POUR UNE MEILLEURE GESTION DES NAMELISTS ET DES VALIDATIONS.

YESSAD Karim, affecté à CNRM/GMAP/ALGO.

June 8, 2010

1 Introduction:

Cette note est une réactualisation d'une note qui avait été écrite en juin 2004 (suite à des difficultés de validation du cycle cy28t0_t1 durant le printemps 2004). Plus généralement nous connaissons assez souvent des difficultés de validation qui se manifestent par l'un au moins des deux syndromes suivants:

- on passe au moins 2 mois (voire davantage) à valider un cycle, surtout pour les configurations d'assimilation, et on peut être confronté à une ou plusieurs bugs difficiles à trouver (exemples: le cycle cy28t0_t1 au printemps 2004, le cycle 34 à l'été 2008).
- on dispose d'un cycle qui semble bien validé au premier abord mais on découvre au fil des mois plusieurs bugs dormantes, cette découverte se faisant lors d'un changement d'environnement de validation (validation avec une version plus récente de MITRAILLETTE ou avec une version plus récente de l'opérationnel). Les cycles 35t2 et 36t1 en fournissent des exemples pertinents mais il y a eu d'autres cas dans le passé.

Ces difficultés de validation récurrentes motivent la rédaction d'une version réactualisée de cette note.

2 Raisons possibles de difficultés de validation:

La liste présentée ci-dessous n'est pas exhaustive (la plupart des points listés étaient déjà mentionnées dans la version de juin 2004) mais elle comprend des sources de difficultés constatées à plusieurs reprises.

- validation insuffisante du cycle de base de développement sur lequel a été bâti le cycle à valider (bugs dormantes qui ne se sont révélées que tardivement).
- impossibilité de reproduire au bit près le cycle de référence en raison de modifications introduites en dur (cela s'est produit notamment pour la physique, mais également pour le traitement d'observations satellitaires).
- difficultés à réactualiser proprement les namelists qui servent à la validation, surtout en ce qui concerne les namelists qui sont utilisées sous OLIVE.
- entrée dans le cycle de développement de modifications insuffisamment validées (par exemple oubli de tests eulériens ou SL3TL pour valider des modifications d'interface dynamique-physique).
- entrée à la volée de modifications de dernière minute bien après la date limite, pouvant entraîner l'apparition de bugs par effets collatéraux.
- certaines routines n'ont pas été examinées par un comparateur de source par rapport à un cycle de référence avant la validation informatique, alors que c'est la première chose à faire (syndrome qui a été pénalisant lors de la validation difficile du cycle 34 au cours de l'été 2008, mais on pourrait citer d'autres exemples).
- validation pas assez analytique, notamment pour l'assimilation (on passe directement au gros mastodonte qu'est le 4DVAR sans intermédiaire car on manque encore de configurations simplifiées de briques de 4DVAR sous OLIVE voire sous MITRAILLETTE).
- nombre insuffisant de "généralistes" à GMAP, c.a.d. de personnes ayant une bonne vision d'ensemble du code, et un certain manque de motivation pour la validation informatique.
- validation annoncée a priori comme facile et finalement recelant plus de pièges que prévu, ce qui peut conduire à un manque de mobilisation.
- imperfection du logiciel de nettoyage automatique du code, pour les cas où on fait du nettoyage automatique. Ce syndrome a par exemple été constaté lors de la mise des attributs d'INTENT aux variables passées en argument.
- dans certaines modifications consistant à réécrire du code (souvent à l'initiative du CEP), il arrive que le gros-oeuvre soit bien fait mais qu'il manque les finitions (lesquelles peuvent représenter un travail important). Ce syndrome a été noté lors de l'introduction du flux de données GMV-GFL mais également lors de l'introduction du nouveau flux de surface à l'automne 2006.
- dogme consistant à passer systématiquement dans tout le set-up de la physique du CEP (incluant les routines du projet SUR = schéma de surface du CEP), y compris dans des configurations adiabatiques ou des configurations n'utilisant pas la physique du CEP (ou une toute petite partie). Ce point peut nous valoir des plantages dans des parties de code où il est inutile de passer.
- valideurs faisant trop de choses différentes à la fois, ce qui ne leur permet pas de bien se concentrer sur un problème (cela a été un point handicapant lors de la validation difficile de l'été 2008).

- validation de configurations variationnelles débutant souvent avec 2 ou 3 semaines de retard par rapport à ce qui pourrait être fait.
- à tout cela s'ajoutent la complexification excessive du code et également de l'environnement opérationnel.

Ceci nous appelle à rappeler certaines règles pour la validation et pour la gestion des namelists.

