Centre de recherche fondamentale et appliquée spécialisé dans
la modélisation et la simulation numériques

Rapport de stage

Master 1

Evaluation et comparaison des interpolations numériques des librairies SCRIP et ESMF

Working Notes Cerfacs WN-CMGC-16-227

Auteur :
Hakim SENHAJI
Etudiant en deuxième année
d'école d'ingénieurs

Maître de stage
Sophie VALCKE
Ingénieur de recherche

Résumé

L'objectif de cette étude est d'évaluer et de comparer la qualité des interpolations numériques entre les librairies d'interpolation SCRIP et ESMF. La comparaison se fait pour deux configurations principales : "avec" et "sans" recherche du plus proche voisin. La première partie du travail a consisté à coder les configurations précédentes pour la SCRIP (en Fortran) et pour ESMF (avec NCL), en définissant une erreur d'interpolation. Ensuite on a mené une batterie de tests pour la SCRIP et ESMF, pour différentes interpolations (bilinéaire, bicubique, linéaire et conservative), pour nos deux configurations de recherche ou non du plus proche voisin. On a affiché les résultats de manière à comparer facilement les résultats de la SCRIP et ESMF, et de surcroît nous avons affiché l'écart entre les résultats de la SCRIP et ESMF (en soustrayant les résultats des deux librairies). Les résultats pour les interpolations entre grilles structurées sont directs, par contre un travail de remaillage supplémentaire doit être entrepris pour les interpolations impliquant des grilles non-structurées. Cette étape est explicitée à la fin du rapport.

Abstract

The objective of this study is to evaluate and compare the quality of the computational interpolation between the interpolation libraries : SCRIP and ESMF. The comparison was done for two main configurations : "with" and "without" search for the nearest neighbor. The first part of the work consisted in coding the previous configurations for the SCRIP (in Fortran) and ESMF (with NCL), through defining an interpolation error. Then we conducted a set of tests for SCRIP and ESMF, for various interpolations (bilinear, bicubic, linear and conservative), for both with and without nearest neighbor search. The results are displayed so that we can easily compare the results of the SCRIP and ESMF, and we moreover featured the gap between the results of the SCRIP and ESMF (by subtracting the results of the two libraries). The results of the interpolations between structured grids are automatic, however an additional work have been done to carry out interpolations involving unstructured grids. This step is explained at the end of the report.

Sujet du stage

Evaluer et comparer la qualité des interpolations numériques, pour les librairies d'interpolation SCRIP et ESMF. Cette étude devra se faire pour plusieurs configurations, pour des grilles structurées et non-structurées.

Informations sur le stage

Ce stage est un stage technique de deuxième année d'école d'ingénieurs qui a duré 3 mois (60 jours de présence effective). Les résultats de ce stage sont présentés sur trois supports : ce rapport, des répertoires de travail enregistrés sous svn (sur une machine du CERFACS) et un "wiki" complémentaire.

Présentation du CERFACS

Le CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique) est un centre de recherche dont le rôle est de développer des méthodes de calcul avancées pour les simulations numériques et la résolution algorithmique de problèmes scientifiques et technologiques aussi bien pour la recherche que pour l'industrie. Le CERFACS est dirigé par un Conseil de Gérance constitué par les représentants de ses actionnaires, et un Conseil Scientifique qui préconise les orientations à suivre. Les sept actionnaires du CERFACS sont : le CNES (Centre National d'Etudes Spatiales), AIRBUS, EDF (Electricité de France), Météo-France, l'ONERA (Office National d'Etudes et de Recherches Aérospatiales), SAFRAN (groupe international de haute technologie), TOTAL (multinationale dans le domaine de l'énergie).

Le CERFACS regroupe plusieurs équipes interdisciplinaires, à la fois pour la recherche et pour la formation, dans les domaines suivants : physique, mathématiques appliquées, analyse numérique, développement de logiciels. Environ 115 personnes y travaillent, dont plus de 95 chercheurs et ingénieurs venant de dix pays différents. Ils travaillent sur des projets spécifiques dans neuf grands domaines de recherche : l'algorithmique parallèle, le couplage de codes, l'aérodynamique, les écoulements dans les turbines, la combustion, l'électromagnétisme, l'impact environnemental, l'assimilation de données et le climat.

Table des matières

1 Introduction et défintions

A l'heure où le changement climatique est tenu pour être l'une des menaces les plus sérieuses pesant sur l'environnement, il est nécessaire de disposer des outils scientifiques adaptés pour le modéliser et le simuler, afin de l'assimiler et prévoir ses conséquences. En effet, la compréhension de notre système climatique fait partie des domaines où le seul outil d'analyse et de compréhension est la modélisation, la réalisation d'expériences sur la Terre elle-même étant impossible.

1.1 Couplage numérique et coupleur OASIS

L'approche optimale pour simuler le climat terrestre consiste à développer des modèles de circulation générale pour l'atmosphère et l'océan. Or on sait que les équations qui régissent les phénomènes atmosphériques et océaniques sont couplées, d'où l'intérêt de construire un coupleur numérique qui permet de mettre en place les différentes interactions entre le modèle atmosphérique et océanique. Le coupleur utilisé dans notre étude est le **coupleur OASIS** ; c'est un logiciel développé par le CERFACS depuis 1991 qui permet de coupler des codes numériques représentant les diverses composantes du système climatique terrestre (océan, atmosphère, glace de mer, surfaces continentales, ...), c'est-à-dire d'échanger de l'information (des "champs de couplage") de façon synchronisée à l'interface de ces composantes. **OASIS** ("Ocean Atmosphere Sea Ice Soil") permet d'effectuer des couplages statiques (dans le sens où toutes les composantes doivent s'exécuter du début à la fin de la simulation) en assurant toutes les transformations requises pour exprimer sur le maillage des composantes cibles les champs de couplage fournis par les composantes sources sur son maillage.

1.2 L'interpolation

Les interactions entre l'océan et l'atmosphère se traduisent, dans les modèles, par des échanges de surface intervenant au premier niveau des grilles océanique et atmosphérique. En général, les modèles de circulation générale atmosphérique et océanique possèdent des pas de temps différents, des grilles spatiales différentes, ainsi que des définitions des lignes de côtes qui ne coïncident pas, d'où la nécessité de faire des **interpolations**. L'interpolation consiste à déterminer, à partir des valeurs d'un champ sur la grille source (océanique par exemple), une approximation de sa valeur en tout point de la grille cible (atmosphérique par exemple).
Les interpolations utilisées dans cette étude sont les suivantes :

- **Interpolation bilinéaire :** c'est une méthode d'interpolation pour les fonctions de deux variables. Elle permet de calculer la valeur d'une fonction en un point quelconque, à partir de ses deux plus proches voisins dans chaque direction (quatre voisins).
- **Interpolation bicubique :** c'est une méthode d'interpolation qui contrairement à l'interpolation bilinéaire, considère un voisinage de 16 valeurs pour les quatre voisins (on utilise en plus la dérivée dans les deux directions et les dérivées croisées).
- **Interpolation linéaire :** c'est une méthode d'interpolation qui consiste à rechercher pour chaque point cible, l'information à partir de son plus proche voisin sur la grille source.
- **Interpolation conservative :** c'est une méthode qui permet d'assurer la conservation de l'énergie et de l'eau du système climatique couplé. C'est une méthode basée sur la fraction d'aire intersectée entre les mailles source et cible.

1.2.1 Définition de l'erreur d'interpolation

Pour effectuer ces interpolations, nous disposons de deux librairies : la **SCRIP** et **ESMF**. La SCRIP fonctionne avec OASIS qui lui même contient des programmes codés en Fortran 90, alors que ESMF fonctionne avec des programmes codés en langage NCL.

L'objectif de cette étude est d'évaluer et de comparer la qualité des différentes interpolations entre la librairie SCRIP et ESMF. Pour tester la qualité d'une interpolation entre la grille source et la grille cible, nous avons défini une erreur d'interpolation (ε_{interp}) avec la formule suivante :

$$\varepsilon_{interp} = \left| \frac{f_{recu} - f_{analytique}}{f_{recu}} \right|$$

Où f_{recu} est le champ défini par une fonction analytique (sur une grille source) interpolé sur une grille cible, et $f_{analytique}$ est le champ défini par la même fonction analytique en chaque point de la grille cible. L'interpolation est parfaite si l'erreur d'interpolation est nulle et par conséquent si le champ interpolé est confondu avec le champ défini par la fonction analytique.

1.2.2 Le champ reçu pour la SCRIP et ESMF

Pour la SCRIP le champ reçu est le champ résultant de la compilation d'OASIS qui procède, via la SCRIP, à l'interpolation du champ analytique (calculé pour la grille source) sur la grille cible. Ce champ reçu est comparé au champ analytique calculé sur les points de la grille cible.

Pour ESMF le raisonnement est le même que pour la SCRIP, sauf que les programmes d'interpolations et de calculs d'erreur et de champ sont écrits en langage NCL. Et le logiciel OASIS est substitué par la fonction `ESMF_regrid` qui effectue les interpolations.

1.3 Les grilles

Les grilles utilisées dans notre étude sont les suivantes, et recouvrent en fait le globe terrestre :

Nom de la grille	Type de grille	Taille de la grille
teo1	Rectilinéaire	294 x 362
bggd	Régulière lon-lat	144 x 143
ssea	Gaussienne réduite	24472
bt42	Gaussienne réduite	6232

La grille **teo1** est une grille rectilinéaire avec 2 pôles de convergence au nord ramenés sur les continents et un pôle de convergence au sud. La grille **bggd**, elle, est une grille régulière, c'est à dire que toutes les mailles ont la même taille dans l'espace longitude-latitude. En revanche la grille **ssea** est une grille gaussienne réduite non structurée où il y a de moins en moins de mailles quand on avance vers les pôles.

Zones masqués : Chaque grille contient des zones appelées "zones masquées" en lesquelles les champs échangés par le modèle n'ont pas une valeur "valide". Par exemple pour une grille océanique, la terre est considérée comme un masque. Puisque les interpolations étudiées concernent les modèles océan-atmosphère, les zones masquées sont les continents.

1.4 Interpolation avec et sans plus proche voisin additionnel non-masqué

L'option avec plus proche voisin additionnel non-masqué est souvent abrégée dans la suite du document en "nnei", et l'option sans est souvent abrégée en "snnei". Pour effectuer une interpolation, certains points cible **non-masqués** ont des points voisins **masqués**. Pour accéder à "l'information" (au champ) pour ces points, il faut utiliser l'option avec plus proche voisin additionnel non-masqué, mais cette option, n'est pas appliquée de la même manière, qu'on soit sur la SCRIP ou ESMF (quand on parle de points dans la suite on sous-entend points cible non-masqués) :

Pour la SCRIP :
La configuration "nnei" donne des résultats différents de la configuration "snnei", uniquement pour les points qui ont leurs **quatre** voisins originaux masqués. C'est uniquement dans ce cas (quatre voisins masqués) que la SCRIP va chercher un nouveau point voisin non-masqué pour l'option "nnei". En effet, il suffit qu'il y est un seul point voisin non masqué, pour que la SCRIP l'utilise pour faire l'interpolation. S'il y a plus d'un point voisin non-masqué, la SCRIP fait une moyenne sur ces points non-masqués, pour faire son interpolation.
Si un point a ses quatre voisins originaux masqués, on dit en configuration "snnei" **qu'il ne reçoit pas de valeur**.

Pour ESMF :
Les quatre points voisins originaux doivent être non-masqués pour effectuer une interpolation. Dans la configuration "nnei", la recherche du plus proche voisin additionnel non-masqué se fait pour les points qui ont au moins un voisin original masqué. S'il y a plus d'un voisin non-masqué, ESMF choisit un des points voisins non-masqués pour faire l'interpolation. La configuration "snnei" ne donne donc des valeurs de champ, que pour les points qui ont leur quatre voisins originaux non-masqués.
En configuration "snnei" toujours, si un point a au moins un voisin original masqué, on dit **qu'il ne reçoit pas de valeur**.

2 Méthode de calcul de l'erreur et du champ

2.1 Pour la SCRIP

2.1.1 Introduction

Tous les programmes utilisés pour toutes les interpolations SCRIP avec le coupleur OASIS sont écrits en Fortran 90 `model1.f90` pour le modèle source et `model2.f90` pour le modèle cible. Pour désactiver la recherche du plus proche voisin additionnel, il faut aller dans le programme OASIS et précisément dans la section qui nous intéresse (par exemple `remap_bili.f`) et mettre l'option `ll_nnei` à `false`. Le répertoire tests.../SCRIP.. est séparé en deux parties `avec_nnei` et `sans_nnei` qui compile automatiquement le code OASIS avec l'option avec ou sans plus proches voisins. Dans la section suivante on présente la manière dont est calculée l'erreur quand on active ou pas la recherche du plus proche voisin additionnel non-masqué.

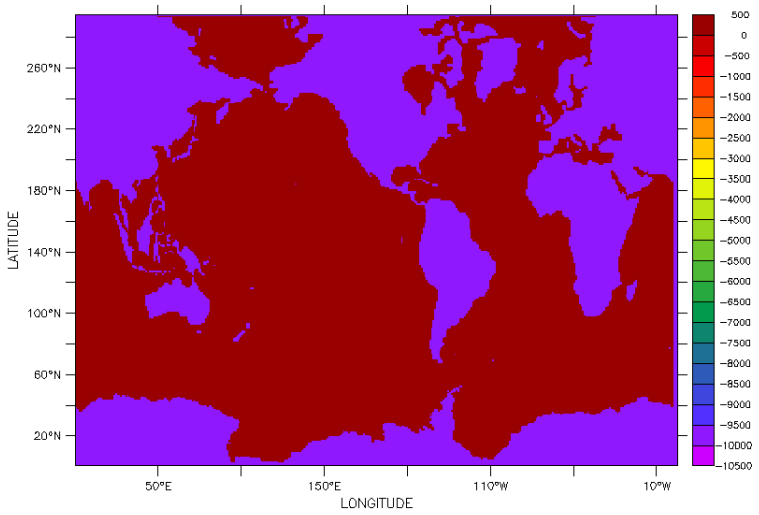
2.1.2 Consignes avec plus proche voisin additionnel non-masqué (nnei)

Calcul de l'erreur

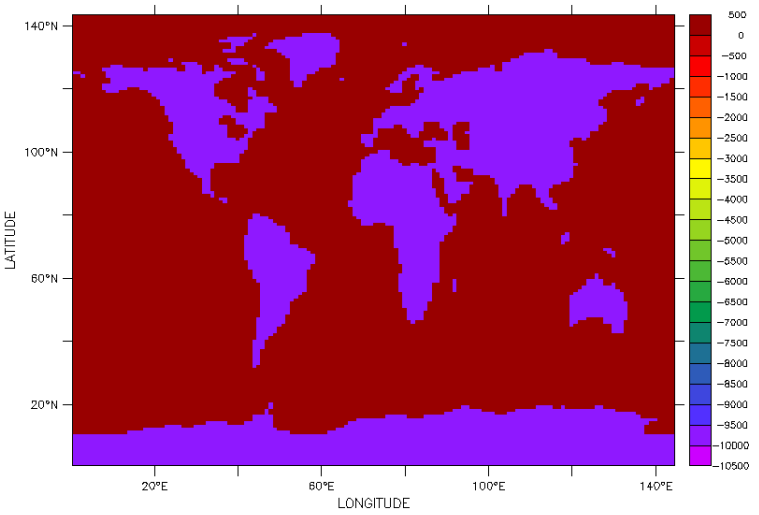
On rappelle que dans cette configuration, tous les points cibles non-masqués reçoivent une valeur. Ce qu'on fait pour la SCRIP avec plus proches voisins, c'est qu'on calcule l'erreur sur les zones non masquées et on la fixe à -10000 sur les zones masquées. Le champ reçu lui n'est pas modifié. Ci-dessous les lignes de code en Fortran tirés de `model2.F90` correspondant à cette consigne (où `err_msk=-10000`) :

```
error=err_msk
!
WHERE (RESHAPE(gg_mask(indi_beg:indi_end,indj_beg:indj_end),(/ var_sh(2), var_sh(4) /)) == 0)
error = ABS(((field_ana - field_recv)/field_recv))*100
END WHERE
```

Voici ce qu'on obtient concrètement pour l'erreur dans les deux sens d'interpolation (de bggd vers teo1 et de teo1 vers bggd) :



(a) Erreur représentative à grande échelle bggd-teo1-scrip



(b) Erreur représentative à grande échelle teo1-bggd-scrip

Ici on a laissé l'échelle de couleur s'ajuster automatiquement, on voit donc bien que les points masqués sont à -10000.

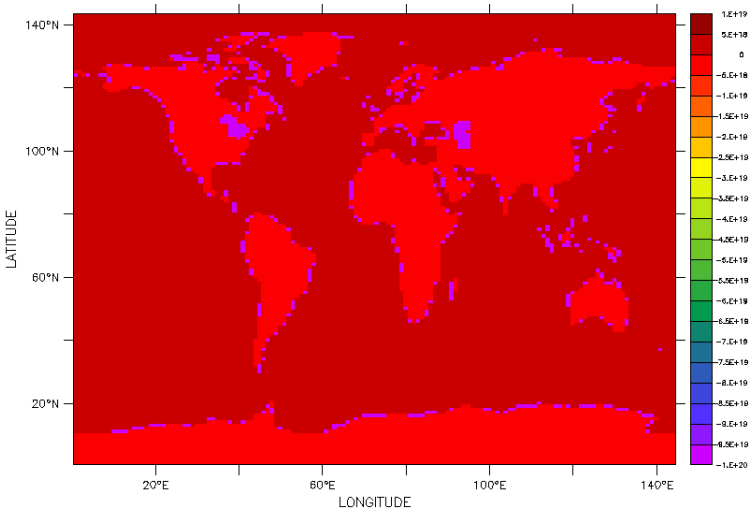
2.1.3 Consignes sans plus proche voisin

Calcul de l'erreur

On rappelle que certains points non-masqués peuvent ne pas recevoir de valeur ; en l'occurrence au contour des continents (là où tous les points ont des voisins masqués). Voici les différentes modifications opérées pour obtenir l'erreur sans plus proches voisins :

1. On demande au programme de calculer l'erreur, là où le champ n'est pas masqué, tout en retranchant à cette zone les points qui ne reçoivent pas de valeur, c'est à dire en considérant, en plus, uniquement les zones où le champ reçu n'est pas nul `WHERE (field_recv /= 0.)`. Ceci nous évite de tomber sur une erreur infinie due à une division par un champ reçu nul (cf. expression de l'erreur).
2. Sur les zones masquées, on met l'erreur à -10000.
3. Sur les points qui ne reçoivent pas de valeur, on met l'erreur à -1e20 (pour pouvoir bien les identifier).

