Compte-rendu des réunions et ateliers du projet MARMO, Toulouse juillet 2010

Sébastien Theetten (sebastien.theetten@ifremer.fr)

Patrick Marsaleix (patrick.marsaleix@aero.obs-mip.fr)

Rachid Bensila (rachid.benshila@locean-ipsl.upmc.fr)

Version 1.1, 2septembre 2010

Introduction

Une des conclusions de la réunion du projet COMODO (COmmunauté de MODélisation Océanographique) qui s'est déroulée à Autrans en mars 2010 est qu'**il est indispensable de fixer un cadre technique pour que les échanges entre modèles puissent se faire**. Les échanges entre modèles couvrent de nombreux aspects comme l'échange de fichier et l'utilisation des outils de pre- et post- processing commun.

Afin de poser des bases communes, le projet MARMO (Moyens d'Aide à la Réalisation de la Modélisation Océanique) qui est un des volets techniques du projet COMODO se propose de rédiger les premières spécifications. Une réunion du projet MARMO s'est tenue à Toulouse en juillet 2010 pour **rassembler les intervenants des différentes communautés de modèlisation au sein de COMODO afin de converger vers des choix consensuels**.

Ce document résume l'ensemble des réflexions qui ont eu lieu entre les différents participants. Ces discussions ont amenés des conclusions que nous avons concrétisé en proposant des spécifications techniques sur les fichiers. **Nous présentons un en-tête de fichier informatique synthétisant les choix** qui ont été faits au cours de ces ateliers. Il reste encore quelques spécifications à définir. Ces **dernières décisions seront prises en concertation avec tous les membres de COMODO.**

Ces réunions d'échanges inter-modèles ont aussi permis de tenir une **liste des outils de pre- et post-traitement utilisés dans les différentes communautés** qui seraient susceptibles d'être disponible par les membres des autres communautés.

L'ADOPTION PAR L'ENSEMBLE DE LA COMMUNAUTE COMODO DES CONVENTIONS DETAILLEES DANS CE DOCUMENT PERMETTRA ALORS DE FACILITER DAVANTAGE LES ECHANGES SCIENTIFIQUES ET TECHNIQUES ENTRE LES DIFFERENTES COMMUNAUTES.

Les participants

Le nom des personnes ayant participé à ces ateliers figure sur la liste ci-dessous. Ces ateliers ont permis de réunir **une vingtaine de chercheurs et d'ingénieurs** qui participent au développement des principaux outils numériques utilisés par la communauté française : **MARS, NEMO, codes SIROCCO, TUGO, HYCOM, ROMS**. Le nombre conséquent de participants et leur implication au sein des différentes communautés a permis de converger vers un consensus dans les spécifications techniques.

Rachid Benshila (LOCEAN, Paris, code NEMO) et **Sébastien Theetten** (IFREMER, Brest,

code MARS) se sont déplacés à Toulouse du 28 juin au vendredi 2 juillet accueillis par **Patrick Marsaleix** (Laboratoire d'Aérologie pour les codes de la tâche de service SIROCCO) pour rencontrer (par ordre d'apparition) :

Florent Lyard (LEGOS, code TUGO, Xscan) florent.lyard@legos.obs-mip.fr
Cyril Nguyen (OMP Laboratoire d'Aérologie /LEGOS, codes de la tâche de service SIROCCO) Cyril.Nguyen@aero.obs-mip.fr
Caroline Ulses (Laboratoire d'Aérologie, codes de la tâche de service SIROCCO) caroline.ulses@aero.obs-mip.fr)
Gildas Cambon (IRD, code ROMS, pre et post traitement (roms_tools, roms_gui)) Gildas.Cambon@ird.fr
Patrick Marchesiello (IRD, code ROMS) Patrick.Marchesiello@ird.fr
Francis Auclair (Laboratoire d'Aérologie, tâche de service SIROCCO: modélisation Non-Hydrostatique et outils de pre- et post- traitement (VIFOP, Sview) francis.auclair@aero.obs-mip.fr
Laurent Roblou (LEGOS), laurent.roblou@legos.obs-mip.fr
Stéphanie Corréard (SHOM, pré et post traitement HYCOM) stephanie.correard@shom.fr
Rémy Baraille (SHOM, code HYCOM) remy.baraille@shom.fr
Flavien Gouillon (SHOM) flavien.gouillon@shom.fr
Edmée Durand (Mercator) durand.edmee@mercator-ocean.fr
Julien Paul (Mercator) paul.julien@mercator-ocean.fr
Guillaume Reffray (Mercator) guillaume.reffray@mercator-ocean.fr
Gaetan Vinay (Mercator) gaetan.vinay@mercator-ocean.fr
Guillaume Riflet (MOHID)

Etat de l'art

Dans chaque communauté, des outils *ad hoc* effectuent le travail de pré-traitement pour chaque code. Les technologies utilisées dans la confection de ces outils sont variées (Fortran, C, Python, Java, Matlab, IDL, Ferret, librairies spécifiques, etc). Les modélisateurs utilisent très pragmatiquement l'outil qui leur est proche. La bonne adéquation des outils avec les problèmes rencontrés est un des critères importants, sachant qu'un outil développé spécifiquement pour un code permet en principe de bien répondre au besoin de chaque communauté. D'autres critères pris en compte pour le choix d'un outil sont les affinités d'un modélisateur avec tel ou tel langage informatique, la disponibilité d'un logiciel payant, les contraintes techniques de la plateforme, etc.

Un autre constat est que les fichiers produits après cette étape de prétraitement ont des caractéristiques que l'on retrouve assez systématiquement dans toutes les équipes de développement. **L'utilisation du format Netcdf et de certaines conventions reconnues par les communautés internationales est par exemple un dénominateur commun**.

Objectifs

Les obstacles pour que ces outils soient utilisés au delà de leur groupe d'origine sont loin d'être infranchissables. Notre objectif est de proposer des solutions pour faciliter leur diffusion. L'idée est par exemple qu'un logiciel de génération de grille, initialement conçu pour un modèle en particulier, puisse être indifféremment utilisé pour tous les modèles. L'intérêt est d'une part de décloisonner les outils pour que les développeurs, et bien plus

largement la communauté des utilisateurs, puissent profiter de la diversité actuelle en ayant un accès facile aux meilleurs outils et d'autre part, pour ce qui concerne les outils futurs (voir par exemple la tidal tool box proposée par le LEGOS), d'éviter des efforts redondants aux équipes qui souhaiteront se recentrer sur leurs objectifs prioritaires (ceux ci concernent rarement les outils de Post/Pre Traitement).

Nous proposons des solutions afin que chaque communauté et chaque modélisateur puissent **utiliser n'importe quel outil pour préparer ses données qui seront compatibles avec son modèle**.

L'ambition est d'arriver à mettre d'accord tout le monde sur l'idée d'une convention pour le maillage bathymétrique et le fichier de coordonnées horizontale avec pour démonstration de cette réussite de standardisation la possibilité d'utiliser un même outil pour les différents codes. Cette interopérabilité n'est pas qu'un exercice formel. Le fait de pouvoir utiliser d'autres outils et de développer les siens de manière à ce qu'ils soient utilisable par d'autre est une possibilité de confronter facilement différentes solutions et de les améliorer éventuellement.

Nous avons traité uniquement le cas d'un fichier simple n'ayant que deux dimensions horizontales avec un nombre restreint de variable. Si cette tentative de standardisation s'avère concluante pour ce simple fichier d'entrée, un travail similaire pourrait être envisagé pour les autres fichiers d'entrées et surtout pour les fichiers de sorties des différents modèles.

