ITS Alignment with MillePede2

Difference between the original MP1 and MP2

Comparison of ITS alignment with MP1 and MP2

Constraints on shifts

SDD alignment + t_0, V_D calibration

R.Shahoyan, 18/03/09

Differences between original MillePede1 and MillePede2

MP1	MP2 (on top of all features of MP1)
➢ Solution is obtained by matrix inversion:	➢ Solution may be obtained by iterations: (note: manual mentions GMRes method but actually only MinRes is implemented)
+ Errors on parameters are provided - CPU expensive, may fail for large matrices or matrices with large condition number, round-off error.	+ More CPU efficient, precise and stable for large matrices - Errors on parameters are not provided, May require matrix preconditioning
➢ Uses dense matrix format: + Fast access to matrix elements - Number of global parameters is limited by RAM	➢ Sparse matrix format may be used: - Slower access to matrix elements + Huge economy in RAM
➢ Information about derivatives and constraints is stored in RAM (~70 bytes/residual) - Number of processed points is limited by RAM	➢ Information about derivatives is stored in the intermediate file + No limit on number of processed points
➢ Outliers rejection is done only by track χ^2 or by rejection of large residuals	➢ Outliers rejection can be done by assigning non Gaussian weights to residuals (Cauchi..)
➢ Track processing and solution is done in single step	➢ Requires separate steps for track processing (C code) and solution (Fortran code)

Implementation of MillePede2 in AliRoot (AliMillePede2)

- For compatibility reasons the interface of AliMillepede was preserved as much as possible.
- Almost all modifications are (AliMillepede-)user transparent.

Basic differences wrt AliMillepede:

- AliMillepede uses hardwired dense matrices: (needs ~3 GB of RAM for 13188 parameters of ITS)
  AliMillePede2 may use 2 kind of dynamically expandable matrices
  (derived from TMatrixD $\rightarrow$ AliMatrixSq):
  AliSymMatrix: dense but symmetric matrix format.
  Built-in fast solution by Cholesky decompostion (only for pos.-definite matrices)
  AliMatrixSparse: fast sparse matrix format (optionally symmetric).
- AliMillePede2 allows Iterative solution of matrix equation (for both kind of matrices) using separate solver class AliMinResSolve.
  It includes 2 separate solvers: MinRes and FGMRes and choice of a few preconditioners
  (details on following slides)
- AliMillePede2 stores processed track data (derivatives over the local and global parameters) in the special AliMillePedeRecord in the disk based buffer (TTree in the TFile).
  In opposite to standard MP2 the solution is obtained in the same session where the data processing is done.
  Nevertheless, this buffer can be saved and reused for independent alignment step with the same setup (for instance, with different constraints).
- Extra features: AliMillePede2 may be requested to automatically fix parameters with insufficient statistics (involved in less than given number measurements)

Using iterative solvers (AliMinResSolve).

Large matrices often have very bad condition number (ratio between largest and smallest eigenvalues)

⇓

The solution of corresponding matrix equation is almost impossible without preconditioners:

instead of **Ax=b** one solves **MAx=Mb** or **MAM^{-1}Mx=Mb** where matrix **M** is such that **MA** has better condition number than **A**.

Standard MP2 implements only incomplete band-Cholesky preconditioner: inversion of **A** using Cholesky decomposition (**A=LL^{T}**) with keeping on elements in the diagonal band.

Caveat: Cholesky decomposition works only for positive-definite matrices.

This is not the case for systems with Lagrangian constraints!

AliMinResSolve implements few preconditioners

[via `static AliMillePede2::SetMinResPreconType(int id)`]

id=0: No preconditioning. OK for small systems (a few 100 parameters) w/o constraints.

Id=1: Jacobi preconditioner: **M = D^{-1}** where **D** is the diagonal of **M** (0's are substituted by ε). Very fast, but works only for systems with a few ~1000 parameters w/o L-constraints.

Id>=100 ILU(k) preconditioners family with k=Id-100 (Y.Saad, Iterative Methods for sparse linear systems). Uses **M=(L+D)(I+U)** incomplete decomposition, where **L** and **U** are strictly triangular matrices and **D** is strictly diagonal.
k stands for the level of "fill-in"'s – hierarchical definition of the 0 elements of **A** which will become non-0 in **M**. The higher is **k**, the more powerful the preconditioner is,
BUT more RAM and CPU demanding its calculation ⇒ one should start with as small as possible **k** and increase it only when the solution does not converge…

Using iterative solvers (AliMinResSolve).

- Standard MP2 implements only **MinRes** iterative solution algorithm (but wrongly calls it **GMRes**)
  (C. C. Paige and M. A. Saunders, SIAM J. Numer. Anal. 12(4), pp. 617-629)

  Implemented in AliMinResSolve too.

- **MinRes** is designed for symmetric matrices, including indefinite ones (standard situation for out **A**).
  But it requires the preconditioner matrix **M** to be positive-definite, which is very diffcult to guarantee when **A** itself is indefinite (e.g. due to the Lagrangian constriants).
  In practice this means that one should test higher and higher values of **k** in **ILU(k)** preconditioner (i.e. more and more dense L,U matrices) until the solutions becomes possible

- For this reason **FGMRes** algorithm was added:
  (adapted from ITSOL_1 package by Y.Saad: http://www-users.cs.umn.edu/~saad/software)

- It may work with general matrices (including non-symmetric) and requires slightly more memory than **MinRes**. But the advantage is that positive-definiteness of the preconditioner is not required.

- In my experience, **ILU(1)** or **ILU(2)** was enough for convergence of system with all 13188 parameters of ITS with Lagrangian constraints added.

Interface to ITS Alignment (AliITSAlignMille2)

- Mostly inherits the interface of AliITSAlignMille but uses AliMillePede2
- Principal differences wrt MP1:
  - allows for the hierarchy of the volumes to align: one can put in the configuration file SPD Sectors, Half-Staves, Ladders, SSD Ladders, SSD Modules etc. all at the same time.
  - allows for different types of constraints: gaussian, fixing the mean or median shift of the sub-volumes of any volume
  - can work both with local and global parameters
  - aligned modules may have arbitrary number of parameters: the minimization is not restricted to geometrical alignment only.
  - currently the SDD t0 and Vdrift calibration in parallel to its alignment is implemented.
- MP2 run with the configuration file from MP1 (with slight modifications) produces results similar to MP1

Comparison of AliITSAlignMille and AliITSAlignMille2 (with SPD)

MP1: SPD alignment is done in separate steps at each hierarchy level:
HB, sectors, staves, hstaves, ladders.
Global deltas are defined on the *ladders* level, everything else is in ideal position.
(`AliITSAlignMille_SPD_021208.root` alignment file provided by M.Lunardon)

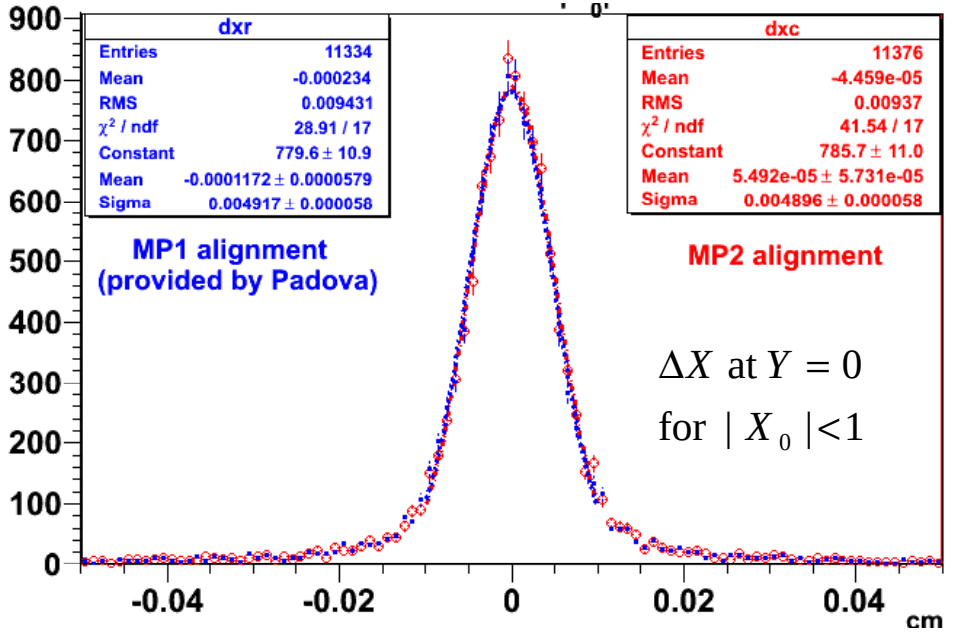
MP2: SPD alignment is done for the hierarchy *HB->sectors->hstaves->ladders* in single step.
Constrains imposed: the global movement of the sub-units of any unit is 0, i.e. within each *sector*:

