NTIC

Projet : Développements complémentaires AMDA-NG

NOTES TECHNIQUES POUR LA PRESTATION 'DEVELOPPEMENTS COMPLEMENTAIRES AMDA-NG'

Rédigé par : Benjamin Renard	Diffusé à : CDPP / IRAP
Approuvé par : Chef de projet AKKA – N. Lormant Responsable projet CNES – N. Dufourg	

LISTE DES MODIFICATIONS DU DOCUMENT

Vers.	Date	Paragraphe	Description de la modification
01.0	21/10/16		Création du document

SOMMAIRE

1 INTRODUCTION

La prestation « Développements complémentaires AMDA-NG » s'est déroulée du lundi 22 février 2016 au lundi 17 octobre 2016 (64 jours de charge, pour un ½ ETP AKKA) dans les locaux de l'IRAP.

Cette prestation concernait un ensemble de tâches diverses, toutes liées au projet AMDA du CDPP / IRAP, référencées dans la proposition commerciale [R0]. Il est à noter que des tâches supplémentaires ont été traitées au cours de la prestation, alors que d'autres ont été modifiées, en accord avec l'équipe technique du CDPP / IRAP.

Ce document constitue un ensemble de notes techniques couvrant l'ensemble des activités réalisées sur la durée de la prestation.

2 COMPLEMENTS RELATIFS A LA PROCEDURE D'INSTALLATION

2.1 INSTALLATION DEPUIS UNE VERSION MINIMALE

L'installation s'est effectuée depuis la version minimale de CentOS 6.3 (Centos-6.3-x86_64-minimal.iso), ce qui a impliqué l'installation d'un certain nombre de prérequis supplémentaires.

Les commandes suivantes doivent être exécutées avant l'étape « 5.3 Installation des prérequis du noyau AMDA-NG » de [R1]):

```
sudo yum install redhat-lsb
sudo yum install wget
sudo yum install glibc-i686
sudo yum install glibc-devel-i686
sudo yum install tcsh.x86_64
```

2.2 VERSION DE PHP

La version de PHP installée avec CentOS 6.3 est la version 5.3.3.

Le module « AMDA_Integration » exige l'utilisation de la version 5.4.45 de PHP. Cette étape doit être effectuée avant l'étape « 4.3 Configuration pour l'installation » de [R2].

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm && sudo rpm -Uvh epel-release-latest-6.noarch.rpm
wget http://rpms.famillecollet.com/enterprise/remi-release-6.rpm && sudo rpm -Uvh remi-release-6*.rpm
sudo yum --enablerepo=remi update php*
```

2.3 AJOUT DU PREREQUIS « CSPICE »

L'installation du prérequis « cspice » doit s'effectuée avant la compilation du module « AMDA_Kernel », c'est-à-dire avant l'étape « 3.3 Installation du noyau AMDA-NG » de [R2].

Les commandes suivantes sont à exécuter :

```
cd AMDA_COTS/cspice
./buildAndInstall.sh
```

2.4 INITIALISATION DE L'ESPACE « SHARED OBJECTS » DANS LE MODULE « AMDA_IHM »

A l'issue de l'installation du module « AMDA_IHM », il est nécessaire d'initialiser l'espace dédié au stockage des « shared objects » de la manière suivante :

```
cd AMDA_IHM/php
php updateSharedObjectsTreeFile.php
```

3 CORRECTIONS COMPLEMENTAIRES

Les corrections suivantes ont été effectuées « au fil de l'eau » lors de la prestation.

3.1 DECLINAISON DE LA FONCTION « MAGNITUDE » POUR LES PARAMETRES DE TYPE « DOUBLE », « LONG DOUBLE », « INT » ET « SHORT »

La fonction « magnitude » définie au niveau de « AMDA_Kernel » n'acceptait que des paramètres de type « float » en entrée. Lorsqu'elle était appliquée sur un paramètre d'un autre type, la compilation du paramètre dérivé ne pouvait s'effectuer et l'exécution de la requête sortait en exception.

Cette fonction a été déclinée de manière à accepter tous les types de paramètre (ed4f5039).

3.2 CORRECTION DU « GAP THRESHOLD » POUR UN PARAMETRE RESAMPLE

3.2.1 Calcul du « gap threshold » pour un paramètre resamplé

La définition d'un « gap » (ie. Intervalle de temps à partir duquel une absence de données est à considérer comme un trou de données pour un paramètre) se fait via la définition du « gap threshold » avec la relation suivante :

« gap » = « gap threshold » * minSampling

Lorsqu'un paramètre est ré échantillonné, son « gap » n'est pas modifié. Nous avons donc la relation suivante :

« gap » = « gap threshold resampled » * minSamplingResampled

D'où l'égalité suivante :

« gap threshold resampled » = « gap threshold » * minSampling / minSamplingResampled

La définition du "gap threshold" pour un paramètre ré-échantillonné a été corrigée dans AMDA_Kernel (31af243f).

3.2.2 Prise en compte du cas où le temps de ré échantillonnage d'un paramètre est supérieur au « gap »

Lors du ré échantillonnage d'un paramètre, il arrive que le temps d'échantillonnage soit supérieur à la définition du « gap » du paramètre.

Lorsque cela est le cas, la valeur du « gap threshold resampled » doit être corrigée à « 1 » (c391ab45).

3.3 BUG DE TRANSTYPAGE DANS LA FONCTION « AVERAGE »

Considérons la situation suivante :

```
int b;
unsigned int c;

b = -25;
c = 5;

int res = b / c;
```

Le résultat obtenu pour “res” est « 858993454 ».

En effet, en langage « C », un transtypage est effectué avant la division sur le paramètre « b » en « unsigned int ». Ainsi « -25 » devient « 4294967271 », puis la division s’effectue pour obtenir « 858993454 » (après transtypage en « int »).

Pour éviter ce genre de comportement, il suffit de forcer le transtypage de « c » en « int » :