En résumé, voici les trois objectifs que nous nous sommes fixé :

- se mettre d'accord sur une convention dans le contenu des fichiers en particulier les fichiers d'entrée. Nous avons mis l'accent sur le fichier de grille horizontale, de bathymétrie et de masque prioritairement.

- faire un bilan des outils de pré-traitement (en priorité) et de post-traitement disponible pour le reste de la communauté.

- recenser les efforts de développement nécessaires afin que les outils de chaque communauté puissent être utilisés par les autres communautés.

Spécifications pour le fichier grille horizontale, bathymétrie et masque

Format informatique

Actuellement, la communauté utilise une des versions de la **librairie netCDF 3.6.x**[1]**.** C'est donc naturellement ce format informatique qui est retenu. L'adoption de la version netCDF 4 est discutée au sein de plusieurs communautés. L'utilisation de cette version 4 ne doit pas conduire à des régressions.

Conventions

Nous avons pu constater que l'ensemble des communautés utilise plus ou moins des conventions qui sont celles d'une convention existante, à savoir celle de « Climate and Forecast[2] ». Cette convention est une extension et une généralisation des précédentes conventions « GDT » et « COARD » utilisées par certaines communautés dès l'adoption du format netCDF. Par ailleurs, les contraintes très fortes sur les systèmes opérationnels PREVIMER et MERCATOR ont conduit ces équipes à adopter cette convention pour pouvoir garantir les différents services en aval. Par exemple, Mercator a des contraintes de convention qui sont imposées par MyOcean. La « Climate and Forecast » version 1.0 est obligatoire et la version 1.4 est conseillée. Le passage de la version 1.0 à la version 1.4 est essentiellement une extension de la liste de `standard_name`. L'utilisation de cette convention est indispensable pour les services de diffusion via opendap. De plus, le développement de la majeure partie des outils permettant de manipuler les fichiers netcdf qui proviennent d'Unidata se base sur les spécifications de cette convention Climate and Forecast. Concernant le Centre de Données d'Océanographie Côtière Opérationnel de PREVIMER, le format Netcdf s'appuie aussi sur les conventions de la communauté « Climate and Forecast ». Cela nous conforte dans l'adoption de cette convention. Nous proposons de généraliser l'utilisation de cette convention en amendant éventuellement cette convention avec des conventions communes et spécifiques à la communauté COMODO. Il s'agit de bénéficier très facilement des apports de la convention « Climate and Forecast » et de rester très libre sur des ajouts dans cette convention. **Nous proposons donc d'utiliser la version 1.4 de la convention « Climate and Forecast » avec des ajouts propres à COMODO qui devront être adoptés d'une façon unanime par l'ensemble de la communauté.**

Nombre de fichier

Il a été convenu qu'un seul et unique fichier contienne les informations concernant la grille horizontale ainsi que les variables associées à la bathymétrie et aux masques (toutes ces variables étant bidimensionnelles). Un fichier unique a l'avantage de faciliter les échanges et le chargement dans les différentes applications. De plus, les risques d'erreurs lors des transferts sont minimisés. D'autre part, il existe des commandes simples permettant de manipuler rapidement des fichiers netcdf comme la concaténation de plusieurs fichiers ou l'extraction d'une ou plusieurs variables. Cette souplesse permet donc que chaque

[1] http://www.unidata.ucar.edu/software/netcdf/

[2] http://www.cfconventions.org/

communauté ne change pas ses habitudes.

Type de grille de calcul

Chacun des modèles HYCOM, MARS, NEMO, ROMS, codes SIROCCO utilise un certain « type » de grille ARAKAWA-C sur laquelle sont renseignées différentes variables**. Il s'agit donc de renseigner l'information concernant le type de grille ARAKAWA-C.**

Des propositions pour un décodage rapide des règles de numérotation :

Tout d'abord une description succincte du problème:

Les variables ne sont pas co-localisées dans le cas de la grille C. Les règles de numérotions des points de grille varient d'un modèle à l'autre selon que la grille commence par un point de vitesse ou par un point de traceur. Si la grille commence par un point de vitesse, la numérotation est par exemple du type: u(1) t(1) u(2)....u(i) t(i) u(i+1)... auquel cas un point i de température est situé entre les vitesses u aux points i et i+1. Inversement si la grille commence par un point de température (t(1) u(1) t(2).....) un point i de température est situé entre les vitesses aux points i-1 et i. On retrouve ce problème dans l'archivage des sorties des modèles dans les fichiers netcdf. Ceci ne pose pas de difficultés particulières à l'intérieur d'un même système de modélisation car modèle et outils de Post/Pre Processing (P3-tools) sont en principe cohérents. Par contre cette cohérence n'existe pas à l'échelle de la communauté nationale. La diversité de convention de numérotation conduit en effet dans la plupart des cas à ce qu'un P3-tool développé pour un modèle en particulier ne reconnaît pas spontanément les conventions des autres modèles, ce qui entrave ses possibilités de diffusion au sein de la communauté nationale.

L'attribut `Coordinates`

Si la construction du fichier netcdf est bien faite, c'est à dire si la position géographique des variables (longitude, latitude, profondeur) est correctement renseignée (avec l'attribut `coordinates`), un P3-tool peut toutefois interpréter ces informations de positionnement afin de rétablir une règle de numérotation cohérente indispensable pour mener à bien ses calculs. Le bilan de cette réunion COMODO permet déjà de dire que presque tous les modèles ont adopté un archivage géoréférencé complet. C'est un bon point de départ : moyennant quelques adaptations, presque tous les P3-tools dont il a été question lors de cet atelier pourront être utilisés par tous les modèles de la communauté.

Ceci étant dit, retrouver la règle de numérotation à partir de l'analyse de la position géographique des points la grille peut soulever quelques critiques. Cette méthode suppose une étape préalable qui nécessite la lecture des longitudes, latitudes et profondeurs de tous les points de grille. Si tant est que ces informations soient présentes (critique 1: *et si ces infos ne sont pas contenues dans le fichier?*), l'opération a un coût en cpu et en mémoire vive (critique 2). Ensuite, sans être excessivement complexe, le codage de l'opération peut paraître fastidieux (critique 3: *les équipes R&D auront elles la détermination de l'implémenter?*) et comporte des risques (à ne jamais négliger) de bugs (critique 4).

Un nouvel attribut pour décrire le type de grille C

Nous avons réfléchis à la création d'attributs qui faciliterait le décodage de la grille C (c'est à dire qui éviteraient que les P3tools aient à passer par l'interprétation des informations

géographiques). A ce stade très préliminaire il ne s'agit que de simples propositions sur lesquelles la communauté COMODO pourra s'exprimer. Deux types d'attributs ont été envisagés

L'attribut global `arakawa_grid_type`

Cet attribut global jouerait le rôle de nomenclature pour les différentes manières d'archiver une grille C. Il pourrait par exemple prendre pour valeur C0 ou C1 ou C2, etc (les participants COMODO s'exprimeront sur le sujet) selon le type de grille. Nous présentons plus loin un état des lieux des différentes formes d'archivages recensées au cours de notre atelier. On y voit que les configurations sont nombreuses, selon que l'archivage commence par une variable ou une autre, ou que les fichiers incluent ou non des zones latérales fantômes, ou compte un nombre de point de grille identique ou non selon les variables.