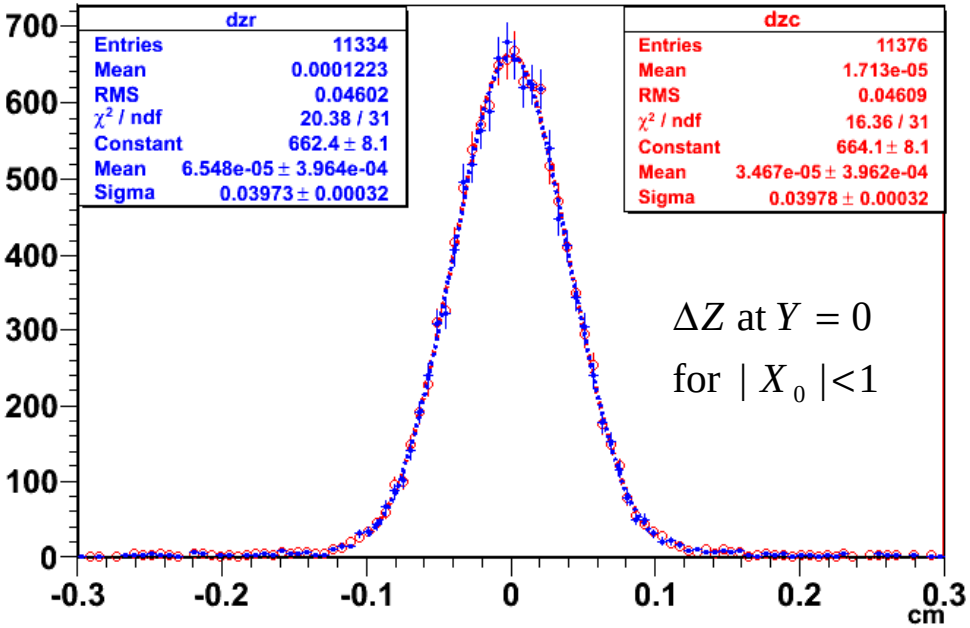
$$\sum_{i=1}^{n_{\text{hstave}}} \delta_i^{\text{hstave}} = 0$$

Global deltas are defined on the level of all alignable objects (*sectors, hstaves, ladders*)

⇓

Residuals of MP2 are fully compatible with those of MP1



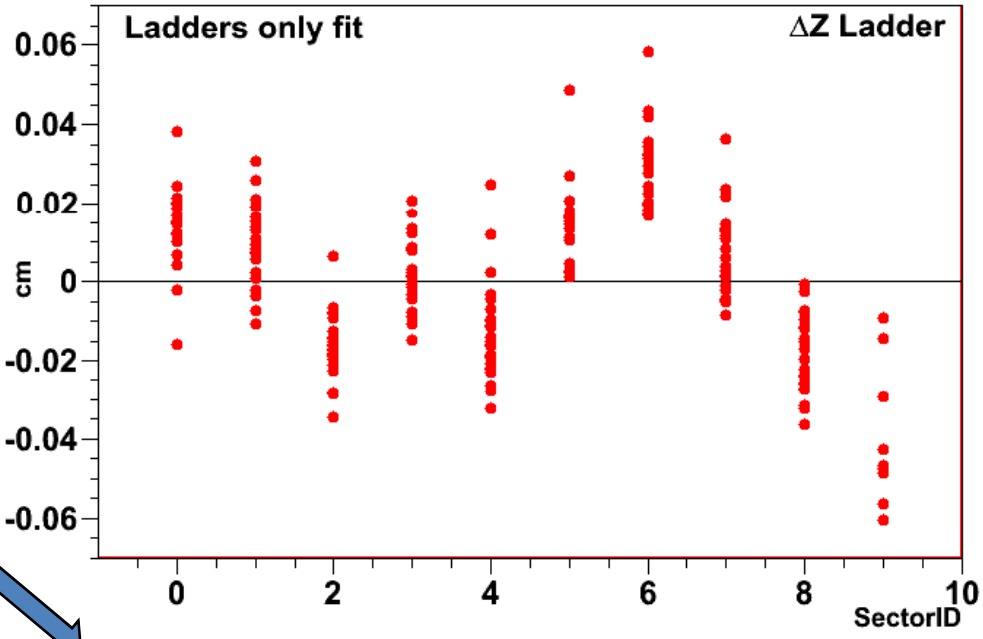


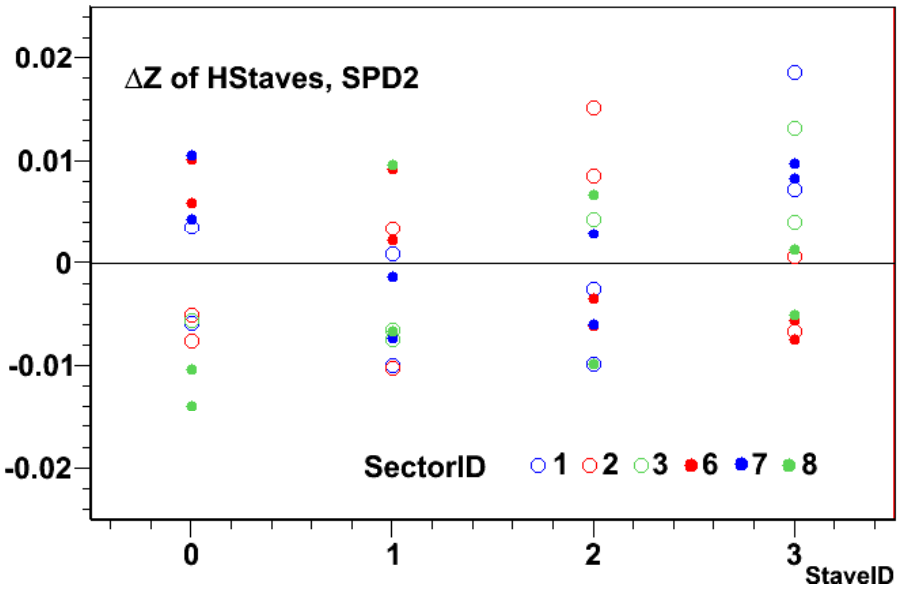
Comparison of AliITSAlignMille and AliITSAlignMille2 (with SPD)