```
int res = b / (int)c;
```

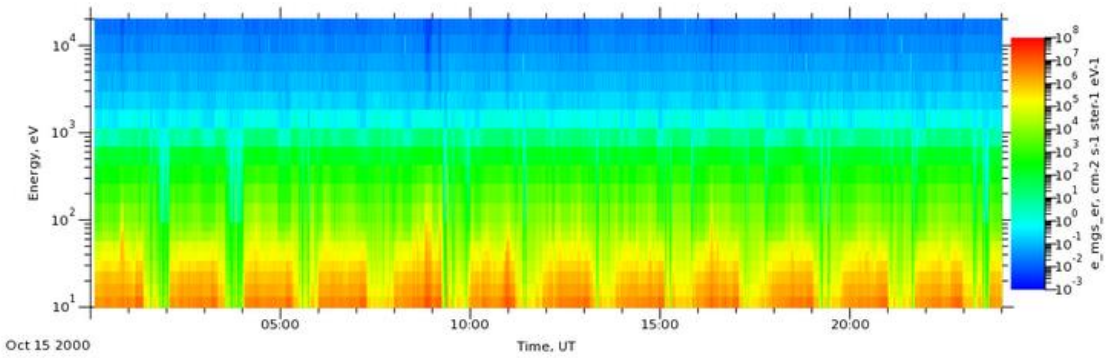
Cette correction a été effectuée dans « DataTypeMath.hh » (db1e413d et 11e094a7).

3.4 CORRECTION DE L’AFFICHAGE DES LEGENDES DES AXES D’UN PLOT

Une incohérence a été détectée lorsque l’échelle d’un plot était définie comme étant « logarithmique ».

Dans ce cas-là, la mention « log » était ajoutée au niveau de l’unité affichée dans la légende.

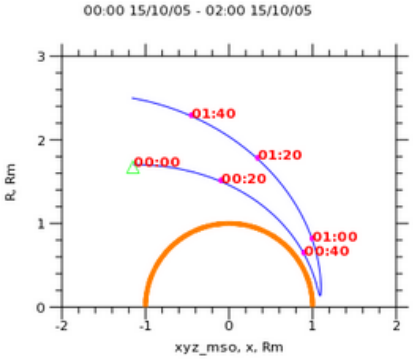
Hors, le fait de modifier la légende de l’axe ne modifie pas l’unité des données tracées, mais seulement la manière dont elles sont tracées. Ainsi, la mention « log » a été supprimée (4a39d79c).



Par ailleurs, le nom du paramètre tracé a été ajouté dans la légende du « Color Axis ».

3.5 CORRECTION AUTOUR DE L’AFFICHAGE DE TICKS LE LONG D’UNE COURBE

Des « ticks » peuvent être ajoutés au niveau du tracé d’une série afin d’indiquer une information temporelle le long de cette courbe.



Il peut arriver que la position du tracé de l'un de ces ticks tombe sur un trou de données.

Ce cas là n'était pas détecté, et l'appel de la fonction plPlot « pltex » avec un « NaN » provoquait des instabilités du contexte d'affichage. Cela a été corrigé (6fc1d86b).

3.6 RECUPERATION DES INFORMATIONS « NAME » ET « UNITS » POUR UNE TABLE « VARIABLE »

Une table variable dépend d'un, ou de plusieurs, paramètres.

Les informations portées par ce (ou ces) paramètre(s) peuvent être utilisées afin de déterminer le « name » et le « units » de la table (uniquement si elles ne sont pas définies au préalable).

Ce mécanisme a été implémenté (3fe8b83d)

4 INITIALISATION DES OPTIONS DE TRACES D'UN PARAMETRE DANS L'IHM DE DEFINITION D'UNE REQUETE DE PLOT

Le fichier de définition d'un paramètre accepte la définition d'un « plot » par défaut.

Cette définition par défaut est à prendre en considération au niveau de l'interface de définition d'une requête de type « plot ».

4.1 LECTURE DU « PLOT » PAR DEFAUT DANS AMDA_INTEGRATION

Le module « AMDA_Integration » implémente une interface entre le module « AMDA_IHM » et le module « AMDA_Kernel ».

Lors d'une requête de type « ParamsRequest », c'est-à-dire une requête s'effectuant sur des paramètres (par exemple pour un « plot » ou un « download »), un ensemble de classes (cf. « AMDA_Integration/src/Request/ParamsRequestImpl/Nodes/Requests/ ») sont utilisées afin **d'écrire** le fichier « XML » de définition de la requête pour le module « AMDA_Kernel ».

Dans le cadre de cette prestation, ces classes ont évolué afin qu'elles soient également en mesure **de lire** la définition d'une requête (966bd5f8).

Ces évolutions permettent au module « AMDA_Integration » de lire la définition d'un « plot » par défaut.

4.2 IMPLEMENTATION DE LA REQUETE « PARAMINFO » DANS AMDA_INTEGRATION

La requête de type « PARAMINFO » a été introduite dans le module « AMDA_Integration » (966bd5f8). Elle a pour vocation de fournir un ensemble d'information concernant un paramètre.

Lorsqu'elle est appelée avec la valeur « plot_init » pour le paramètre « type », elle fournit la définition du plot par défaut.

Elle se base notamment sur l'utilisation des implémentations décrites dans §4.1.

4.3 INITIALISATION DES OPTIONS DE TRACE DANS AMDA_IHM

L'ajout d'un paramètre dans l'IHM de définition d'une requête de type plot s'effectue via l'utilisation de la fonction « createNewParam » de la classe « PlotPanelObject ».

Cette fonction a été modifiée (ced82260) afin qu'elle soit en mesure d'effectuer une requête de type « PARAMINFO » (cf. §4.2) permettant de récupérer la définition du plot par défaut pour le paramètre.

Les différents objets définissants une requête de type plot sont ensuite initialisés via l'appel aux méthodes « loadFromData ».

5 AMELIORATION DU TRAITEMENT DE L'ABSENCE DE DONNEES POUR UN PARAMETRE

Lorsqu'une requête sur un paramètre s'effectue avant le « global start », ou après le « global stop », ou au milieu d'un trou de données, une exception était émise au niveau des différentes implémentations de « ParamGet » du module « AMDA_Kernel ».

Cette exception était « attrapée » par les implémentations de « ParamOutput », et les requêtes correspondantes sortaient en erreur.

Une requête pouvant s'effectuer sur plusieurs paramètres à la fois, ce comportement n'était pas satisfaisant.

L'émission de cette exception a donc été supprimée (a6490f4d).

La principale difficulté de cette tâche a été la correction des différents « Process » et « ParamOutput » afin de supprimer les nombreux effets de bords qui sont apparus.

6 SELECTION D'UNE COMPOSANTE D'UN PARAMETRE LORS DE LA DEFINITION D'UNE REQUETE

6.1 GENERATION DU FICHIER D'INFORMATION POUR UN PARAMETRE

Lors de l'exécution d'une requête, le module « AMDA_Kernel » est en mesure de récupérer les informations sur le type de données portées par un paramètre (« scalaire », « Tab1D » ou « Tab2D »), ainsi que les éventuelles informations sur les tables associées aux différentes dimensions du paramètre.

Le nouvel exécutable « amdaParameterInfo » a été implémenté au niveau du module « AMDA_Kernel » (ad920fd6). Il a pour vocation de, à partir de l'identifiant d'un paramètre fournit en entrée, récupérer l'ensemble de ces informations et de les écrire dans un fichier XML de sortie.

6.2 SCRIPT DE GENERATION DES FICHIERS D'INFORMATION POUR UNE BASE DE PARAMETRES

Le script « script/generate_param_info.sh » a pour vocation de générer l'ensemble des fichiers d'information associés à une base de paramètre.

Ce script a été fourni à titre d'exemple (d04038f4). Il devra être inclus dans la procédure de génération des paramètres, qui est de la responsabilité de l'IRAP.

Ce script s'appuie sur la définition de la configuration spécifique « app-generate-paraminfo ».

Il peut être appelé de la manière suivante :