L'attribut `c_grid_i_axis_shift`

Cette seconde méthode suppose de créer un attribut netcdf de "décalage" pour chaque variable. Par convention, on décide arbitrairement que la grille des traceurs est la grille de référence (dans notre méthode cela se traduit par "*décalage nu*"*l* ou "*shift nul*"). Rappelons que dans le cas d'une programmation fortran classique, le premier point trouvé par le logiciel X sur la grille horizontale des traceurs est en principe numéroté (i,j) = (1,1). L'idée est que lorsque le logiciel X lit une variable située sur la grille des vitesses il interroge également l'attribut netcdf de décalage associé à la variable. Si cette variable est la vitesse u, l'interrogation de l'attribut *c_grid_i_axis_shift* renvoie + ou - ½, indiquant ainsi si le premier point de grille u est situé ½ maille en avant ou en arrière du premier point de traceur (vis à vis de l'axe de direction des indices i). Le logiciel X saura alors numéroter le premier de vitesse (en pratique soit i=0, soit i=1) en fonction de ses propres règles de numérotation. Des figures récapitulant les différentes configurations peuvent être consultées sur :

(http://sirocco.omp.obs-mip.fr/outils/Symphonie/Documentation/SymphonieDocnetcdf.htm)

Le Pour et le Contre:

L'avantage de la deuxième solution est d'obtenir une solution unique et non ambiguë pour le décodage des différentes grilles. Cela évite donc les séries de tests dans les applications tel que les tests conditionnels `if then`. Cette solution potentiellement puissante proposée par Patrick Marsaleix est encore en gestation.

La première solution est beaucoup plus simple à mettre en œuvre dans la constitution du fichier puisque la définition d'un attribut global reporte en fait le décodage des grilles dans les applications qui devront comporter une interface basée sur le test de la valeur de l'attribut global `arakawa_grid_type`.

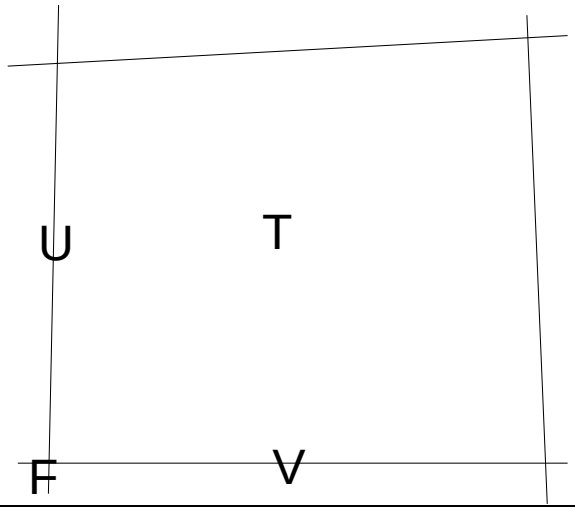
Ces deux solutions ne sont pas exclusives. En effet, le format netCDF permet d'ajouter très simplement des attributs supplémentaires. Il est donc envisageable que ces deux solutions cohabitent dans le même fichier

Dans la suite de ce document nous utilisons la première solution qui est l'utilisation d'un nouvel attribut global `arakawa_grid_type`

Description des différents types de grille ARAKAWA

- **`arakawa_grid_type` = « C0 » option « u » ou option « t »**

C'est l'une des deux conventions principalement adoptées pour les codes du groupe **SIROCCO** et par la communauté du modèle **ROMS,**



On note que le point F n'est pas systématiquement présent dans le fichier grille car il n'y a en principe pas de variables d'état définies au point F dans ce type de codes. Cependant, la connaissance des coordonnées de ce point peut être utile, selon les approches, pour la construction de la grille où pour certaines contingences de la représentation graphique.

Ce qui caractérise ce mode d'archivage c'est qu'il respecte l'esprit de la grille C dans le fait que le nombre total de points de grille n'est pas le même pour toutes les variables d'état.

Les tableaux de variables sont dimensionnés aux vraies dimensions, c'est-à-dire qu'il existe plusieurs paires de dimension (ni_t,nj_t), (ni_u, nj_u) , (ni_v, nj_v) et (ni_f, nj_f) avec :

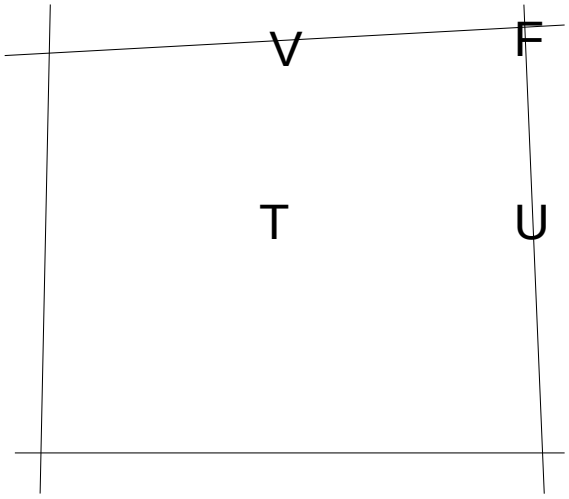
ni_u = ni_t + 1
ni_v = ni_t
ni_f = ni_t +1
nj_u = nj_t
nj_v = nj_t +1
nj_f = nj_t + 1

Cette première convention est basée sur une grille commençant et finissant par les points de vitesse (C0 « u »). A celle-ci s'ajoute une seconde variante, également exploitée par ces mêmes équipes, basée sur une grille commençant et se terminant par un point de traceur (CO « t »). Dans ce cas les 4 paires de dimensions sont telles que:

ni_u = ni_t - 1
ni_v = ni_t
ni_f = ni_t -1
nj_u = nj_t
nj_v = nj_t -1
nj_f = nj_t - 1

- **arakawa_grid_type = « C1 »**

C’est le type de grille utilisée par **MARS**



Toutes les variables ont les mêmes dimensions (ni_t+1, nj_t +1)

A la lecture, on néglige la première ligne (j=0) de T et première colonne (i=0) de T

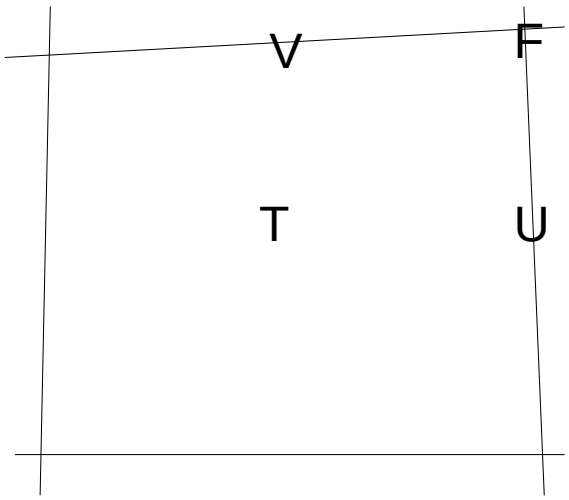
on néglige la première ligne (j=0) de U

on néglige la première colonne (i=0) de V

Les points T sont entourés de points U.

- arakawa_grid_type = « C2 »

C’est le type de grille utilisée par **NEMO**



Toutes les variables ont les mêmes dimensions (ni_t + 2, nj_t +2)

Pour les points T, on masque la première ligne (j=1),

la dernière ligne (nj_t+2),

la première colonne (i=1)

et la dernière colonne (ni_t+2)

Pour les points U, on masque l'avant-dernière colonne i=ni_t+1,

la première ligne (j=1),

la dernière ligne (nj_t+2),

la première colonne (i=1),

et la dernière colonne (ni_t+2)

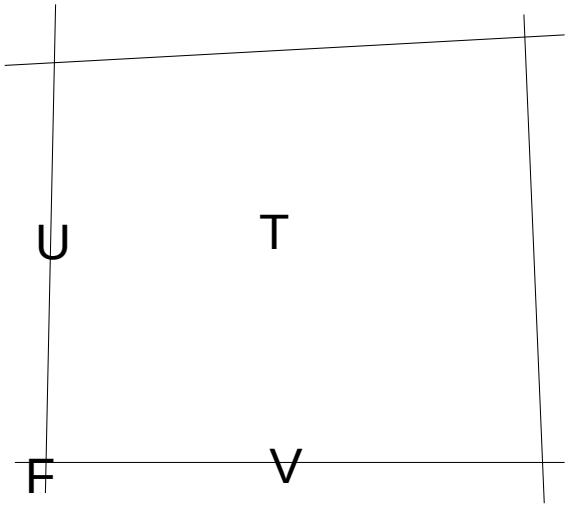
Pour les points V, on masque l'avant-dernière ligne (j=nj_t+1),

la première ligne (j=1),
la dernière ligne (nj_t+2),
la première colonne (i=1),
et la dernière colonne (ni_t+2)

Les points U sont « entourés » de points T.

