Spécifications du format de postraitement LATA 2.0

Benoit Mathieu

25 mars 2010

1 Introduction

Ce document décrit le format de postraitement LATA 2.0, introduit à partir de la version 1.5.6 de Trio_U.

Le format LATA est destiné à enregistrer des maillages et des champs à différents pas de temps lors d'un calcul à des fins de postraitement. Le but n'étant pas de faire des reprises de calcul, les valeurs sauvegardées peuvent avoir une précision inférieure à celles du calcul.

La conception du format LATA est guidée par les objectifs suivants :

- disposer d'un format de sauvegarde extrêmement simple, lisible par un code de quelques lignes seulement (d'où l'utilisation d'un fichier maître au format ASCII et de fichiers de données ASCII ou binaires ayant une structure minimaliste),
- permettre la sauvegarde de très gros volumes de données (d'où l'écriture d'un fichier différent pour chaque champ et pour chaque pas de temps),
- offrir de façon simple certaines fonctionnalités pour les maillages dynamiques (utilisation d'un maillage différent à chaque pas de temps),
- disposer d'un format extensible (ajoute de champs localisés aux faces, description des faces, des arêtes, attributs supplémentaires des éléments géométriques, stockage de maillages structurés, raffinement dynamique, etc).

On distingue dorénavant deux niveaux d'abstraction pour les données écrites dans le fichier lata :

- On appellera LataDB (LataDataBase) le conteneur de bas niveau comparable à HDF par exemple : on spécifie comment stocker de façon hiérarchique des tableaux d'entiers ou de nombres flottants. Les spécifications de ce conteneur sont conçues pour être pérennes et ne plus changer avec les versions des codes.
- Les maillages et le champs (qui sont des objets de niveau d'abstraction plus haut) sont stockés dans le conteneur LataDB selon des conventions qui peuvent éventuellement évoluer d'une version du code à l'autre (notamment par ajout de nouveaux types d'objets).

Ce document comporte deux parties :

- les spécifications du format LataDB proprement dit (appelé Lata V2),
- une description de l'utilisation que Trio_U fait de ce format (de quelle façon les informations sur les maillages, champs, bords et les informations de calcul parallèle sont stockées).

2 Définition du format LataDB

Le code de référence pour l'interprétation des fichiers lata est fourni dans le code lata2dx (voir la classe `LataDB`).

Les données au format LataDB sont stockées dans plusieurs fichiers :

- un fichier "maître", au format ASCII, contenant les informations de directory (liste des domaines, des pas de temps et des champs, taille des tableaux, noms des fichiers de données),
- des fichiers de données au format ASCII ou binaire.

2.1 Spécifications des fichiers de données "data"

Un fichier de données contient un ou plusieurs *blocs* contenant des matrices n*m de valeurs entières ou flottantes. On notera n le nombre de lignes et m le nombre de colonnes.

Chaque bloc est caractérisé par :

- la position absolue en octets du début de bloc dans le fichier,
- l'encodage des valeurs (ascii, binaire big-endian ou binaire little-endian : avec l'encodage big-endian, l'octet de poids fort est stocké à l'adresse mémoire la plus basse),
- le type des valeurs (int32, int64, ieee-real32 ou ieee-real64),
- la taille du tableau de valeurs (valeurs n et m),
- l'ordre d'écriture des valeurs (fortran : valeurs écrites colonne par colonne ou C : ligne par ligne),
- la présence ou pas de marqueurs de début et de fin de "bloc fortran" et si oui, la matrice est-elle écrite en un seul "bloc fortran" ou en m "blocs fortran",
- le type binaire de ce marqueur de bloc (entier 32 ou 64 bits).

S'il est présent, le marqueur de bloc fortran correspond à la valeur attendue par l'instruction fortran `READ` si le fichier était écrit en binaire. Il s'agit du nombre d'octets de données (attention, si le bloc de données est encodé en ascii, le marqueur est écrit en ascii et la valeur écrite est le nombre d'octets qu'aurait l'intérieur du bloc, sans les marqueurs, s'il avait été écrit en binaire).

2.2 Spécifications du fichier maître

Le fichier lata maître est un fichier ASCII qui contient une base de données hiérarchisée pointant vers des blocs « data » situés dans d'autres fichiers.