Voici un exemple de ce qu'on obtient concrètement pour l'erreur pour l'interpolation bilinéaire de la grille teo1 vers bggd :



(a) Erreur représentatif à grande échelle
teo1-bggd-scrip

On voit bien que les points non-masqués qui ne reçoivent pas de valeur (en violet sur le contour des continents) ont une erreur forcée à -1e20. Et que les points masqués (rouge pâle : sur les continents) ont une erreur forcée à -10000. Et enfin, les points qui reçoivent une valeur (en rouge foncé : sur l'océan) ont une erreur supérieur à 0. Ci-dessous les lignes de code en Fortran tirées de `model2.F90` définissant cette consigne :

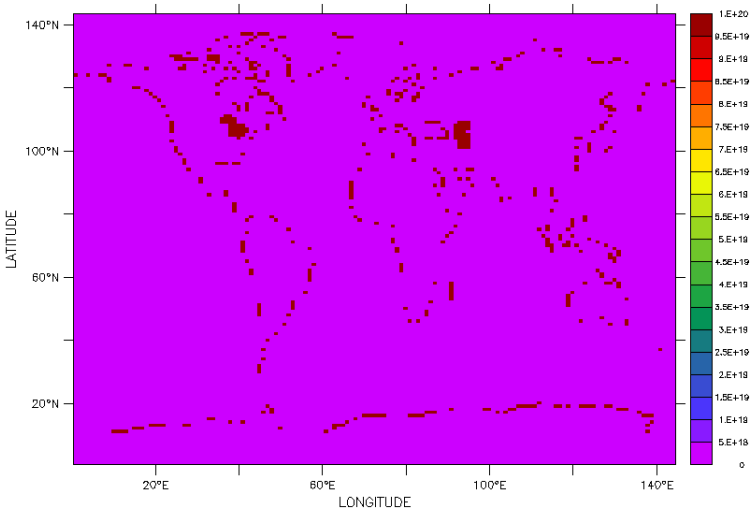
```
error=-10000
!
WHERE (RESHAPE(gg_mask(indi_beg:indi_end,indj_beg:indj_end),(/ var_sh(2), var_sh(4) /)) == 0)
WHERE (field_recv /= 0.)
error = ABS(((field_ana - field_recv)/field_recv))*100
END WHERE
END WHERE
WHERE (RESHAPE(gg_mask(indi_beg:indi_end,indj_beg:indj_end),(/ var_sh(2), var_sh(4) /)) == 1)
field_recv=10000.
END WHERE
!
temp=field_recv
WHERE (field_recv == 0.)
temp=1.e20
error=-1.e20
END WHERE
field_recv=temp
WHERE (RESHAPE(gg_mask(indi_beg:indi_end,indj_beg:indj_end),(/ var_sh(2), var_sh(4) /)) == 1)
field_recv=0.
END WHERE
```

Modification du champ

Sur l'extrait du programme Fortran précédent, on distingue aussi les modifications amenées au champ reçu. Ci-dessous les différentes manipulations effectuées :

1. Le champ est égal au champ interpolé sur les zones non-masquées.
2. Sur les zones masquées le champ est fixé à 10000 (l'erreur est initialisée en fait à -10000), pour permettre d'isoler les points qui ne reçoivent pas de valeurs (cf. $4^{ème}$ point).
3. On met le champ à 1e20, là où il ne reçoit pas de valeur.
4. Après les manipulations précédentes et uniquement après, on fixe finalement le champ reçu à 0 sur les zones masquées (pour être conforme aux réglages effectués sur ESMF pour la même configuration).

Voici ce qu'on obtient concrètement pour le champ pour l'interpolation bilinéaire de la grille teo1 vers bggd :



(a) Champ reçu représentatif à grande échelle
teo1-bggd-scrip

Ici on a laissé l'échelle de couleur s'ajuster automatiquement, et donc on voit bien que les points non-masqués qui ne reçoivent pas de valeur sont en rouge foncé (à 1e20).

2.2 Pour ESMF

2.2.1 Introduction

Pour tester les interpolations avec ESMF, on appelle les fonctions correspondantes "en langage NCL". Donc pour ESMF, il faut modifier le fichier NCL qui fait les interpolations et le *regriding* de la grille source. Dans la section suivante on présente la manière dont est calculée l'erreur avec plus proches voisins (le champ reçu est prédéfini en l'occurrence `field_ou`), puis la manière dont est calculée l'erreur mais aussi le champ sans activer la recherche du plus proche voisin additionnel non-masqué.

2.2.2 Consignes avec plus proche voisin additionnel non-masqué

Calcul de l'erreur

On rappelle une nouvelle fois que tous les points non-masqués reçoivent une valeur dans cette configuration (avec plus proches voisins). Le champ lui n'est pas modifié. On effectue ici les même manipulations que pour la SCRIP. Ci-dessous l'extrait du programme NCL, qui fait le calcul d'erreur :

```
msk_field_value=10000
msk_err_value=-10000

func_ana_tg_msk
= where((tgt_msk_interp) .eq. 1, func_ana_tgt, msk_field_value)

error = where((tgt_msk_interp) .eq. 1,
abs(((func_ana_tgt_msk -func_regrid_msk_new)/func_regrid_msk_new))*100, msk_err_value)
```

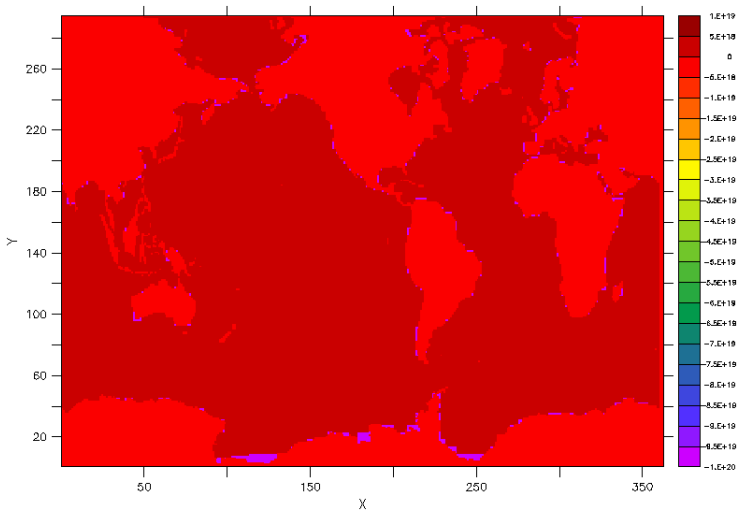
Ceci veut dire qu'on calcule l'erreur sur tous les points non-masqués et on fixe l'erreur à -10000 sur les points masqués.

2.2.3 Consignes sans plus proche voisin

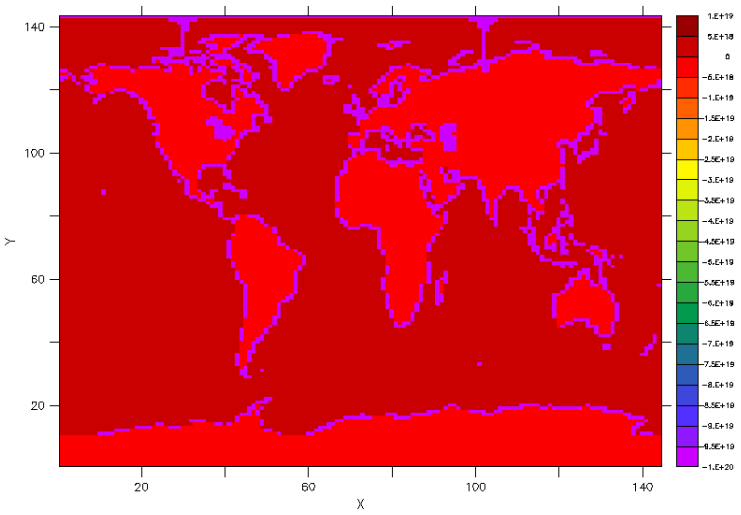
Calcul d'erreur

On rajoute dans le fichier NCL, "qui fait les interpolations", une nouvelle erreur "sans plus proche voisins" (erreur snnei), définie de la même manière que la SCRIP :

Voici ce qu'on obtient concrètement pour l'erreur dans les deux sens d'interpolation (ie. de bggd vers teo1 et de teo1 vers bggd) :



(a) Erreur représentative à grande échelle bggd-teo1-esmf



(b) Erreur représentative à grande échelle teo1-bggd-esmf

On voit bien que les points non-masqués qui ne reçoivent pas de valeur (en violet sur le contour des continents) ont une erreur forcée à -1e20. Comme on pouvait s'y attendre ces points (cf. figure (b)) sont plus nombreux que pour la SCRIP (cf. figure (a) page 8 section SCRIP). Les points masqués, eux (rouge pâle : sur les continents), ont une erreur forcée à -10000. Et enfin, les points qui reçoivent une valeur (en rouge foncé : sur l'océan) ont une erreur supérieur à 0.
Ci-dessous, l'extrait du programme NCL, qui fait ce réglage :

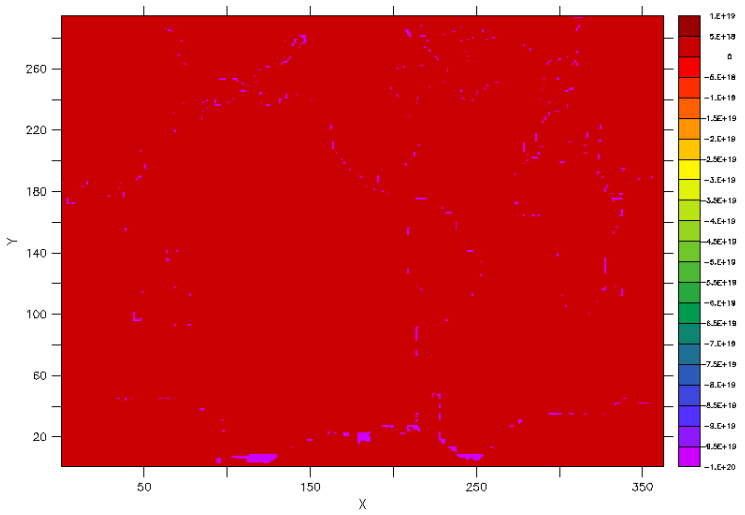
```
msk_ismissing=-1e20
msk_err_value=-10000
msk_field_ou=0
error_ismissing=-1e20

func_ana_tgt_msk = where((tgt_msk_interp) .eq. 1, func_ana_tgt, msk_field_value)
error_snnei = where((tgt_msk_interp) .eq. 1 .and. (.not.ismissing(func_regrid_msk_fillvalue)),
abs(((func_ana_tgt_msk -func_regrid_msk_fillvalue)/func_regrid_msk_fillvalue))*100,msk_err_value)

error_snnei = where(ismissing(func_regrid_msk_fillvalue),error_ismissing,error_snnei)

func_regrid_msk_fillvalue_snnei = where((tgt_msk_interp) .eq. 0, msk_field_ou,
func_regrid_msk_fillvalue)

func_regrid_msk_fillvalue_snnei = where(ismissing(func_regrid_msk_fillvalue),msk_ismissing,
func_regrid_msk_fillvalue_snnei)
```

On a remplacé ici la fonction `func_regrid_new` qui représentait le champ reçu avec l'option plus proche voisin par la fonction `func_regrid_msk_fillvalue` qui remplace les points qui ne reçoivent pas de valeurs par `FillValue` et qui est le champ reçu par définition. On calcule ensuite l'erreur sur les zones masquées (`tgt_msk_interp .eq. 1`) tout en retranchant les zones qui ne reçoivent pas de valeurs `ismissing(func_regrid_msk_fillvalue)`, puis en fixant l'erreur à -10000 pour les zones restantes. Enfin les points qui ne reçoivent pas de valeur, sont fixés à -1e20.

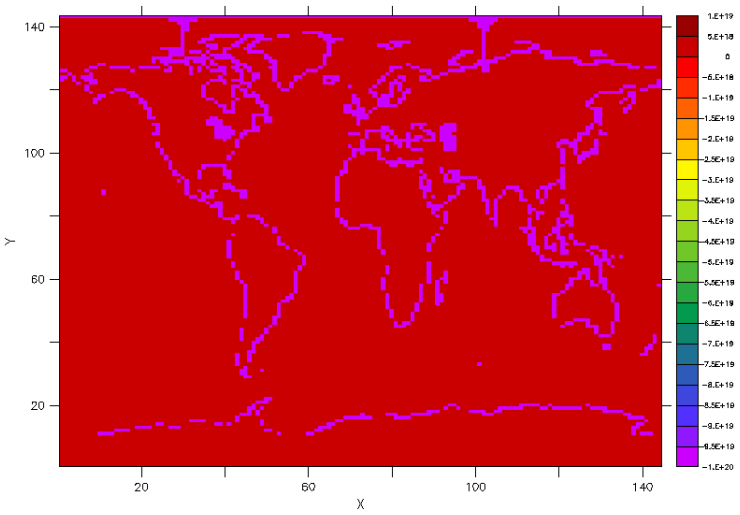
Calcul de champ :
Le champ reçu est `func_regrid_msk_fillvalue_snnei` :

1. On le met à zéro sur les zones masquées.
2. On calcule le champ interpolé sur les zones non-masquées.
3. On met le champ des points qui ne reçoivent pas de valeurs à -1e20.

Voici ce qu'on obtient concrètement pour le champ reçu dans les deux sens d'interpolation (ie. de teo1 vers bggd et de bggd vers teo1) :



(a) Champ reçu représentatif à grande échelle bggd-teo1-esmf



(b) Champ reçu représentatif à grande échelle teo1-bggd-esmf

3 Maillages structurés

3.1 Introduction

L'objectif de cette section est de comparer les résultats d'erreur et de champ obtenus pour la SCRIP et ESMF. Les résultats sont disposés de façon à ce qu'on puissent facilement comparer, sur la même page, les résultats de la SCRIP et d'ESMF. La présentation des résultats se fait dans l'ordre suivant (du plus large au plus spécifique) :

1. Le couple de grilles (de bggd vers teo1 ou le contraire).
2. Le genre d'interpolation (bilinéaire, bicubique, linéaire, conservative).

3.2 Couple de teo1 vers bggd

3.2.1 Introduction

Dans cette partie nous allons étudier l'erreur pour différentes interpolations de la grille rectilinéaire "teo1" vers une autre grille regulière lon-lat "bggd". Nous allons comparer les résultats d'erreur entre la SCRIP et ESMF, dans la configuration avec et sans plus proches voisins.
Le champ reçu quant à lui dépend très peu de l'interpolation. On choisit par conséquent d'illustrer, tout d'abord, le champ reçu sur la grille rectangulaire "bggd" avec et sans plus proches voisins (respectivement champ nnei et champ snnei) pour la SCRIP et ESMF :

Résultats de champ pour la SCRIP	**Résultats de champ pour la ESMF**

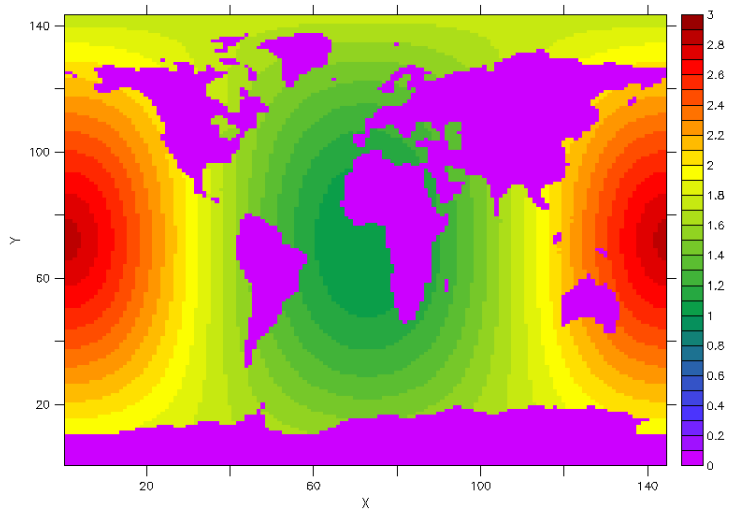
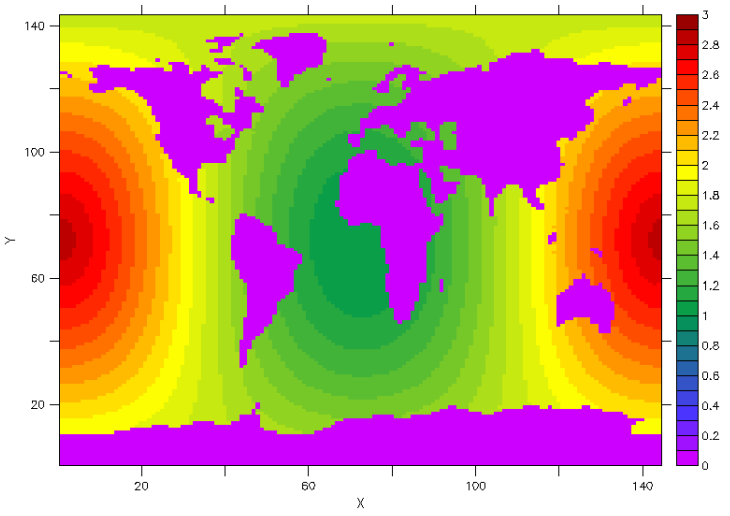

	

Figure (a) Champ nnei sur bggd Interpolation conservative	**Figure (b) Champ nnei sur bggd Interpolation bilinéaire**

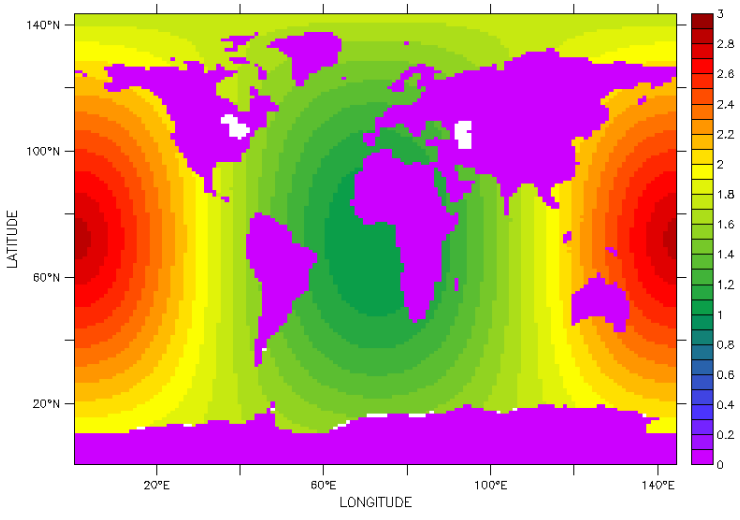
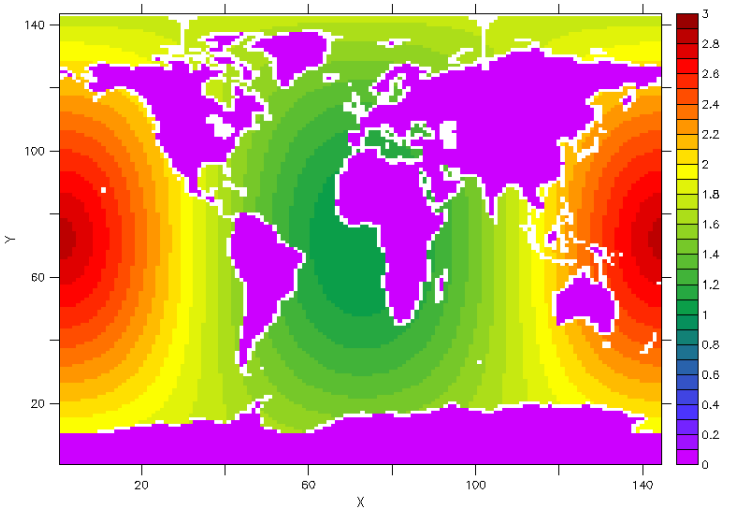

	

Figure (c) Champ snnei sur bggd Interpolation conservative	**Figure (d) Champ snnei sur bggd Interpolation bicubique**

Les champs reçus pour les autres types d'interpolations sont tout à fait similaires et ne peuvent pas être distingués visuellement, nous ne les avons donc pas illustrés dans ce rapport. Les interpolations ci-dessus, ont été choisi de manière aléatoire, sauf pour l'option snnei, où on a choisi les interpolations qui mettent le plus en relief (ceci est très subtile visuellement) les points qui ne reçoivent pas de valeur. Encore une fois, les différences entre champs sont très sensibles d'une interpolation à l'autre.