- **arakawa_grid_type = « C3 »**

C'est le type de grille utilisée par **HYCOM**



Toutes les variables ont les mêmes dimensions ni_t, nj_t

Il existe donc à peu près le même nombre de grille ARAKAWA-C qu'il existe de modèles dans la communauté COMODO. Aucun changement dans le type de grille utilisée par chaque modèle n'est envisagé pour converger vers le même type de grille ARAKAWA-C. Si cette idée devait quand même se concrétiser dans le long terme, il semble que le type C0 qui définit chaque variable sur ses propres dimensions paraît être

une bonne solution. Le fait d'avoir plusieurs dimensions suivant les points de maille rendrait quand même sensiblement plus compliqué les tâches de post-traitement.

Contenu du fichier

Une fois défini le type de grille ARAKAWA, nous décrivons les conventions qui sont communes et standard au sein de la communauté COMODO. Nous rappelons que les conventions « Climate Forecast » V1.4 sont utilisées par défaut et si besoin nous rajoutons d'une façon consensuelle des attributs supplémentaires. Ici, les nouveaux noms d'attributs sont issus d'une première réflexion dans le cadre de la réunion MARMO-COMODO à Toulouse, juin 2010.

Les **noms de variables et les noms des dimensions** utilisés par la suite sont **INDICATIFS.** Chaque communauté continue à utiliser les noms de variables et les noms de dimension qui leur sont propres. En revanche, la présence de l'attribut **`standard_name` est obligatoire** pour chaque variable présente. Tous comme les noms des attributs, les valeurs de ces attributs respectent la convention CF éventuellement amendée. **Les applications utilisant ce fichier n'ont qu'à interroger la valeur de cet attribut `standard_name` pour identifier la variable.**

C'est ce mécanisme d'association entre chaque variable et le `standard_name` qui est capital.

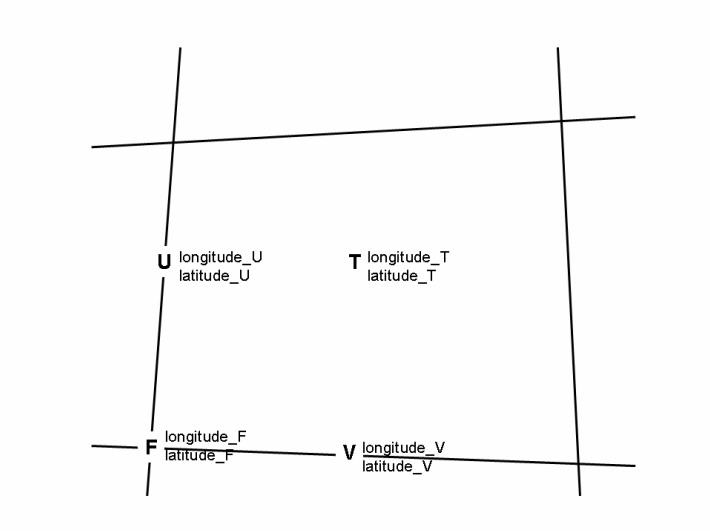
Précision numérique des variables

Nous rappelons que nous traitons que les variables contenues dans le fichier de grille horizontale et de maillage bathymétrique.

Concernant la précision des variables de coordonnées et des taille de maille (appelé aussi "facteurs d'échelle", il est fortement recommandé d'utiliser la double précision (**double**). Pour les masques la précision pourra être au choix **int** ou **byte**. La déclaration en simple précision (**float**) pour la bathymétrie est suffisante mais ne doit pas être un critère bloquant pour l'utilisation de ce fichier par les applications. L'idée de généraliser l'utilisation de la double précision à toutes les variables de ce fichier (sauf pour le masque) permet d'homogénéiser ce fichier tout en n'augmentant pas de façon démesurée la taille de ce fichier.

Description des variables contenues dans le fichier

Variable de coordonnées :



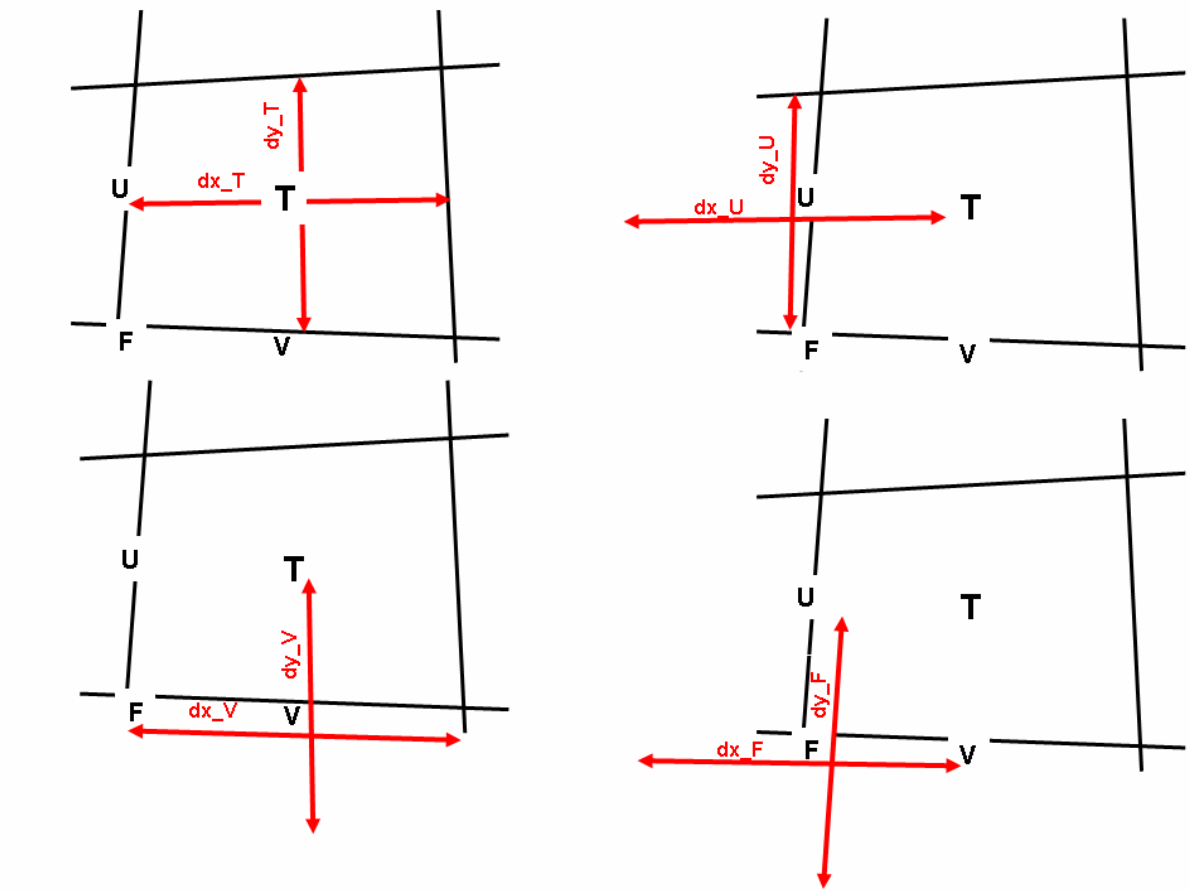
En chaque point T, U et V de la grille sont définies les coordonnées géographique longitude et latitude de ces points.