```
cd AMDA_Kernel/app-generate-paraminfo
../script/loginDD_Server.sh
../script/generate_param_info.sh
```

6.3 EVOLUTION DE LA REQUETE « PARAMINFO » DE AMDA_INTEGRATION

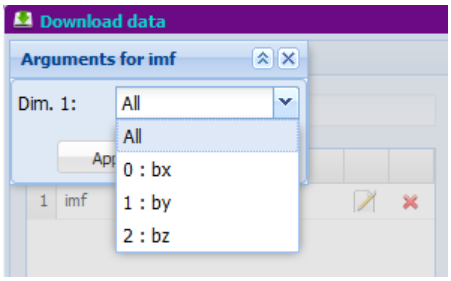
La requête de type « PARAMINFO » a été introduite dans le module « AMDA_Integration » lors de §4.2. Lorsqu'elle est appelée avec la valeur « param_info » pour le paramètre « type », elle fournit les informations du paramètre décrites dans §6.1 (f28f7c0e).

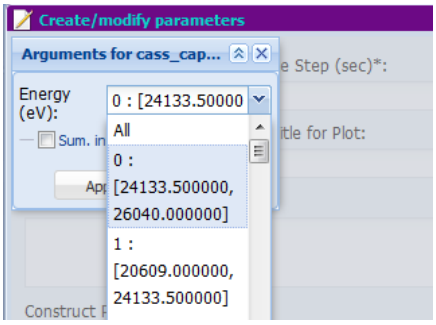
6.4 IMPLEMENTATION DE LA SELECTION D'UNE COMPOSANTE AU NIVEAU DE L'IHM

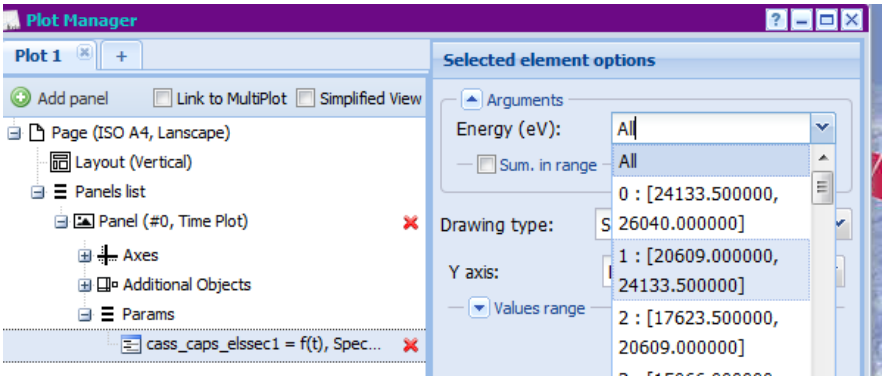
La sélection d'une composante pour un paramètre se fait via l'utilisation de l'IHM générique définie dans « js/app/views/ParamArgumentsUI.js ».

Le contenu de cette IHM est construit en fonction des informations retournées par un appel à la requête « PARAMINFO » décrite dans §6.3.

Cette IHM est dorénavant utilisée par les modules de « Plot », de « Download », de « Data Mining » et de « Parameter Definition » (7ac3ce50 et 33ce72a7).







Cela a impliqué également des modifications dans le module « AMDA_Integration » (3182799a).

6.5 SELECTION D'UNE COMPOSANTE PAR DRAG & DROP DEPUIS L'ARBRE

Pour certains paramètres (notamment les vecteurs), il est possible de réaliser un « Drag & Drop » d'une composante depuis l'arbre de l' « explorer » vers une IHM de définition d'une requête.

Dans ce cas, l'IHM de sélection d'une composante (cf. §6.4) est initialisée de manière à ce que la composante soit sélectionnée (63ac7745).

6.6 MISE EN CACHE DES INFORMATIONS DEJA CHARGEES DANS LE MODULE AMDA_IHM

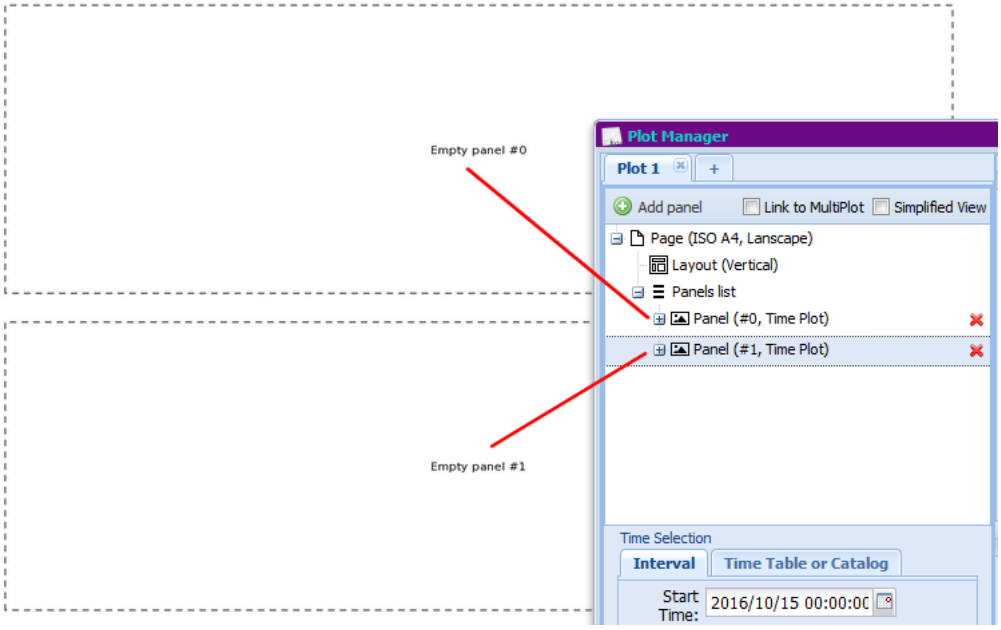
Les informations sur les paramètres qui sont chargées par AMDA_IHM sont conservées en cache afin de limiter le nombre d'appels à la requête « PARAMINFO ».

Cela se fait par le maintien d'un « registre » au niveau de l'« explorer » (0314bd32).

7 MODIFICATION DE L'ORDRE D'AFFICHAGE DES PANELS DANS UNE PAGE DE PLOT

La méthode virtuelle « computePanelsPosition » de la classe abstraite « Layout » a été introduite. Elle a pour vocation de fournir la position d'un panel sur une page.

Une implémentation de cette méthode a été réalisée pour les différents « layouts » (« LayoutAuto » et « LayoutVertical »). C'est dans ces implémentations que le nouvel ordre d'affichage des panels est établi (2f0b202e).



8 OPTIMISATION DU POSITIONNEMENT DES ELEMENTS D'UN PLOT

Après analyse des différents problèmes relatifs au positionnement des éléments d'un plot, trois causes majeures ont été relevées :

- Le calcul de la taille d'un caractère alphanumérique était erroné,
- Le calcul de la zone de tracé sur la page pour les panels était incorrect,
- Un ensemble d'erreurs et d'incohérences existaient dans le calcul de la place à réserver pour les éléments d'un panel,

Des erreurs dans l'héritage de la font de l'élément « parent » ont également été corrigés.

Au final, ce travail a demandé une refonte assez complète du « PlotOutput » (f6eaec4e).

Suite à ces corrections, une modification de la configuration par défaut a été effectuée au niveau du module « AMDA_Integration » (9937b488).

8.1 CORRECTION DU CALCUL DE LA TAILLE D'UN CARACTERE ALPHANUMERIQUE

Ce calcul est primordial dans le positionnement des différents éléments sur une page.

Il permet, par exemple de déterminer la hauteur qui sera à réserver pour l'écriture d'une légende, d'un titre, etc.

Ce calcul était erroné, et a été corrigé (cf. méthode « getCharacterSizeInPlPage » de la classe « PlPlotUtil »).

8.2 CORRECTION DU CALCUL DE LA ZONE DE TRACE SUR LA PAGE POUR LES PANELS

La zone dédiée au tracé d'un panel est déterminée à partir de l'espace restant après application de la marge et de l'écriture du titre de la page, et de l'application du « layout » sélectionné.

La fonction « getBoundsInPlPage » de la classe « Panel » permet dorénavant de calculer de manière précise la zone dédiée au tracé d'un panel.

8.3 CORRECTION DU CALCUL DE LA PLACE A RESERVER POUR LES ELEMENTS D'UN PANEL

En raison d'un grand nombre d'erreurs dans le calcul de la place à réserver pour les éléments d'un panel, ce calcul a été entièrement réécrit (cf. fonction « calculatePlotArea » de la classe « PanelPlotOutput »).

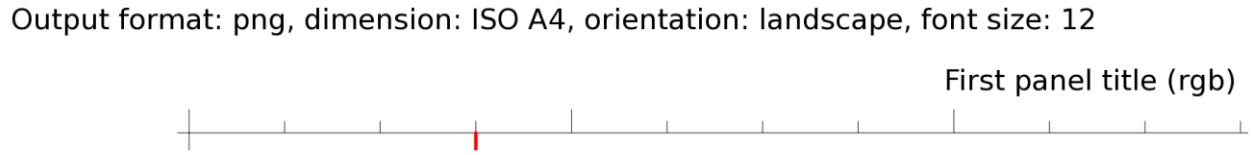
8.4 APERÇU DES CORRECTIONS

Cette section montre un « échantillon » des optimisations apportées.

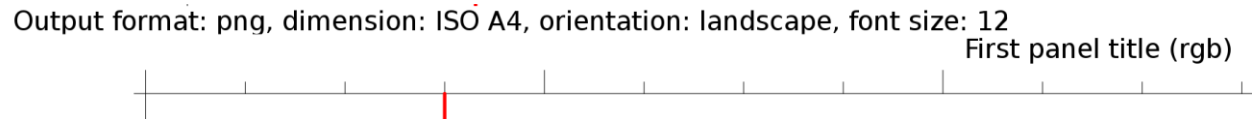
8.4.1 Espacement entre les titres des différents éléments

Le nouveau calcul de la taille d'un caractère alphanumérique permet de maitriser l'espacement entre l'écrire des titres des différents éléments d'un plot.

Avant correction :



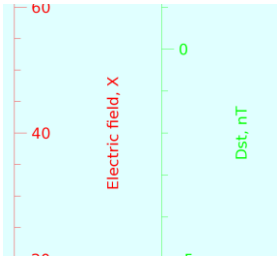
Après correction :



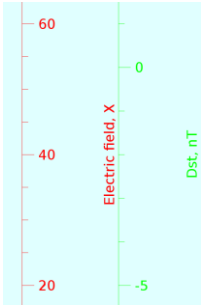
8.4.2 Espacement entre les légendes des axes

Le nouveau calcul de la taille d'un caractère alphanumérique permet de maitriser l'espacement entre les légendes des différents axes d'un plot.

Avant correction :



Après correction :

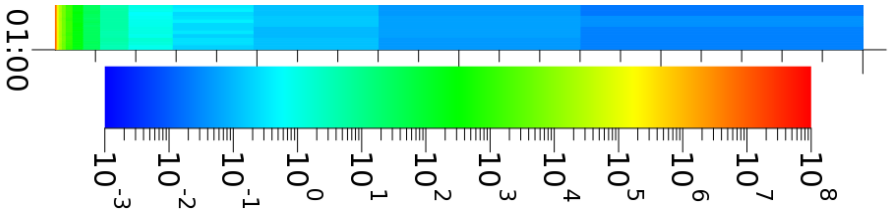


8.4.3 Espacement entre la zone d'un plot et un « Color Axis »

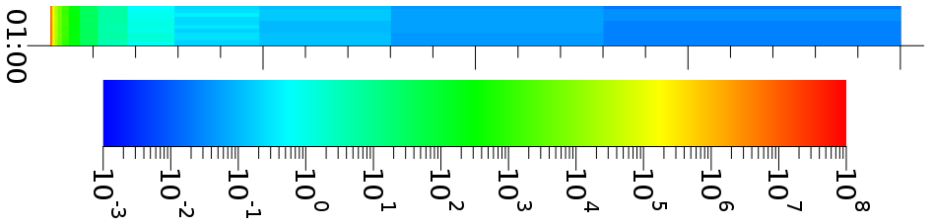
L'espacement entre la zone d'un plot et celle de sa « Color Axis » a été corrigé.

En effet, il arrivait que cet espacement soit trop faible, et que le « Color Axis » soit tracé sur les « ticks » du plot (principalement en mode « portrait »).

Avant correction :



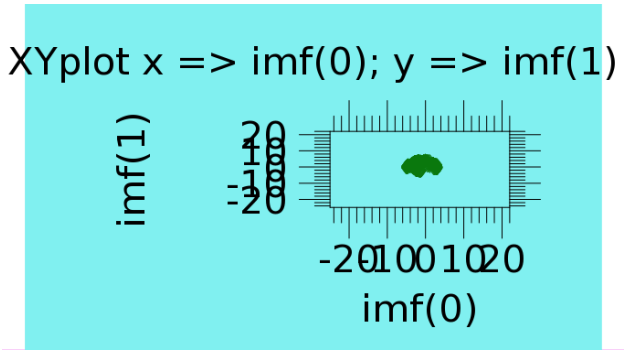
Après correction :



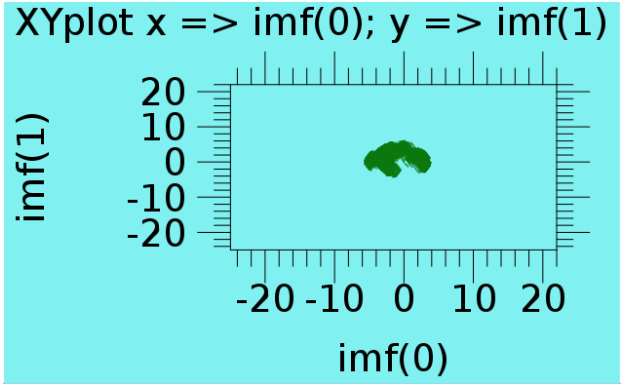
8.4.4 Optimisation de la zone de tracé d'un plot

Les différentes corrections ont permis une optimisation de la zone restante pour le tracé d'un plot.

Avant correction :



Après correction :



9 IMPLEMENTATION DU PARAMETRE « MORSHHAUSER_MARS_MODEL_MAGNETIC_FIELD » LE LONG D'UNE ORBITE ET INTRODUCTION DES « TEMPLATED PARAMETERS »

9.1 IMPLEMENTATION DU PROCESS DANS LE MODULE AMDA_KERNEL

Le calcul du modèle de Morshhauser a été introduit dans « AMDA_Kernel » sous forme d'une librairie externe (dans « AMDA_Kernel/src/ExternLib/Morschhauser) contenant la définition du « process » (a98e6fc1).

Il s'agit d'une implémentation d'un « SingleParamProcess » dont le paramètre d'entrée est un paramètre désignant une orbite (un paramètre de type « vector » est donc attendu, tout autre type de paramètre sortira en exception).

Le calcul est porté par la classe « MorschhauserCommon » qui constitue une réécriture en « C++ » du code fourni par l'IRAP.

Par soucis d'optimisation, les constantes qui étaient initialement lues dans des fichiers sont dorénavant définies dans le fichier « morschhauser_constants.hh » et compilées avec le module.

Le calcul du changement de repère de « MSO » vers « aerocentric » a également été implémenté (413a6873).

9.2 INTRODUCTION DES « TEMPLATED PARAMETERS » DANS LES MODULES AMDA_IHM ET AMDA_INTEGRATION

9.2.1 Définition des « templated parameters » dans AMDA_IHM

Les « templated parameters » disponibles sont définis dans un fichier XML situé à l'emplacement suivant : « AMDA_IHM/generic_data/ParamTemplate/ParamTemplateList.xml ».

Voici la définition des paramètres pour le modèle de Morshhauser :