Notons que pour la configuration avec plus proches voisins, les maximums d'erreur entre la SCRIP et ESMF sont identiques. Ceci provient du fait que pour ces points (à maximum d'erreur) l'extrapolation du plus proche voisin source additionnel non-masqué a été appliquée. On choisit alors, dans un premier temps, plutôt que de comparer les résultats d'erreur pour nos deux librairies, d'illustrer l'erreur obtenue pour les 4 types d'interpolation pour la SCRIP en laissant l'échelle de couleur s'ajuster automatiquement et donc en incluant le maximum de l'erreur. Les résultats pour ESMF sont très similaires et ne peuvent pas être distingués visuellement, nous avons choisi de ne pas les illustrer ici. On obtient alors le résultat suivant :

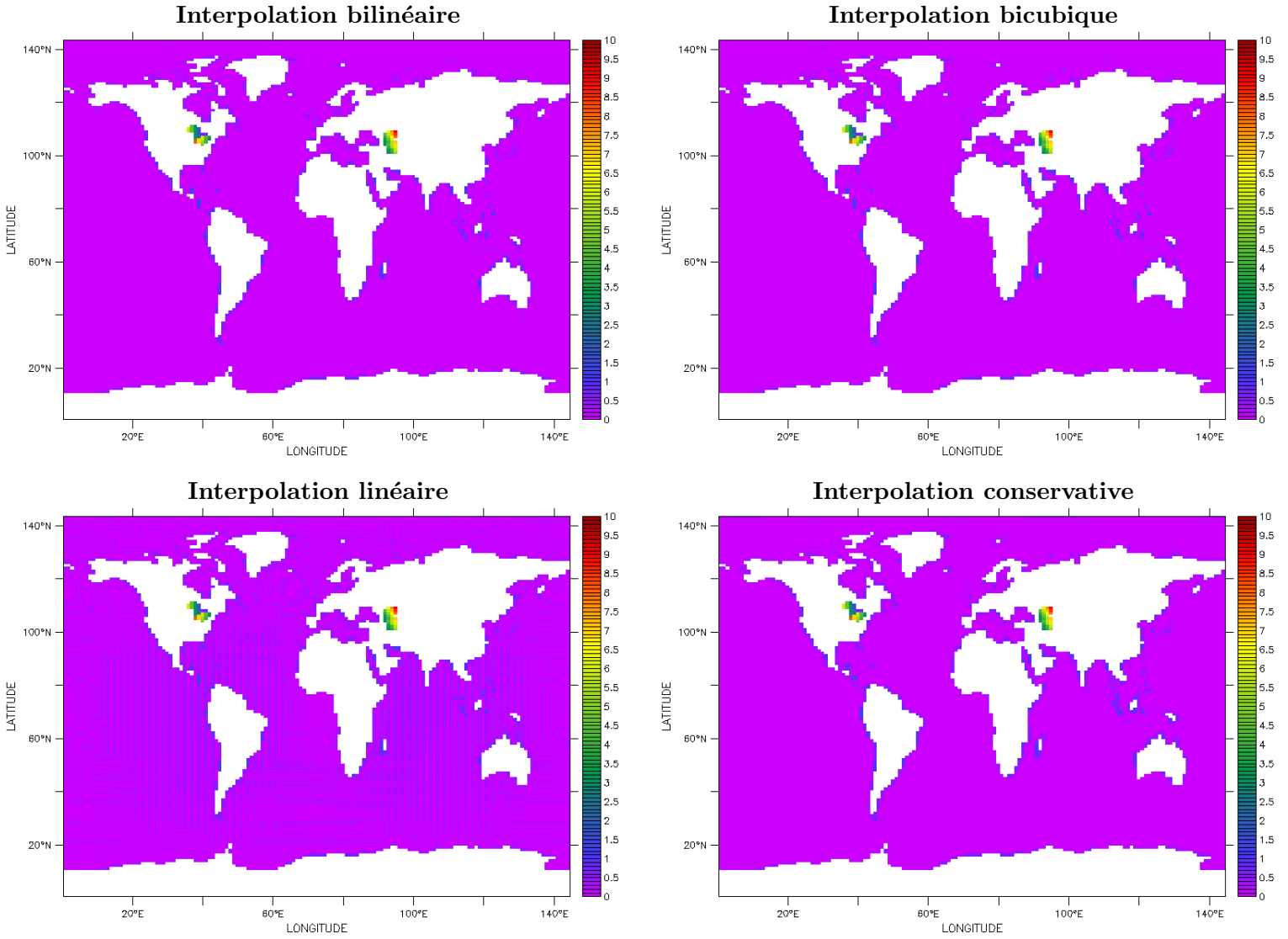


Figure : Pour les cas nnei, on retrouve la même erreur maximale de 9.03% pour les différentes interpolations dans cette configuration
teo1-bggd- pour la scrip et esmf

On choisit ensuite de fixer la valeur maximale de l'échelle de couleur pour les résultats d'erreur à 1 % pour permettre une comparaison judicieuse entre la SCRIP et ESMF dans les zones où l'extrapolation du plus proche voisin source non-masqué n'est pas appliquée. On présente ci-après l'erreur pour la SCRIP et ESMF, dans la configuration avec et sans plus proches voisins :

3.2.2 Interpolation bilinéaire

Résultats d'erreur pour la SCRIP	Résultats d'erreur pour la ESMF

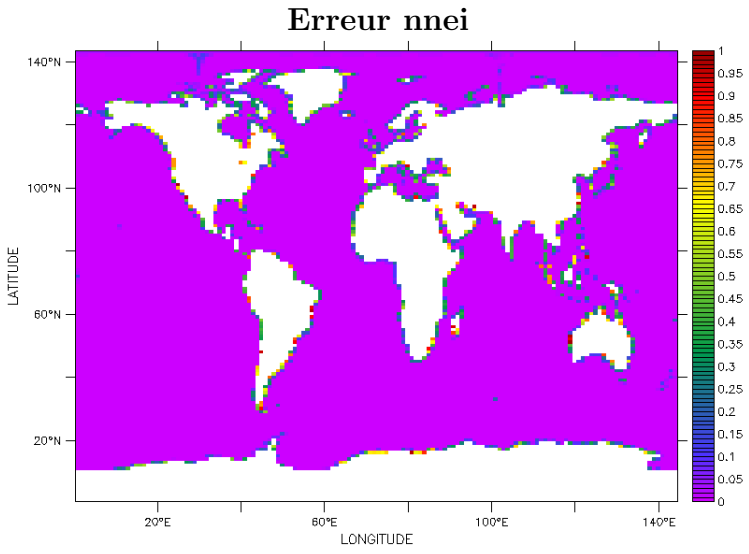


Figure (a) Echelle ajustée à une valeur maximale de 1%
teo1-bggd-bilinéaire-scrip-nnei

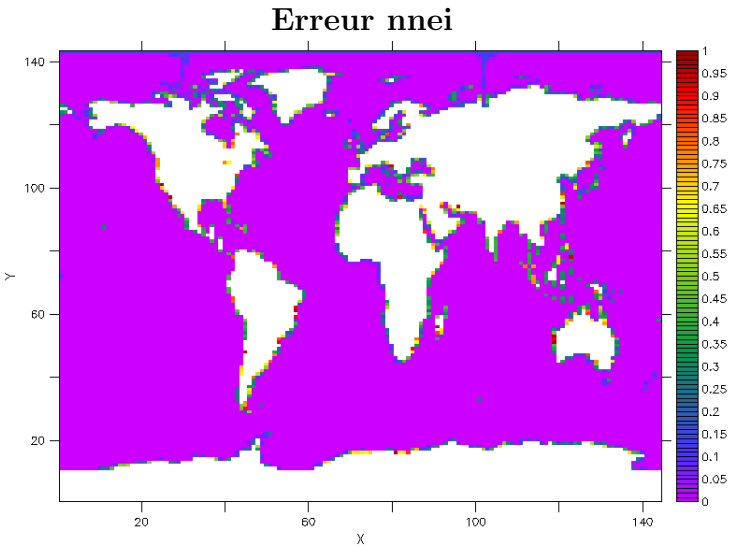


Figure (b) Echelle ajustée à une valeur maximale de 1%
teo1-bggd-bilinéaire-esmf-nnei

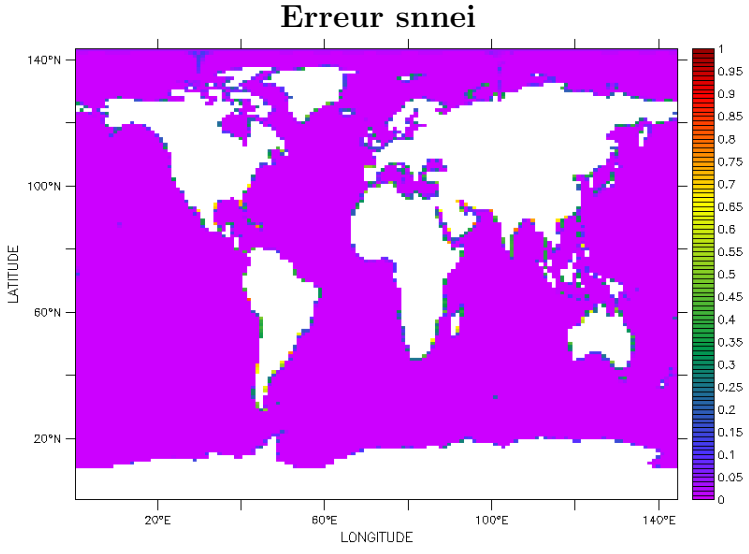


Figure (c) Erreur maximale de 0.8%
teo1-bggd-bilinéaire-scrip-snnei

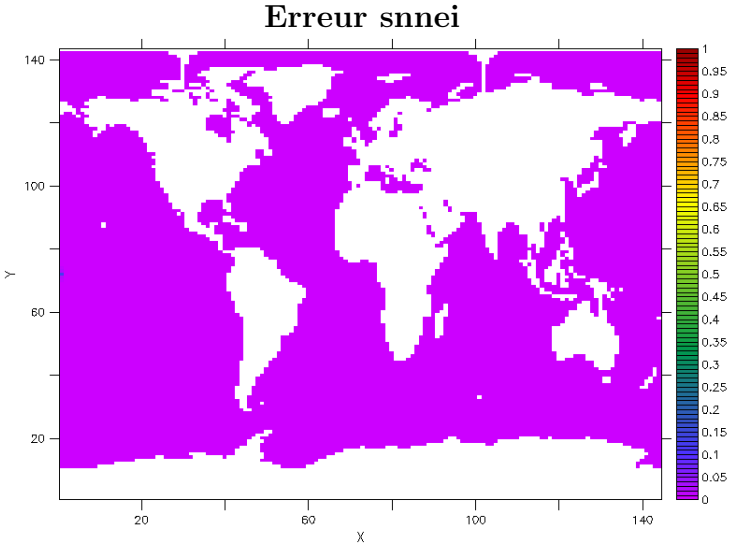


Figure (d) Erreur maximale de 0.13% sur les points recevant une valeu
teo1-bggd-bilinéaire-esmf-snnei

Commentaire

L'erreur snnei pour la SCRIP semble plus élevée que pour ESMF, mais ceci revient simplement au fait que dans le cas où au moins un des voisins bilinéaires est masqué, ESMF ne calcule pas de valeur alors que la SCRIP fait une moyenne sur les voisins bilinéaires non-masqués.

On décide après chaque interpolation d'illustrer la différence de l'erreur (l'écart d'erreur) entre la SCRIP et ESMF (en nnei et snnei), pour pouvoir évaluer précisément les différences entre les deux librairies. Voici ce qu'on obtient pour l'interpolation bilinéaire :

Différence en nnei	**Différence en snnei**

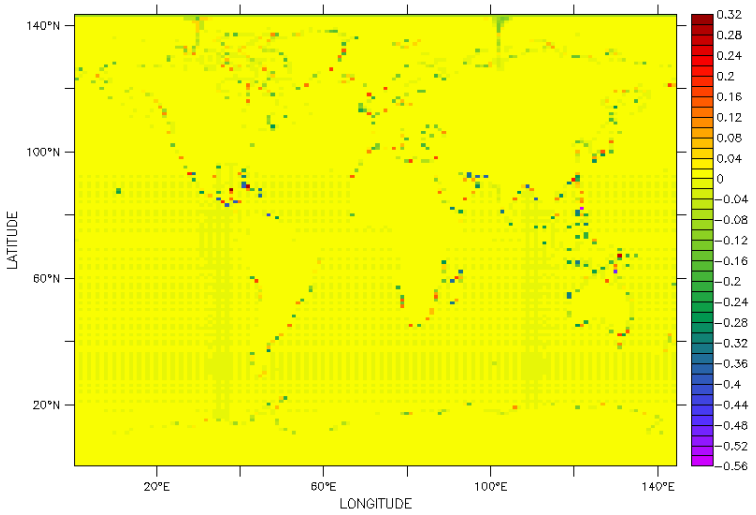
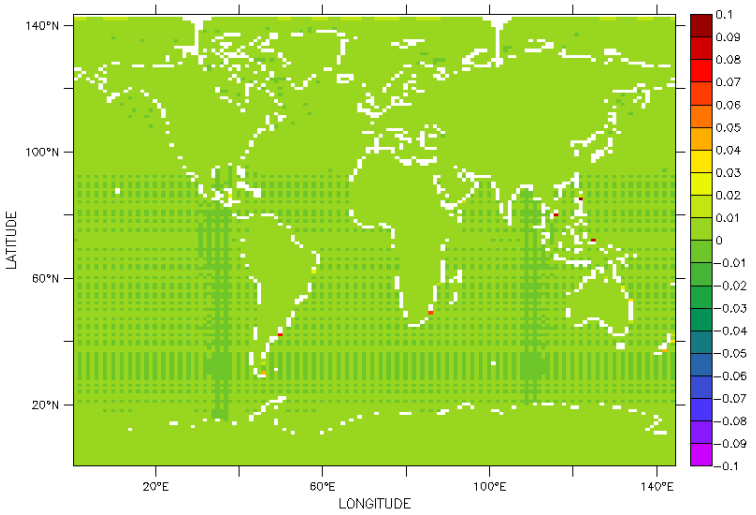

	

Figure (a) Différence d'erreur automatique teo1-bggd-bilinéaire-nnei	Figure (b) Différence d'erreur ajustée teo1-bggd-bilinéaire-snnei

Sur la figure nnei, l'échelle de couleur est ajustée automatiquement ; par contre pour la configuration snnei il faut la forcer à 0.1%. En effet si on laisse l'échelle de couleur s'ajuster automatiquement, elle ira jusqu'à -1e20, car en snnei l'erreur des points qui ne reçoivent pas de valeur (le contour des continents) est fixée à -1e20 (cf. section consignes sans plus proche voisin). L'échelle de couleur est fixée ici à -0.1/0.1%, car c'est l'écart maximal repéré visuellement pour le reste des points.

3.2.3 Interpolation bicubique

Résultats d'erreur pour la SCRIP	Résultats d'erreur pour la ESMF

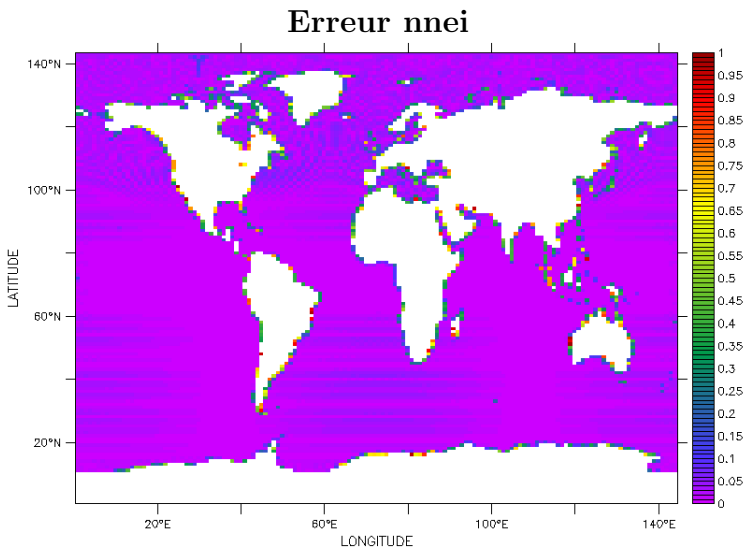


Figure (a) Echelle ajustée à une valeur maximale de 1%
teo1-bggd-bicubique-scrip-nnei

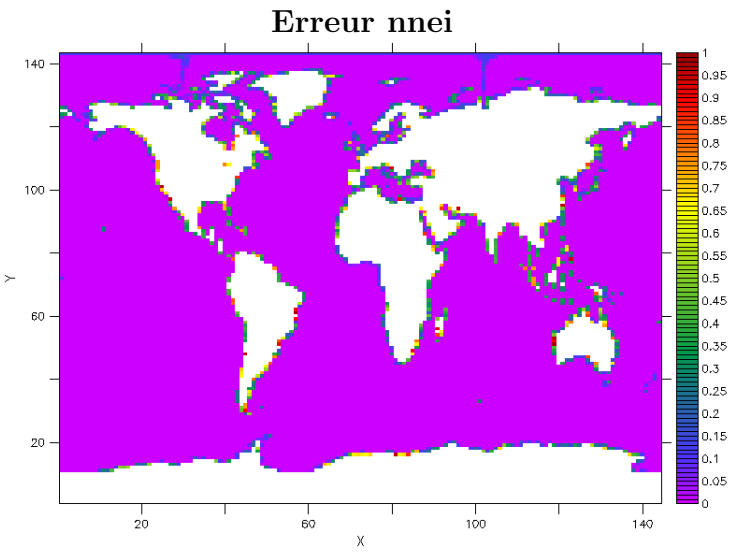


Figure (b) Echelle ajustée à une valeur maximale de 1%
teo1-bggd-bicubique-esmf-nnei

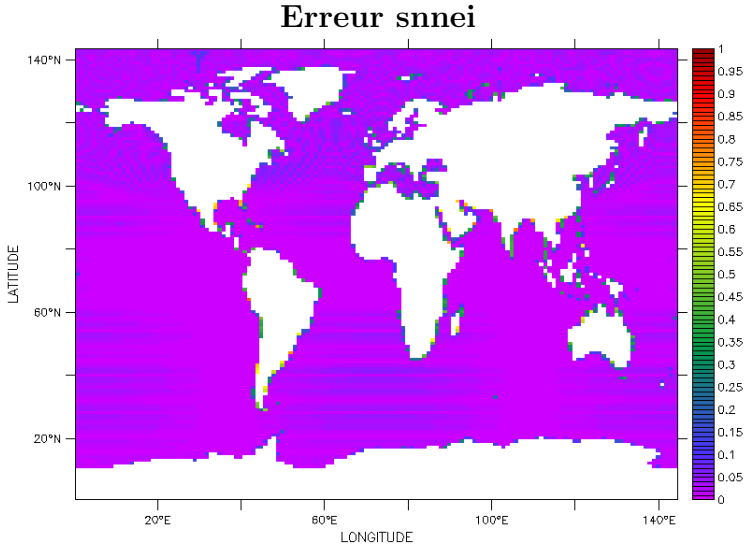


Figure (c) Erreur maximale de 0.81%
teo1-bggd-bicubique-scrip-snnei

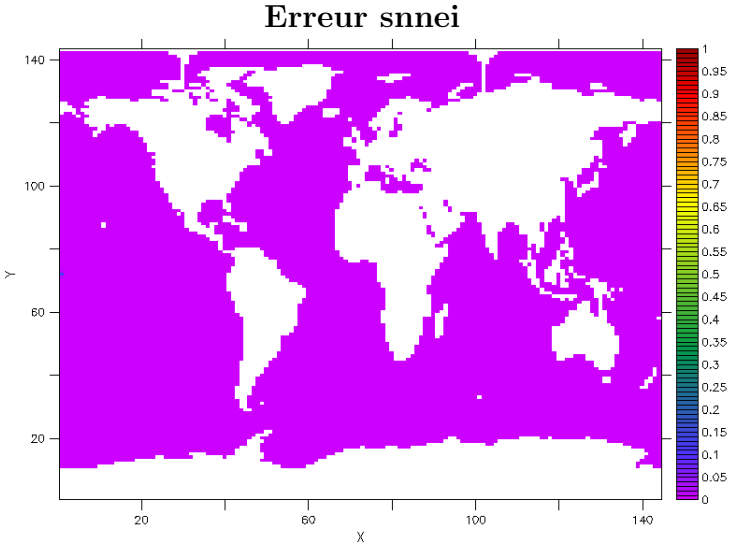


Figure (d) Erreur maximale de 0.12%
teo1-bggd-bicubique-esmf-snnei

Commentaire

L'erreur snnei pour la SCRIP semble plus élevée que pour ESMF, mais ceci revient simplement au fait que dans le cas où au moins un des voisins bicubiques est masqué, ESMF ne calcule pas de valeur alors que la SCRIP fait une moyenne sur les voisins bicubiques non-masqués.