Ces variables de coordonnées peuvent porter n'importe quel nom. Seul l'attribut `standard_name` décrit précisemment la variable avec une valeur pour ce `standard_name` qui doit correspondre à une des valeurs de la liste ci-dessous :

`longitude_t`
`longitude_u`
`longitude_v`
`longitude_f`
`latitude_t`
`latitude_u`
`latitude_v`
`latitude_f`

Ces valeurs de cet attribut `standard_name` ne sont pas des valeurs standard de la convention CF. En effet, la convention CF ne permet pas de stocker dans un seul fichier plusieurs systèmes de coordonnées correspondant à la grille Arakawa. Les seules valeurs standard concernant les coordonnées sont « longitude » et « latitude ». Nous proposons d'utiliser les valeurs ci-dessus qui constituent le **premier amendement à la convention CF**.

Variable de taille de maille ou ″facteur d'échelle″ :

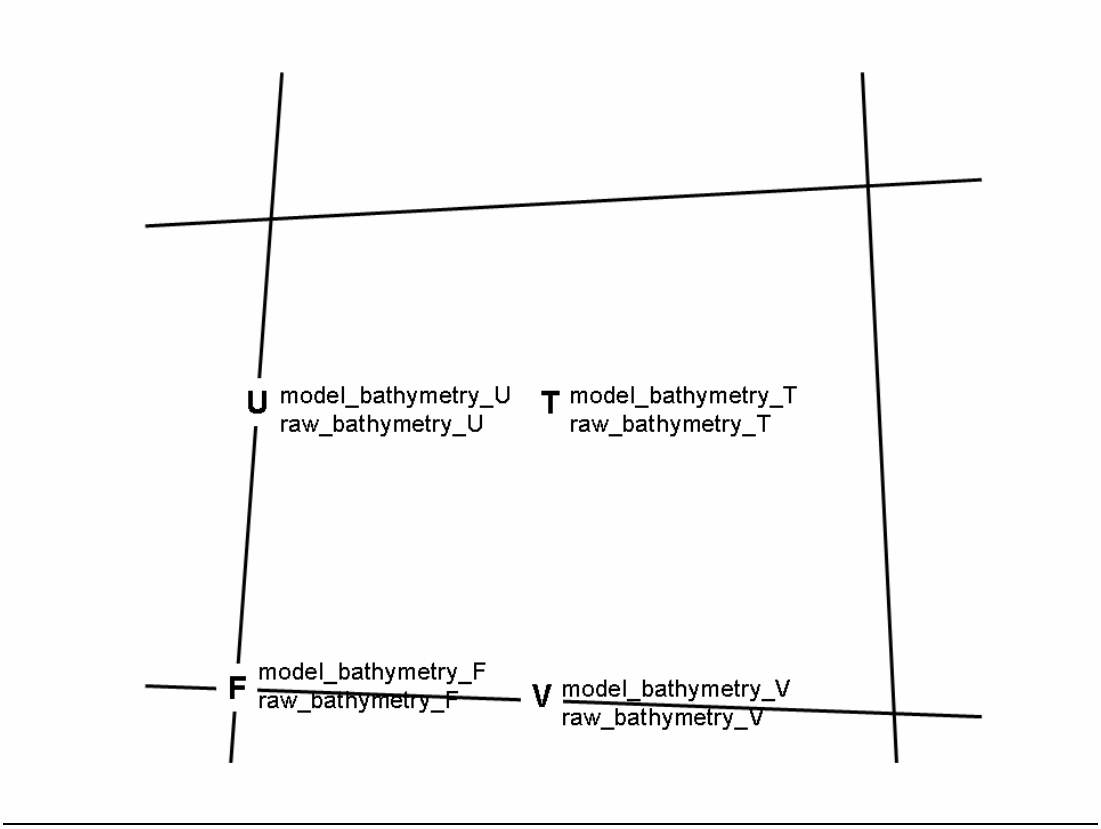


En chaque point T, U et V de la grille sont définies les tailles de maille ou "facteur d'échelle". Ces variables peuvent porter n'importe quel nom (*e.g.* sur la figure ci-dessus d[x,y]_[T,U,V,F]). Seul l'attribut `standard_name` décrit précisemment la variable avec une valeur pour ce `standard_name` qui doit correspondre à une des valeurs de la liste ci-dessous :

`mesh_size_along_x_axis_at_t_point`
`mesh_size_along_x_axis_at_u_point`
`mesh_size_along_x_axis_at_v_point`
`mesh_size_along_x_axis_at_f_point`
`mesh_size_along_y_axis_at_t_point`
`mesh_size_along_y_axis_at_u_point`
`mesh_size_along_y_axis_at_v_point`
`mesh_size_along_y_axis_at_f_point`

Ces valeurs de cet attribut `standard_name` ne sont pas des valeurs standard de la convention CF. Nous proposons d'utiliser les valeurs ci-dessus qui constituent le **deuxième amendement à la convention CF.**

Variable de bathymétrie :



A l'exception du code MARS, la bathymétrie est uniquement renseignée au point central de la maille (point T). La définition de la bathymétrie au point U et V est nécessaire pour MARS. Pour anticiper un éventuel besoin nous définissons aussi la bathymétrie au point F. Seul la variable de bathymétrie est obligatoirement définie au point T. Les autres variables sont facultatives dans le fichier.

On distingue deux types de variables de bathymétrie : la bathymétrie issue de l'interpolation des données (c'est la bathymétrie dite « brute ») et une bathymétrie qui est issue d'un processus de prétraitement de cette bathymétrie brute (e.g. filtrage) qui sera la bathymétrie vue par le modèle (c'est la bathymétrie « modèle »)

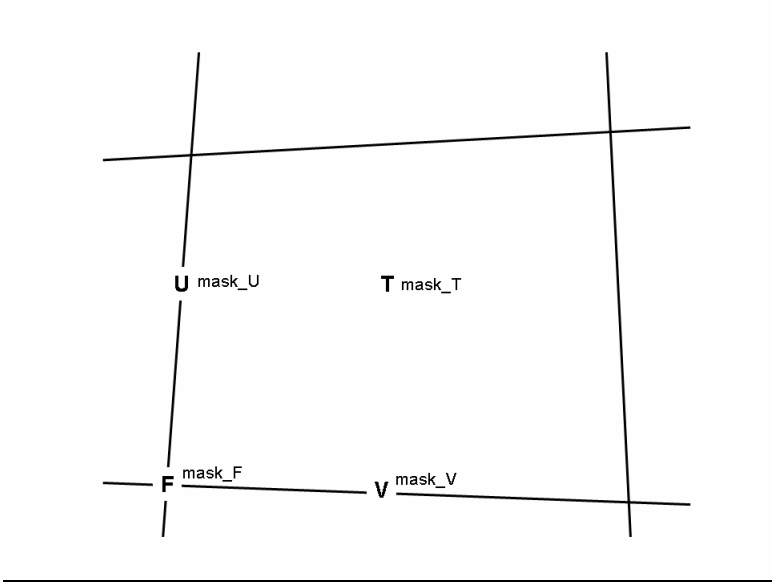
En pratique, suivant les modèles soit c'est la bathymétrie brute qui est entrée dans le modèle et cette bathymétrie brute est modifiée dans le code (on peut alors stocker la bathymétrie modifiée par la suite dans ce fichier), soit on s'impose rigoureusement qu'il n'y ait aucune modification de bathymétrie dans le code et ces deux bathymétries coexistent dans ce fichier.