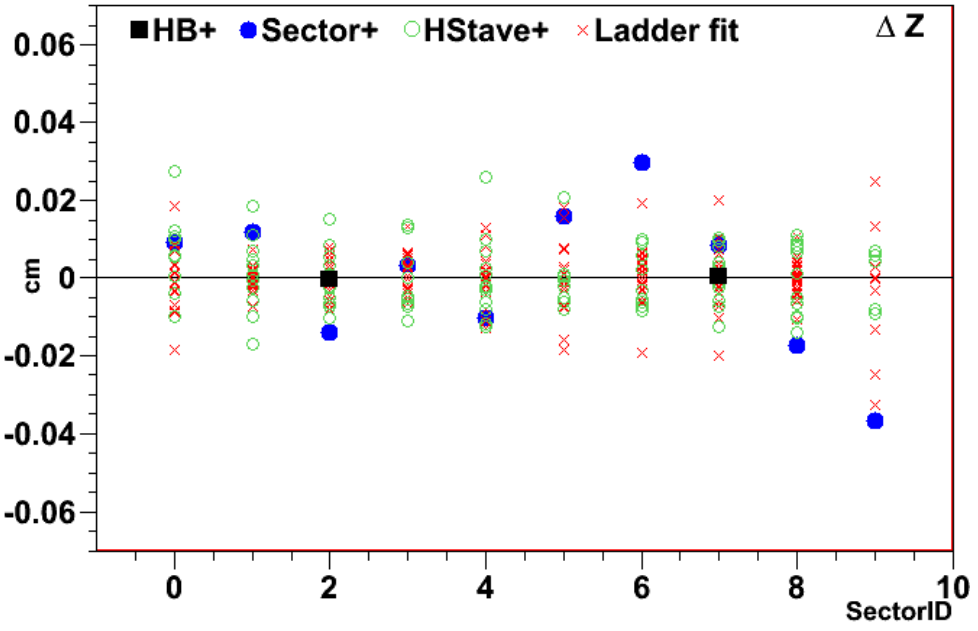
MP1 – like fit (obtained with MP2 configuration file containing the *ladder* degrees of freedom). Produces the corrections and the residuals similar to official MP1 results.
Clear pattern of Z shifts specific for each *sector*.

MP2 with *HBarrel+sectors+hstaves+ladders* constrained fit.
The *sector* shifts are correctly attributed.
The *sector* dependence is removed on finer level of hierarchy (*hstaves, ladders*)

Correlations among the *hstaves* within the *stave* are apparent.





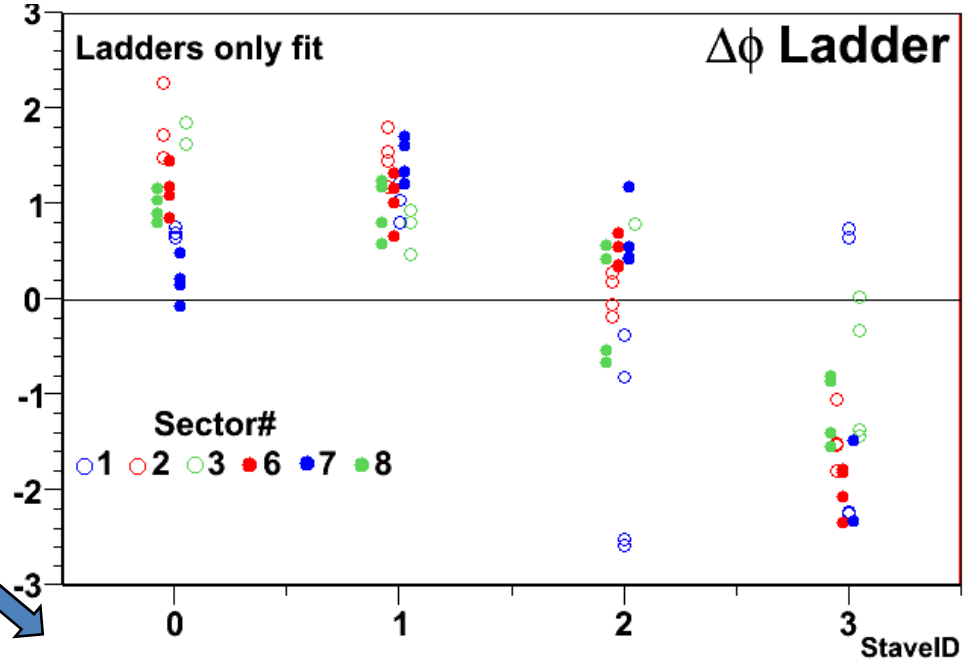


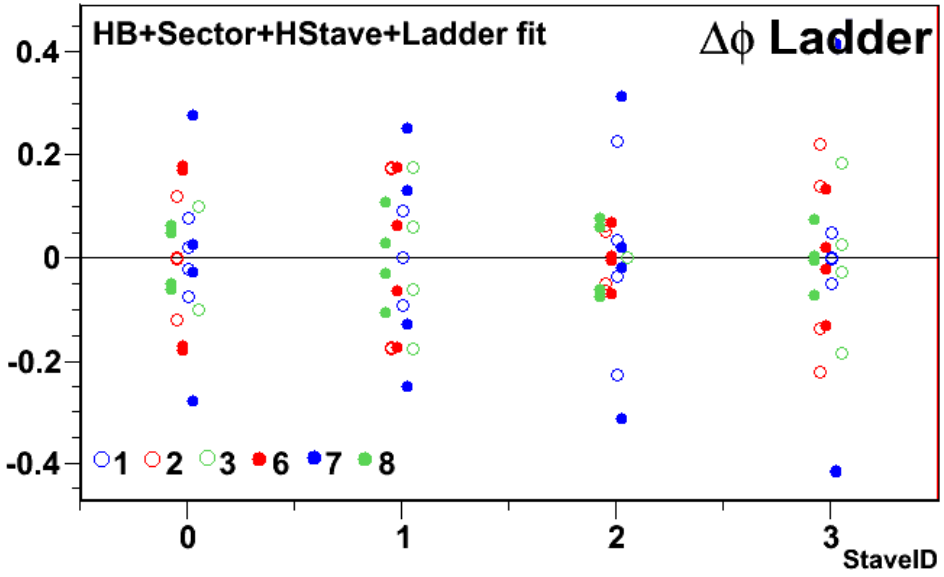
Comparison of AliITSAlignMille and AliITSAlignMille2 (with SPD)

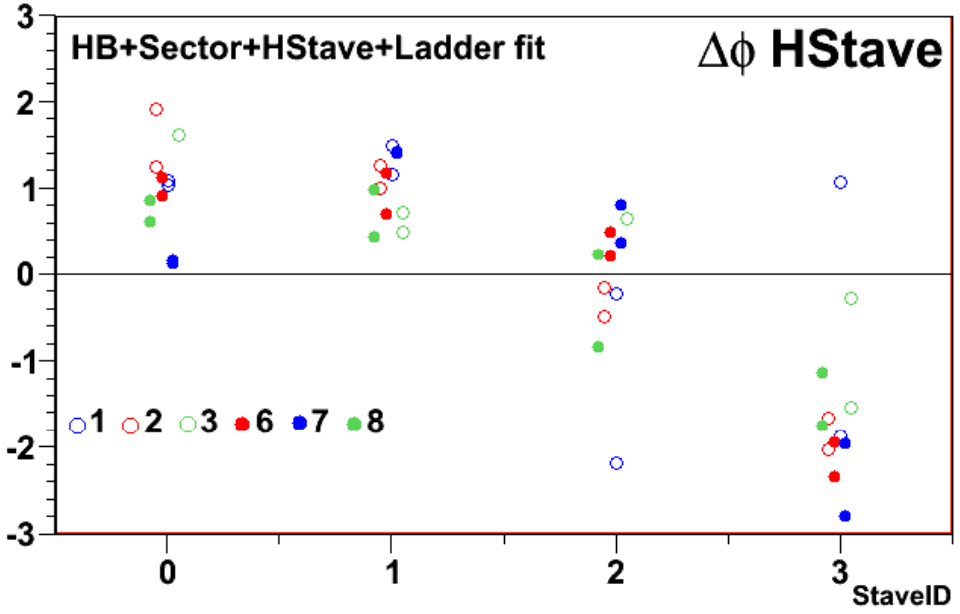
MP1-like fit: apparent systematic difference between the ideal and real curvature of the *sector* surface.

MP2 resolves in a single step the *hstave* (since the *staves* are not alignable) systematic inclinations

and randomizes the ladders corrections.