```
<?xml version="1.0"?>


<paramTemplateList>
    <paramTemplate paramId='bmorsch_mso' fileName='bmorsch_mso_##sampling##_##orbit##'>
        <arguments>
            <argument key='sampling' name='Sampling Time' type='float' default='60'/>
            <argument key='orbit' name='Spacecraft Orbit' type='list' default='mex_xyz'>
                <item key='mex_xyz' name='MEX'/>
                <item key='mav_xyz_mso' name='MAVEN'/>
                <item key='xyz_mgs_mso' name='MGS'/>
            </argument>
        </arguments>
    </paramTemplate>
    <paramTemplate paramId='bmorsch_tot' fileName='bmorsch_tot_##sampling##_##orbit##'>
```

```
            <arguments>
                <argument key='sampling' name='Sampling Time' type='float' default='60'/>
                <argument key='orbit' name='Spacecraft Orbit' type='list' default='mex_xyz'>
                    <item key='mex_xyz' name='MEX'/>
                    <item key='mav_xyz_mso' name='MAVEN'/>
                    <item key='xyz_mgs_mso' name='MGS'/>
                </argument>
            </arguments>
    </paramTemplate>
</paramTemplateList>
```

L'attribut « paramId » du nœud « paramTemplate » désigne l'identifiant du paramètre, utilisé notamment dans l'arbre de l'« explorer » de AMDA_IHM pour désigner ce paramètre.

L'attribut « fileName » du nœud « paramTemplate » désigne le nom du fichier de définition du paramètre (par exemple «bmorsch_mso_##sampling##_##orbit##.xml ») qui doit être présent dans le même répertoire que le fichier « ParamTemplateList.xml ».

Le nœud « arguments » définit les différents arguments du paramètre « templaté ». Dans le cas des paramètres du modèle de Morshhauser, deux arguments sont attendus :

- « sampling » : temps d'échantillonnage à appliquer,
- « orbit » ; identifiant du paramètre d'orbite à utiliser en entrée.

Dans le fichier de définition d'un paramètre, ainsi que dans son nom, les mots encadrés par « ## » doivent faire référence à l'un des arguments définis.

Exemple du fichier de définition du paramètre pour « bmorsch_mso_##sampling##_##orbit## » :

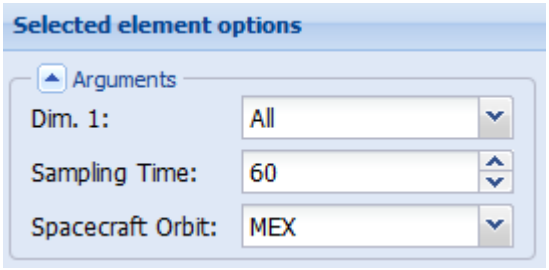
```
<?xml version="1.0" encoding="UTF-8"?>
<param xml:id="bmorsch_mso_##sampling##_##orbit##">
  <get>
      <amdaParam name='##orbit##'/>
  </get>
  <process                                                    description="bmorsch_mso(##sampling##,
##orbit##)">#morschhauser_bfield(#sampling_classic($##orbit##;##sampling##;5))</process>
  <output/>
</param>
```

9.2.2 Enrichissement du résultat de la requête « PARAMINFO » avec les informations d'un paramètre templaté

La requête « PARAMINFO » du module « AMDA_Integration », appelée avec la valeur « param_info » pour le paramètre « type » (cf. §6.4), fournit la définition des différents arguments disponibles pour un paramètre templaté (bf27ba04).

9.2.3 Sélection des valeurs des arguments d'un paramètre templaté depuis AMDA_IHM

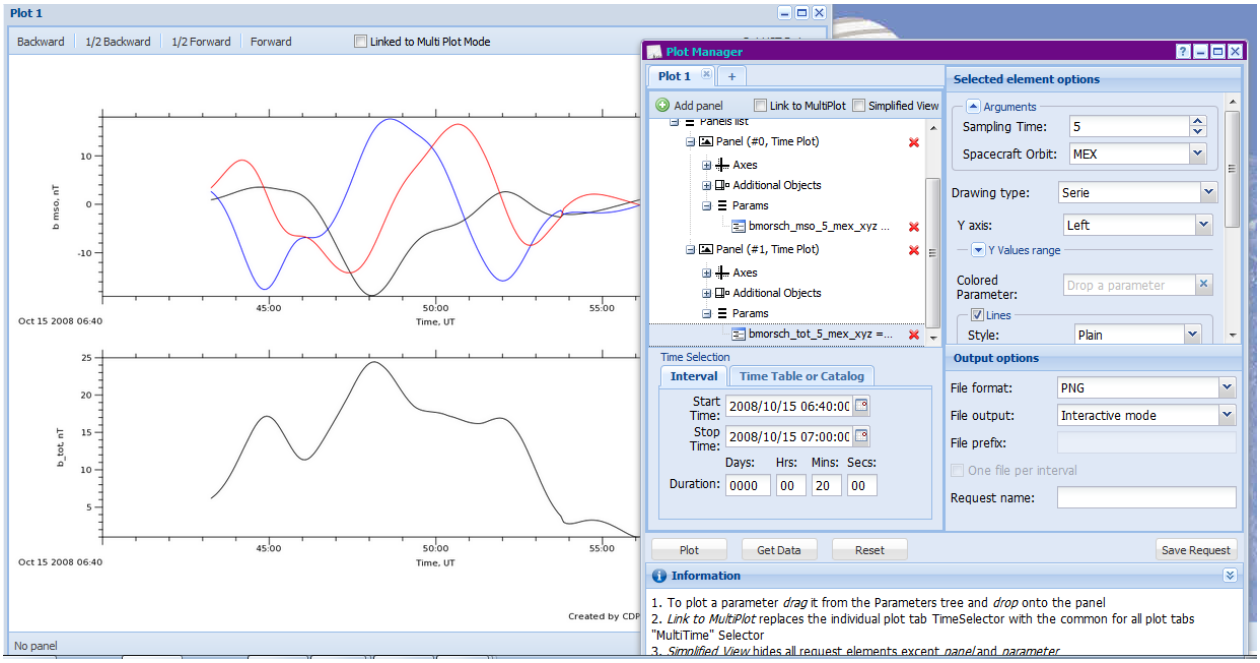
L'IHM générique (définie dans « js/app/views/ParamArgumentsUI.js ») utilisée pour la sélection d'une composante (cf. §6.4), contient également la construction des différents champs permettant la définition des valeurs des arguments d'un paramètre templaté (51b7c77c) :



9.2.4 Construction du paramètre « réel » à partir des valeurs des arguments d'un paramètre templaté dans AMDA_Integration

Au moment de l'exécution d'une requête, et lorsque l'utilisation d'un paramètre templaté est détecté, un mécanisme du module AMDA_Integration permet de substituer les mots encadrés par « ## » (cf. §9.2.1) par les valeurs définies par l'utilisateur depuis l'IHM (cf. §9.2.3), construisant ainsi le fichier de définition du paramètre « réel » à utiliser pour la suite de l'exécution de la requête (bf27ba04 et e4545ed5).

9.3 EXEMPLE DE TRACE DU PARAMETRE « MORSHHAUSER MARS MODEL MAGNETIC FIELD »



10 CHARGEMENT DYNAMIQUE DES COMPOSANTS DE L'IHM DE DEFINITION D'UN PLOT

Les différents composants utilisés pour définir les différents éléments d'un plot étaient chargés lors de l'ouverture du module de définition d'un « Plot ».

Une évolution de la classe « js/app/views/PlotComponents/PlotElementPanel.js » a permis de supprimer ce chargement systématique pour le remplacer par un chargement « à la demande ».

Ainsi, ces composantes ne sont chargées que lorsque l'utilisateur les sélectionne dans l'arbre de définition d'une requête de plot (0a295c3e).

11 INTRODUCTION DES « SHARED OBJECTS »

Avec l'accord des équipes techniques de l'IRAP, et compte tenu des difficultés rencontrées lors de la phase de spécification, il a été acté que cette tâche devait se concentrer sur :

- L'implémentation du mécanisme de base permettant à un utilisateur de partager un objet de son espace de travail avec les autres utilisateurs,
- La mise en place du mécanisme uniquement pour les « TimeTable » et les « Catalog », mais s'appuyant sur une conception permettant d'envisager aisément l'ajout de nouveaux types d'objets à partager,

Aucun mécanisme de « modération » n'a été implémenté. En revanche la conception établie a pris en considération cette possibilité pour l'avenir.

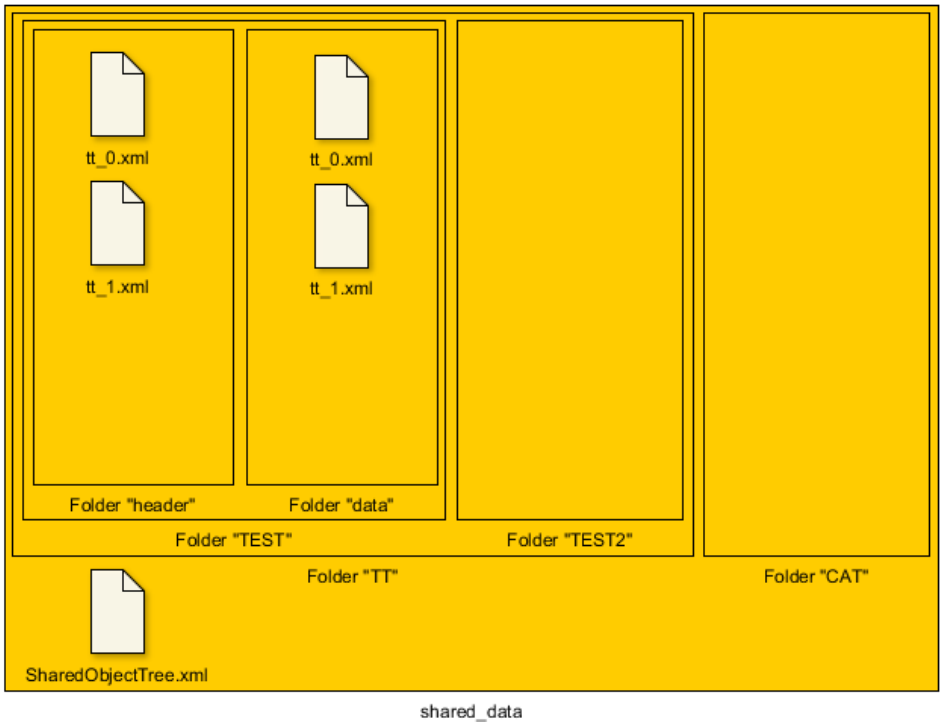
(169f14d2, a242857a & 09eefb2d).

11.1 ORGANISATION DE L'ESPACE « SHARED OBJECT »

Les objets partagés sont stockés dans le répertoire « AMDA_IHM/shared_data ».

Ce répertoire contient :

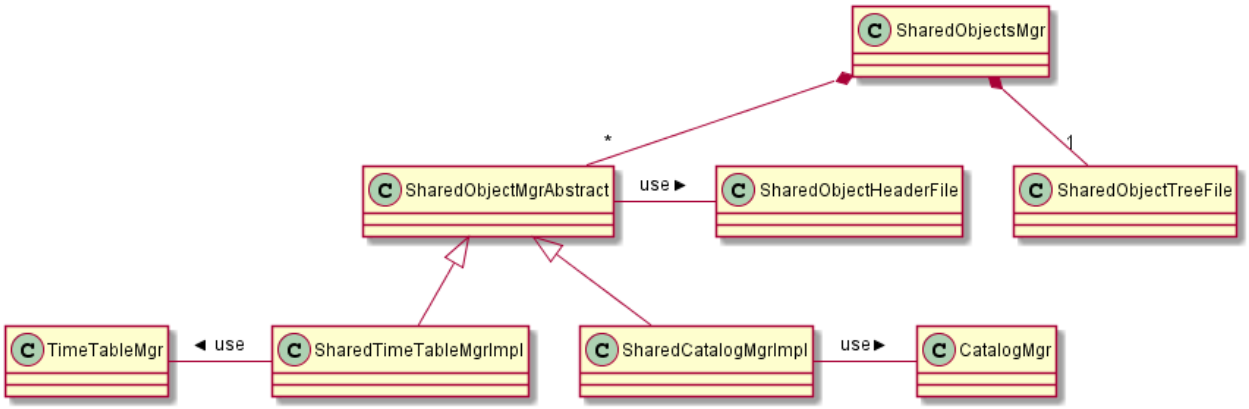
- Un fichier XML « SharedObjectTree.xml » définissant la liste des objets partagés et des répertoires disponibles. Ce fichier est mis à jour par « synchronisation » avec l'espace disque (cf. §11.3).
- Un répertoire par type d'objet partagé disponible (cf. §11.4).



Dans les répertoires liés aux types d'objets (dans notre exemple « TT » et « CAT »), un administrateur peut créer autant de répertoires qu'il le souhaite (cf. §11.5). Chacun de ces répertoires contiendra (après synchronisation, cf. §11.3) :