Différence d'erreur entre la SCRIP et ESMF :

Différence en nnei	Différence en snnei

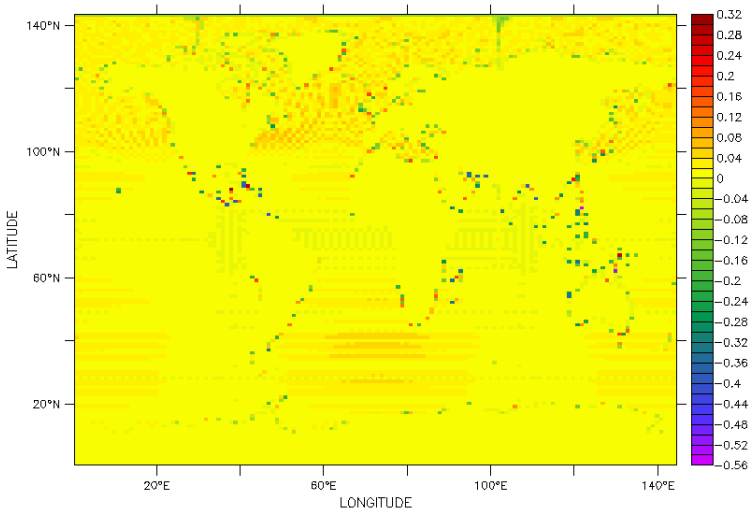
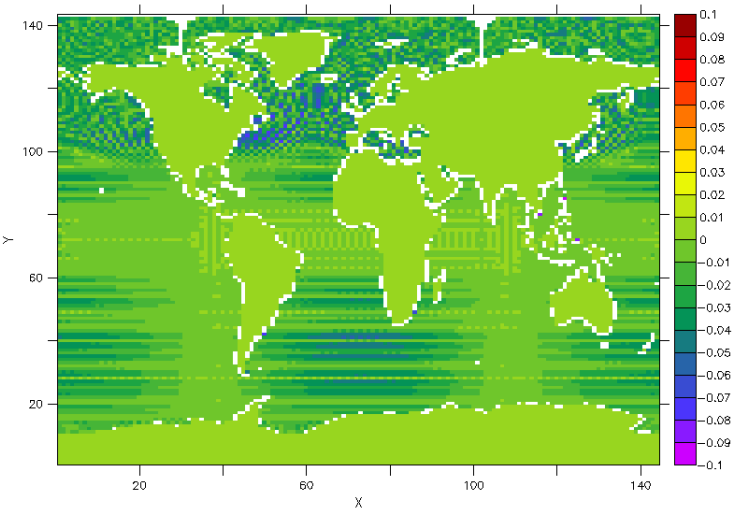

	

Figure (a) Différence d'erreur automatique teo1-bggd-bicubique-nnei	Figure (b) Différence d'erreur ajustée teo1-bggd-bicubique-snnei

De la même manière que pour l'interpolation bilinéaire, sur la figure nnei, l'échelle de couleur est ajustée automatiquement ; par contre pour la configuration snnei il faut la forcer à 0.1%. En effet si on laisse l'échelle de couleur s'ajuster automatiquement, elle ira jusqu'à -1e20, car en snnei l'erreur des points qui ne reçoivent pas de valeur (le contour des continents) est fixée à -1e20 (cf. section consignes sans plus proche voisin). L'échelle de couleur est fixée ici à -0.1/0.1%, car c'est l'écart maximal repéré visuellement pour le reste des points.

3.2.4 Interpolation conservative

Résultats d'erreur pour la SCRIP	Résultats d'erreur pour la ESMF

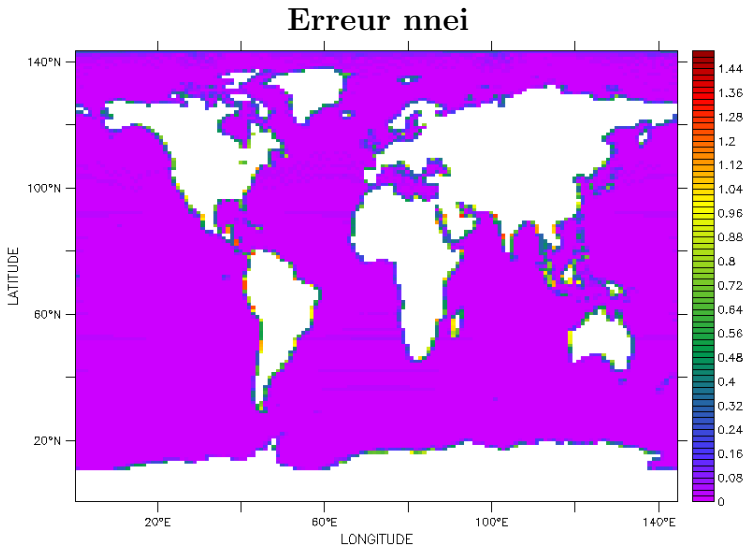


Figure (a) Echelle ajustée à une valeur maximale de 1.5%
teo1-bggd-conserve-scrip-nnei

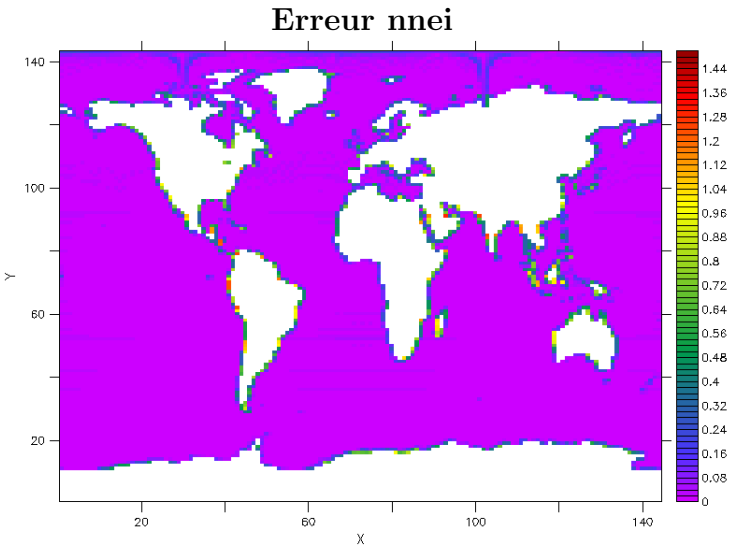


Figure (b) Echelle ajustée à une valeur maximale de 1.5%
teo1-bggd-conserve-esmf-nnei

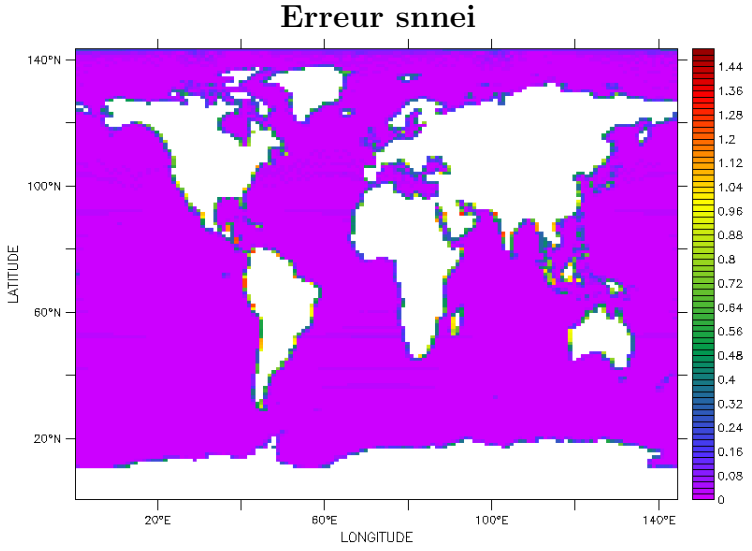


Figure (c) Erreur maximale de 1.29%
teo1-bggd-conserve-scrip-snnei

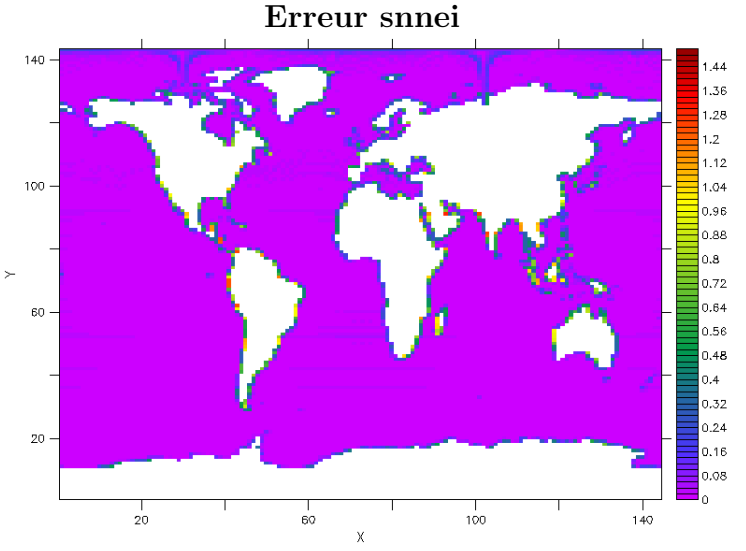


Figure (d) Erreur maximale de 1.29%
teo1-bggd-conserve-esmf-snnei

Commentaire
La comparaison des figures (c) et (d) nous permet de conclure que, en dehors des zones où les librairies vont chercher la valeur de la plus proche maille non-masquée, l'erreur est de même ordre.

Différence d'erreur entre la SCRIP et ESMF :

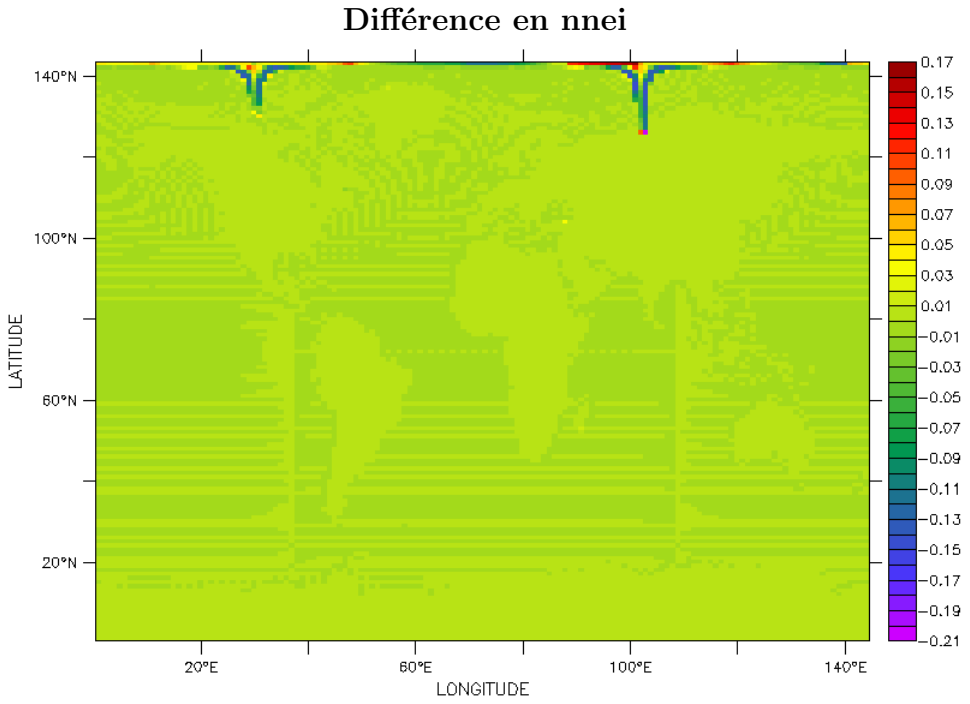


Figure (a) Différence d'erreur automatique
teo1-bggd-conserve-nnei

L'échelle de couleur est ici ajustée automatiquement à +0.17% et -0.21%. Il est difficile de juger quelle librairie est la plus précise ; la différence d'erreur présentant des zones positives (couleurs chaudes) et des zones négatives (couleurs froides).

3.3 Couple de bggd vers teo1

3.3.1 Introduction

Dans cette partie nous allons étudier l'erreur pour différentes interpolations de la grille regulière lat-lon "bggd" vers une autre grille rectilinéaire "teo1". Nous allons comparer les résultats d'erreur entre la SCRIP et ESMF, dans la configuration avec et sans plus proches voisins.
Commençons tout d'abord par donner le champ reçu sur la grille rectangulaire "teo1" avec et sans plus proches voisins pour la SCRIP et ESMF pour l'interpolation conservative (on exclut ici le champ pour l'interpolation bilinéaire et bicubique -sauf dans le cas snnei- qui présente une anomalie détaillée plus loin) :

Résultats de champ pour la SCRIP	**Résultats de champ pour la ESMF**

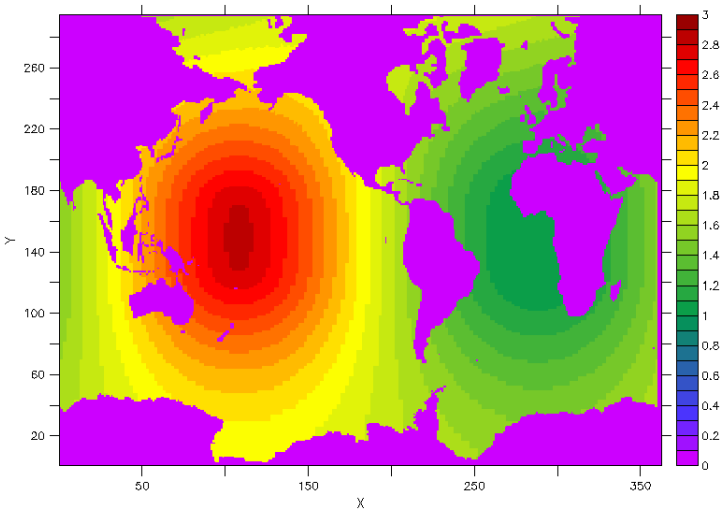


Figure (a) Champ nnei sur teo1
Interpolation linéaire

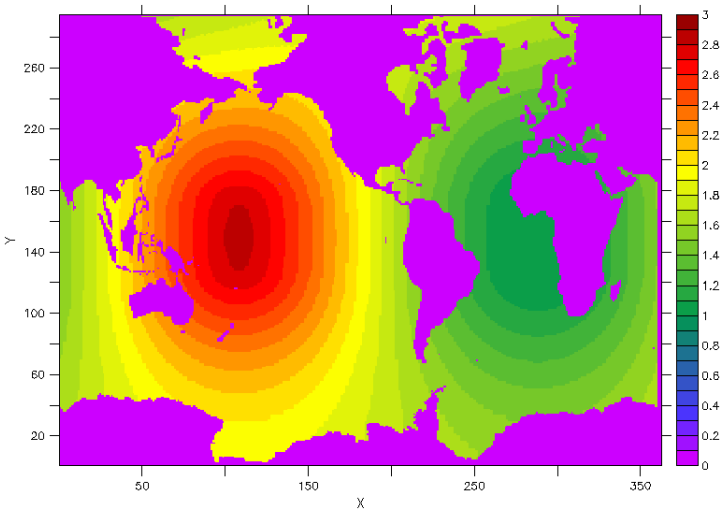


Figure (b) Champ nnei sur teo1
Interpolation conservative

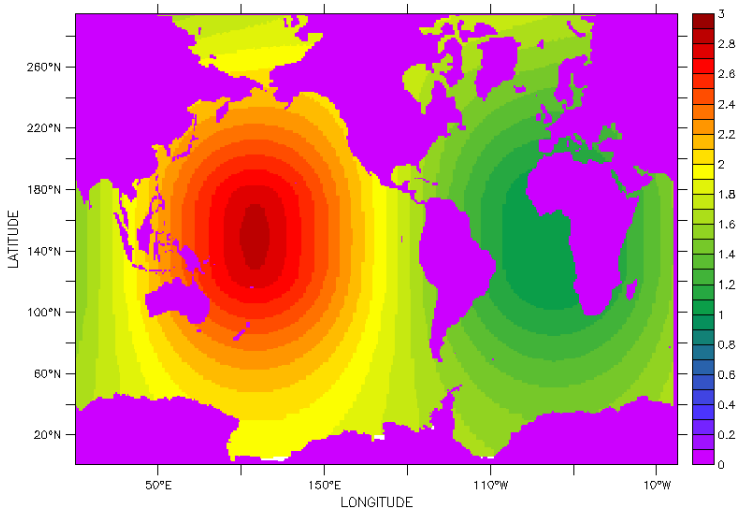


Figure (c) Champ snnei sur teo1
Interpolation conservative

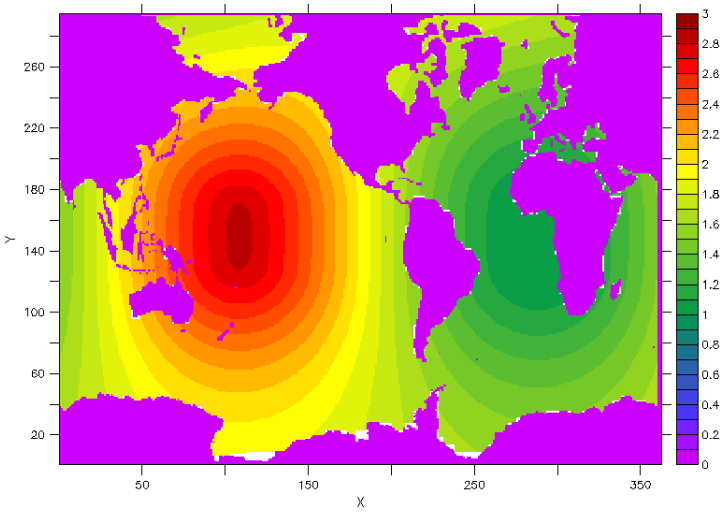


Figure (d) Champ snnei sur teo1
Interpolation bilinéaire

Les zones en blanc sur les figures "snnei" viennent du fait que certaines mailles non-masquées n'intersectent aucune maille source non-masquée. Pour avoir une explication concernant le choix des interpolations, il faut se référer à l'introduction de la partie présentant les interpolations de teo1 vers bggd.

3.3.2 Interpolation bilinéaire

On commence dans cette section par donner les résultats d'erreur avec une erreur maximale problématique de 173% (pour les deux configurations avec et sans plus proches voisins pour la SCRIP), on donnera ensuite les raisons derrière ce résultat. On présentera plus loin les résultats avec une échelle avec une valeur maximale réajustée pour pouvoir faire des comparaisons pertinentes pour les autres points du domaine.