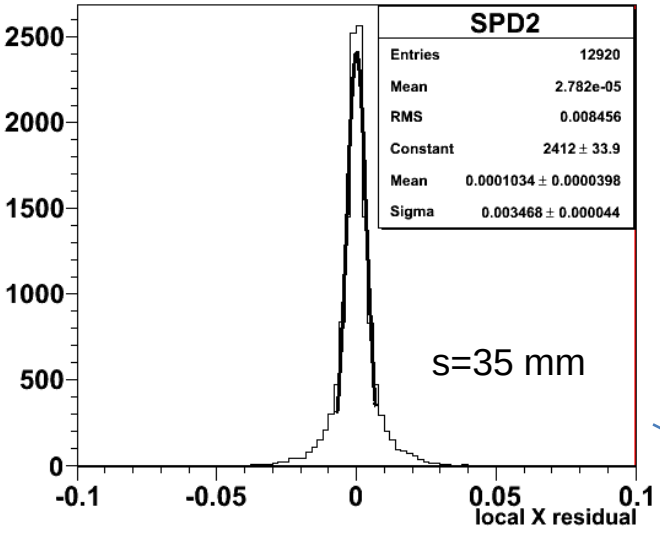




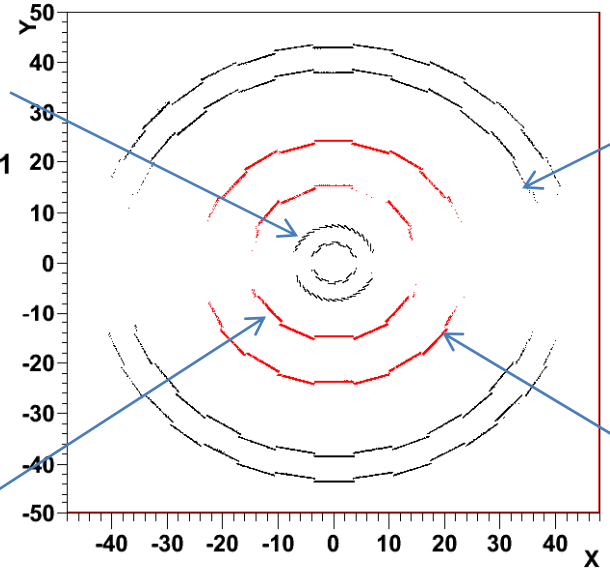


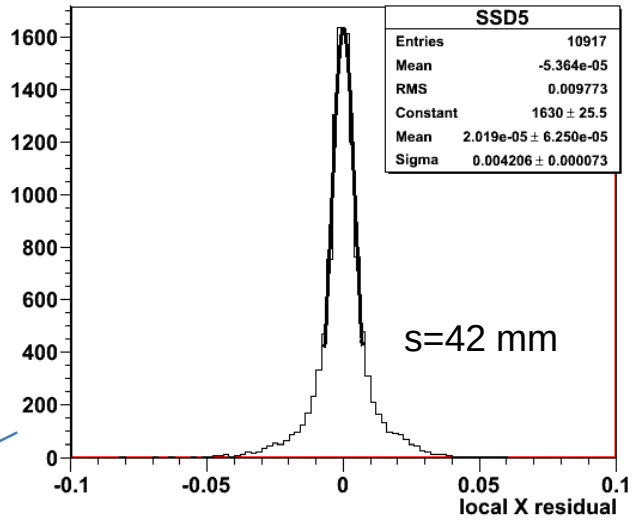
SDD alignment + calibration of t_0 and V_{drift}

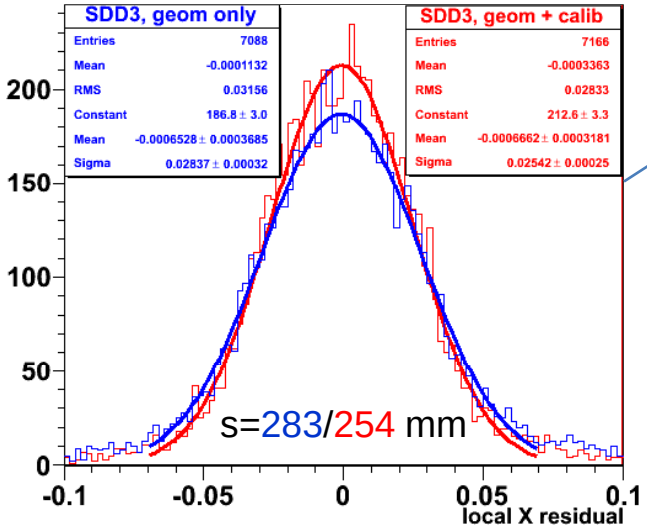
SPD and SSD are pre-aligned and fixed. Results are fully compatible with those from Padova (MP1)



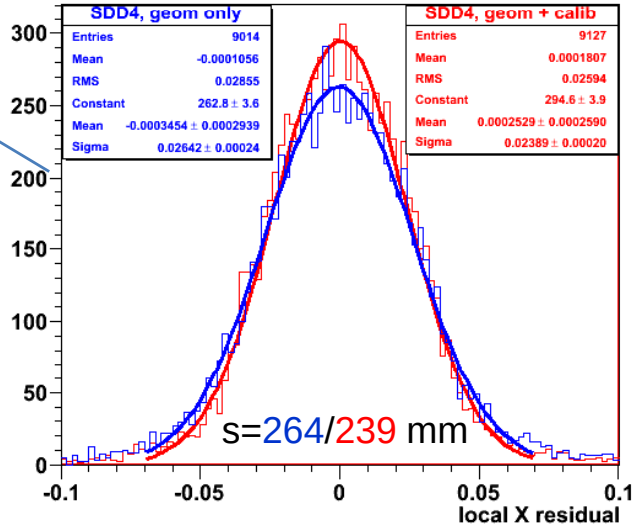
SDD varied with:
1) geometry free
2) geometry + t_0, V_D free.







~115 mm improvement in the SDD resolution after the calibration. Still very far from ~40 mm (trigger jitter)



SDD alignment + calibration of t_0 and V_{drift}

Assumed coordinate in the drift direction: $x_l = \pm\left(L - (t - t_0)V_D\right)$

Assumed error for each sensor time0: δt_0 and drift speed: δV_D

$\Downarrow$

Error on the measured local coordinate (in linearized form)

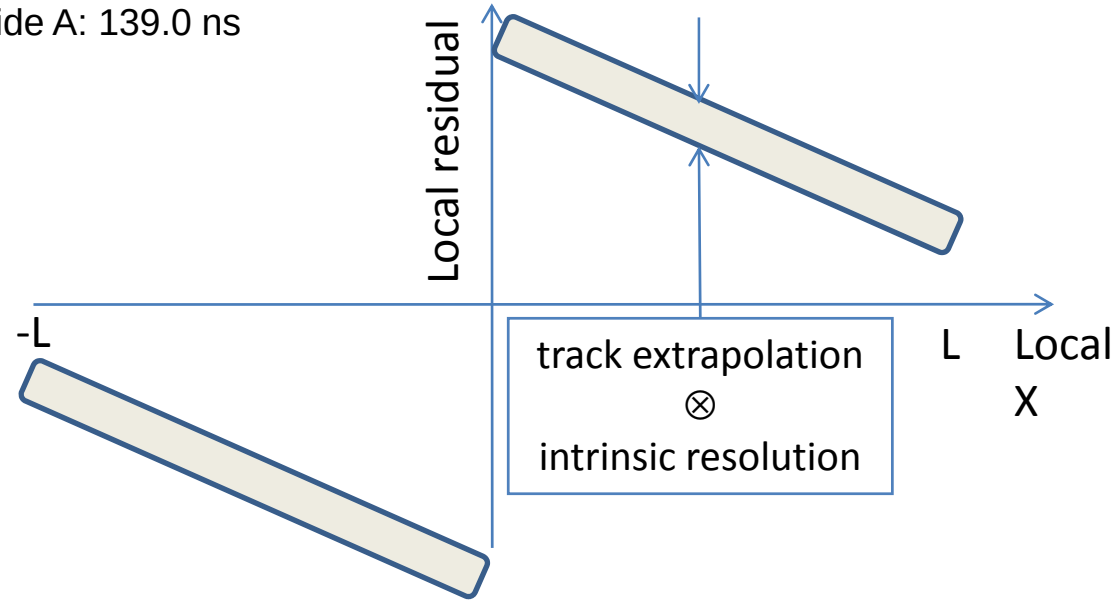
$$\delta x_l = \pm\left(\delta t_0 V_D - \delta V_D (t - t_0)\right)$$

Currently the obtained corrections to t_0 and V_D are stored in the temporary calibration object (per module).