- Un répertoire « header » contenant un ensemble de fichiers de description des objets partagés,
- Un répertoire « data » contenant un ensemble de fichiers de données des objets partagés.

11.2 CONCEPTION GENERALE



La classe « SharedObjectsMgr » est la classe gérant l'ensemble des mécanismes liés aux « shared objects ». Elle instancie :

- Un objet « SharedObjectTreeFile » dédié à la manipulation du fichier XML de définition de l'arbre des « shared objects »,
- Un ensemble d'objets héritants de la classe « SharedObjectMgrAbtract », représentants les différents « managers » utilisés pour manipuler les différents types de « shared objects ».

La classe « SharedObjectMgrAbtract » utilise une instance de « SharedObjectHeaderFile » pour manipuler le fichier « header » d'un « shared object ».

11.3 MECANISME DE SYNCHRONISATION

Le mécanisme de synchronisation consiste à un « scan » du répertoire des objets partagés afin de mettre à jour le fichier « SharedObjectTree.xml » en conséquence (cf. §11.1).

Elle s'effectue en appelant la méthode « updateTree » du « SharedObjectsMgr », ou en appelant directement le script php « AMDA_IHM/php/updateSharedObjectsTreeFile.php ».

11.4 AJOUT D'UN NOUVEAU TYPE D'OBJET PARTAGE

L'ajout d'un nouveau type d'objet partagé revient à implémenter une nouvelle classe héritant de « SharedObjectMgrAbtract » et à la déclarer dans la fonction « init » du « SharedObjectsMgr ».

La nouvelle classe doit contenir une implémentation des méthodes abstraites suivantes :

- « getDataInfo » : il s'agit d'une méthode qui retourne, sous forme d'un tableau associatif « clé / valeur », les informations qui peuvent être lues dans le fichier de données
- « addData » : il s'agit de la méthode qui va copier le fichier de données du type d'objet considéré dans l'espace des objets partagés.

Cette classe doit également déclarer la valeur de « rootdirname » de « SharedObjectMgrAbtract », afin de préciser le nom du répertoire à utiliser pour ce type d'objet partagé.

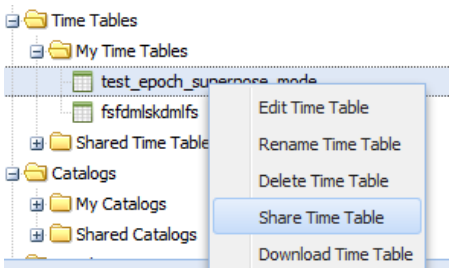
11.5 AJOUT / SUPPRESSION D'UN REPERTOIRE

L'administrateur du système peut ajouter ou supprimer manuellement des répertoires dans le répertoire lié à un type d'objet (cf. §11.1).

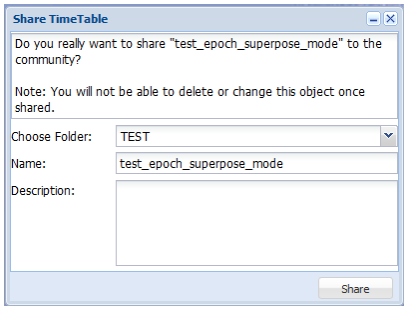
Après une telle opération, une synchronisation de l'espace des objets partagés est nécessaire (cf. §11.3).

11.6 AJOUT D'UN OBJET PARTAGE DEPUIS AMDA_IHM

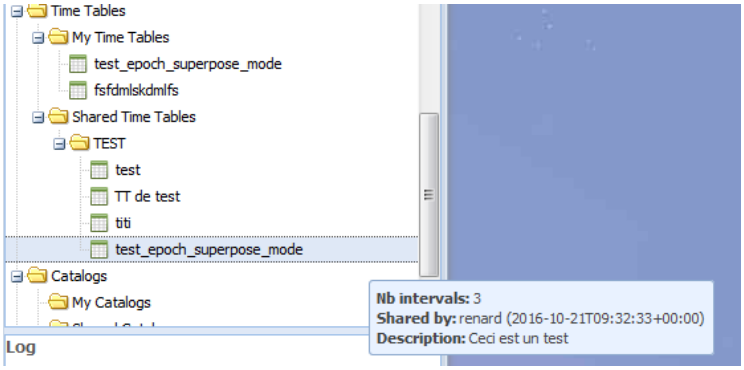
Un objet de type « TimeTable » ou « Catalog » peut être ajouté dans l'espace des objets partagés depuis l'action « Share … » du menu contextuel :



Une fenêtre permet à l'utilisateur de sélectionner le répertoire dans lequel déposer son objet (cf. §11.5), ainsi que le nom et la description de l'objet (informations requises) :



Un clic sur « Share » aura pour effet de copier son objet dans l'espace des objets partagés.



11.7 SUPPRESSION D'UN OBJET PARTAGE

La suppression d'un objet se fait en le supprimant à la fois son fichier « header » et son fichier « data » (cf. §11.1) dans l'espace des objets partagés.

Après une telle opération, une synchronisation de l'espace des objets partagés est nécessaire (cf. §11.3).

12 TRACE « EPOCH PLOT » DEPUIS UNE « TIME TABLE »

Cette tache a consisté à donner la possibilité à un utilisateur de fournir une « TimeTable » en entrée d'un tracé de type « Epoch Plot ».

Dans ce cas-là, le temps central considéré pour chaque intervalle de temps correspond au temps central réel de l'intervalle (db4a6b56).

13 CORRECTION DU BUG LORS D'UN CHANGEMENT D'INTERVALLE DANS LE MODULE « AMDA_KERNEL »

Ce bug apparaissait dans la configuration suivante :

- AMDA_Kernel réclame un paquet de données à DDServer ,
- DDServer lui retourne un paquet de données. La dernière donnée retournée coïncide avec la dernière donnée d'un fichier de la base,
- AMDA_Kernel réalise les traitements adéquates aux données reçues, et réclame ensuite un nouveau paquet de données à DDServer ,
- DDServer réalise qu'en ouvrant le fichier suivant de la base, la date de fin de l'intervalle de temps est atteinte. Il n'a donc pas de données à fournir DDServer.

Dans cette configuration, le mécanisme de changement d'intervalle de temps du module « AMDA_Kernel » ne fonctionnait pas correctement.

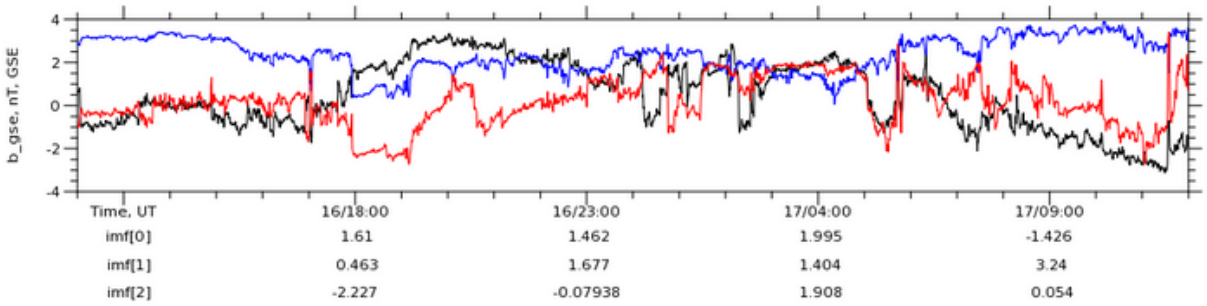
Ce bug a été corrigé en parti par les développements effectués pour §5 et par un ensemble de corrections effectuées sur les « Process » et les « ParamGet » (a6490f4d & fa4b7852).

14 CORRECTION D'UN BUG LORS D'UN TRACE D'UN PLOT DE TYPE « TICKMARKS »

Cette correction a consisté en une refonte du mécanisme implémenté dans « src/ParamOutputImpl/Plot/TickPlot/TickMarkDecorator.cc ».

Avant la correction, les valeurs des paramètres à tracer dans un plot « TickMarks » étaient déterminées en partant sur le principe que le premier « major tick » coïncidait avec le « Start Time » de l'axe. Hors cela n'est pas toujours le cas.

Dorénavant, les valeurs à tracer sont correctement déterminées au moment du tracé des « major tick » (225ac17f).



15 CALCUL D'UN CHANGEMENT DE REPERE A LA VOLEE DANS LE MODULE AMDA_KERNEL

15.1 PLUGIN DE CALCUL D'UNE TRANSFORMATION DE REPERE DANS AMDA_KERNEL

Le calcul d'une transformation de repère a été implémenté sous forme d'un plugin de « AMDA_Kernel ». Son code source est inclus dans le répertoire : « AMDA_Kernel/src/SpiceKernel ».

Il se base sur l'utilisation des « Spice Kernel ». La compilation de ce plugin nécessite donc l'installation du prérequis « cspice » (cf. §2.3).

(cebd3f0e & 063337d8).

15.1.1 Configuration dédiée au chargement des « SpiceKernel »

Deux nouvelles clés ont été introduites dans le fichier « app.properties » :