Résultats erreur SCRIP (avec ou sans plus proche voisin)

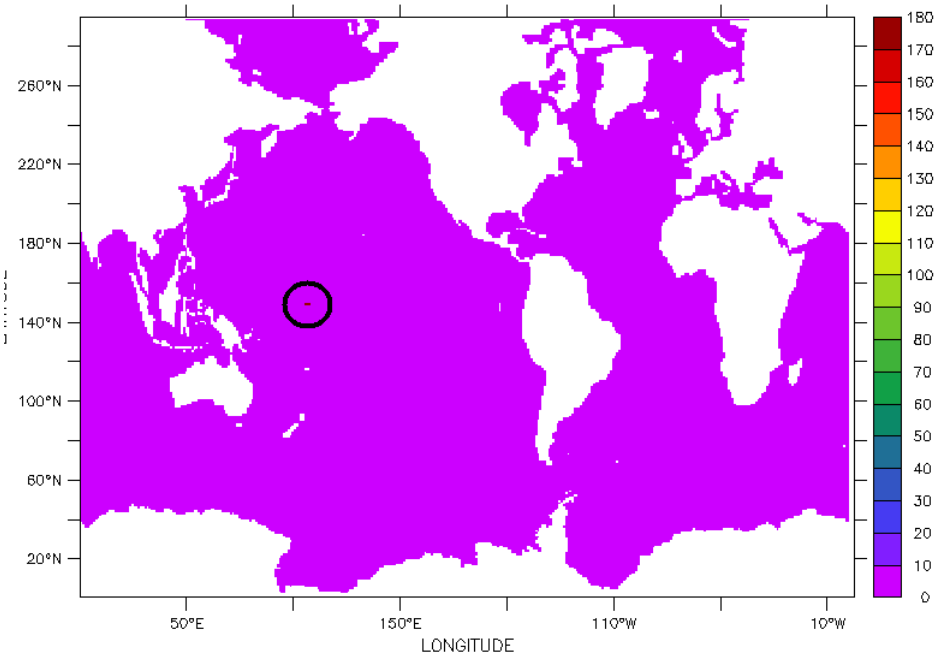


FIGURE 1 – Erreur abhérante de 173% dans la zone du pacifique entourée en noir

Résultats champ SCRIP (avec ou sans plus proche voisin)

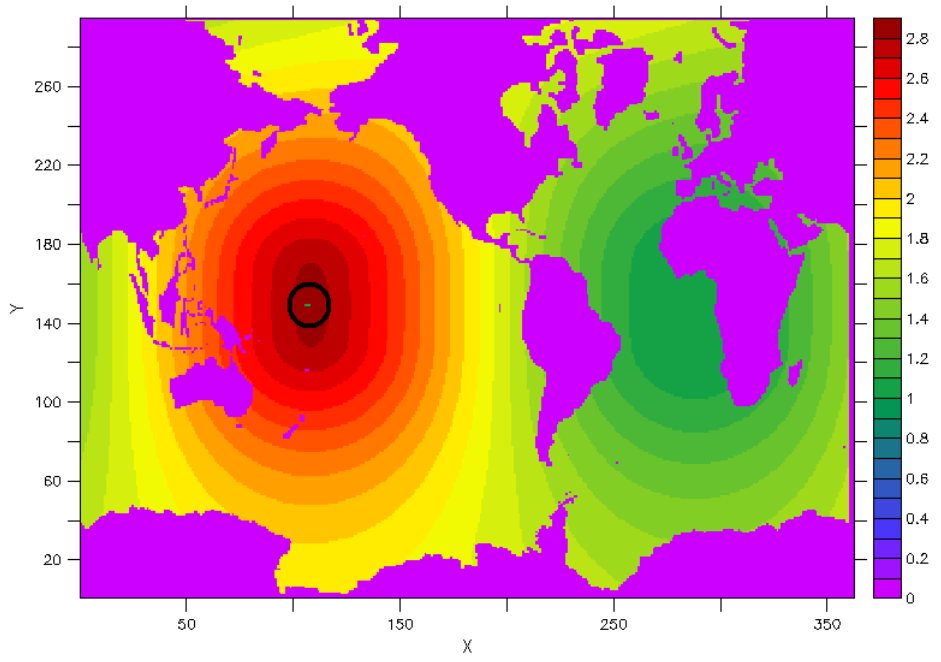


FIGURE 2 – Champ anormal cf. zone entourée en noir

Interprétations et commentaires progressifs (erreur)

1. Il y a un problème concernant la SCRIP (la région entourée en noir et qui contient une erreur grossière) ; l'enjeu et de trouver l'origine de l'erreur et y remédier.
2. La lecture des valeurs des champs reçus dans la région entourée en noir nous a permis de repérer les points cibles problématiques. Il s'agit en fait de trois points qui se trouvent à l'équateur de coordonnées dans teo1 (y=148,x=105 ; y=148,x=106 ; y=148,x=107), et d'adresses de destination (53681, 53682, 53683).
3. Un fichier NCL a été créé pour remonter aux points sources des points anormaux pour essayer d'identifier le problème.

4. Ces trois points se trouvent à une longitude environnant les 180° ; et pourtant deux de ces points utilisent pour l'interpolation un quadruplet source qui se trouvent aux "antipodes", d'une longitude environnant 0°. En effet ces points là ont pour adresses source :
   — 10152 (de coord. en radian lon :0, lat :0.2) et de poids : 0
   — 10153 (de coord. en radian lon :0.4, lat :0.2) et de poids : 0
   — 10296 (de coord. en radian lon :0.4, lat :0.2) et de poids : 73.
   — 10297 (de coord. en radian lon :0, lat :0) et de poids :-72.
   On peut remarquer que la somme des poids est égale à 1, mais que ces poids sont aberrants en tant que tels.
5. L'enjeu est de suivre le parcours de ces trois points dans le programme OASIS (dans `remp_bilinear.f`) et trouver l'origine de ce problème.
6. On a ainsi compris que le problème vient du fait que la SCRIP et ESMF modifie les coordonnées des points pour éviter les sauts de périodicité.

On affiche ici les résultats avec une échelle limitée à 1%, on exclut ainsi les points à problèmes.

Résultats d'erreur pour la SCRIP	Résultats d'erreur pour la ESMF

Erreur nnei

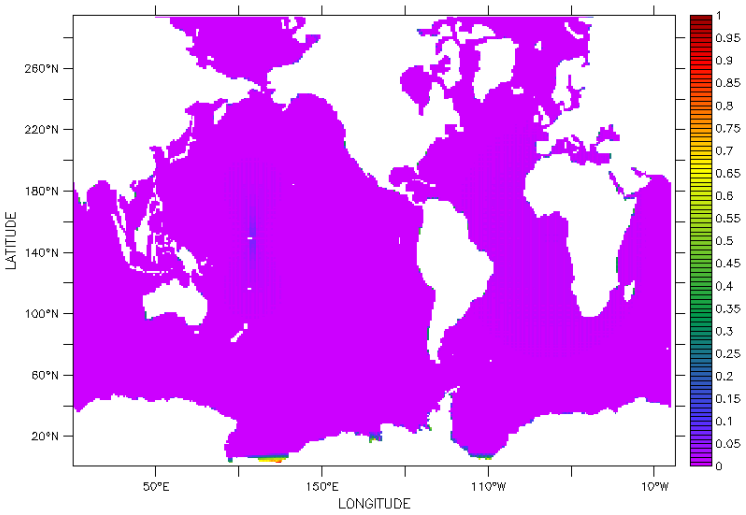


Figure (a) Echelle ajustée à une valeur maximale de 1%
bggd-teo1-bilinéaire-scrip-nnei

Erreur nnei

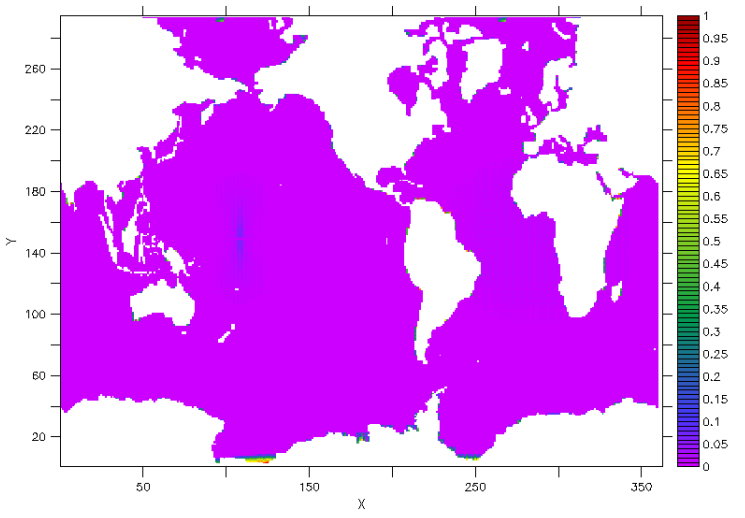


Figure (b) Echelle ajustée à une valeur maximale de 1%
bggd-teo1-bilinéaire-esmf-nnei

Erreur snnei

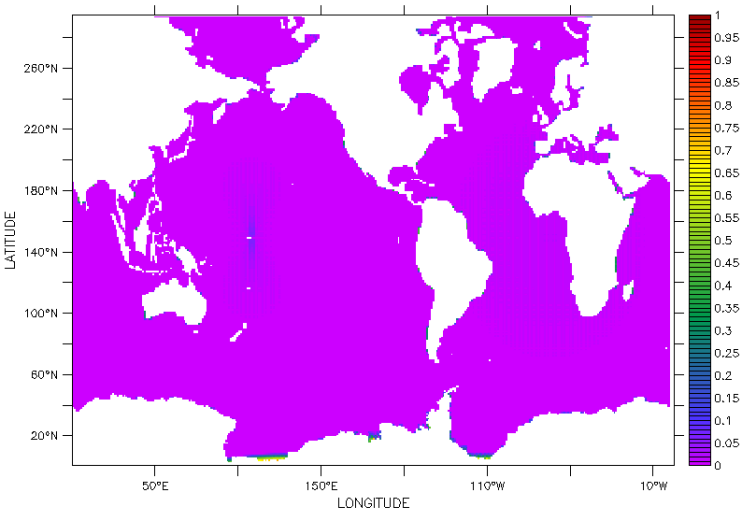


Figure (c) Erreur maximale de 0.94%
teo1-bggd-bilinéaire-scrip-snnei

Erreur snnei

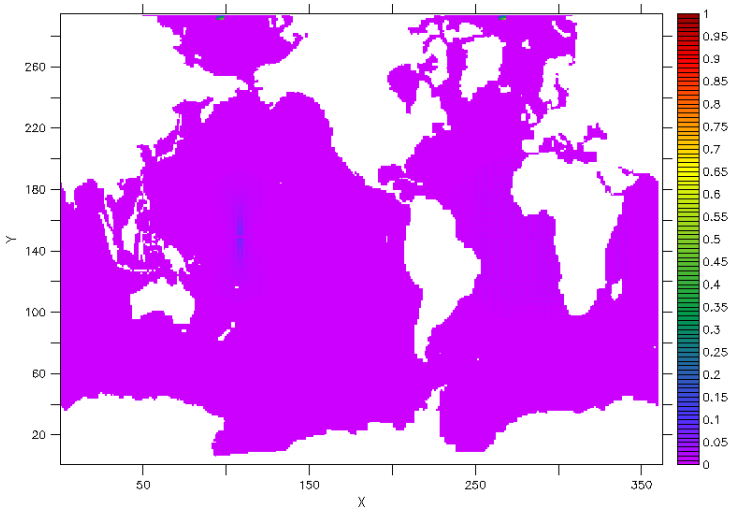


Figure (d) Erreur maximale de 0.49%
teo1-bggd-bilinéaire-esmf-snnei

Commentaire :
Comme pour l'interpolation bilinéaire de teo1 vers bggd, l'erreur snnei semble plus elevée que pour ESMF, mais ceci revient simplement qu fait que dans le cas où au moins une des mailles voisines est masquée, ESMF ne calcule pas de valeur alors que la SCRIP fait une moyenne sur les maille voisines non-masquées.

Différence d'erreur entre la SCRIP et ESMF :

Différence en nnei	Différence en snnei

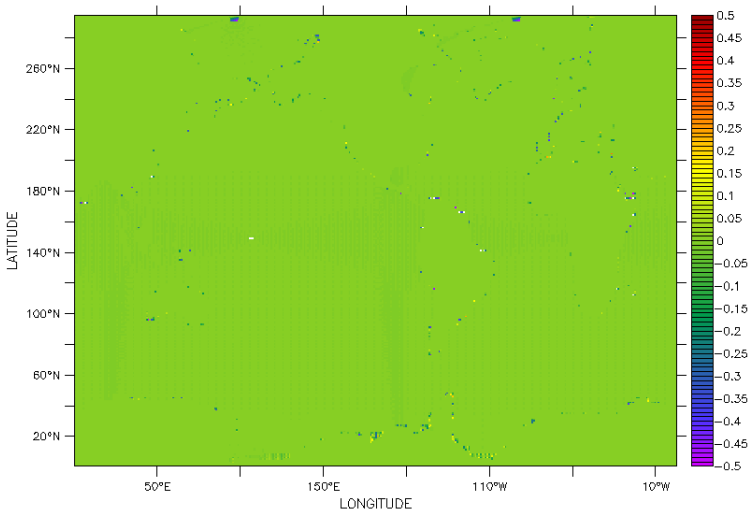
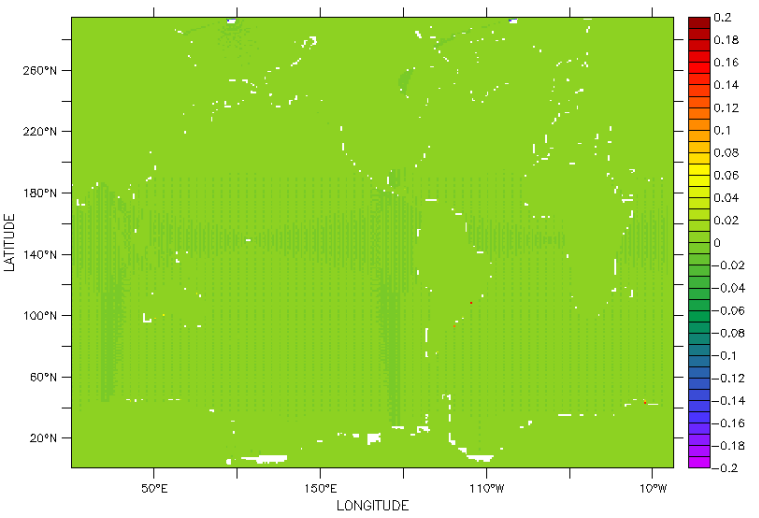

	

Figure (a) Différence d'erreur ajustée bggd-teo1-bilinéaire-nnei	Figure (b) Différence d'erreur ajustée bggd-teo1-bilinéaire-snnei

Pour l'interpolation bilinéaire de bggd vers teo1, on a fixé la différence à -0.5/0.5%. En effet l'ajustement automatique de l'échelle de couleur irait jusqu'à l'erreur maximale anormale vue précédemment et on aurait, de ce fait, des résultats biaisés. Comme l'écart maximal repéré visuellement pour le reste des points est de 0.5%, on a choisit de fixer la différence d'erreur à -0.5/0.5%. Pour la configuration snnei il faut la forcer à 0.2%. En effet si on laisse l'échelle de couleur s'ajuster automatiquement, elle ira jusqu'à -1e20, car en snnei l'erreur des points qui ne reçoivent pas de valeur (le contour des continents) est fixée à -1e20 (cf. section consignes sans plus proche voisin). L'échelle de couleur est fixée ici à -0.2/0.2%, car c'est l'écart maximal repéré visuellement pour le reste des points.

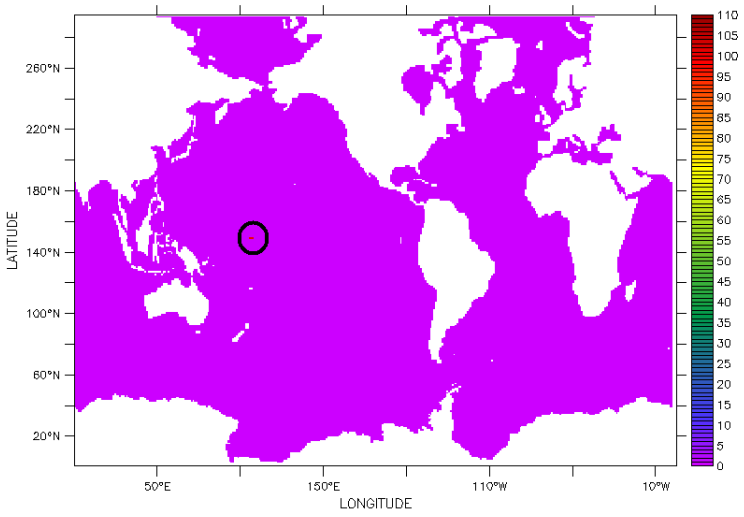
3.3.3 Interpolation bicubique

Dans cette section :

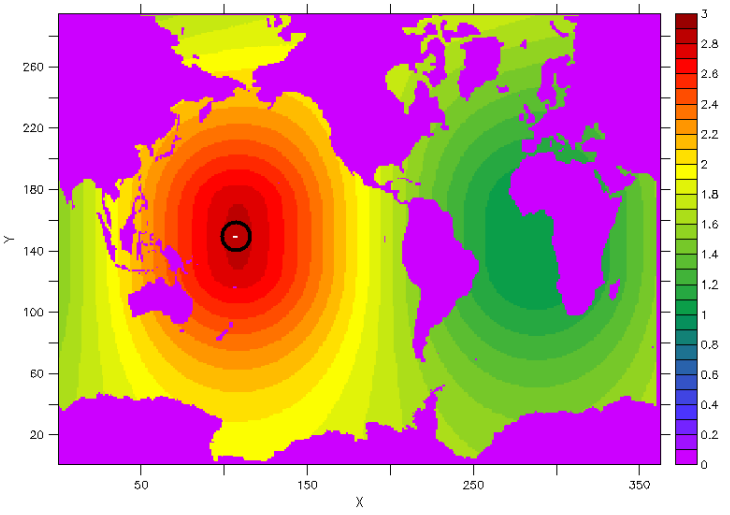
1. On retrouve la même problématique qu'en bilinéaire.
2. On utilise la même démarche pour identifier les points qui posent problème, et remonter à leurs points sources. On obtient alors exactement les mêmes résultats à la différence prêt que les poids sont cette fois 100 fois plus grands.
3. Il faut maintenant de la même manière que précédemment, suivre le parcours de ces trois points dans les programmes d'OASIS (dans `remap_bicubic.f`) et trouver l'origine de ce problème.
4. L'origine du problème est la même que pour l'interpolation bilinéaire.

Ci-dessous les différents résultats disposés de la même manière que le bilinéaire.

Résultats SCRIP (avec plus proches voisins : nnei)



(a) Erreur abhérante de 100% (bggd-teo1-bicu-scrip)



(b) Champ reçu anormal cf. zone entourée en noir (bggd-teo1-bicu-scrip)

On affiche ici les résultats avec une échelle limitée à 1%, on exclut ainsi les points à problèmes.