Sa structure est la suivante :

2.2.1 Entête

L'entête est constituée d'exactement trois lignes, plus une ligne optionnelle de description du format d'encodage. Le premier mot indique le nom du format de bas niveau utilisé (LATA_V2.1). La suite de la première ligne, la deuxième ligne et le premier mot de la troisième ligne sont au choix du code. Par exemple, le code Trio_U écrit ceci :

```
LATA_V2.1 Trio_U Version1.4.9
nom_du_cas
Trio_U
Format LITTLE_ENDIAN,INT32,F_INDEXING,C_ORDERING,F_MARKERS_SINGLE,REAL32
```

A l'heure actuelle, si on ne trouve pas "LATA_V2." au début, on suppose que le fichier est écrit à l'ancien format.

La suite de la troisième ligne de l'entête est interprêtée par l'outil lata2dx et peut contenir des options (les mêmes options que celles en ligne de commande). Par exemple, la ligne suivante force la reconnection des maillages parallèles en un bloc unique et l'affichage d'informations de débuggage lors de la lecture du fichier dans Visit.

```
Trio_U reconnect=1e-6 verbosity=4
```

Le mot-clé `Format` définit le format d'encodage par défaut des champs. Ce format peut être modifié champ par champ (par exemple, un champ peut être écrit en ascii alors que tous les autres sont en binaire). Les mots-clés autorisés après `Format` sont les suivants :

- `BIG_ENDIAN`, `LITTLE_ENDIAN` ou `ASCII`. Pour mémoire, le format binaire utilisé sur les processeurs Intel est little-endian.
- `INT32` ou `INT64` (définit la taille par défaut des champs d'entiers)
- `REAL32` ou `REAL64` (définit la précision par défaut des champs de valeurs réelles). Attention : l'outil lata2dx supporte très peu le type `REAL64` pour l'instant.
- `F_INDEXING` ou `C_INDEXING`. Si le champ est un champ d'entiers, cet attribut indique si le contenu du champ est un index C ou fortran. Par exemple, le tableau des éléments contient des indices de sommets, le premier sommet du maillage portant le numéro zéro en C et un en Fortran.
- `F_ORDERING` ou `C_ORDERING`. Pour les tableaux ayant plusieurs composantes (coordonnées des sommets, champs vectoriels, etc...), cet attribut indique si les valeurs sont stockées dans l'ordre C (pour chaque point de discrétisation, valeurs de toutes les composantes) ou dans l'ordre Fortran (composante x de tous les points de discrétisation, puis composante y, etc...).
- `F_MARKERS_NO`, `F_MARKERS_SINGLE` ou `F_MARKERS_MULTIPLE`. Cet attribut indique s'il y a ou non des marqueurs de début et de fin de bloc de données, et si oui, si toutes les composantes sont écrites avec une seule instruction `WRITE` (un seul marqueur au début et un à la fin) ou avec autant d'instructions que de composantes (un bloc par composante).

2.2.2 Corps du fichier maître

Le corps du fichier contient une succession de blocs `TEMPS`, `GEOM`, et `CHAMP`. Les mots peuvent être arbitrairement séparés par des espaces, des tabulations ou des retours à la ligne (il est conseillé d'utiliser une ligne par bloc temps, geom ou champ, cela simplifie l'interprétation du fichier par des scripts).

Exemple :

```
GEOM  dom type_elem=HEXAEDRE
CHAMP SOMMETS post.lata.dom geometrie=dom size=3745182 composantes=3
CHAMP ELEMENTS post.lata.dom geometrie=dom size=3538944 composantes=8
 format=INT32 file_offset=44942192
CHAMP FACES post.lata.lata.dom geometrie=dom size=10819592 composantes=4
 format=INT32 file_offset=158188408
CHAMP ELEM_FACES post.lata.lata.dom geometrie=dom size=3538944 composantes=6
 format=INT32 file_offset=331301888
TEMPS 0
CHAMP VITESSE post.lata.lata.VITESSE.FACES.dom.pb.0.0000000000
 geometrie=dom size=10819592 composantes=1 localisation=FACES nature=scalar
CHAMP TEMPERATURE post.lata.lata.TEMPERATURE.ELEM.dom.pb.0.0000000000
```

```
 geometrie=dom size=3538944 composantes=1 localisation=ELEM nature=scalar
TEMPS 5.5923e-06
CHAMP VITESSE post.lata.lata.VITESSE.FACES.dom.pb.0.0000055923
 geometrie=dom size=10819592 composantes=1 localisation=FACES nature=scalar
CHAMP TEMPERATURE post.lata.lata.TEMPERATURE.ELEM.dom.pb.0.0000055923
 geometrie=dom size=3538944 composantes=1 localisation=ELEM nature=scalar
FIN
```

Dans cet exemple, on définit une géométrie nommée `dom` valable à tous les pas de temps (car le bloc `GEOM` apparaît avant le premier bloc `TEMPS`) et discrétisée en hexaedres. On définit juste après les coordonnées des sommets de ce maillage, les éléments (pour chaque maille hexaedrique, indices de ses huit sommets) qui sont stockés dans un champ de type `INT32`, les faces (pour chaque face, indices de ses sommets) et la connectivité éléments-faces (pour chaque élément, indices de ses six faces).