The fit starts from t_0,V_D values used to produce the AliTrackPoints:

SDD3	sideC: 169.5 ns	side A: 140.1 ns
SDD4	sideC: 158.3 ns	side A: 139.0 ns

The SPD and SSD are pre-aligned beforehand and fixed in this test.



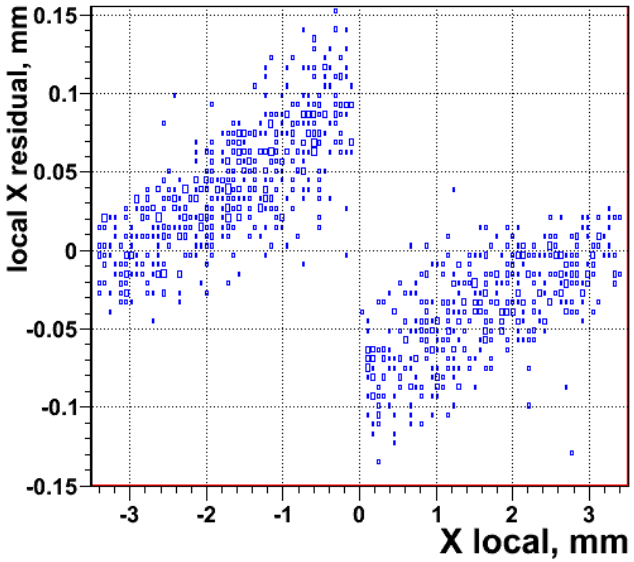
Expected pattern of local residuals vs local coordinate along the drift direction.

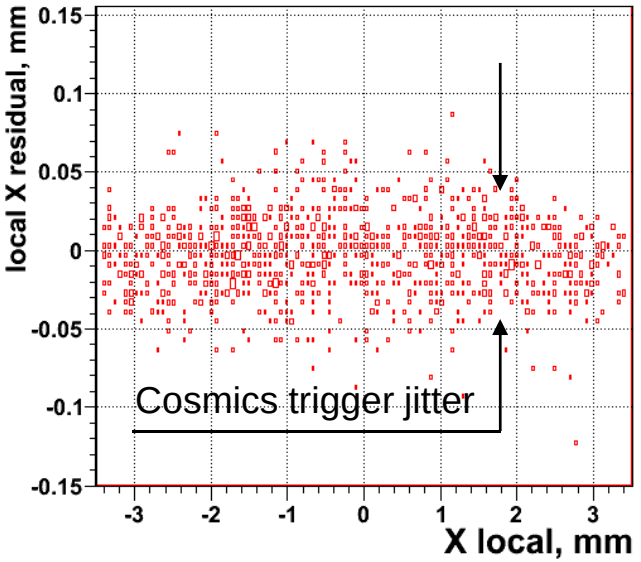
SDD alignment + calibration of t_0 and V_{drift}

Geometry only

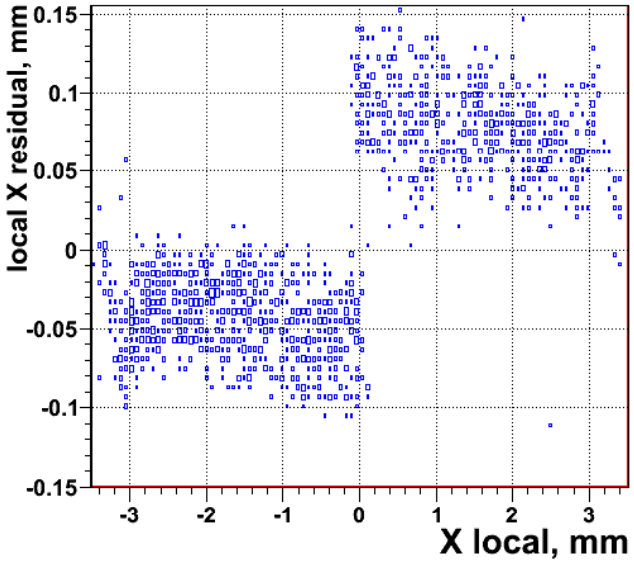
Geometry + Calibration

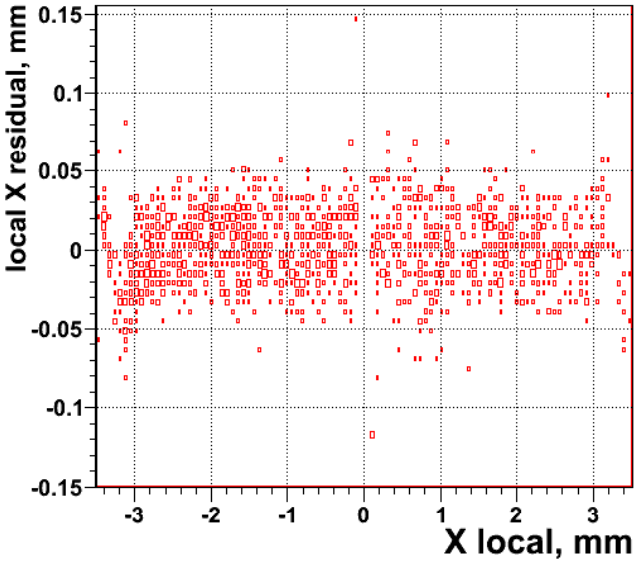
SDD 266





SDD 298



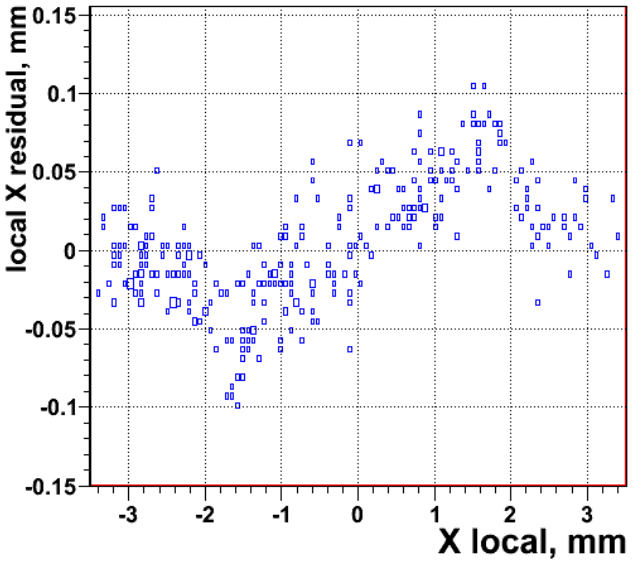


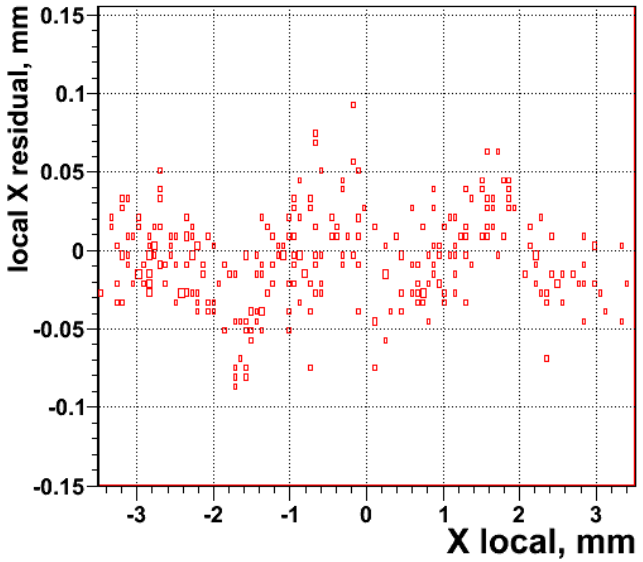
SDD alignment + calibration of t_0 and V_{drift}