Ces variables de bathymétrie peuvent porter n'importe quel nom. Seul l'attribut `standard_name` décrit précisemment la variable avec une valeur pour ce `standard_name` qui doit correspondre à une des valeurs de la liste ci-dessous :

`model_bathymetry_t`
`model_bathymetry_u`
`model_bathymetry_v`
`model_bathymetry_f`
`raw_bathymetry_t`
`raw_bathymetry_u`
`raw_bathymetry_v`
`raw_bathymetry_f`

Ces valeurs de cet attribut `standard_name` ne sont pas des valeurs standard de la convention CF. Nous proposons d'utiliser les valeurs ci-dessus qui constituent le **troisième amendement à la convention CF.**

Variable de masque



En chaque point T, U, V et F sont définis les masques. Ces variables de masque peuvent porter n'importe quel nom. Seul l'attribut `standard_name` décrit précisément la variable avec une valeur pour ce `standard_name` qui doit correspondre à une des valeurs de la liste ci-dessous :

`mask_t`
`mask_u`
`mask_v`
`mask_f`

Il existe un nom de `standard_name` standardisée dans la convention CF qui est `land_binary_mask`. Or ce `standard_name` fait exclusivement référence au masque terre-mer et ces variables de masque peuvent avoir d'autre utilisation en utilisant d'autre valeur comprise en 0 et 255. Nous proposons des noms de `standard_name` plus générique. C'est le **quatrième amendement de la convention CF.**

Il y a donc 4 séries de `standard_names` propres à COMODO qui sont utilisé en plus des `standard_names` défini dans la convention CF 1.4.

Autres attributs utilisés

Outre la présence de l'attribut `standard_name` qui est obligatoire et capital pour garantir

l'interopérabilité des fichiers et des applications, d'autres attributs sont utilisés.

Pour toutes les variables :

Présence obligatoire de :

- **`_FillValue`**

La valeur de `_FillValue` doit être une valeur n'étant pas contenue dans le domaine de validité des données définie par `[valid_min,valid_max]`
Nota bene : Cet attribut doit être présent pour les variables de coordonnées dans le cas de grille irrégulière incomplète en certains points. Cela déroge à la convention CF qui stipule que Les variables de coordonnées ne doivent pas contenir cet attribut _FillValue

- **`units`**

Présence fortement recommandée de :

- **`long_name`**

- **`valid_min`**
- **`valid_max`**

Les valeurs de `valid_min` et `valid_max` sont utiles pour certaines applications afin de gagner du temps calcul. Si ces valeurs ne sont pas présentes, alors les applications doivent faire le calcul du minimum et maximum.

Présence recommandée ou, si les variables sont compressées, présence obligatoire de :

- **`scale_factor`**
- **`add_offset`**

```
valeur_réelle = (scale_factor * valeur_compressée) + add_offset
```

Pour les variables de coordonnées :

- **`axis`**

Cet attribut est utile à certains logiciels de visualisation pour l'automatisation des procédures de représentation graphique. Il ne peut qu'être valorisé que par "X" ,"Y", "Z" ou "T".

Pour les variables de bathymétrie :

Présence obligatoire de :

- **`coordinates`**

- **`bathymetry_vertical_reference`**

`bathymetry_vertical_reference` est un **nouvel_attribut** qui décrit la référence verticale de la bathymétrie interpolée (par exemple : zéro hydrographique, niveau des plus

hautes mers, etc)

Présence recommandée de :

- **`mask`**

mask est un **nouvel attribut** spécifique au logiciel XSCAN ; il fait référence à la variable de masque.

- **`references`**

L'attribut **`references`** décrit les informations sur la méthode de traitement (filtrage) de la bathymétrie interpolée

- **`source`**

L'attribut **`source`** décrit les données utilisées pour l'interpolation (par exemple : GEBCO, ETOPO, etc)

Attributs globaux :

Présence obligatoire de :

- **`arakawa_grid_type`**

Ce **nouvel attribut** décrit le type de grille C utilisé.

Présence recommandée de :

- **`Conventions`**
- **`model_name`**

Ce **nouvel attribut renseigne** sur le modèle numérique utilisé.

Description détaillée de l'en-tête d'un fichier type "coordonnées horizontales – bathymétrie – masque" :

`dimensions:`

`ni =  ;` (pour le cas des grilles C1,C2 et C3)
`nj =  ;`

ou

`ni_t =  ;` *(pour le cas des grilles C0)*
`nj_t =  ;`
`ni_u =  ;`
`nj_u =  ;`
`ni_v =  ;`
`nj_v =  ;`
`ni_f =  ;`

```
  nj_f =  ;
```

(ci-après est traité le cas des grilles C1,C2 et C3)

variables:

définitions des variables de coordonnées :

```
double longitude_t(nj, ni) ;
  longitude_t:standard_name = "longitude_t" ;
  longitude_t:axis = « X »
  longitude_t:_FillValue = -9.e+33f
  longitude_t:long_name = "mon nom de variable préféré";
  longitude_t:units = "degrees_east"  ;
  longitude_t:valid_min = -10.f
  longitude_t:valid_max = 80.f ;
  longitude_t:scale_factor = 1.f ;
  longitude_t:add_offset = 0.f ;

double longitude_U(nj, ni) ;
  longitude_U:standard_name = "longitude_u" ;
  longitude_U:axis = "X";
  longitude_U:_FillValue = -9.e+33f ;
  longitude_U:long_name = "mon nom de variable préféré";
  longitude_U:units = "degrees_east" ;
  longitude_U:valid_min = -10.f ;
  longitude_U:valid_max = 80.f ;
  longitude_U:scale_factor = 1.f ;
  longitude_U:add_offset = 0.f ;

double longitude_V(nj, ni) ;
  longitude_V:standard_name = "longitude_v" ;
  longitude_V:axis = "X" ;
  longitude_V:_FillValue = -9.e+33f ;
  longitude_V:long_name = "mon nom de variable préféré";
  longitude_V:units = "degrees_east" ;
  longitude_V:valid_min = -10.f ;
  longitude_V:valid_max = 80.f ;
  longitude_V:scale_factor = 1.f ;
  longitude_V:add_offset = 0.f ;

double longitude_F(nj, ni) ;
  longitude_F:standard_name = "longitude_f";
  longitude_F:axis = "X" ;
  longitude_F:_FillValue = -9.e+33f ;
  longitude_F:long_name = "mon nom de variable préféré";
  longitude_F:units = "degrees_east" ;
  longitude_F:valid_min = -10.f ;
  longitude_F:valid_max = 80.f ;
  longitude_F:scale_factor = 1.f ;
```

```
  longitude_F:add_offset = 0.f ;

double latitude_T(nj, ni) ;
  latitude_T:standard_name = "latitude_t" ;
  latitude_T:axis = "Y" ;
  latitude_T:_FillValue = -9.e+33f ;
  latitude_T:long_name = "mon nom de variable préféré" ;
  latitude_T:units = "degrees_north ";
  latitude_T:valid_min = -10.f
  latitude_T:valid_max = 80.f ;
  latitude_T:scale_factor = 1.f ;
  latitude_T:add_offset = 0.f ;

double latitude_U(nj, ni) ;
  latitude_U:standard_name = "latitude_u" ;
  latitude_U:axis = "Y" ;
  latitude_U:_FillValue = -9.e+33f ;
  latitude_U:long_name = "mon nom de variable préféré" ;
  latitude_U:units = "degrees_north" ;
  latitude_U:valid_min = -10.f ;
  latitude_U:valid_max = 80.f ;
  latitude_U:scale_factor = 1.f ;
  latitude_U:add_offset = 0.f ;

double latitude_V(nj, ni) ;
  latitude_V:standard_name = "latitude_v";
  latitude_V:axis = "Y" ;
  latitude_V:_FillValue = -9.e+33f ;
  latitude_V:long_name = "mon nom de variable préféré »;
  latitude_V:units = "degrees_north" ;
  latitude_V:valid_min = -10.f ;
  latitude_V:valid_max = 80.f ;
  latitude_V:scale_factor = 1.f ;
  latitude_V:add_offset = 0.f ;

double latitude_F(nj, ni) ;
  latitude_F:standard_name = "latitude_f"
  latitude_F:axis = "Y"
  latitude_F:_FillValue = -9.e+33f ;
  latitude_F:long_name = "mon nom de variable préféré";
  latitude_F:units = "degrees_north";
  latitude_F:valid_min = -10.f ;
  latitude_F:valid_max = 80.f ;
  latitude_F:scale_factor = 1.f ;
  latitude_F:add_offset = 0.f ;
```