Ensuite, on trouve deux pas de temps, avec pour chacun un champ de vitesse scalaire aux faces (composante normale de la vitesse à la face) et un champ de température discrétisé aux éléments.

Le mot-clé `FIN` est optionnel.

2.2.3 Entrée TEMPS

Ce bloc déclare un nouveau pas de temps. Les géométries et champs définies ensuite sont spécifiques à ce pas de temps. Le temps physique correspondant est indiqué juste après :

```
TEMPS 5.5923e-06
```

2.2.4 Entrée GEOM

Ce bloc déclare un nouveau nom de géométrie. Attention : pour l'instant, le plugin de lecture Visit ne supporte que deux cas : des géométries statiques (indépendantes du temps), déclarées avant le premier pas de temps, ou des géométries dynamiques qui changent à chaque pas de temps. Dans ce cas, la géométrie doit être déclarée et décrite (sommets, éléments etc...) à l'intérieur de chaque pas de temps.

Le modèle est le suivant

```
GEOM nom type_elem=type
```

Le type est une chaîne de caractères et ne fait pas partie de la spécification du format de bas niveau LataV2 (elle n'est pas interprêtée ni utilisée par la classe LataDB).

2.2.5 Entrée CHAMP

Cette entrée est utilisée pour stocker tous les types de tableaux (coordonnées des sommets du maillage, description des éléments, des faces, connectivité de joint des maillages parallèles, champs de valeurs physiques).

Le modèle est le suivant :

```
CHAMP nom_du_champ nom_fichier
 geometrie=nom_geometrie
 size=n
 [ composantes=m ]
 [ localisation=SOM|ELEM|FACES ]
```

```
[ nature=scalar|vector ]
[ format=... ]
[ file_offset=... ]
[ reference=... ]
[ noms_compo=nom1,nom2,... ]
```

- `nom_du_champ` est le nom du champ. Certains noms sont interprétés de façon particulière par le plugin Visit (ceci ne fait pas partie de la spécification du format de bas niveau), par exemple sommets, elements, faces, elem_faces, sommets_ijk_i, etc.
- `nom_fichier` est le nom du fichier qui contient les données.
- `file_offset=N` (optionnel, par défaut : zéro) indique l'offset en octets du début du bloc de données à partir du début du fichier. Attention à utiliser des entiers longs pour écrire cette valeur dans le fichier maître (support de fichiers de plus de 2Go).
- `size=N` nombre de lignes du tableau de valeurs.
- `composantes=M` nombre de colonnes du tableau (par défaut, une).
- `localisation=...` localisation du champ, lorsque cela est pertinent.
- `nature=...` scalaire par défaut, indique si le champ doit être proposé sous forme de champ vectoriel dans la visualisation (le nombre de composantes doit alors être égal au nombre de composantes des coordonnées des sommets).
- `format=...` modificateur du format d'écriture par défaut (même syntaxe que dans l'entête).
- `reference=...` lorsque le champ contient des indices d'items dans un autre champ (tableau des éléments, des faces ou autre connectivité), ce mot-clé permet d'indiquer le nom du champ en référence (non exploité pour l'instant).
- `noms_compo=...` permet d'indiquer les noms de chaque composante du champ.

3 Stockage des maillages et des champs dans le format LataV2

Ce chapitre indique comment Trio_U, Visit et lata2dx utilisent le format de bas niveau LataV2 pour stocker les maillages et les champs.

3.1 Maillages structurés

Les maillages structurés sont décrits à l'aide de trois champs qui fournissent les coordonnées X, Y, et Z des sommets du maillage. La taille (size) de ces champs définit donc implicitement la taille du maillage. Un champ d'entiers optionnel nommé `INVALID_CONNECTIONS` et contenant un ou zero pour chaque maille permet de définir les mailles invalides. L'attribut NO_INDEXING indique que l'entier stocké dans le champ n'est pas un index dans un autre tableau (l'attribut par défaut peut être F_INDEXING et qui conduirait à retirer une unité à chaque valeur lors de la lecture par le plugin écrit en C++).