3 Propositions pour améliorer les choses:

∗ **Avant d'entrer toute modification:** Bien la valider, et pas seulement en terme de code direct ou en se limitant au schéma semi-Lagrangien à deux pas de temps. D'une façon générale, un développeur doit cerner les effets collatéraux de ses modifications pour déterminer la liste adéquate de configurations à valider. Pour citer quelques exemples:

- une modification d'interface physique-dynamique doit être systématiquement validée en eulérien, SL3TL et SL2TL.
- ne pas oublier de tester les configurations utilisant du code tangent linéaire ou adjoint (conf 501, 401, voire 601, 801) pour des développements touchant aux codes tangent linéaire ou adjoint.

On peut rappeler que les outils MITRAILLETTE et MITRARP sont publics et que leur utilisation n'est pas réservée à un sous-ensemble restreint de valideurs réguliers.

∗ **Reproductibilité:** On devrait toujours être capable de reproduire le cycle de référence (cela s'applique également à la physique et au 4DVAR). Ceci implique que toute option nouvelle susceptible de modifier les résultats doit être codée sous clé.

∗ **Examen par un comparateur de sources de toute modification:** A l'arrivée d'un nouveau cycle, celui-ci doit être examiné routine par routine par un comparateur de sources par rapport au cycle précédent, pour les raisons suivantes:

- bonne vision des modifications introduites (cela sert notamment pour la remise à jour de la documentation).
- repérage des bugs évidentes, des problèmes de phasage, des lignes dupliquées ou mal placées, des incohérences.
- quand il s'agit d'un modset venant du CEP, cela permet d'évaluer les difficultés de phasage à prévoir avec nos contributions.

J'ai jusqu'à présent pris en charge ce travail pour une bonne partie du code, mais ce travail doit être mieux partagé.

∗ **Namelists:** Un effort a été fait sur ce point depuis 2004, mais il reste encore du chemin à faire:

- Une norme de présentation des namelists a été définie et a été décrite dans différentes documentations publiques. Toutes les namelists utilisées par MITRAILLETTE, MITRARP, OLIVE, la chaîne opérationnelle et la chaîne en double doivent respecter ces normes.
- La gestion des namelists de l'opérationnel et d'OLIVE n'est pas encore assez centralisée, et devrait inclure davantage d'outils automatiques pour la maintenance.
- La rentrée sous OLIVE de namelists servant à la validation de configurations d'assimilation prend encore beaucoup trop de temps et un effort important doit être fait sur ce point afin que la validation de ces configurations commence à temps.

∗ **Entrée à la volée de modifications de dernière minute bien après la date limite:** Ceci devra être prohibé à l'avenir, en dehors des bugfixes bien sûr.

∗ **Validation complète:** Un effort doit être fait pour que le cycle de référence sur lequel sont faites les modifications soit convenablement validé, et qu'on ne se retrouve pas à y retrouver moult bugs dormantes après coup. En particulier il doit fonctionner pour la chaîne opérationnelle et la chaîne en double en cours.

∗ **Validation en mode "indef" et débordement de tableau:** De telles validations doivent être faites plus souvent, et un effort doit être fait pour permettre d'utiliser facilement l'option débordement de tableau. Concernant les points qui posent problème pour l'option débordement de tableau, il reste au moins deux points à traiter en priorité sur la base du cycle CY36T2:

- allouer tous les groupes de tableaux de surface à au moins un champ (le fait que certains groupes ne soient pas alloués est l'un des points bloquants qui génèrent beaucoup de fausses alarmes, car ces tableaux sont passés en argument d'appel à de nombreux endroits du code, avec des pointeurs).
- réduire le nombre d'arguments d'appels des routines appelées par CPG (CPG_GP, CPG_DYN, CPG_DIA, et surtout MF_PHYS) par l'introduction de nouvelles structures de types dérivés, car ce nombre excessif d'arguments d'appels empêche actuellement l'utilisation de CPG en mode débordement de tableau.

∗ **Validation avec un pas de temps nul, ou très petit:** L'expérience a montré à plusieurs reprises que ce genre de validation permet de trouver des bugs dans des configurations qui semblent marcher à peu près (pas mal de bugs trouvées dans l'option "NH avec LGWADV" en 2008 grâce à ce type de validation), mais ce type de validation se fait à une fréquence insuffisante.

Avec un pas de temps nul le modèle doit être stationnaire, et avec un pas de temps de 10^{-8} secondes il doit être quasi-stationnaire. Le cycle 36T2 comporte un certain nombre de divisions par le pas de temps, ce qui en pratique impose de valider avec un pas de temps très petit mais pas nul. Il serait souhaitable à l'avenir de prohiber complètement les divisions par le pas de temps, et de retrouver la fonctionnalité permettant de tourner le modèle avec un pas de temps nul.

