News in the alignment framework

Raffaele Grosso

October 10, 2007

Updates in the alignment framework

The recent updates in the framework concern:

- moving look-up tables from `AliGeomManager` to `TGeo`
- dealing with full/partial geometry
- dealing with alignment objects for structures
- introduction of correlation matrix in the alignment objects

Study of impact of saving global/local delta transformation in the alignment object classes.

Look-up tables moved into **TGeo**

Since root v5-17-02, `TGeo` has introduced some data members and methods to host the look-up tables allowing fast access to alignable volumes (**TGeoPNEntries**) and correspondent physical nodes and original matrixes, by means of the global unique index (UID) of the volume.

AliRoot will take advantage of this when using a tag of root more recent than the present one (v5-16-00).
At that point few small changes will be needed in AliRoot detector's code (in **AddAlignableVolumes** methods), then it will be possible to remove the look-up tables from `AliGeomManager`.

Look-up tables moved into TGeo

New or modified methods:

- **TGeoManager::SetAlignableEntry**
  was taking the symname and volume path as argument, now can take also the UID.
- **TGeoManager::GetAlignableEntryByUID(Int_t uid)**
  is a new method which allows to get the alignable volume by means of the UID
- **TGeoManager::GetNAlignable(Bool_t with_uid=kFALSE)**
  if the boolean argument is set to kTRUE it returns the number of alignable volumes declared with index
- the **TGeoPNEntry** class, which already had a pointer to the corresponding physical node, has now an additional pointer to the original local matrix, which is set in the class constructor. For this to work correctly all alignable volumes need to be created before applying any alignment. The original local matrix can than be get by means of:
  TGeoMatrix *TGeoPNEntry::GetMatrixOrig()

Look-up tables moved into `TGeo`

This will allow:

- to remove the look-up tables from `AliGeomManager`*:
  - index-symbolic name
  - index-`TGeoPNEntry`
  - index-original matrix

  `AliGeomManager` will continue to provide the indexing of the detectors (layers) and the mapping of alignment objects initialized from the geometry.
- to remove the duplication of code defining the alignable volumes and assigning them an index (in detector code and in `AliGeomManager`)
- to have the look-up tables as an integral part of the geometry.

* The current scheme with the duplication has been important to assist the detectors in changes in the indeces. The scheme we are finalizing stresses the responsability of each detector (not to change the indexes a part from improvements proven to be necessary).

Full/partial geometry

We are currently using two different set-ups for the geometry

- the full geometry: all full detectors are in, as in `Config.C`
- the partial geometry: TOF and TRD construct only some of their supermodules (11/18 for TOF, 8/18 for TRD) to reproduce the initial set-up, as in `Config_PDC06.C`.

Full/partial geometry

The choice between the two geometries has been introduced:

- in the macros producing misalignment and the ideal geometry:
  - the steering macros **macros/MakeAllDETs<type>MisAlignment.C**
  - the macros for the concerned detectors: **DET/MakeDET<type>MisAlignment.C**, DET in {TOF,TRD}
  - **GRP/UpdateCDBIdealGeom.C**
- in the OCDB storages: for the current production there are two different storages (`/alice/simulation/2007/PDC07_1/`), one for full and one for partial geometry.

Alignment objects for structures

The misalignment of structures (non-sensitive modules in the ALICE geometry) can be in several cases something which is needed to be included in the real geometry, for several reasons:

- the space-frame supports sensitive detectors
- the beam-pipe, fixed to ITS, will determine the position of the interaction point
- structures can need to be aligned to avoid overlaps or to give the correct material budget

Introduction of correlation matrix in the alignment objects

Preliminary introduction of two data members in the `AliAlignObj` class storing the diagonal and off-diagonal elements of the correlation matrix:
Double32_t fDiag[6]; // diagonal elements
Double32_t fODia[15]; // [-1, 1,8] off-diagonal elements (in 8 bit precision)
based on the **AliAODRedCov** class.
Was requested by MUON.
Still the use-cases are not clear e.g.:

- how to compose the correlation matrices for volumes in the same geometry branch
- how to fill the correlation matrix

Alignment objects for structures

The general schema for alignment objects for structures has been defined:

- the macros to produce them are in **$ALICE_ROOT/GRP/**
- the alignment objects for structures are stored in the GRP directory of the CDB storage
- GRP is added to the list of detectores for which simulation and reconstruction ask the `AliGeomManager` to load the alignment objects from the default/specific CDB storage

Presently the schema is applied for the alignment objects reproducing the expected deformation of the space-frame.

Storing global vs local delta transformation

This delta transformation, saved in the alignment objects, can be expressed in two ways:

1. as the global delta transformation
2. as the local delta transformation

The two expressions for the delta transformation are related to each other by geometrical relations and depend on the global transformation for the given volume. For a given ideal geometry + set of alignment objects the two expressions of the delta transformation give equivalent information and can be converted into one another.

$\Rightarrow$ The choice is a matter of opportunity, not a matter of correctness.

The consideration of pros and cons has still to lead us to a final decision.

Misalignment on one single level

Consider first a misalignment on one single level (for volume C):

$$\mathcal{G}_C = ABC$$

$$\begin{aligned} \mathcal{G}_C^a &= ABC\delta_C \quad (1) \\ &= \Delta_C ABC \quad (2) \end{aligned}$$

$\mathcal{G}_C$ ideal global transformation for volume C
$\mathcal{G}_C^a$ aligned global transformation for volume C
δ_C local delta transformation for volume C
Δ_C global delta transformation for volume C

Misalignment on more levels

The most frequent case is probably misalignment on more levels of the geometry tree, i.e. a volume is misaligned but also some of its parent volumes are. In the following example volume C and mother volume B.

$$\mathcal{G}^a_C = AB\delta_B C\delta_C \quad (3)$$
$$= \Delta_C AB\delta_B C \quad (4)$$
$$= \Delta_C \Delta_B ABC \quad (5)$$

$\mathcal{G}^a_C$ aligned global transformation for vol C
δ_B (δ_C) local delta transformation for B (C)
Δ_B (Δ_C) global delta transformation for B (C)

Pros and cons for storing the global/local delta

1. With both choices the interface allows to set the alignment object using both deltas. Nevertheless it is critical to set(read) the global delta when starting from (quering) the local delta:
   - introduction of unintuitive correlations in the parameters
   - objects not readable without open geometry and ”translation” by TGeo
   - wrong precisions

   This argument is reversible, so it is in favour of the most common occurence (local in my opinion).

Pros and cons for storing the global/local delta

2. ”The global delta is not affected by changes in the geometry, while the local is”.
   I think the global delta is less affected by changes in the geometry:
   - alignment objects must be bound to a given geometry in any case;
   - we should ”freeze” the code related to geometry at a given point (”AliDET::CreateGeometry” methods), hopefully before real data are taken.
3. We loose the previously produced alignment objects (they can be rewritten if we have the macros which produced them).
4. The global delta introduces dependencies between alignment objects pertaining to all volumes in the same branch of the geometry tree (cfr eq.(3) - eq.(5)). The important implication is:

   if storing globalΔ, when aligning a parent volume, the objects for all daughter volumes need to be rewritten (while the contrary does not hold).

Open issues

1. The concerned classes in the alignment framework have already been rewritten (leaving the interface unchanged) and are under test.
2. Would these changes be transparent for the realignment framework?