Résultats d'erreur pour la SCRIP	Résultats d'erreur pour la ESMF

Erreur nnei

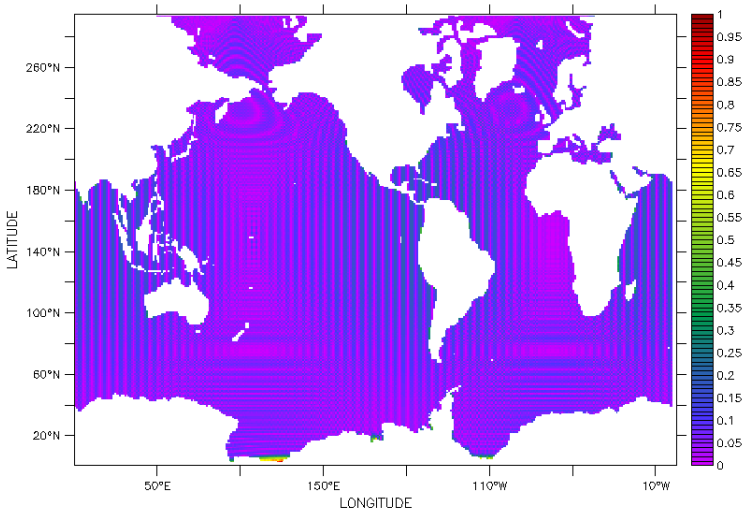


Figure (a) Erreur ajustée à 1%
bggd-teo1-bicubique-scrip-nnei

Erreur nnei

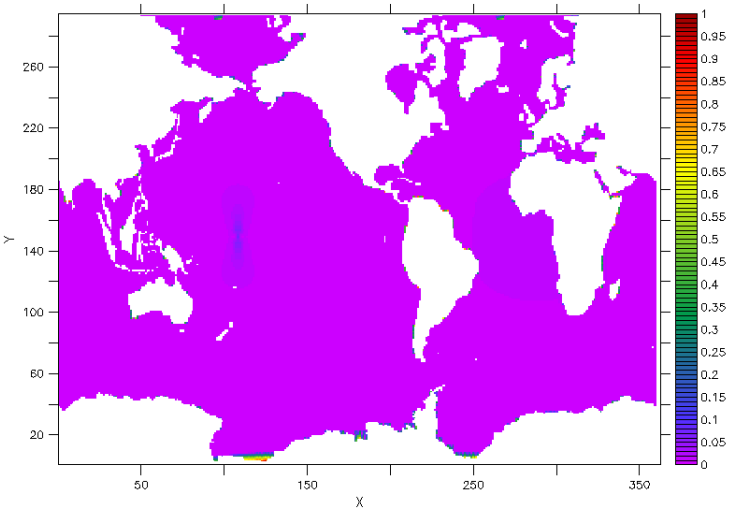


Figure (b) Erreur maximale de 0.94%
bggd-teo1-bicubique-esmf-nnei

Erreur snnei

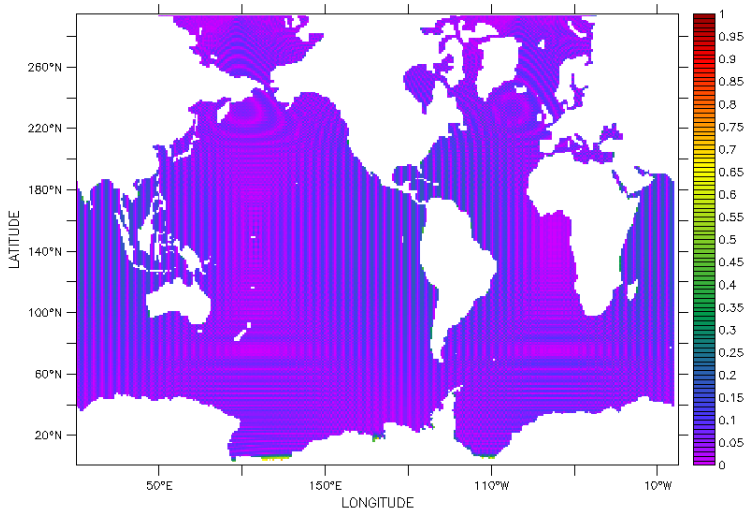


Figure (c) Erreur ajustée à 1%
bggd-teo1-bicubique-scrip-snnei

Erreur snnei

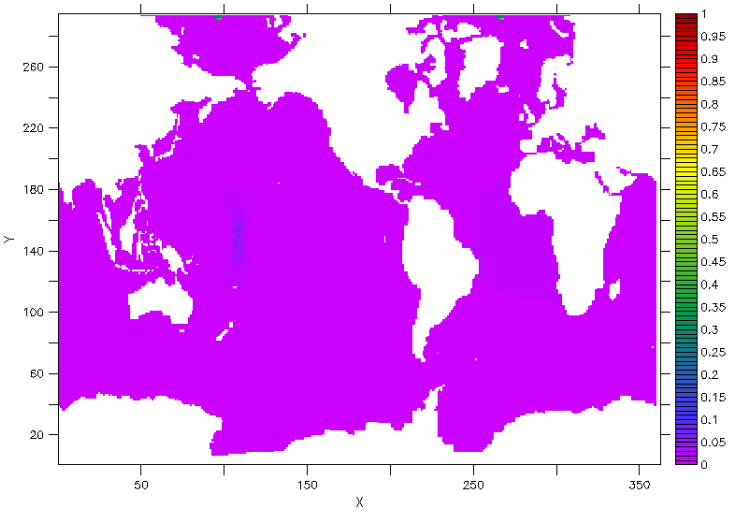


Figure (d) Erreur maximale de 0.49%
bggd-teo1-bicubique-esmf-snnei

Différence d'erreur entre la SCRIP et ESMF :

Différence en nnei	Différence en snnei

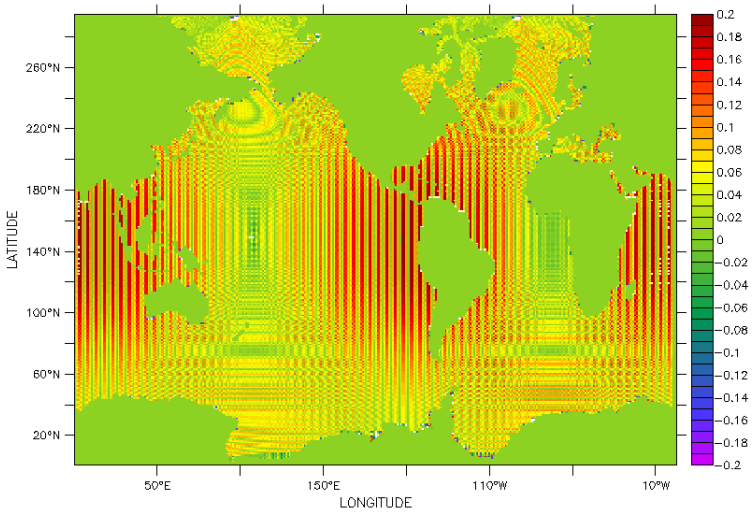
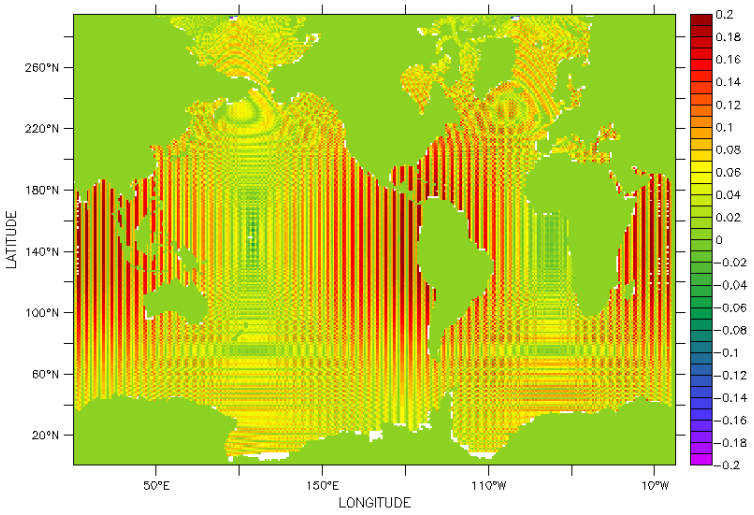

	

Figure (a) Différence d'erreur ajustée teo1-bggd-bicubique-nnei	Figure (b) Différence d'erreur avec une échelle de couleur ajustée teo1-bggd-bicubique-snnei

Pour les mêmes raisons que l'interpolation bilinéaire de bggd vers teo1, on a fixé la différence à -0.2/0.2%. En effet l'ajustement automatique de l'échelle de couleur irait jusqu'à l'erreur maximale anormale vue précédemment (pour l'interpolation bicubique) et on aurait, de ce fait des résultats biaisés. Comme l'écart maximale repéré visuellement pour le reste des points est de 0.2%, on a choisit de fixer la différence d'erreur à -0.2/0.2%. Pour la configuration snnei il faut la forcer à 0.2%. En effet si on laisse l'échelle de couleur s'ajuster automatiquement, elle ira jusqu'à -1e20, car en snnei l'erreur des points qui ne reçoivent pas de valeur (le contour des continents) est fixée à -1e20 (cf. section consignes sans plus proche voisin). L'échelle de couleur est fixée ici à -0.2/0.2%, car c'est l'écart maximal repéré visuellement pour le reste des points.

Sur la majeure partie des bassins océanique (en dehors des points aberrants discutés dans la dernière section), la SCRIP donne des résultats plus précis que ESMF, les graphes de différences étant dominés par les couleurs chaudes.

3.3.4 Interpolation conservative

Résultats d'erreur pour la SCRIP	Résultats d'erreur pour la ESMF

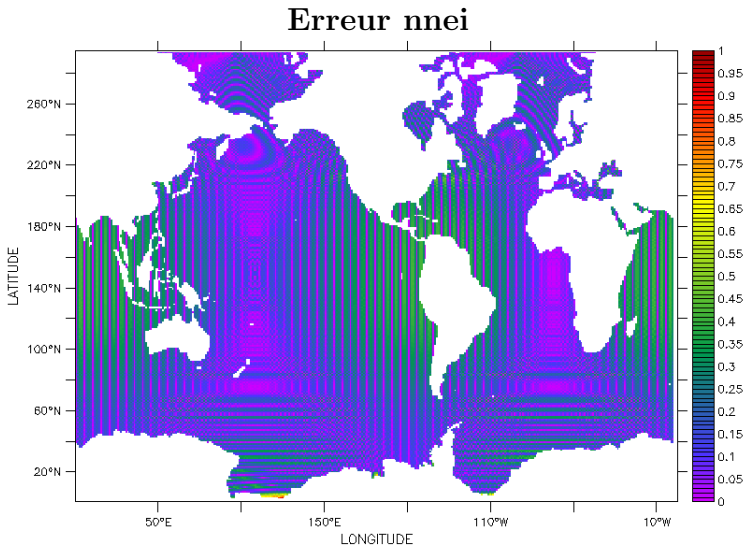


Figure (a) Erreur maximale de 0.94%
bggd-teo1-conserve-scrip-nnei

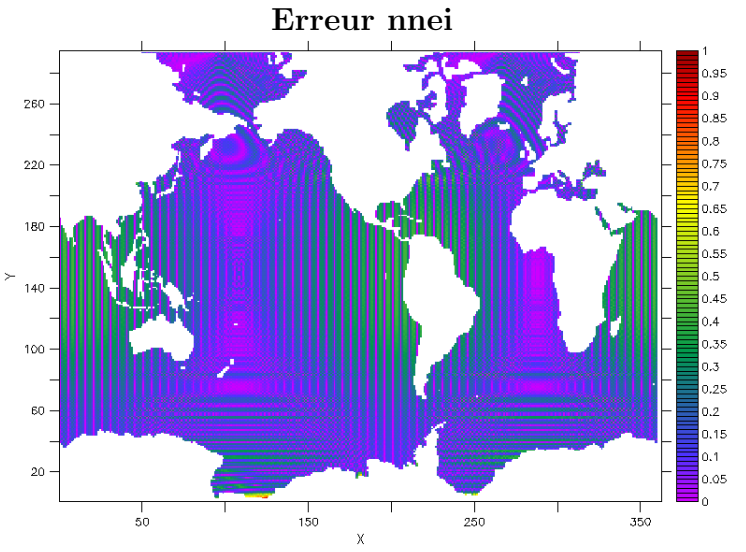


Figure (b) Erreur maximale de 0.94%
bggd-teo1-conserve-esmf-nnei

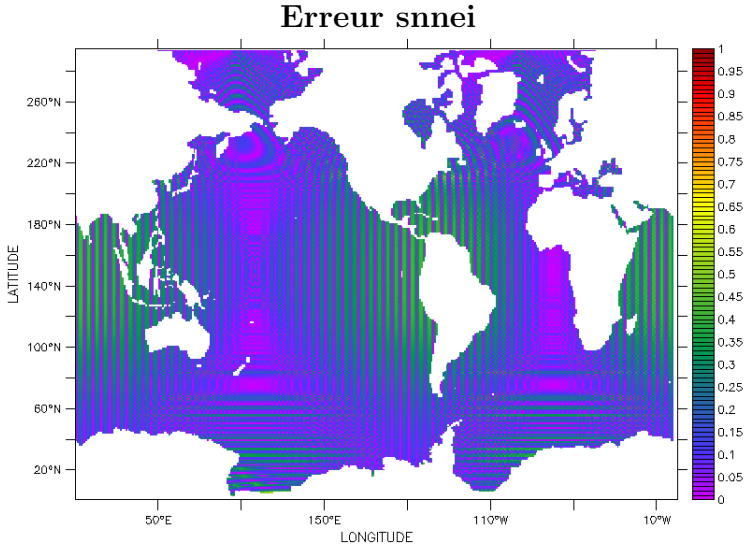


Figure (c) Erreur maximale de 0.54%
bggd-teo1-conserve-scrip-snnei

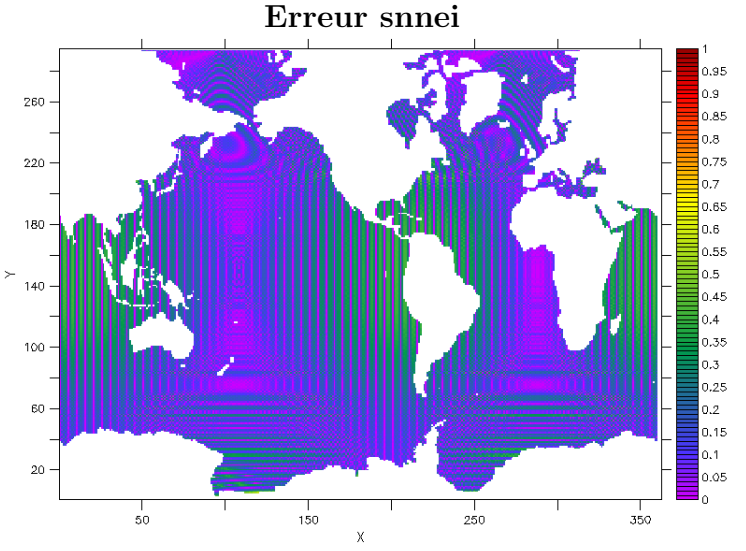


Figure (d) Erreur maximale de 0.58%
bggd-teo1-conserve-esmf-snnei

Différence d'erreur entre la SCRIP et ESMF :

Différence en nnei	Différence en snnei

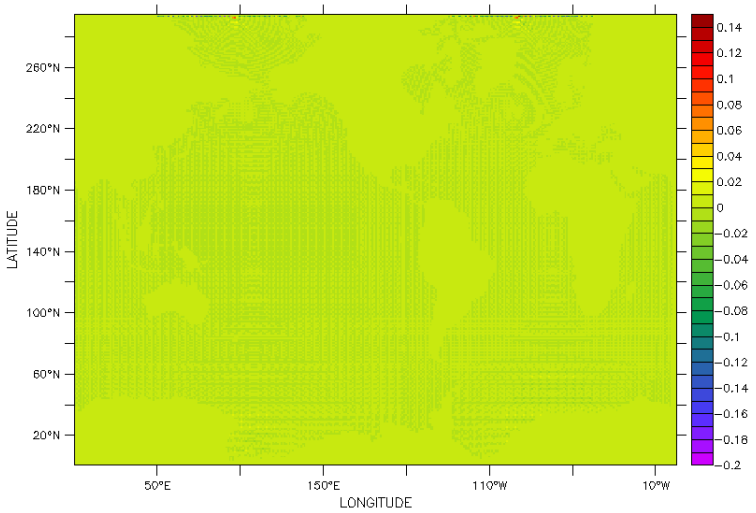
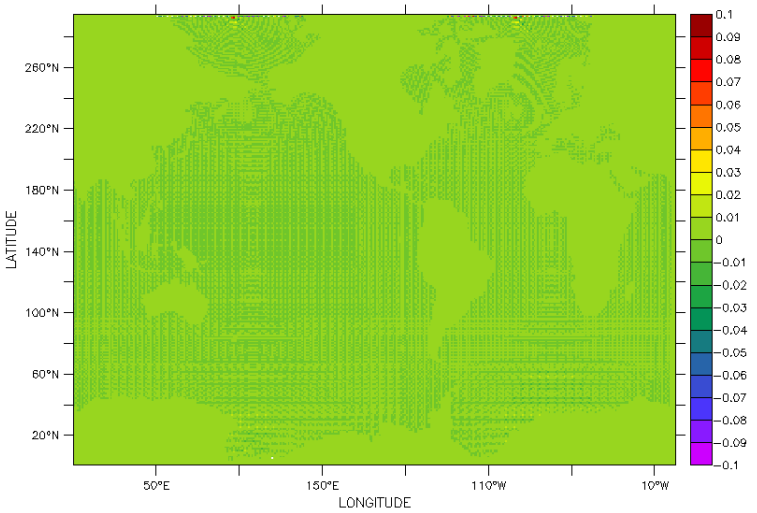

	

Figure (a) Différence d'erreur automatique teo1-bggd-conserve-nnei	Figure (b) Différence d'erreur ajustée teo1-bggd-conserve-snnei

Pour l'interpolation conservative, sur la figure nnei, l'échelle de couleur est ajustée automatiquement ; par contre pour la configuration snnei il faut la forcer à 0.1%. En effet si on laisse l'échelle de couleur s'ajuster automatiquement, elle ira jusqu'à -1e20 (cf. section consignes sans plus proche voisin). L'échelle de couleur est fixée ici à -0.1/0.1%, car c'est l'écart maximal repéré visuellement pour le reste des points.

ESMF est meilleur que la SCRIP au nord (ligne à points de couleur froide) sauf en quelques points rouges-orangés.

3.3.5 Interpolation linéaire

Résultats d'erreur pour la SCRIP

Erreur nnei

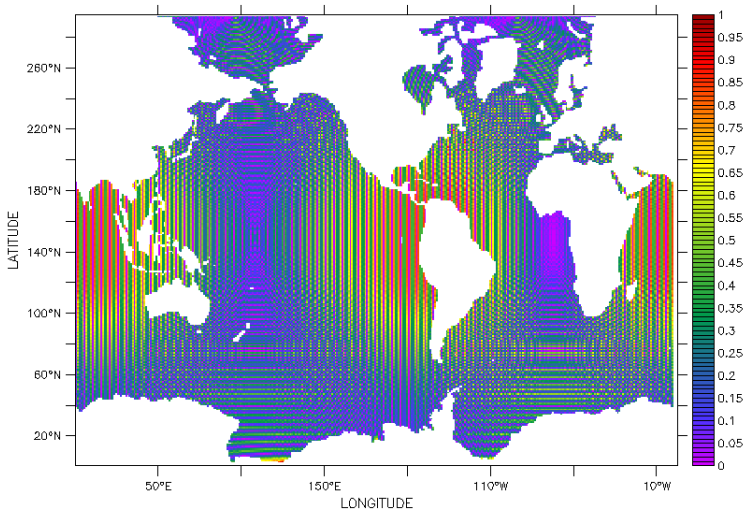


Figure (a) Erreur maximale de 0.94%
bggd-teo1-nn-scrip-nnei

Résultats d'erreur pour la ESMF

Erreur nnei

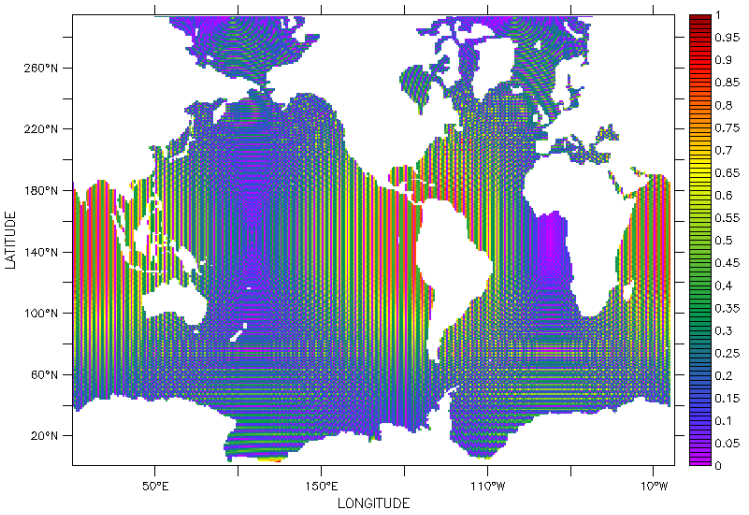


Figure (b) Erreur maximale de 0.94%
bggd-teo1-nn-esmf-nnei

Ici il n'y pas d'erreur "sans plus proche voisin" (snnei), car le principe même de cette interpolation est de rechercher les plus proches voisins non-masqués. On choisit, comme pour les interpolations précédentes, d'illustrer la différence entre l'erreur de la SCRIP et ESMF. Cependant, puisque qu'on a naturellement pas de résultats en configuration snnei, on a décidé, exceptionnellement, d'afficher la différence d'erreur de la grille teo1 vers bggd et vice versa.