Geometry only

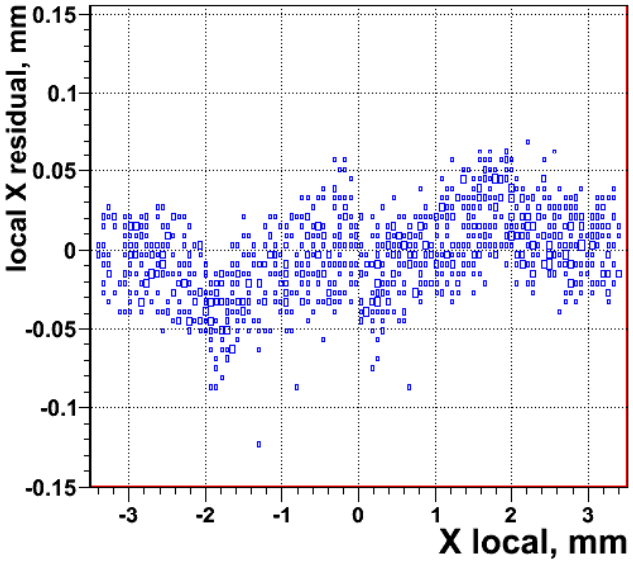
Geometry + Calibration

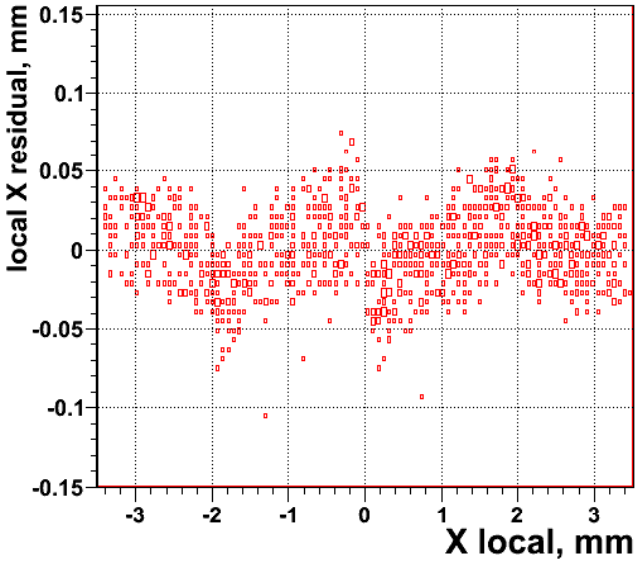
SDD 373





SDD 302





The modules with anomalous patterns were checked by Francesco, they all happen to have voltage problems.

Constraining local shifts (preliminary)

In certain situations it is preferable to restrict the local shifts of some modules:

- To avoid overlaps
- To stay close to the survey results (within the survey declared precision)

Under the local transformation $\Delta x, \Delta y, \Delta z, \Delta\psi, \Delta\theta, \Delta\varphi$ the corners of the module $L_x \times L_z$ are shifted by

$$\delta x = \Delta x \pm L_z \, \Delta\theta / 2 \qquad \delta y = \Delta y \pm L_x \, \Delta\varphi / 2 \pm L_z \, \Delta\psi / 2 \qquad \delta z = \Delta z \pm L_z \, \Delta\theta / 2$$

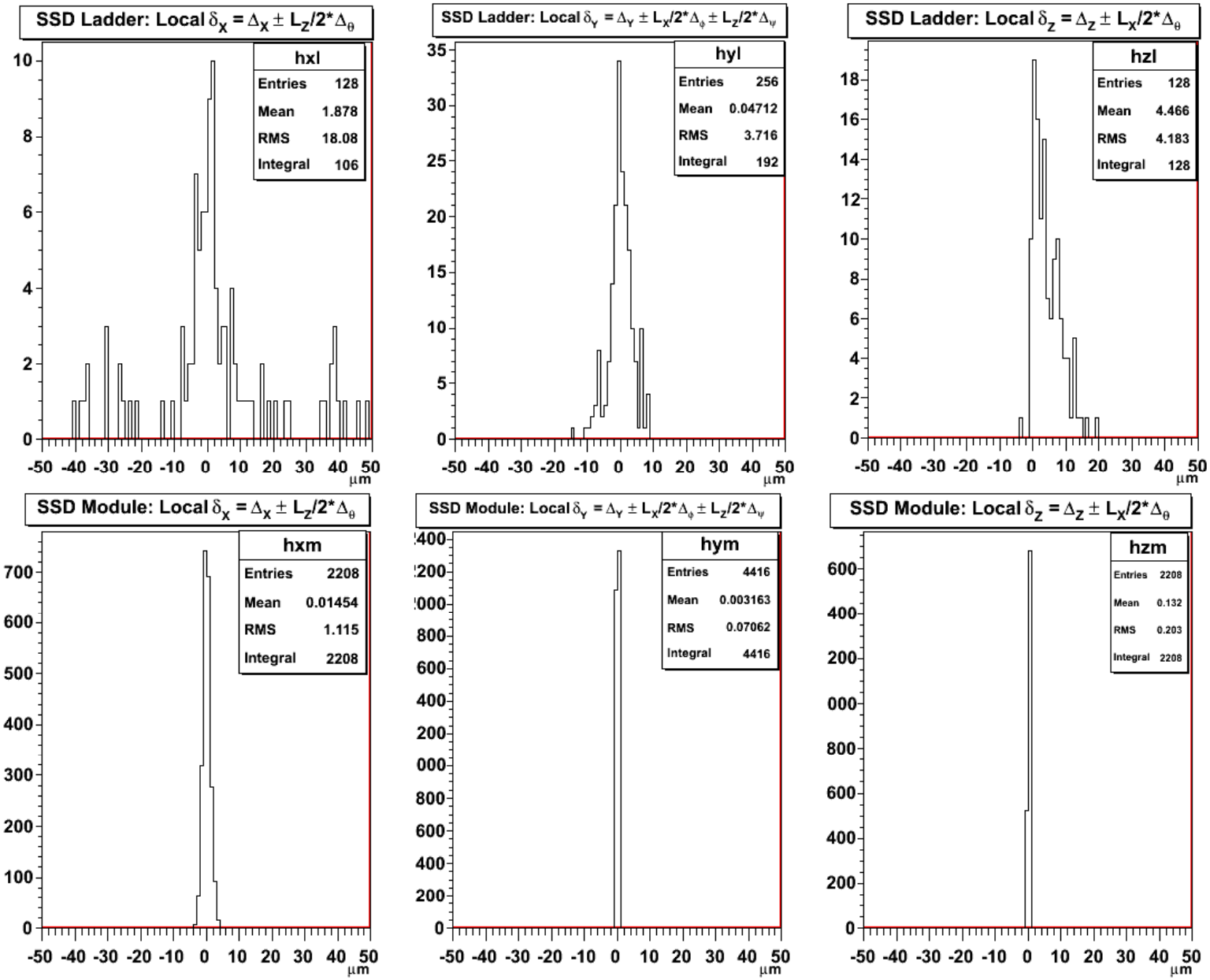
Two alternatives to constraint them:

- Strict inequalities $\max\{\delta x\} = |\, \Delta x + L_z \Delta\theta / 2 \,| < \varepsilon_x$ … the best to prevent the overlaps but very difficult to implement (the solution by Simplex algorithm is a complicated version of the Gaussian elimination which is prohibitive for large matrices). Existence of the solution is not guaranteed.
- Gaussian constraints $\overline{\chi}^2 = \chi^2 + w\left(\delta x / \varepsilon_x\right)^2 \ldots$: natural choice for the “survey-friendly” alignment.

