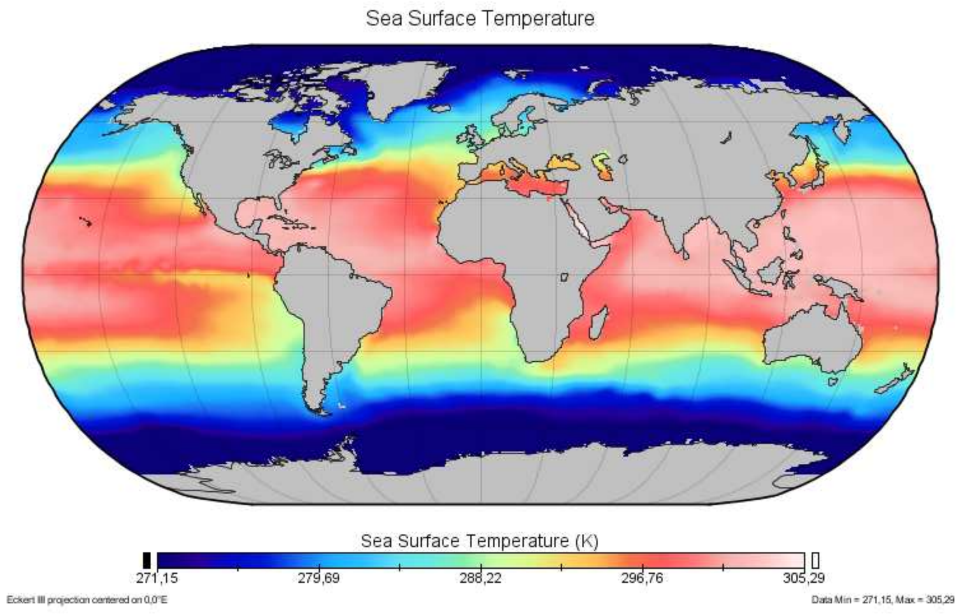

FIG. 4.2 – Température de surface

Chapitre 5

Conclusion

Ce nouvel outils d'interpolation permet de faire les interpolations en s'affranchissant d'IDL. Il est maintenant possible de faire les interpolations de n'importe quel ordinateur sous Unix ou Linux.

De plus, les schémas d'interpolation ont été améliorés avec l'utilisation d'une interpolation bicubique.

Les sources sont propres et commentées et l'écriture du script Korn Shell a été soigné afin de traiter correctement les erreurs et les arguments.