Différence d'erreur entre la SCRIP et ESMF (de bggd vers teo1 et de teo1 vers bggd) :

Différence entre la SCRIP et ESMF	Différence entre la SCRIP et ESMF

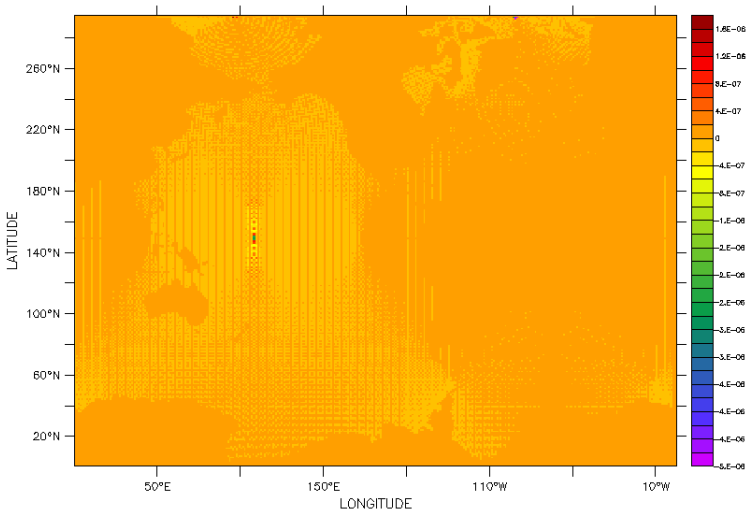


Figure (a) Différence d'erreur automatique
bggd-teo1-linéaire

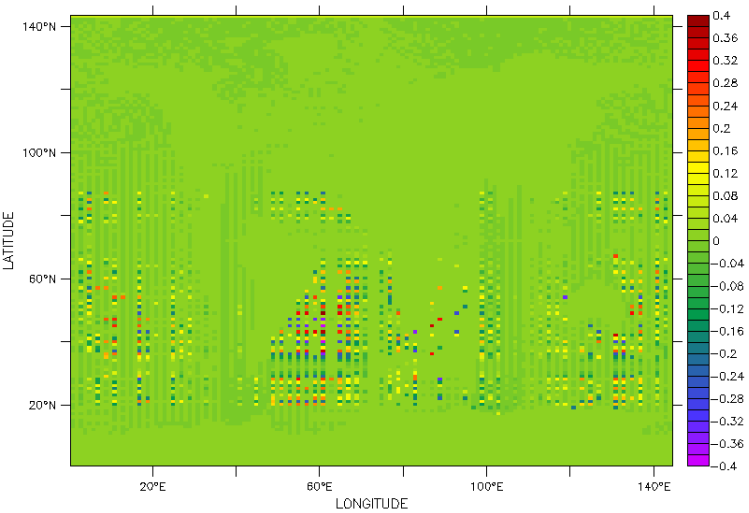


Figure (b) Différence d'erreur automatique
teo1-bggd-linéaire

La différence d'erreur pour ces interpolations est ajustée automatiquement.

4 Maillages non-structurés

4.1 Introduction

Dans cette section nous allons étudier, de la même manière que dans la partie précédente, l'erreur et le champ pour différentes interpolations mais cette fois-ci entre une grille non-structurée "ssea" (une grille gaussienne réduite) et notre grille rectilinéaire "teo1".
Nous allons dans un premier temps présenter les résultats pour la SCRIP, puis dans un deuxième temps les différentes modifications amenées au maillage non-structuré pour pouvoir faire des interpolations sur ESMF.
Commençons d'abord par donner le champ reçu sur la grille non-structurée "ssea" avec et sans plus proches voisins. Le champ reçu sur la grille teo1 est le même que dans la partie précédente (notez que les champs reçus dépendent peu de l'interpolation, et que c'est l'interpolation conservative qui est utilisée dans les figures suivantes) :

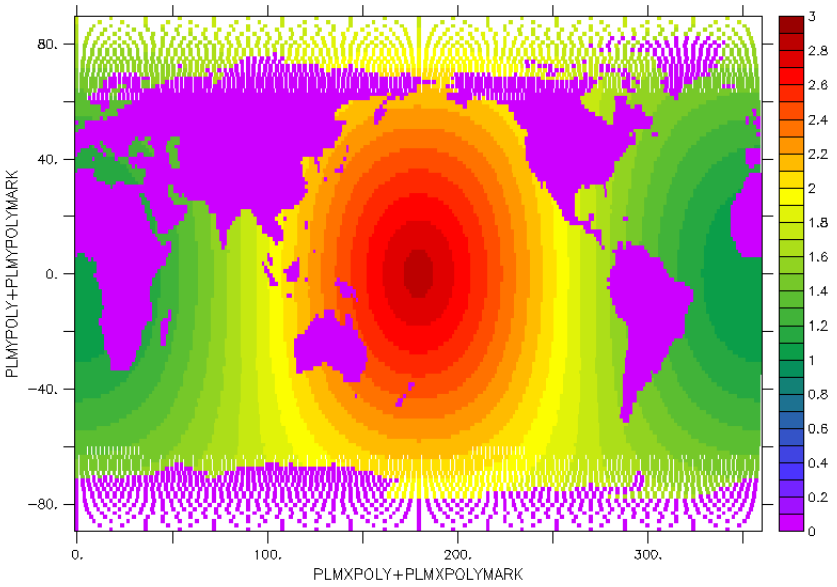
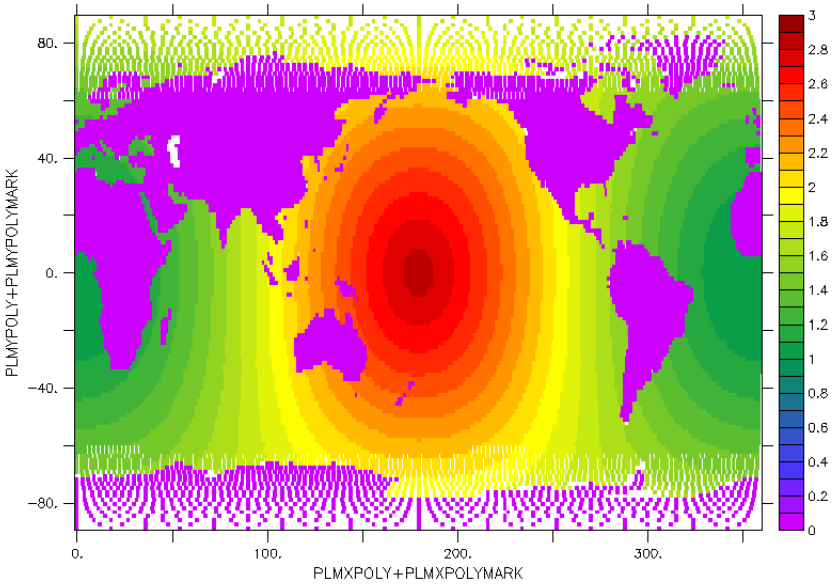


Figure (a) : Champ reçu nnei teo1-ssea-scrip



Figure(b) : Champ reçu snnei teo1-ssea-scrip

4.2 Résultats de la SCRIP

4.2.1 Interpolation de teo1 vers ssea

Interpolation bilinéaire Ici on présente l'erreur avec et sans plus proches voisins :

Résultats d'erreur pour la SCRIP

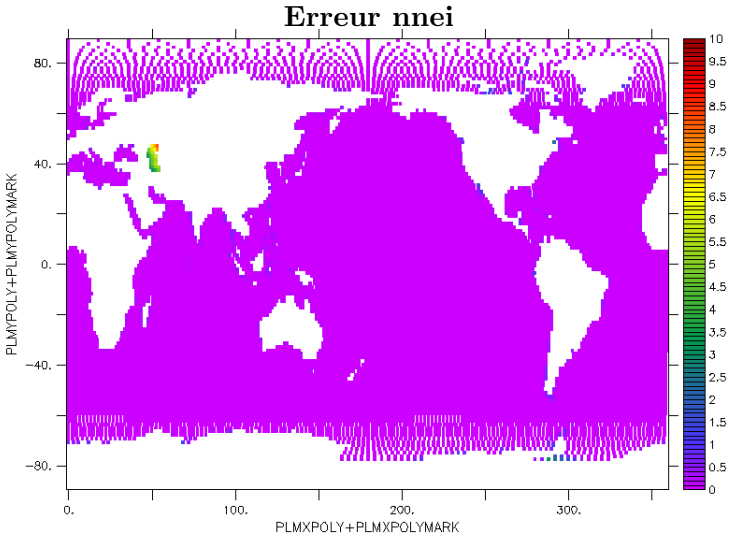


Figure (a) : Erreur maximale de 8.12%
teo1-ssea-bilinéaire-scrip-nnei

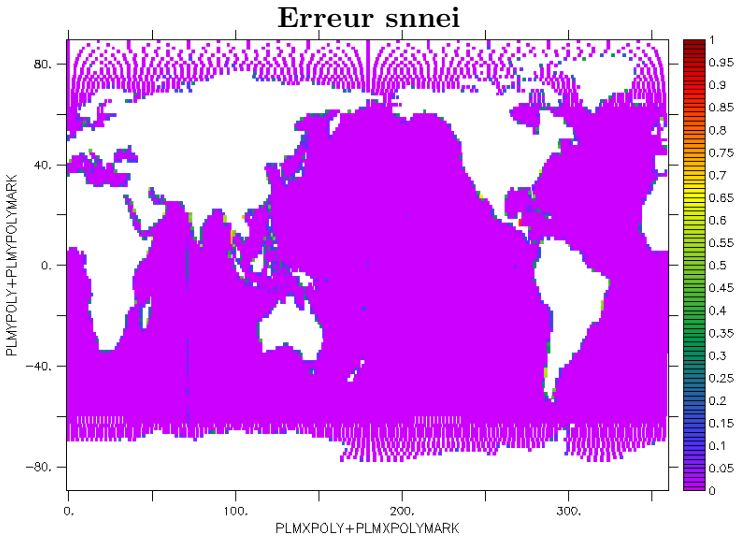


Figure (b) Erreur maximale de 0.86%
teo1-ssea-bilinéaire-scrip-snnei

Interpolation conservative Ici on présente l'erreur avec et sans plus proches voisins :

Résultats d'erreur pour la SCRIP

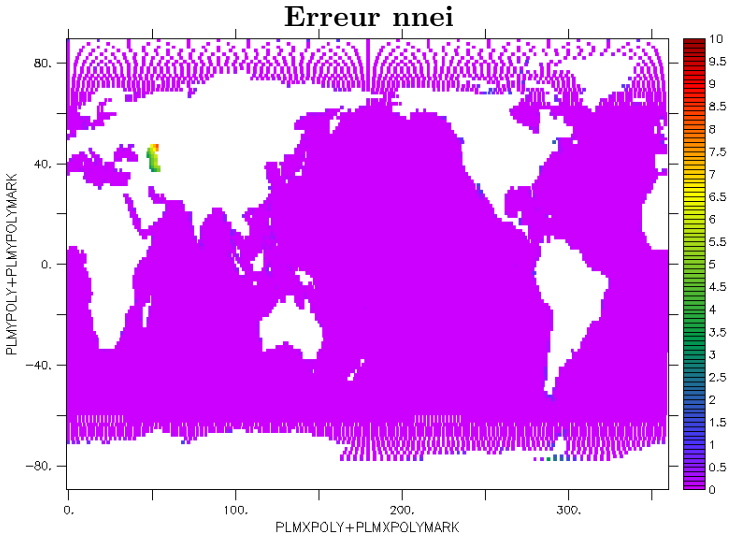


Figure (a) : Erreur maximale de 8.12%
teo1-ssea-conserve-scrip-nnei

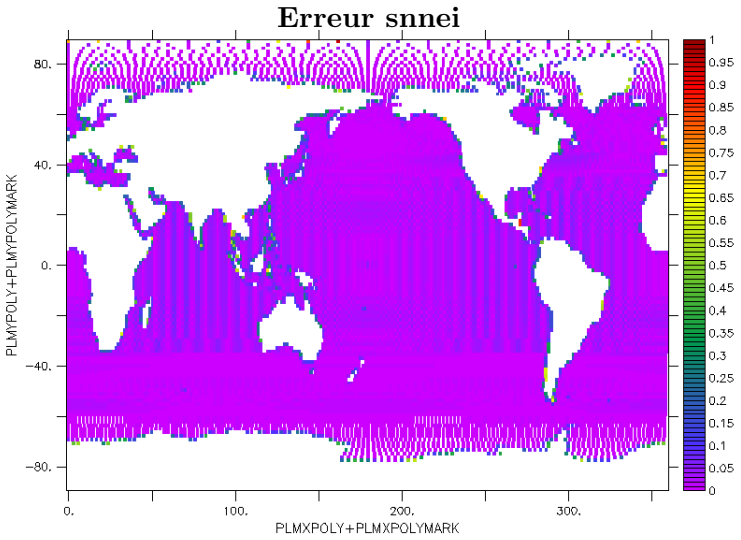


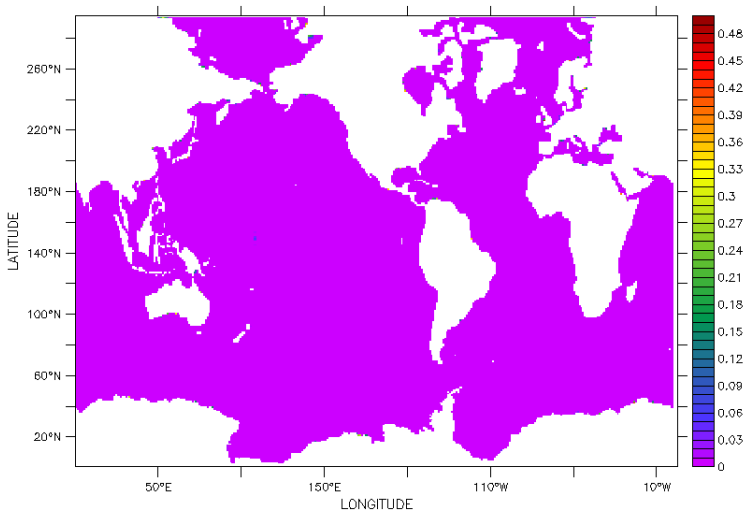
Figure (b) Erreur maximale de 0.86%
teo1-ssea-conserve-scrip-snnei

4.2.2 Interpolation de ssea vers teo1

Interpolation bicubique

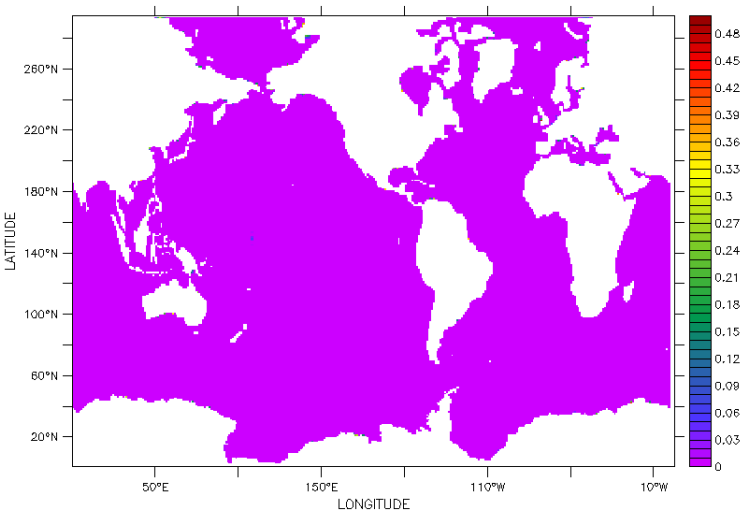
Résultats d'erreur pour la SCRIP

Erreur nnei



(a) Erreur maximale de 0.46%
ssea-teo1-bicubique-scrip-nnei

Erreur snnei

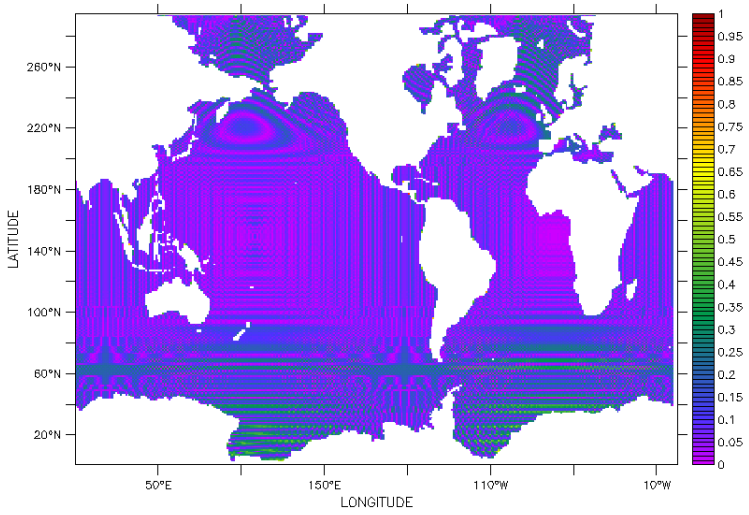


(b) Erreur maximale de 0.46%
ssea-teo1-bicubique-scrip-snnei

Interpolation conservative

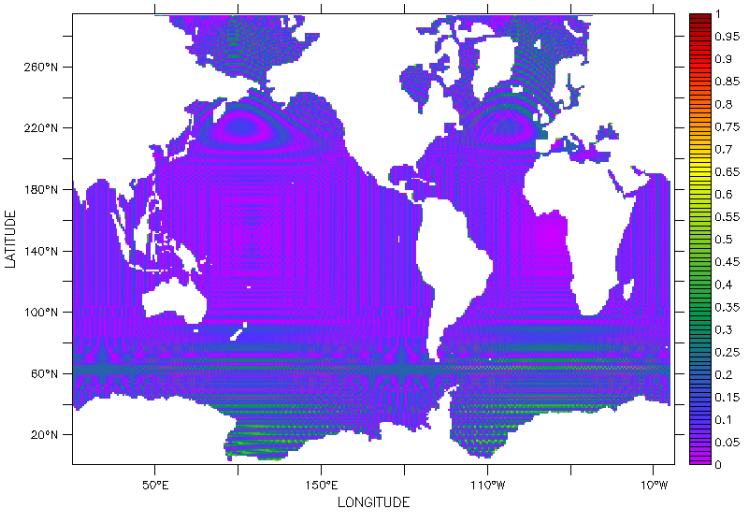
Résultats d'erreur pour la SCRIP

Erreur nnei



(a) Erreur maximale de 0.86%
ssea-teo1-conserve-scrip-nnei

Erreur snnei



(b) Erreur maximale de 0.86%
ssea-teo1-conserve-scrip-snnei

4.3 Travail mené pour ESMF

4.3.1 Introduction

Pour pouvoir poursuivre notre étude comparative, nous devons -comme ce qui a été entrepris précédemment pour la SCRIP- effectuer des interpolations entre notre grille rectilinéaire "teo1" et notre grille non-structurée "ssea" (gaussienne réduite). Cependant, toutes les interpolations d'une grille structurée vers une grille non-structurée, ne sont pas "automatiques" sur ESMF. En effet pour l'interpolation conservative, tous les coins de la maille doivent être définis. Ceci ne posait pas de problème pour les interpolations entre grilles rectilinéaire et régulière étudiées précédemment (de teo1 vers bggd et vice versa) car toutes les mailles étaient "alignées" d'une latitude à l'autre, et donc les quatre coins prédéfinis de chaque cellule étaient suffisants pour définir la grille en entier et mener à bien les interpolations. Or, ce n'est pas le cas pour la grille "ssea" (non-structurée), car le décalage des mailles entre elles d'une latitude à l'autre crée des coins en plus, qu'il faut absolument prendre en compte pour pouvoir effectuer l'interpolation conservative en particulier, sur ESMF. Les figures suivantes expliquent cette contrainte supplémentaire :