  Implemented, constraints on local parameter combinations can be used both with local and global parameters definition. Needs some optimization of the relative weights for the constraints.

```
#                          val sigma    Wx    Wy    Wz    Wpsi      Wtheta        Wphi
# survey constraints on SSD modules wrt ladders
CONSTRAINT_LOCAL SSDM_X1   0.  7.e-4    1.    0.    0.    0.        3.66e-2       0.
CONSTRAINT_LOCAL SSDM_X2   0.  7.e-4    1.    0.    0.    0.       -3.66e-2       0.
CONSTRAINT_LOCAL SSDM_Y1   0.  7.e-4    0.    1.    0.    3.66e-2   0.            6.54e-2
. . .
CONSTRAINT_LOCAL SSDM_Z1   0.  7.e-4    0.    0.    1.    0.        6.54e-2       0.
CONSTRAINT_LOCAL SSDM_Z2   0.  7.e-4    0.    0.    1.    0.       -6.54e-2       0.
# survey constraints on SSD5 ladders wrt endcones
CONSTRAINT_LOCAL SSD5L_X1  0. 20.e-4    1.    0.    0.    0.        7.52e-1       0.
CONSTRAINT_LOCAL SSD5L_X2  0. 20.e-4    1.    0.    0.    0.       -7.52e-1       0.
. . .
CONSTRAINT_LOCAL SSD5L_Z2  0. 20.e-4    0.    0.    1.    0.       -6.54e-2       0.
```

Constraining local shifts (preliminary)



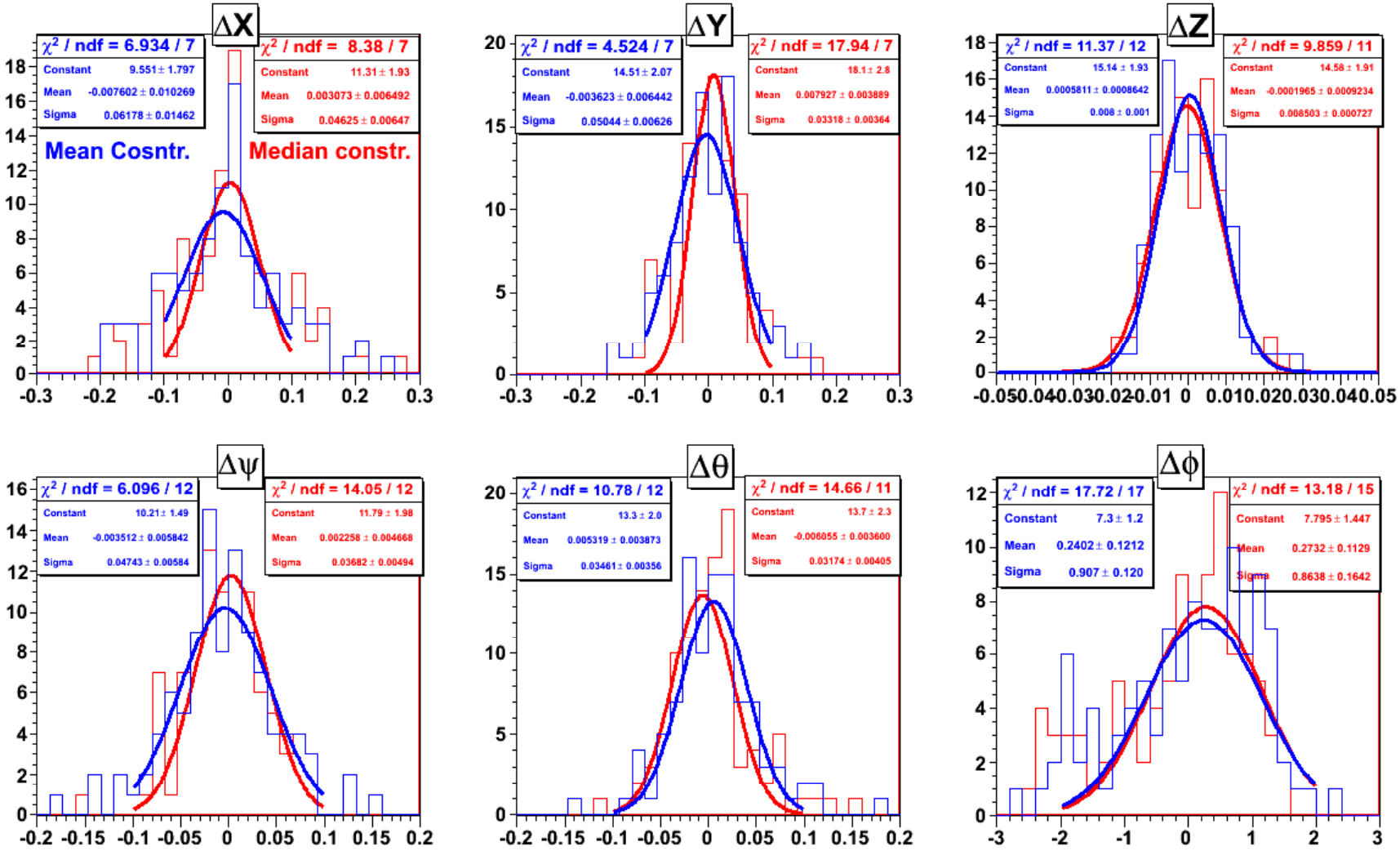
Quality of the alignment seems not to suffer, need further checks…

Summary

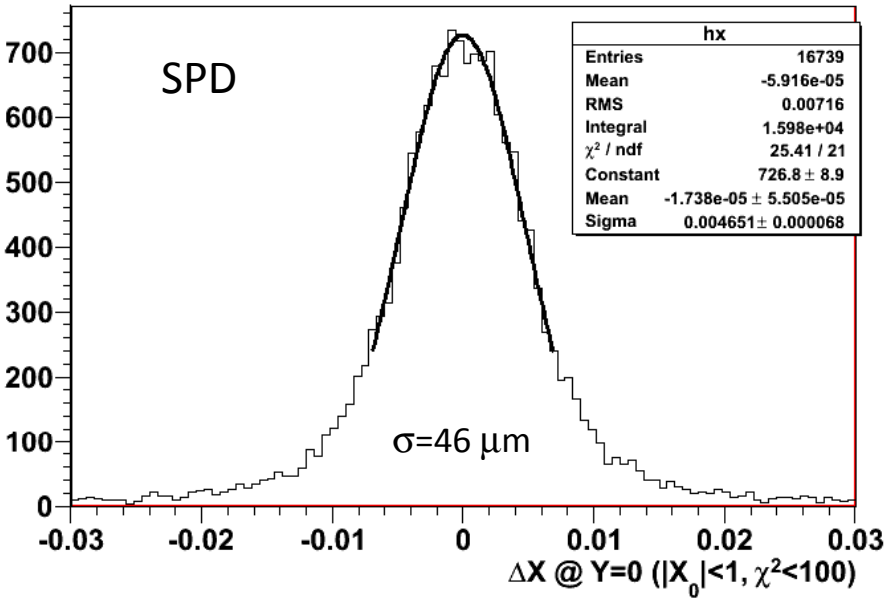
- MP2 and its interface to ITS is mostly ready.
  - Large economy in RAM/CPU
  - Simultaneous alignment of the full detector hierarchy is possible
  - SDD t0 and Vdrift calibration works
  - Gaussian constraints on local parameters linear combinations works but still needs some tuning

