OASIS4 - Nouveau mode d'interpolation definie par l'utilisateur

juillet 2008

J. Latour

Rapport technique CERFACS
TR/CMGC/08/79

OASIS4

Nouveau mode d'interpolation définie par l'utilisateur

Juin / Juillet

2008

Jean Latour

HPC Consultant

Index

1) DEFINITION DE L' INTERPOLATION

Dans un couplage classique entre deux modèles, lors de l'échange d'un champ physique comme la température de surface de l'océan, il est nécessaire d'interpoler cette fonction entre la grille source et la grille cible. Cette opération est faite automatiquement dès lors que les descriptions des grilles sont connues, ainsi que leur partitionnement entre les processeurs. Le modèle cible reçoit les valeurs de la fonction source interpolées aux points définis dans sa propre grille.

Cependant certains couplages impliquent le passage de valeurs d'un champ entre quelques sous ensembles de points définis dans la grille source vers d'autres sous ensembles de points définis dans la grille cible. Les méthodes classiques d'interpolation ne peuvent être utilisées ici, car ces sous ensembles de points sont disjoints, et sont distribués de façon aléatoire sur les grilles. Pour ce type de couplage, l'utilisateur doit définir chaque lien nécessaire entre un point de la grille source et un point de la grille cible, et affecter ce lien d'un poids qui servira à calculer la fonction sur la grille cible, à partir des valeurs choisies sur la grille source.
Cette description est faite dans un fichier contenant les poids et les adresses des points sources et cibles

Un exemple très simple sur la figure 1 permet de comprendre le principe de cette interpolation définie par l'utilisateur.

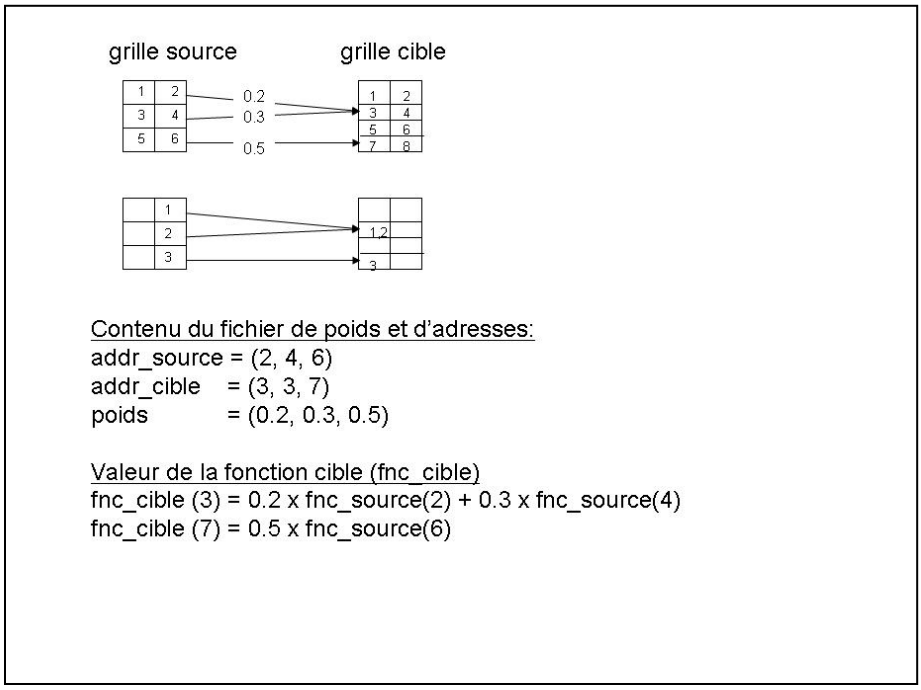

Figure 1

A gauche et à droite de la figure sont représentées des composantes modèles très simplifiées dont les grilles géographiques ont respectivement 6 et 8 mailles. Un fichier de poids et

d’adresse prédéfini par l’utilisateur décrit 3 liens entre la grille source et la grille cible. Les liens 1, 2, 3 associent respectivement les mailles sources 2, 4, 6 aux mailles cibles 3, 3, et 7 avec des poids de 0.2, 0.3, et 0.5. Le remaillage donné par ce fichier de poids et d’adresses doit donc donner la fcn_cible(x) à partir des valeurs de la fnc_source(y) :

fnc_cible(3) = 0.2 x fnc_source(2) + 0.3 x fnc_source(4)

fnc_cible(7) = 0.5 x fnc_source(6)