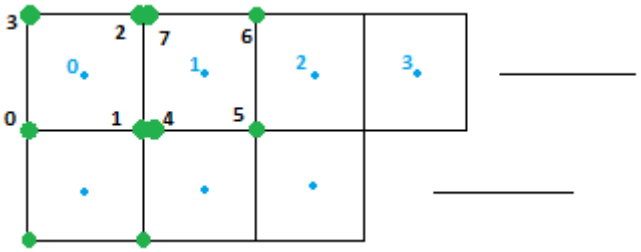


Figure 1 : Coins sur une grille structurée

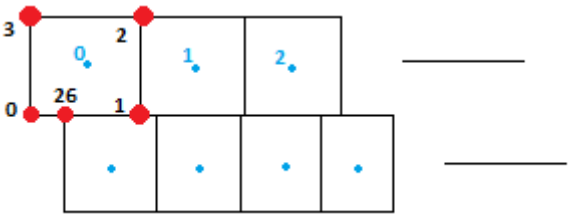
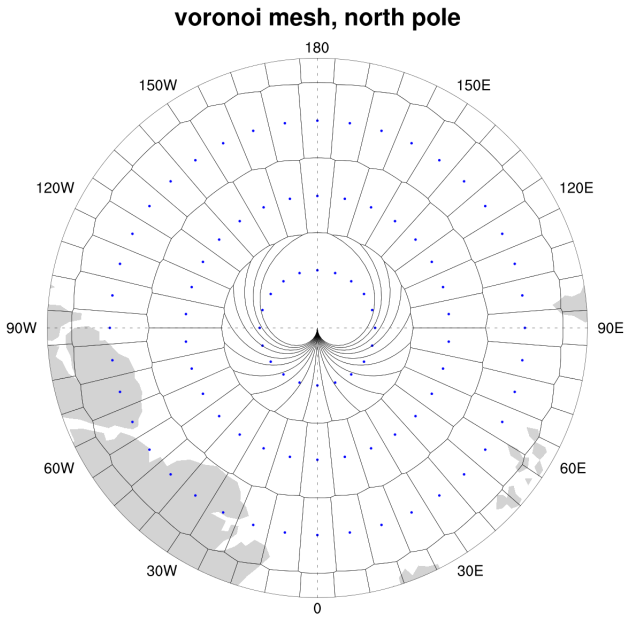


Figure 2 : Coins sur une grille non structurée

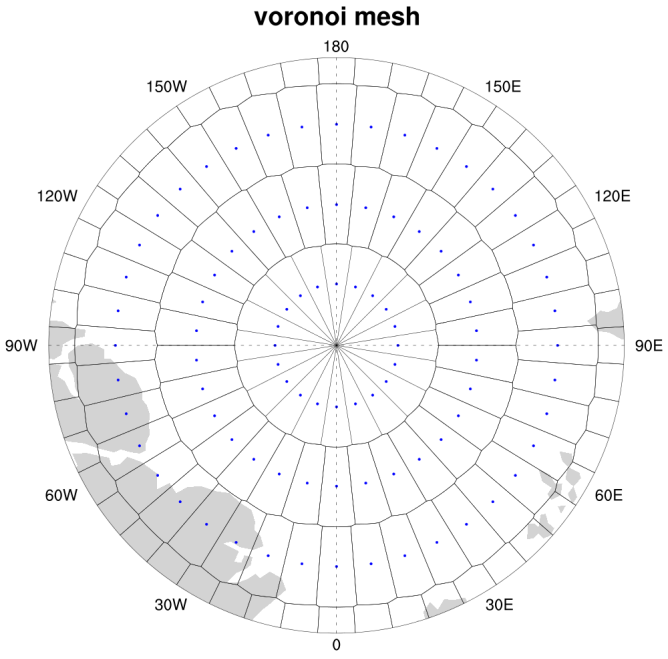
On voit bien la nécessité de définir des coins en plus pour la grille non-structurée. Pour la grille structurée, la première maille (notée 0) aura pour coins, conformément à la Figure 1, [0,1,2,3] (les coins sont définis dans le sens trigonométrique). La deuxième maille aura pour coins [4,5,6,7] etc. En revanche pour la grille non-structurée, la première maille devrait avoir 5 coins [0,26,1,2,3].
La SCRIP se contente (pour la grille non-structurée) des quatre coins ([0,1,2,3] par exemple) pour effectuer les interpolations sans se soucier des coins de la maille se trouvant à la latitude juste inférieure, on a pu ainsi trouvé des résultats (cf. la section précédente). Par contre, pour ESMF, a besoin de tous les coins ([0,26,1,2,3] par exemple) pour effectuer l'interpolation (conservative en particulier).

4.3.2 La fonction csvoro

Pour effectuer cette opération de redéfinition des "bons" coins, indispensable sur ESMF, nous avons utilisé la fonction `csvoro` présente dans les librairies de NCL (langage adapté utilisé pour ESMF). Cette fonction permet de générer automatiquement les coins nécessaires à notre interpolation (comme expliqué précédemment), en procédant à un découpage du plan (pavage) en cellules, où tous les coins sont pris en compte du moment qu'ils satisfont une distance minimale du centre de la maille. On va choisir dans la suite, de mener nos tests d'abord sur la grille non-structurée "bt42" et non pas "ssea", car elle contient 4 fois moins de cellules, ce qui réduit considérablement le temps de calcul.
Voici ce qu'on obtient avec la grille "bt42" dans un premier temps (au pôle nord) :



Il est clair que les coins au pôle nord sont "anormaux". On a en fait, 18 coins à la lat=90 et lon=0 pour 20 cellules (points bleus). Pour régler ce problème on rajoute deux coins (aux 18 coins) en fixant leurs longitudes (la longitude de ces 20 coins donc) aux longitudes des coins qui leur correspondent à la latitude juste inférieure (cf. `interp_esmf_plot_voronoi_modif_bt42.ncl` qui opère ces modifications présent dans `/scratch/globc/senhaji/tests_to_compare_esmf_scrip/.../curv_to_bt42_grids`). Ce programme sort un premier fichier `voronoi_bt42.nc` avec les coins originaux non modifiés, qu'il transforme pour obtenir un deuxième fichier `voronoi_bt42_new.nc` avec les coins arrangés. On obtient alors la figure suivante :



Une deuxième étape consiste à écrire le fichier `voronoi_bt42_new.nc`, en format ESMF, avant de pouvoir faire une interpolation. Le format d'un fichier ESMF est le suivant :

```
dimensions:
nodeCount = 8305 ;
coordDim = 2 ;
elementCount = 6232 ;
maxnodeperelement = 10 ;

variables:
double nodeCoords(nodeCount, coordDim) ;
nodeCoords:units = "degrees_east" ;

double centerCoords(elementCount, coordDim) ;

byte numElementConn(elementCount) ;

int elementConn(elementCount, maxnodeperelement) ;
elementConn:_FillValue = -1 ;

double elementArea(elementCount) ;

double elementMask(elementCount) ;
```

Les éléments les plus importants de ce fichier sont `elementConn`, `nodeCoords` et `numElementConn` :

- `elementConn` : c'est la matrice qui stocke l'adresse des coins de chaque maille. Chaque ligne représente une maille de la grille, et chaque colonne contient l'adresse du coin correspondant à cette maille. Cette matrice contient par défaut 10 colonnes, mais seules les 7 ou 8 premières colonnes sont remplies (7 ou 8 coins au maximum pour une cellule), le reste est fixé à une valeur spéciale appelée `Fill_Value` qui fait office de coefficient indisponible (et qui est représenté dans le fichier par un trait d'union).
- `numElementConn` : c'est le vecteur qui contient le nombre de coins par maille. Il a autant d'éléments que de cellules de la grille.
- `nodeCoords` : c'est une matrice à deux colonnes, où la première colonne contient les latitudes des coins des cellules et la deuxième colonne contient les longitudes des coins des cellules.

Le travail précédent s'est avéré insuffisant, puisque nous ne sommes pas parvenu à effectuer des interpolations avec les coins donnés par la fonction `csvoro`. Deux messages d'erreur revenaient sans cesse :

- **"clockwise polygons" :** ce message d'erreur remet en cause le sens de disposition des coins. De nouveaux fichiers ont été crées, avec des coins disposés dans le sens horaire et dans le sens trigonométrique, en refermant ou non la maille (en reprenant ou non le premier coin de chaque maille). On a aussi suspecté les coins des mailles se trouvant au pôle sud. On a donc crée un fichier avec les coins au pôle sud et un fichier sans. Mais dans tous les cas le problème persistait.
- **"entries are lower than 1" :** ce message renvoie au fait que certaines adresses de coins (dans `elementConn`) sont inférieures à 1. On a donc rajouté 1 à toutes les adresses, puis on s'est rendu compte que cela pouvait provenir des `Fill_Value` qui étaient fixés symboliquement à -1, et qui étaient considérés comme des coins. On a donc forcé `numElementConn` (le nombre de coins par maille) à ne prendre en compte que les "vrais coins", mais sans succès, on avait toujours le même problème ou, parfois, un retour au problème précédent.

Tous les fichiers utilisés pour faire ces tests sont disponibles dans le répertoire `curv_to_bt42_grids` et documentés dans le wiki. Le fait de ne pas connaître "réellement" la source du problème et de ne pas savoir la méthode utilisée concrètement par la fonction `csvoro` pour trouver les coins des mailles, nous a poussé à inspecter de nouvelles pistes pour résoudre notre problème.

4.3.3 Remaillage manuel

Puisque nous avons en notre disposition la définition des quatre coins (les coordonnées des coins) de chaque maille dans notre fichier de grille `"grids.nc"` ; nous avons décidé d'entreprendre un remaillage à la main. En effet, nous disposons dans le fichier de définition de la grille `"bt42"`, de deux tableaux de coordonnées des coins : `"bt42_clo"` et `"bt42_cla"` qui contiennent respectivement les longitudes et les latitudes des coins des cellules de la grille. Chaque tableau possède quatre colonnes, qui correspondent chacune à un coin de la cellule (colonne 1 pour le coin 0, ... cf. figures de l'introduction). Ces tableaux contiennent autant de lignes que de cellules constituant la grille.

Un programme NCL a été crée (`interp_bt42_to_esmfformat.ncl`) pour générer automatiquement un fichier de grilles en format ESMF (explicité précédemment), qui affecte à chaque maille l'adresse des coins qui lui correspondent comme expliqué dans l'introduction, et ce, à partir des tableaux de coordonnées de coins (`"bt42_clo"` et `"bt42_cla"`). Ce qui nous permettra, d'effectuer enfin nos interpolations. On explique ci-dessous les différentes étapes de ce programme :

Processus de définition des coins

Pour ce faire on a décidé de "déterminer" notre grille uniquement par le premier et le troisième coin de chaque cellule, ceci est suffisant pour obtenir tous les coins dont nous avons besoin sans prendre de coins "dupliqués" (des coins qui ont exactement les mêmes coordonnées). La figure ci-dessous illustre - à titre indicatif - notre méthode (notez que les adresses des coins commencent à 1 et non pas 0 pour respecter l'une des conditions du format ESMF) :

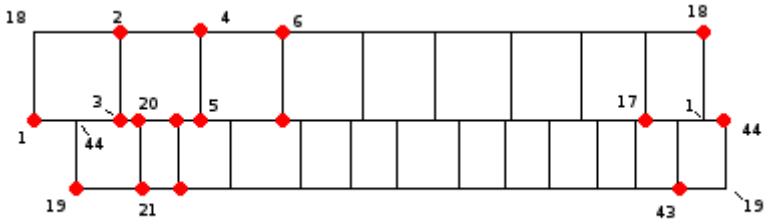


Figure (a) : Schéma explicatif de la définition des coins et de leurs adresses

Il faut imaginer que cette grille est "circulaire", et que donc le troisième coin de la dernière maille (coin 18 pour la première ligne) de chaque rangée se superpose avec le quatrième coin de la première maille correspondante et même chose pour le deuxième coin de la dernière maille et le premier coin de la première maille (coin 1 pour la première rangée).

Il s'est avéré que cette définition n'était pas tout à fait optimale, puisque sur certains intervalles de latitude, les mailles étaient alignées d'une latitudes à une autre. Nous avons décidé en conséquence, sur ces zones là uniquement, de ne prendre que le premier coin qui est suffisant pour la définition de la grille, comme le montre la figure suivante :

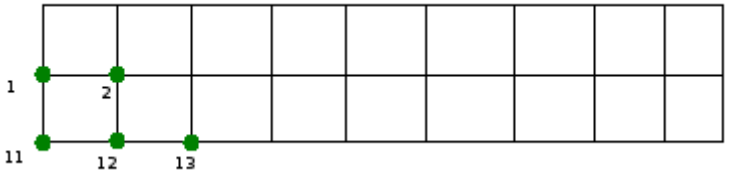


Figure (b) : Schéma explicatif de la définition des coins et de leurs adresses

Finalement on a découvert qu'on avait toujours un problème, car il existe des mailles qui sont alignés bien qu'on ne soit pas sur les zones explicitées précédemment. C'est ce qui ce passe pour le coin jaune ci-dessous (par exemple).

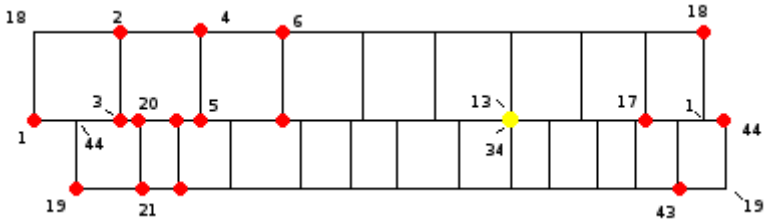


Figure (c) : Schéma explicatif de la définition des coins et de leurs adresses

L'étape de définition des coins n'a pas été achevée, faute de temps. Elle est en cours de développement. Cette partie sert en tous cas à définir notre `nodeCoords`.

Etape de recherche des coins pour chaque maille

On commence par la recherche des coins qui se trouvent sur la latitude minimale de chaque cellule c'est à dire les coins [1,44,3] pour la première maille (cf. figure (c)). Pour ce faire, on fait une boucle sur tous les coins, en imposant une condition sur leur latitude qui doit être égal à la latitude minimale de la maille, et une condition leur longitude qui doit être comprise entre la longitude minimale de la maille (la longitude du premier coin de la maille, ie. le coin 1 pour la première maille) et la longitude maximale de la maille (la longitude du troisième coin, ie. le coin 2 pour la première maille). On utilise la même méthode pour trouver les coins de la latitude maximale. On définit ainsi notre `elementConn` initial.

Etape de tri des coins dans le sens trigonométrique

On commence tout d'abord par classer les coins de la latitude inférieure de la maille par ordre croissant ; en classant leurs longitudes et en leurs réaffectant leurs adresses. On définit ensuite le nombre de coins sur la latitude inférieure, ie un `numElementConn` initial. On passe après, à la latitude maximale et on classe les longitudes des coins de manière décroissante, en leur réaffectant leurs adresses et en mettant à jour le nombre de coins (ie `numElementConn`). On obtient un tableau `elementConn` final, avec les bonnes adresses des coins triées dans le sens trigonométrique. On peut alors procéder aux interpolations.
Cette étape est intercalée dans le programme avec l'étape précédente.

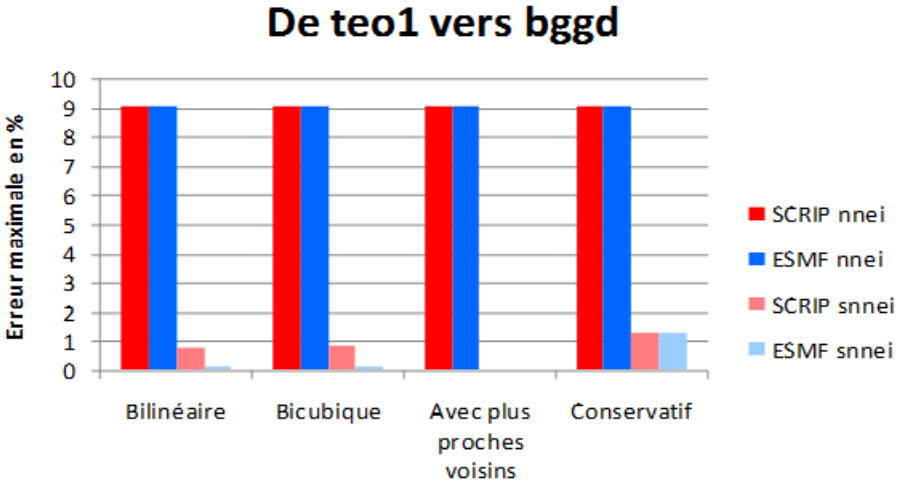
5 Récapitulatif et conclusion

Tableau récapitulatif des erreurs maximales pour chaque interpolation :

Ce tableau regroupe les valeurs des erreurs maximales des différentes interpolations (en noir), et en rouge là où les valeurs maximales fixées à chaque fois pour le tracé de l'erreur :

			bggd → *teo1*		teo1 → *bggd*		teo1 → *ssea*		ssea → *teo1*	
Bilinéaire	nnei	SCRIP	173.2	-	9.03	10/1	8.12	10/1		
		ESMF	0.94	1	9.03	10/1	-			
	snnei	SCRIP	173.2	-	0.8	1	0.86	1		
		ESMF	0.49	1	0.13	1	-			
Bicubique	nnei	SCRIP	100.59	-	9.03	10/1			0.46	0.5
		ESMF	0.94	1	9.03	10/1			-	
	snnei	SCRIP	100.59	-	0.81	1			0.46	0.5
		ESMF	0.49	1	0.12	1			-	
Avec + proches voisins	nnei	SCRIP	0.94	1	9.03	10				
		ESMF	0.94	1	9.03	10				
	snnei	SCRIP	-	-						
		ESMF	-	-						
Conservative	nnei	SCRIP	0.94	1	9.03	10/1.5	8.12	10/1	0.94	1
		ESMF	0.94	1	9.03	10/1.5	-	-		
	snnei	SCRIP	0.54	1	1.29	1.5	0.86	1	0.86	1
		ESMF	0.58	1	1.29	1.5	-	-		

On synthétise ce tableau dans le graphe suivant :



Conclusion

On remarque que les résultats d'erreur maximale sont identiques pour ESMF et la SCRIP, sauf pour les interpolations bilinéaire et bicubique sans plus proche voisin additionnel non-masqué, où les interpolations d'ESMF semblent plus précises que la SCRIP.

Ceci revient au fait que la SCRIP, dans la configuration "snnei", utilise les voisins non-masqués parmi les quatre voisins originaux alors que ESMF ne calcule simplement pas de valeurs pour les points cibles dans ce cas là.

6 Bibliographie

[1] S. Valcke, 2013 : *The OASIS3 coupler : a European climate modelling community software*, Geosci. Model Dev., 6, 373-388, doi :10.5194/gmd-6-373-2013. http ://www.geosci-model-dev.net/6/373/2013/gmd-6-373-2013.pdf

[2] S. Valcke, T. Craig and L. Coquart, 2015. *OASIS3-MCT User Guide, OASIS3-MCT3.0*, Technical Report TR/CMGC/15/38, Cerfacs, France.

[3] Hill, C., DeLuca, C., Balaji, V., Suarez, M., and da Silva, A. : *Architecture of the Earth System Modeling Framework, Comput Sci Eng*, 6(1), 18-28, 2004.

[4] *Description d'un banc d'essai permettant de quantifier la qualité des interpolations avec le coupleur OASIS4*, Sophie Valcke, Laure Coquart, octobre 2007

[5] *OASIS : le couplage océan-atmosphère*, Olivier Thual, Laurent Terray, juin 1995

[6] *Réalisation d'un modèle couplé océan-atmosphère avec le coupleur dynamique de codes parallèles Open-PALM*, Pierre Trespeuch, Septembre 2011