∗ **Vers une validation plus analytique:** Tout comme pour les fractions rationnelles, il convient de décomposer une validation en éléments simples pour bien comprendre d'où peuvent venir les problèmes éventuels. Cette démarche est loin d'être utilisée pour la validation du 4DVAR, qu'on valide globalement sans avoir validé séparément certains constituants. Or débeuguer un tel mastodonte n'est pas chose évidente. Il faut donc avoir une validation plus analytique de ce type d'option, avec des étapes intermédiaires, par exemple:

- la validation de l'interpolateur d'observation dans une configuration simple qui utiliserait peu d'observations, ou un jeu d'observations fictives qui ne passerait pas par le logiciel ODB mais par le set-up d'ARPEGE (dans l'interpolateur d'observation, ce dont on a besoin ce sont des coordonnées des points d'observations, pas du contenu des observations elles-mêmes).
- la validation de versions simplifiées de briques de 3D-VAR ALADIN (écrémage, CANARI, minimisation) qui utiliseraient un nombre très réduit de fichiers d'observations: seules de telles versions pourraient être introduites dans MITRAILLETTE, l'entrée de configurations telles qu'elles se présentent sous OLIVE et dans l'opérationnel comportant un nombre et une diversité de fichiers d'entrée incompatibles avec l'utilisation dans MITRAILLETTE.

Le fait qu'on ne puisse pas utiliser le même fichier ODB pour deux cycles pleins différents (exemple CY35T2 et CY36T1) est un autre point de difficulté sur lequel il faudra apporter une réponse acceptable.

∗ **Au moins un responsable permanent à GMAP pour chaque partie de code:** Il reste encore quelques parties de code qui ne sont bien maîtrisées qu'au CEP ou que par certains visiteurs ALADIN ou HIRLAM de passage. A terme, toute partie de code d'ARPEGE/ALADIN devra avoir au moins un responsable permanent à GMAP (et si possible au moins deux). En particulier la partie adiabatique du modèle (direct + TL et AD) doit être bien maîtrisée par au moins deux personnes de GMAP dont au moins un IT (qui n'est pas nécéssairement dans l'équipe ALGO).

∗ **Personnes impliquées dans la validation:** Elles doivent être un peu plus nombreuses et les jeunes récemment arrivés doivent y participer davantage. Il faudrait davantage de personnes "généralistes", notamment dans les équipes ALGO et PROC.

∗ **Réécritures de code faites par le CEP et gestion des finitions:** Dans les réécritures de code faites par le CEP, il reste souvent des finitions pas faites qui se révèlent être parfois un gros travail. La question va se poser de façon sensible pour les réécritures liées au projet OOPS. Les moyens doivent être convenablement dimensionnés à GMAP pour effectuer dans de bonnes conditions le travail complémentaire à celui du CEP.

4 Rappel de la procédure de validation.

On résume les étapes comme suit:

- V1/ Si nécessaire, mise à jour des configurations de MITRAILLETTE et MITRARP de façon à être le plus proche de la physique opérationnelle et à valider les configurations les plus utiles. Dans ce cas il faut refaire un jeu de simulations sur le cycle de base de développement.
  On refait environ une grosse remise à jour de MITRAILLETTE et MITRARP par an (pas davantage car cela demande pas mal de travail). Les précédentes remises à jour ont eu lieu en avril 2009 et avril 2010. La prochaine interviendra vraisemblablement au début de l'année 2011.
- V2/ Fabrication des namelists MITRAILLETTE et MITRARP pour le cycle à valider.
- V3/ Validation des configurations de MITRAILLETTE et MITRARP. Cette validation se fait toujours par rapport au cycle de développement le plus récent, à options égales permettant une bonne reproductibilité des résultats. Par exemple on a validé cy36t2 par rapport à cy36t1.
- V4/ Validation des configurations ARPEGE-CLIMAT (action GMGEC).
- V5/ Fabrication des namelists OLIVE qui vont servir à valider les configurations qui sont sous OLIVE (4D-VAR ARPEGE, 3D-VAR ALADIN, 3D-VAR AROME, PEARP).
- V6/ Vérifier que tout ce qui est lié à ODB marche bien (entre autres valider un BATOR).
- V7/ Valider les configurations qui sont sous OLIVE (4D-VAR ARPEGE, 3D-VAR ALADIN, 3D-VAR AROME, PEARP), en prenant comme base l'opérationnel, ou une chaîne en double proche du basculement. Dans certains cas la validation par rapport à l'opérationnel et la validation par rapport au double peuvent être nécessaires.
- V8/ Valider la PEARP.

L'étape V7 doit attendre la fin de l'étape V3. Par contre on ne doit pas attendre l'achèvement de l'étape V3 pour commencer l'étape V5.