Dans OASIS4, cette fonctionnalité peut s’implémenter assez simplement en exploitant la possibilité d’OASIS4 de redistribuer des champs de couplage simplement associés à un espace global unique du coté source et du coté cible (c’est ce qu’on appelle des champs « gridless »). Dans l’exemple de la figure 1, il suffira donc d’annoncer du coté source et du coté cible un champ « gridless » de dimension 3 en associant aux points 1, 2, et 3 respectivement les points 2, 4, et 6 de la grille géographique du coté source et les points 3, 3, et 7 de la grille géographique du coté cible.

Une implémentation de cette fonctionnalité nouvelle dans OASIS4 est décrite ici. Elle s'appuie sur les fonctions existantes de la librarie Psmile et définit de nouvelles routines que l'utilisateur doit explicitement appeler dans son application. Dans un deuxième temps cette fonctionnalité pourra être prise en compte implicitement par la librarie Psmile (sous les appels à PRISM_Put et PRISM_get) à partir de la description du fichier de poids et d'adresses associé au champ qui doit être échangé entre les modèles quand elle est spécifiée par l’utilisateur dans le fichier de configuration XML du couplage..

2) IMPLEMENTATION EXPLICITE

2.1) Suite des opérations

Le transfert de l'information entre le champ source et le champ cible par l'intermédiaire des liens définis par l'utilisateur est semblable à un adressage indirect du type :

Target_fnc[Ind_trg(i)] = Source_fnc[Ind_src(i)] * Poids(i) + valeur pour (i-1) (1)

Cette expression est répétée sur l'ensemble des liens : "nliens" définis par l'utilisateur. Le fichier fourni contient donc les tableaux d'indices (éventuellement à 2 et 3 dimensions suivant les grilles) : Ind_src(nliens), Ind_trg(nliens) et Poids(nliens).
Il est important de noter que les indices de ce fichier sont les indices sur les grilles globales, coté source et coté cible. Le calcul du membre de droite de l'égalité (1) se fait sur les processeurs sources, et le report dans la fonction Target_fnc se fait dans les processeurs cibles. La somme "+ valeur pour (i-1)" n'a de sens que coté cible car on peut définir plusieurs liens, à partir de points sources différents, vers une même cellule cible. Elle est donc faite après le Prism_get. Il est possible de définir aussi plusieurs liens partant de la même cellule source, mais cela ne correspond pas à une somme coté source.

L'ensemble des valeurs calculées par les processeurs sources constituent une fonction définie sur les indices 1 à nliens, qui est appelée fonction "gridless". Une "grille gridless" se résume à un indiçage global pour une fonction à transmettre. Cet indiçage global peut être partitionné entre les processeurs cibles et sources, et les fonction Prism_put et Prism_get permettent la reconstitution de ces partitions coté cible. Cependant, dans l'état actuel, les partitions ainsi définies sur des "gridless" doivent constituer un seul sous ensemble continu d'indices dans chaque processeur. La fonction "gridless" à transmettre sera donc partitionnée simplement en intervalles égaux à nliens/NP (plus ou moins une unité) si NP est le nombre de processeurs, nombre qui est en général différent entre la source et la cible.

En ce qui concerne la fonction initiale "Source_fnc", le partitionnement de la grille sur laquelle elle est définie entre les processeurs sources, limite l'usage de la formule (1) aux indices situés dans les limites de la partition de chaque processeur. Un seul processeur source ne peut calculer que la partie du champ "Target_fnc" qui correspond aux indices "Ind_src" de sa partition. Sa contribution à la fonction "gridless" intermédiaire se fera en général pour des indices (de l'intervalle 1-nliens) qui peuvent être discontinus, car l'indiçage des liens dans le fichier de poids et d'adresses est tout à fait arbitraire. Pour pouvoir créer un partitionnement correct de la fonction "gridless" en un seul domaine continu par processeur, il faut donc d'abord reconstituer la fonction gridless complète dans chaque processeur. Ceci peut se faire par la fonction MPI_Allreduce entre les processeurs sources. Après cette opération globale coté source, chaque processeur contient l'ensemble de la fonction gridless pour tous les liens définis par l'utilisateur. Celle-ci peut alors être partitionnée en intervalles égaux à nliens/NP avant d'être transmise par un PRISM_Put.