```
# Spice Kernel config schema
app.spicekernel.configxsd=../config/SpiceKernel/config.xsd
# Spice Kernel config file
app.spicekernel.configfile=spiceKernelConfig.xml
```

La clé "app.spicekernel.configfile" définit le fichier de configuration XML qui permet au système de charger les kernels en fonction de la transformation demandée, alors que la clé « app.spicekernel.configxsd » définit le chemin vers son « schéma » associé.

Le fichier de configuration se présente de la manière suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
<spicekernelconfig>
    <datafiles rootPath="/usr/local/data/ORBITS/Kernels/">
            <datafile bodyId="-1" path="general/naif0011.tls"/>
            <datafile bodyId="-1" path="general/de432s.bsp"/>
            <datafile bodyId="-1" path="general/pck00010.tpc"/>
            <datafile bodyId="-1" path="general/frames.tf"/>
    </datafiles>
</spicekernelconfig>
```

Chaque nœud « datafile » contient les attributs suivants :

- « bodyId » : identifiant « spice » du corps considéré. Si la valeur de cet identifiant est « -1 », le fichier sera systématiquement chargé, sinon il ne le sera que lorsque cela est nécessaire (en fonction de la transformation demandée).
- « path » : chemin relatif (par rapport au « rootPath » défini au niveau du nœud « datafiles ») vers le fichier kernel correspondant.

15.1.2 Plugin « SpiceKernel »

Le plugin « SpiceKernel » est constitué :

- De la classe « SpiceKernelConfig » qui représente l'objet dans lequel est stocké la configuration liée au SpiceKernel (cf. §15.1.1),
- De la classe « SpiceKernelConfigParser » qui constitue le parser du fichier XML de configuration, et dont la responsabilité est d'instancier l'objet « SpiceKernelConfig »,
- De la classe « SpiceKernelStatus », et plus spécifiquement de la méthode statique « CheckErrors », utilisée afin de vérifier le statut du dernier appel à une méthode de « cspice »,
- Et enfin de la classe « SpiceKernelMgr » qui est le « chef d'orchestre » des appels effectués à « cspice » et qui contient notamment la méthode « computeTransformation » de calcul d'un changement de repère.

15.2 PROCESS DE CALCUL DE TRANSFORMATION DANS AMDA_KERNEL

Le process de calcul d'une transformation est porté par le plugin « src\ExternLib\FramesTransformation ».

Ce process utilise le plugin « SpiceKernel » afin d'effectuer le calcul d'un changement de repère.

Un tel process s'écrira de la manière suivante dans un fichier de définition d'un paramètre :

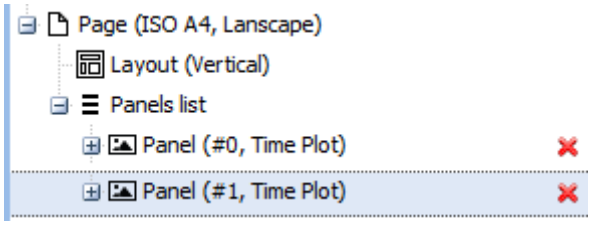
```
<process>#framesTransformation($param_GSE;GSE;GSM;true) </process>
```

Dans cet exemple:

- Le paramètre "$param_GSE" représente le paramètre d'entrée, exprimé dans le repère GSE,
- Le paramètre « GSE » représente le nom du repère d'entrée,
- Le paramètre « GSM » représente le nom du paramètre de sortie,
- Le paramètre « true » précise qu'il s'agit d'une transformation s'appliquant sur un paramètre représentant une position. Dans ce cas-là, les translations nécessaires sont effectuées lors du calcul de changement de repère.

16 IDENTIFICATION D'UN PANEL PAR SON INDEX DANS LE MODULE AMDA_IHM

Dorénavant, un panel est désigné par son index dans l'interface de définition d'une requête de plot (509bf9fa).



Une évolution du module « AMDA_Kernel » autorise la définition d'un index pour un panel au niveau du fichier de définition d'une requête (a88159f7).

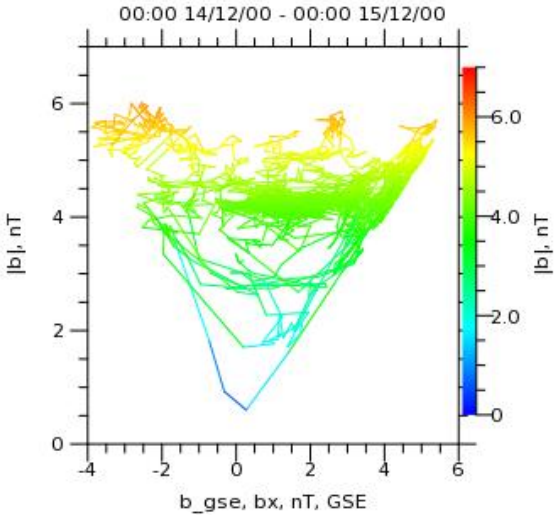
Cette évolution a également impacté le module « AMDA_Integration » (99a73aaf).