Exemple pour un maillage ni=192, nj=192, nk=96 éléments :

```
LATA_V2.1
lata2dx
Trio_U
Format LITTLE_ENDIAN,INT32,F_INDEXING,C_ORDERING,F_MARKERS_SINGLE,REAL32
GEOM  dom_IJK type_elem=HEXAEDRE
```

```
CHAMP SOMMETS_IJK_I titi.lata.SOMMETS_IJK_I.dom_IJK geometrie=dom_IJK size=193 composantes=1
CHAMP SOMMETS_IJK_J titi.lata.SOMMETS_IJK_J.dom_IJK geometrie=dom_IJK size=193 composantes=1
CHAMP SOMMETS_IJK_K titi.lata.SOMMETS_IJK_K.dom_IJK geometrie=dom_IJK size=97 composantes=1
CHAMP INVALID_CONNECTIONS titi.lata.INVALID_CONNECTIONS.dom_IJK geometrie=dom_IJK
  size=3538944 composantes=1 localisation=ELEM format=INT32,NO_INDEXING
TEMPS 5.5923e-06
CHAMP VITESSE titi.lata.dom_IJK_VITESSE_FACES.1 geometrie=dom_IJK size=3613153 composantes=3
  localisation=FACES nature=vector
CHAMP TEMPERATURE titi.lata.dom_IJK_TEMPERATURE_ELEM.1 geometrie=dom_IJK size=3538944
  composantes=1 localisation=ELEM nature=scalar
FIN
```

Les types d'éléments supportés sont `HEXAEDRE` et `QUADRANGLE`.

La numérotation des sommets, éléments et faces dans les champs est telle que les indices i varient le plus rapidement et les indices j et k lentement.

Les tableaux `SOMMETS_IJK_` doivent avoir respectivement ni+1, nj+1 et nk+1 valeurs (sans compter les marqueurs de blocs fortran).

Les tableaux de valeurs aux éléments doivent avoir ni*nj*nk valeurs.

Les tableaux de valeurs aux faces doivent avoir autant de composantes que de directions d'espace (on ne peut stocker qu'une composante par face, et le champ est forcément interprété ensuite comme un champ vectoriel) et doivent avoir autant de lignes qu'il y a de sommets dans le maillage. L'indice de ligne où on trouvera une face dans un champ est l'indice du plus petit sommet de la face, l'indice de colonne est la direction de la normale à la face (x, y, ou z). Certains indices de faces ne sont donc pas utilisés.

Attention, dans cet exemple, le format d'écriture est C_ORDERING, par conséquent les valeurs associées aux faces de directions x et y et z sont entrelacées dans le fichier de données. Si les valeurs sont triées par direction de faces il faut spécifier le format F_ORDERING.

3.2 Maillages non structurés

Les types d'éléments supportés sont
- HEXAEDRE
- QUADRANGLE
- SEGMENT
- TETRAEDRE
- TRIANGLE
- POLYEDRE

Pour les polyedres, le nombre maximal de sommets par polyèdre est le nombre de composantes du champ `ELEMENTS`. Pour chaque polyèdre, s'il y a moins de sommets que le nombre maximal, il faut mettre la valeur -1 comme indice de sommet pour les sommets non utilisés du tableau `ELEMENTS`.

L'ordre des sommets dans les éléments est l'ordre conventionnel du code Trio_U (à détailler).

Il est possible aussi de ne pas spécifier de type d'élément. Dans ce cas on crée uniquement un nuage de points et la seule localisation possible des champs est `SOM`.

Les champs spécifiquement reconnus pour décrire le maillage sont les suivants. Seuls les tabeaux sommets et elements sont obligatoires (elements sera omis uniquement pour les nuages de points avec type_elem non spécifié). Les tableaux faces et elem_faces sont requis s'il existe des champs aux faces. Les tableaux joints sont utilisés pour réutiliser le découpage du maillage lors des visualisations en parallèle. Les

tableaux sont tous des tableaux d'entiers à l'exception du tableau des sommets.

- SOMMETS : coordonnées des sommets, size=nombre de sommets, composantes=dimension d'espace
- ELEMENTS : indices des sommets des éléments, size=nombre d'éléments, composantes=nombre de sommets par élément
- FACES : indices des sommets des faces, size=nombre de faces, composantes=nombre de sommets par face
- ELEM_FACES : connectivité éléments faces, size=nombre d'éléments, composantes=nombre de faces par élément
- JOINTS_SOMMETS : size=nombre de processeurs du calcul, composantes=2. La colonne 1 contient l'indice du premier sommet du i-ième processeur
- JOINTS_ELEMENTS : idem pour les indices d'éléments
- JOINTS_FACES : idem pour les indices de faces

4 A finir

maillages découpés et parallélisme ghost cells