Coté cible, le même problème se pose : chaque processeur va contenir une partie continue (en indices) de la fonction "gridless", mais l'indirection Ind_trg(i) doit être faite pour tous les indices de la fonction finale qui sont dans sa partition de la grille (Reglonlatvrt ou autre) sur laquelle est définie la fonction Target_fnc. Il faut donc que chaque processeur cible traite TOUS les indices de la fonction gridless (1à nliens) pour détecter tous les liens qu'il peut

traiter. Ici encore il faut donc reconstituer, dans chaque processeur cible, par une opération MPI_Allreduce, l'ensemble de la fonction gridless intermediaire, avant de traiter la formule (1) pour tout les indices Ind_trg(i) locaux à sa partition.

2.2) Description des fonctions

Dans un premier temps, nous avons donc développé les routines suivantes qui réalisent cet échange d'un champ « gridless » et qui doivent être utilisées explicitement par l'utilisateur dans les codes des composantes source et cible pour le champ de couplage en question (ceci demande donc de savoir a priori qu'un fichier de poids et d'adresse prédéfini sera utilisé pour l'interpolation de ce champ) :

- prism_init_usefdef.F90 (à appeler dans le code source et cible avant le prism_enddef) :
  - lecture du fichier de poids et d'adresses ; stockage pour chaque lien des adresses sources et cibles et des poids
  - définition et déclaration de la grille « gridless » dimensionnée du nombre de liens (prism_def_grid, prism_def_partition, prism_set_point_gridless)
  - déclaration du champ « gridless » (prism_def_var)
- prism_put_userdef (à appeler dans la boucle temporelle du code source) :
  - constitution du champ « gridless » en fonction du champ géographique source et des informations du fichier de poids et d'adresses
  - envoi du champ « gridless »
- prism_get_userdef (à appeler dans la boucle temporelle du code cible) :
  - réception du champ « gridless »
  - reconstitution du champ géographique cible en fonction du champ « gridless » et des informations du fichier de poids et d'adresses

Dans la version actuelle, l'intégralité du champ gridless est rassemblé sur tous les processeurs sources et cibles avant et après l'envoi, grâce à la fonction MPI_Allreduce; puis chaque processeur envoie ou reçoit une sous-partie continue du champ. Cette étape intermédiaire est nécessaire car la librairie de communication ne supporte pas pour l'instant l'envoi ou la réception par un processeur de plusieurs morceaux non contigus de champ gridless ce qui peut a priori être le cas selon le fichier de poids et d' adresses et la distribution des grilles géographiques source et cible. Nous espérons dans les prochains mois lever cette restriction en permettant l'usage de sous-domaines disjoints dans chaque partition gridless, et supprimer ainsi l'étape du MPI_Allreduce qui peut être couteuse en mémoire.

Ces routines ont pour l'instant été testée dans des cas simples tous concluants (voir les programmes source et cible sur http://www.cerfacs.fr/cicle/trunk/tests/grless_links_mod).

3) IMPLEMENTATION IMPLICITE DANS OASIS4

3.1) Intégration future dans l'interface d'OASIS4

Dans un 2e temps, nous nous proposons d'incorporer ces étapes directement dans les routines d'interface de la librairie de communication existantes. La définition d'une grille et d'un champ «gridless » se feront alors automatiquement lorsque l'utilisateur aura indiqué dans le fichier de configuration XML qu'il choisit ce type d'interpolation pour un champ en particulier ; ce fonctionnement sera plus conforme à la philosophie d'OASIS4 qui veut que les codes sources et cibles n'aient a priori besoin d'aucune connaissance quant à la source, cible, ou transformations des champs qu'ils reçoivent ou envoient.