A noter que la notion d'identifiant d'un panel continue à exister, mais elle n'est plus visible pour l'utilisateur.

17 CORRECTION D'UN BUG LORS DU TRACE D'UN PARAMETRE LE LONG D'UNE SERIE AVEC UN AXE Z DE TYPE « LOGARITHMIQUE »

Ce bug était dû au fait que pour un tracé d'un paramètre coloré le long d'une série avec un axe Z de type « logarithmique », le log n'était pas appliqué sur les données pour déterminer la couleur à appliquer (381f60f0).

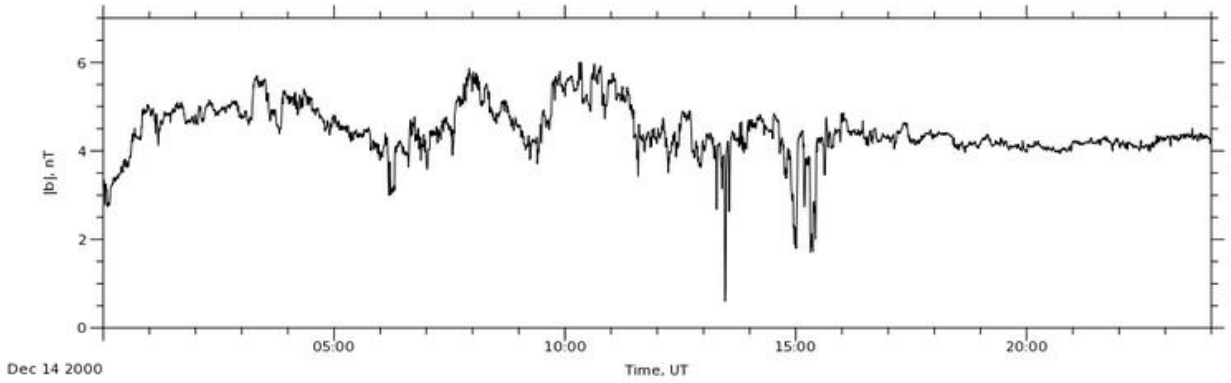
Après correction, voici le plot obtenu :



18 CORRECTION DE L'AFFICHAGE DES TICKMARKS SUR UN AXE TEMPOREL

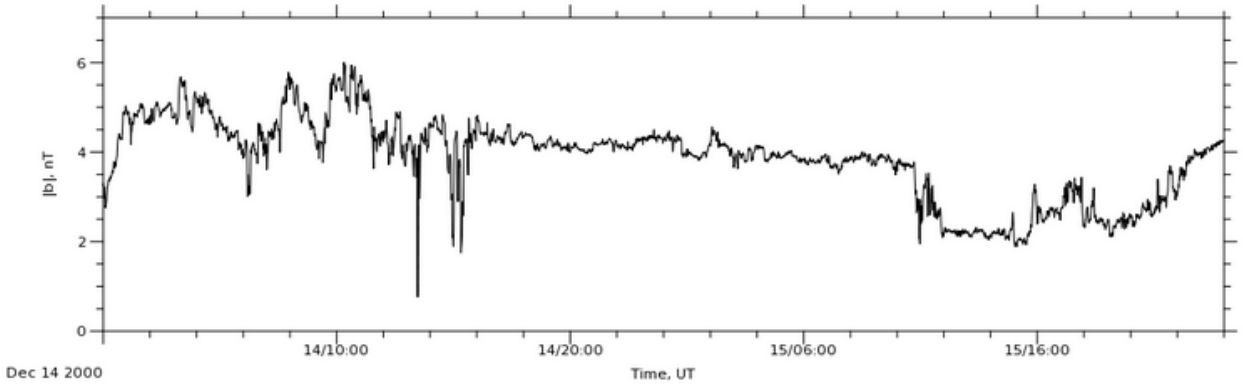
18.1 NE PAS AFFICHER LA DATE DANS LES TICKMARKS LORSQU'ELLE N'EST PAS NECESSAIRE

Lors d'un plot d'une journée, avec le « start time » à « 00h00 », il n'est pas nécessaire d'indiquer la date dans les tickmarks (243565aa).



18.2 NE PAS AFFICHER LE JOUR DANS LA LEGENDE DE L'AXE TEMPOREL LORSQU'IL N'EST PAS NECESSAIRE

Lors d'un plot de plus d'une journée, le jour est indiqué dans les tickmarks, il n'est donc pas nécessaire de l'indiquer dans la légende de l'axe temporel (243565aa).



18.3 LA POSITION DES MAJOR TICKS DE L'AXE TEMPOREL DEPEND DU JOUR CONSIDERE

Lors d'un plot d'une journée, par exemple, deux « major ticks » consécutifs sont espacés de 5 heures.

Ainsi, si on considère un plot dont le premier « major tick » est à « 00H00 », les suivants seront tracés à « 05H00 », « 10H00 », « 15H00 » et « 20H00 ».

Si on trace le même plot au jour suivant, le prochain « major tick » sera à « 01H00 » (car espacé de 5 heures par rapport au précédent).

Nous voyons donc que le problème est lié à la stratégie de répartition des « major tick » sur l'axe temporel. Pour un plot d'une journée, il faudrait que 24 soit un multiple de l'espacement entre deux « major ticks » consécutifs.

Cette stratégie de répartition est définie dans la fonction « computeAutoMajorTickSpace » de la classe « TimeAxis ».

Après correction de cette stratégie, nous nous sommes rendu compte d'une « désynchronisation » avec le mécanisme mis en place dans plPlot (fonction « pldtik », cf. http://plplot.sourceforge.net/doxygen/html/pldtik_8c_source.html).

Il aurait donc fallu modifier également cette fonction plPlot, ce qui n'a pas été fait avec l'accord de l'équipe technique de l'IRAP.

Ainsi, ce problème n'a pas été résolu dans le cadre de cette prestation.

19 MAINTIEN DE L'ETAT D'OUVERTURE DE NŒUD DANS L'INTERFACE DE DEFINITION D'UNE REQUETE DE PLOT

L'état d'ouverture des nœuds est conservé dans les différents « models » décrivant une requête de Plot. Ainsi, lors de la reconstruction de l'arbre de la requête, le système est en mesure de rétablir l'état précédent (2a0b8d2c).

20 PRISE EN COMPTE DES PARAMETRES EISCAT 2D

Les paramètres EISCAT ont été définis comme étant des paramètres « templatés » afin de donner la possibilité à un utilisateur de :

- Sélectionner la table à associer au paramètre : « Range », « Altitude », « Geographic latitude » ou « Geographic longitude ». Pour un plot de type « spectro », cette table est représentée sur l'axe Y.
- Sélectionner un filtrage par mode de l'instrument. Lorsque l'option « All » est sélectionnée, aucun filtrage n'est effectué.

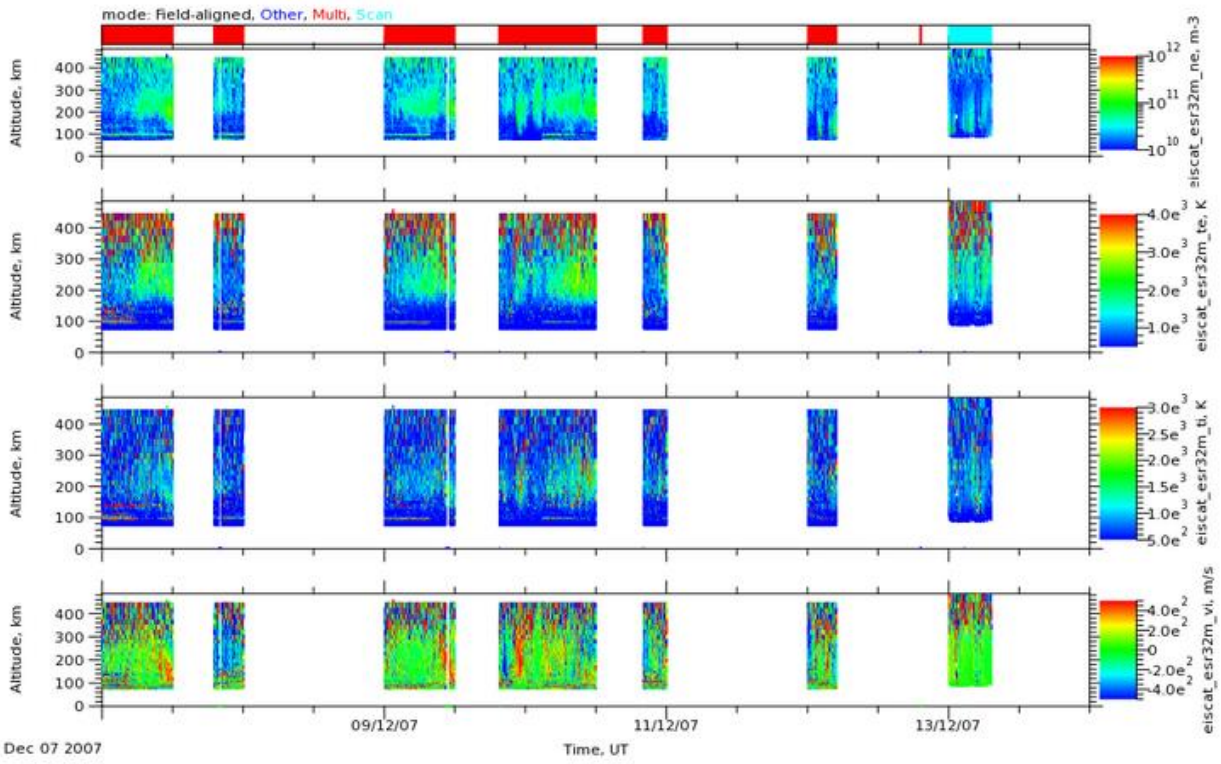
Note : L'option « corrected » a également été ajoutée. En revanche, les données corrigées n'étaient pas disponibles dans la base de données. Cette option n'est donc pas implémentée dans le code.