Définition des bathymétries :

```
double bathymetry_T(nj, ni) ;
  bathymetry_T:standard_name = "model_bathymetry_t" ;
  bathymetry_T:coordinates = "longitude_T latitude_T" ;
  bathymetry_T:_FillValue = -9.e+33f ;
  bathymetry_T:mask =" mask_T"
  bathymetry_T:long_name = "mon nom de variable préféré";
  bathymetry_T:units = "m" ;
  bathymetry_T:valid_min = -10.f ;
  bathymetry_T:valid_max = 80.f ;
  bathymetry_T:references = "type de filtrage, paramètres, etc"

double bathymetry_U(nj, ni) ;
  bathymetry_U:standard_name = "model_bathymetry_u" ;
  bathymetry_U:coordinates = "longitude_U latitude_U" ;
  bathymetry_U:_FillValue = -9.e+33f ;
  bathymetry_U:mask = "mask_U "
  bathymetry_u:long_name = "mon nom de variable préféré";
  bathymetry_U:units = "m" ;
  bathymetry_U:valid_min = -10.f ;
  bathymetry_U:valid_max = 80.f ;
  bathymetry_U:references = "type de filtrage, paramètres, etc"

double bathymetry_V(nj, ni) ;
  bathymetry_V:standard_name = "model_bathymetry_v ";
  bathymetry_V:coordinates = "longitude_V latitude_V" ;
  bathymetry_V:_FillValue = -9.e+33f ;
  bathymetry_V:mask = "mask_V"
  bathymetry_V:long_name = "mon nom de variable préféré";
  bathymetry_V:units = "m" ;
  bathymetry_V:valid_min = -10.f ;
  bathymetry_V:valid_max = 80.f ;
  bathymetry_V:references = "type de filtrage, paramètres, etc"

double bathymetry_F(nj, ni) ;
  bathymetry_F:standard_name = "model_bathymetry_f" ;
  bathymetry_F:coordinates = "longitude_F latitude_F" ;
  bathymetry_F:_FillValue = -9.e+33f ;
  bathymetry_F:mask = "mask_V "
  bathymetry_F:long_name = "mon nom de variable préféré";
  bathymetry_F:units = "m" ;
  bathymetry_F:valid_min = -10.f ;
  bathymetry_F:valid_max = 80.f ;
  bathymetry_F:references = "type de filtrage, paramètres, etc"

double raw_bathymetry_T(nj, ni) ;
  raw_bathymetry_T:standard_name = "raw_bathymetry_t" ;
  raw_bathymetry_T:coordinates = "longitude_T latitude_T" ;
  raw_bathymetry_T:_FillValue = -9.e+33f ;
  raw_bathymetry_T:mask = "mask_T" ;
```

```
  raw_bathymetry_T:long_name = "mon nom de variable préféré" ;
  raw_bathymetry_T:units = "m" ;
  raw_bathymetry_T:valid_min = -10.f ;
  raw_bathymetry_T:valid_max = 80.f ;
  raw_bathymetry_T:bathymetry_vertical_reference = "e.g. zero
    hydrographique, niveaux des plus hautes mers, etc"
  raw_bathymetry_T:source = "e.g. GEBCO,ETOPO , etc"

double raw_bathymetry_U(nj, ni) ;
  raw_bathymetry_U:standard_name = "raw_bathymetry_u" ;
  raw_bathymetry_U:coordinates = "longitude_U latitude_U" ;
  raw_bathymetry_U:_FillValue = -9.e+33f ;
  raw_bathymetry_U:mask = "mask_U" ;
  raw_bathymetry_u:long_name = "mon nom de variable préféré";
  raw_bathymetry_U:units = "m" ;
  raw_bathymetry_U:valid_min = -10.f ;
  raw_bathymetry_U:valid_max = 80.f ;
  raw_bathymetry_T:bathymetry_vertical_reference = "e.g. zéro
    hydrographique,  niveaux des plus hautes mers, etc"
  raw_bathymetry_T:source = "e.g. GEBCO, ETOPO, etc"

double raw_bathymetry_V(nj, ni) ;
  raw_bathymetry_V:standard_name = "raw_bathymetry_v" ;
  raw_bathymetry_V:coordinates = "longitude_V latitude_V" ;
  raw_bathymetry_V:_FillValue = -9.e+33f ;
  raw_bathymetry_V:mask = "mask_V" ;
  raw_bathymetry_V:long_name = "mon nom de variable préféré";
  raw_bathymetry_V:units = "m" ;
  raw_bathymetry_V:valid_min = -10.f ;
  raw_bathymetry_V:valid_max = 80.f ;
  raw_bathymetry_V:bathymetry_vertical_reference = "e.g. zéro
    hydrographique,  niveaux des plus hautes mers, etc"
  raw_bathymetry_V:source = "e.g. GEBCO, ETOPO, etc"

double raw_bathymetry_F(nj, ni) ;
  raw_bathymetry_F:standard_name = "raw_bathymetry_f" ;
  raw_bathymetry_F:coordinates = "longitude_F latitude_F" ;
  raw_raw_bathymetry_F:_FillValue = -9.e+33f ;
  raw_bathymetry_F:mask = "mask_F" ;
  raw_bathymetry_F:long_name = "mon nom de variable préféré";
  raw_bathymetry_F:units = "m" ;
  raw_bathymetry_F:valid_min = -10.f ;
  raw_bathymetry_F:valid_max = 80.f ;
  raw_bathymetry_F:bathymetry_vertical_reference = "e.g. zéro
    hydrographique,  niveaux des plus hautes mers, etc"
  raw_bathymetry_F:source = "e.g. GEBCO, ETOPO, etc"
```