Exemple de définition d'un paramètre EISCAT 2D :

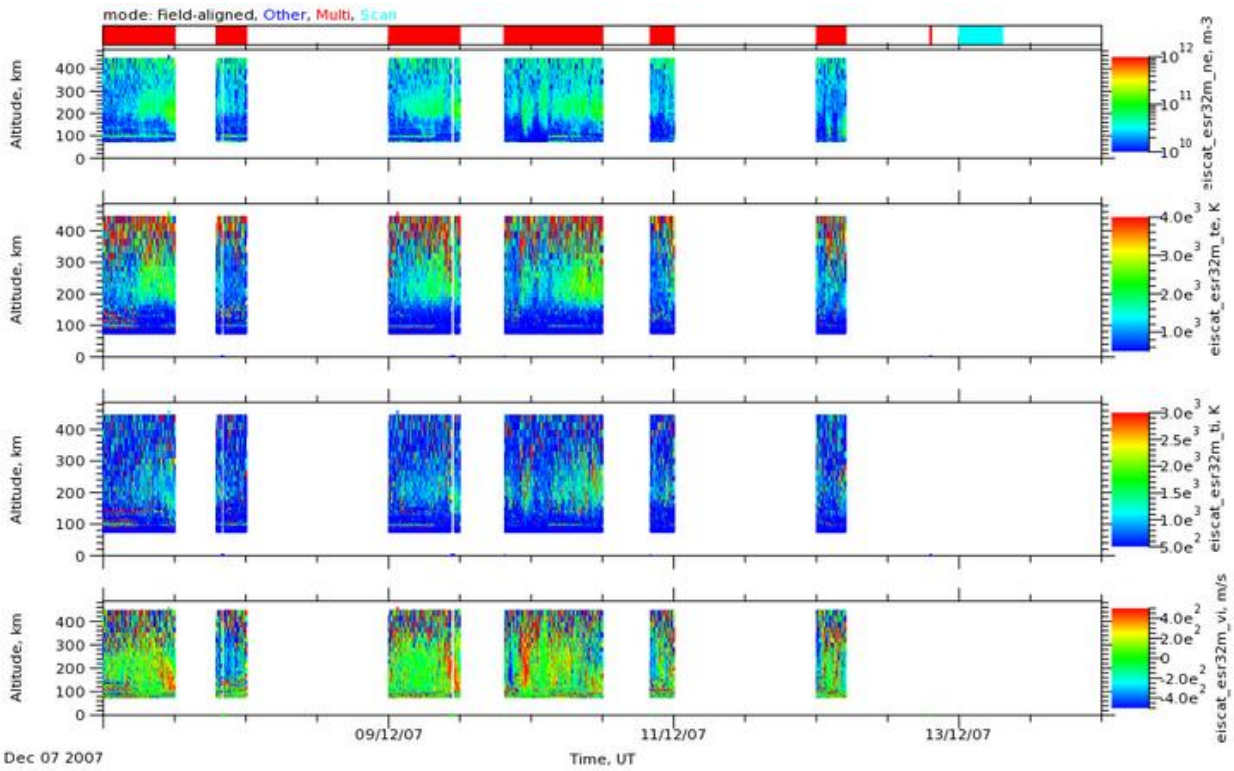
```
<paramTemplate paramId='eiscat_esr42m_ne' fileName='eiscat_esr42m_ne_##mode##_##corrected##_##yaxis##'>
        <arguments>
                <argument key='mode' name='Mode' type='list' default='fieldaligned'>
                        <item key='fieldaligned' name='Fixed pos. : Field aligned'/>
                </argument>
                <argument key='corrected' name='Corrected' type='bool' default='0'/>
                <argument key='yaxis' name='Y Axis' type='list' default='range'>
                        <item key='range' name='Range'/>
                        <item key='alt' name='Altitude'/>
                        <item key='geolat' name='Geographic latitude'/>
                        <item key='geolong' name='Geographic longitude'/>
                </argument>
        </arguments>
</paramTemplate>
```

Exemples de traces:

Sans filtrage sur les modes :



Avec filtrage pour ne voir que le mode « Multi » :



21 DEFINITION DE CONSTANTES DANS LE MODULE « AMDA_KERNEL »

Le fichier « src/Common/AMDA_constants.hh » a été ajouté. Il est dédié à recevoir la définition des différentes constantes internes au module « AMDA_Kernel » (95430126).

Une inclusion de ce fichier est effectuée au moment de la génération des sources d'un paramètre dérivé (cf. fonction « generateCppfile » de la classe « ProcessStandard »), autorisant ainsi l'utilisation de ces constantes pour la définition d'un « process » dans un fichier de définition d'un paramètre.

22 CORRECTION D'UN BUG DDSERVER

L'intitulé initial de cette tâche a été conservé, même si au final il ne s'agissait pas d'un bug DDServer.

En effet, et après de longues prospections, il a été découvert qu'une tache « CRON » s'exécutant sur le poste « manunja » de l'IRAP (poste contenant la base de test, ainsi que le DDServer utilisé par AMDA_Kernel) était responsable de l'échec de l'exécution de la requête.

En effet, cette tâche s'exécutait toutes les 30 minutes et « tuée » tous les process « DDServer » dont le temps d'exécution était supérieur à 2 minutes.

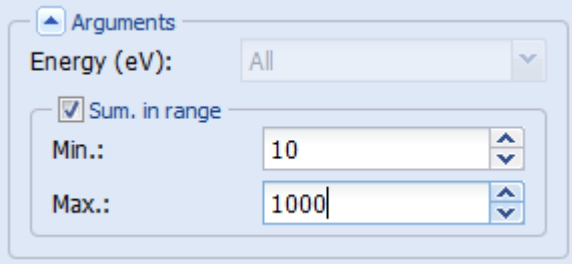
Après suppression de cette tâche CRON, l'exécution des longues requêtes s'effectuent correctement.

23 SOMME DES DONNEES D'UN PARAMETRE SUR UN INTERVALLE DE SA « TABLE »

23.1 DEFINITION DEPUIS AMDA_IHM

Lorsqu'une table est associée à l'une des dimensions d'un paramètre, il est possible de sélectionner un intervalle de cette table sur laquelle les données du paramètre seront sommées (690e0a87).

Cela se passe au niveau de l'IHM générique « ParamArgumentsUI.js » (cf. §6.4) et se présente de la manière suivante :



23.2 MECANISME DANS AMDA_INTEGRATION

Lors de l'exécution d'une requête, lorsqu'un paramètre sur lequel ce type de somme est détecté, il sera automatiquement transformé en paramètre templaté « sum_into_table_range ». Ce paramètre templaté est défini de la manière suivante (f25e55ba et 0bb6e345) :

```
<?xml version="1.0" encoding="UTF-8"?>
<param xml:id="sum_into_table_range_##paramid##_##relateddim##_##min##_##max##">
 <get>
    <amdaParam name='##paramid##'/>
 </get>
 <process description="sum_into_table_range(##paramid##, ##relateddim##, ##min##, ##max##)">#sum_into_table_range($##paramid##;##relateddim##;##min##;##max##)</process>
 <output/>
</param>
```

23.3 IMPLEMENTATION DU PROCESS DANS AMDA_KERNEL

Le process « sum_into_table_range » est défini dans le plugin « AMDA_Kernel\src\ExternLib\ sum_into_table_range » (bd34e02d).

Il s'agit d'un process de type « MultiParamProcess », en effet, lorsque la table du paramètre est variable au cours du temps, il doit être en mesure de « tirer » aussi bien les données du paramètre que les données de sa table.

24 DOCUMENTS APPLICABLES ET DE REFERENCE (A/R)

A/R	Référence	Titre
[R0]	PRC-14421-AMD-AV3	Proposition commerciale – Développements complémentaires AMDA-NG
[R1]	CDPP-MI-32500-463-CS	Manuel d'installation du noyau AMDA-NG (2nde partie)
[R2]	CDPP-MI-32500-505-SIL	Manuel d'installation du noyau AMDA-NG (3ème partie) et d'intégration avec l'IHM

25 GLOSSAIRE ET ABREVIATIONS

25.1 GLOSSAIRE

Terme	Définition

25.2 ABREVIATIONS

Abréviation	Nom détaillé
IRAP	Institut de Recherche en Astrophysique et Planétologie
CDPP	Centre de Données de la Physique des Plasmas
AMDA	Automated Multi-Dataset Analysis