définition des" facteurs d'échelle" ou taille de maille :

```
double dx_T(nj, ni) ;
```

```
  dx_T:standard_name = "mesh_size_along_x_at_t_point" ;
  dx_T:coordinates = "longitude_T latitude_T" ;
  dx_T:_FillValue = -9.e+33f ;
  dx_T:long_name = "mon nom de variable préféré »;
  dx_T:units = "m"  ou  "1/m" ;
  dx_T:valid_min = 54.5f ;
  dx_T:valid_max = 86.4.f ;

double dx_U(nj, ni) ;
  dx_U:standard_name = "mesh_size_along_x_at_u_point" ;
  dx_U:coordinates = "longitude_U latitude_U" ;
  dx_U:_FillValue = -9.e+33f ;
  dx_U:long_name = "mon nom de variable préféré" ;
  dx_U:units = "m" ou "1/m";
  dx_U:valid_min = 54.5f ;
  dx_U:valid_max = 86.4.f ;

double dx_V(nj, ni) ;
  dx_V:standard_name = "mesh_size_along_x_at_v_point ;
  dx_V:coordinates = "longitude_V latitude_V" ;
  dx_V:_FillValue = -9.e+33f ;
  dx_V:long_name = "mon nom de variable préféré";
  dx_V:units = "m" ou "1/m" ;
  dx_V:valid_min = 54.5f ;
  dx_V:valid_max = 86.4.f ;

double dx_F(nj, ni) ;
  dx_F:standard_name = "mesh_size_along_x_at_f_point" ;
  dx_F:coordinates = "longitude_F latitude_F";
  dx_F:_FillValue = -9.e+33f ;
  dx_F:long_name = "mon nom de variable préféré";
  dx_F:units = "m" ou "1/m" ;
  dx_F:valid_min = 54.5f ;
  dx_F:valid_max = 86.4.f ;

double dy_T(nj, ni) ;
  dy_T:standard_name = "mesh_size_along_y_at_t_point" ;
  dy_T:coordinates = "longitude_T latitude_T" ;
  dy_T:_FillValue = -9.e+33f ;
  dy_T:long_name = "mon nom de variable préféré";
  dy_T:units = "m" ou "1/m" ;
  dy_T:valid_min = 54.5f ;
  dy_T:valid_max = 86.4.f ;

double dy_U(nj, ni) ;
  dy_U:standard_name = "mesh_size_along_y_at_u_point" ;
  dy_U:coordinates = "longitude_U latitude_U";
  dy_U:_FillValue = -9.e+33f ;
  dy_U:long_name = "mon nom de variable préféré";
```

```
  dy_U:units = "m" ou "1/m" ;
  dy_U:valid_min = 54.5f ;
  dy_U:valid_max = 86.4.f ;

double dy_V(nj, ni) ;
  dy_V:standard_name = "mesh_size_along_y_at_v_point" ;
  dy_V:coordinates = "longitude_V latitude_V" ;
  dy_V:_FillValue = -9.e+33f ;
  dy_V:long_name = "mon nom de variable préféré";
  dy_V:units = "m" ou "1/m" ;
  dy_V:valid_min = 54.5f ;
  dy_V:valid_max = 86.4.f ;

double dy_F(nj, ni) ;
  dy_F:standard_name = "mesh_size_along_y_at_f_point" ;
  dy_F:coordinates = "longitude_F latitude_F" ;
  dy_F:_FillValue = -9.e+33f ;
  dy_F:long_name = "mon nom de variable préféré" ;
  dy_F:units = "m" ou "1/m" ;
  dy_F:valid_min = 54.5f ;
  dy_F:valid_max = 86.4.f ;
```

definition des masques :

```
byte mask_T(nj, ni) ;
  mask_T:standard_name = "mask_t" ;
  mask_T:coordinates = "longitude_T latitude_T" ;
  mask_T:long_name = "mon nom de variable préféré" ;

byte mask_U(nj, ni) ;
  mask_U:standard_name = "mask_u" ;
  mask_U:coordinates = "longitude_U latitude_U" ;
  mask_U:long_name = "mon nom de variable préféré" ;

byte mask_V(nj, ni) ;
  mask_V:standard_name = "mask_v" ;
  mask_V:coordinates = "longitude_V latitude_V" ;
  mask_V:long_name = "mon nom de variable préféré";

byte mask_F(nj, ni) ;
  mask_F:standard_name = "mask_f" ;
  mask_F:coordinates = "longitude_F latitude_F" ;
  mask_F:long_name = "mon nom de variable préféré";

// global attributes:
  :arakawa_grid_type = "C0" ou "C1", etc ;
  :conventions = "CF-1.4_COMODO-0.1" ;
  :model_name = "MARS V8.15" ;
```

Les outils de pre- et post-traitement :

Nous avons listé plusieurs outils de pre- et post- traitement qui seraient susceptibles d'être mis en commun et de devenir interopérables s'il subissent des modification afin de respecter les convention détaillés ci-dessus.

ROMS TOOLS (pré- et post- traitement)

Développé par équipe ROMS
http://roms.mpl.ird.fr/
Langage utilisé : Matlab

XSCAN (pré- et post-traitement)

Développé par Florent Lyard
http://poc.obs-mip.fr/pages/softs/xscan/xscan.htm

BMGTOOLS (pré-traitement)

Développé par Ifremer
http://www.ifremer.fr/bmgtools
C'est un ensemble d'outils d'aide à la préparation de maillage bathymétrique pour les modèles numériques de circulation océanique côtière et hauturière. Ces outils sont utilisables pour le modèle MARS intégralement et certaines fonctionnalités sont potentiellement compatibles pour d'autres modèles comme Symphonie, NEMO, ROMS, etc ...
Langages utilisés : java pour interface graphique interfacé avec Fortran 90 pour les calculs.
Fonctionne sur Linux, Windows et MacOSX

SIREN (pré-traitement)

VIFOP (pré-traitement)
Développé par le Pôle d'Océanographie Côtière, Toulouse

Le logiciel VIFOP est un outil générique original d'imbrication de modèles océaniques. VIFOP permet l'imbrication de modèles océaniques régionaux et côtiers dans des modèles de circulation générale.

Langage utilisé : Fortran.

http://poc.obs-mip.fr/auclair/Recherche/VIFOP/vifop_index.htm

SVIEW (post-traitement)
Développé par le Pôle d'Océanographie Côtière, Toulouse
SVIEW est un outil de post-traitement et de visualisation sous MATLAB de fichiers Netcdf. SVIEW est auto-documenté et son codage basé sur la bibliothèque MATLAB-GUI permet

l'ajout de fonctionnalités par l'utilisateur. Le module SVIEW-E permet en outre le post-traitement et la visualisation de fichiers Netcdf contenant les suivis de transferts énergétiques.
http://poc.obs-mip.fr/auclair/Recherche/SVIEW/sview.htm
Langage utilisé : Matlab

SAXO (post-traitement)
Développé par IPSL/LOCEAN
http://www.nemo-ocean.eu/About-NEMO/Projects/SAXO-visualize-and-analyze-models-outputs
Langage utilisé : IDL

Efforts de développements

Une première étape fondamentale est la reconnaissance d'une convention commune par toutes les communautés qui permettra les échanges de fichiers et d'applications. Une fois les spécifications bien définies et entérinées, il reste à adapter les différentes applications.
Certaines communautés seraient d'ores et déjà prêtes à directement modifier leurs entrées/sorties dans leur modèle numérique. Certains outils ont aussi à commencer à faire l'effort de pouvoir prendre en compte les conventions d'autres modèles. Avec l'établissement d'une convention commune, le travail d'adaptation de ces outils serait grandement facilité.
Il nous semble important qu'un **outil de vérification de format** soit mis à disposition de toute la communauté. A la manière des "*convention compliance checker*"[3] qui existe pour vérifier que les fichiers respectent bien la convention "Climate Forecast", il serait très utile de créer un outil équivalent qui vérifierait que les fichiers respectent notre nouvelle convention COMODO.
L'amélioration de la modélisation numérique passe d'abord par un investissement maximum dans les codes numériques et les efforts engagés sur le développement et la maintenance des outils périphériques ne sont pas toujours prioritaires au sein de chaque communauté. Pour autant, si l'approche communautaire du projet COMODO doit se concrétiser, alors il faudra **dégager des ressources identifiées qui puissent soutenir cet effort.**

[3] http://titania.badc.rl.ac.uk/cgi-bin/cf-checker.pl