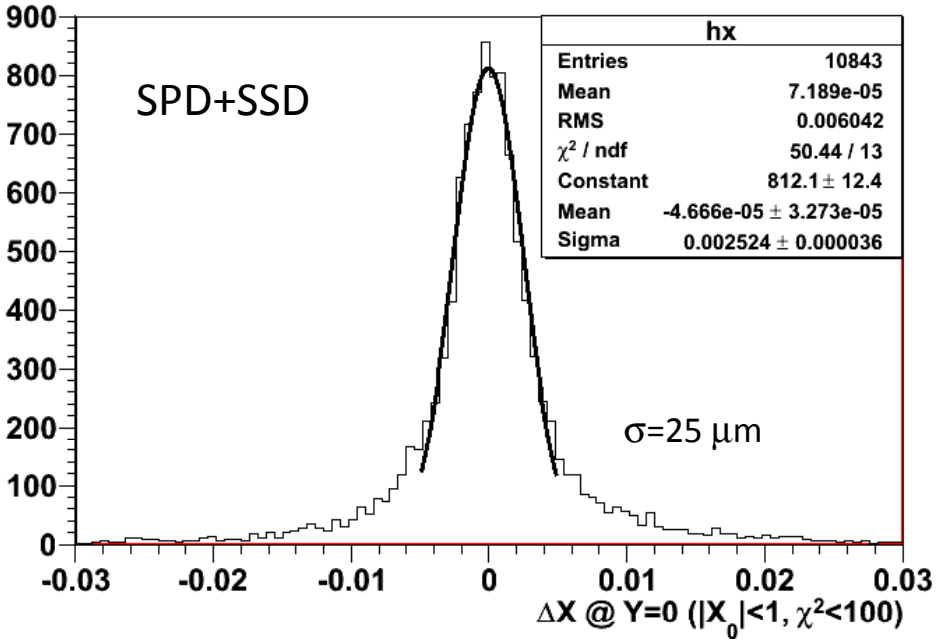
backup

Constraint on the median vs constraint on the mean



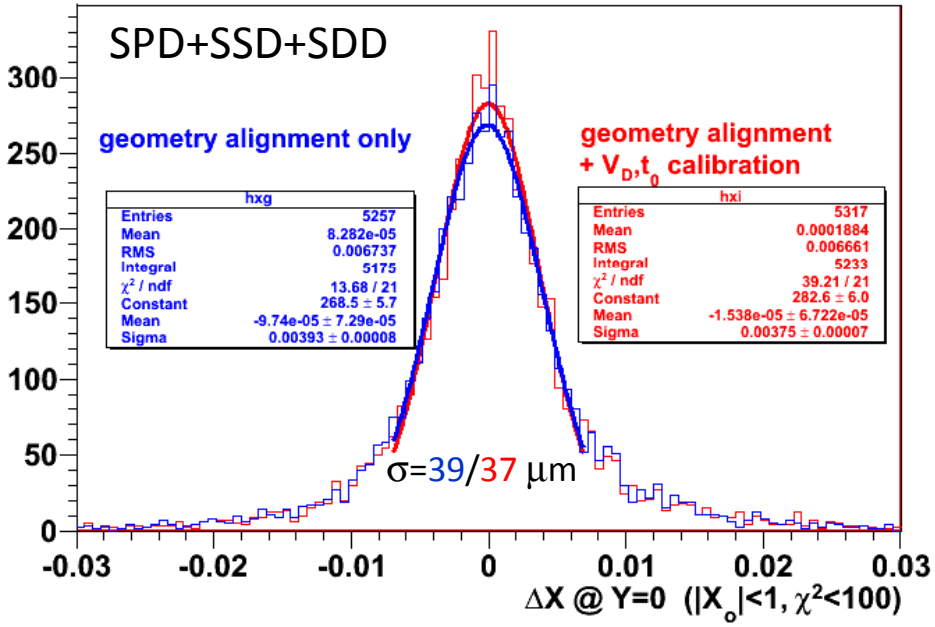
Corrections for *hstave* parameters: the **median** constraints provide more compact distribution of corrections with smaller most probable value than the constraints on the **mean**.



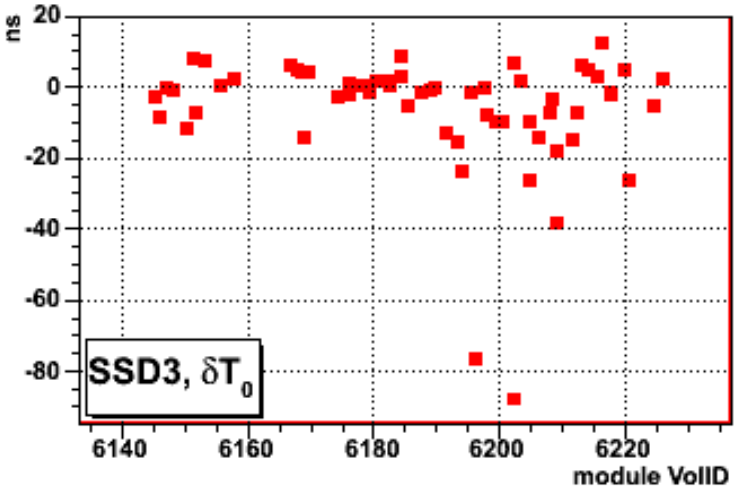


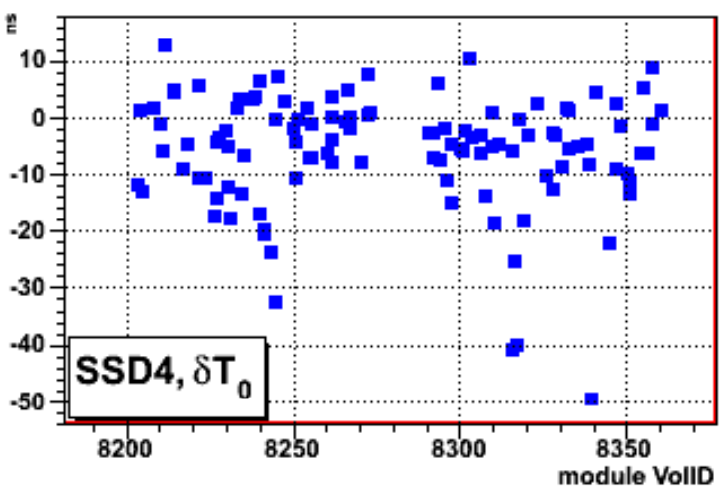
ΔX @ Y=0 for SPD and SPD+SSD are similar to those obtained by MP1 in Padova.

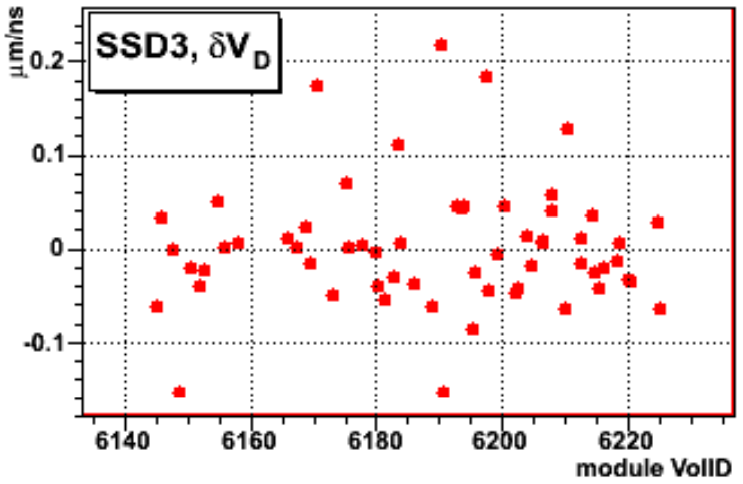
For SPD+SSD+SDD it is slightly better due to the calibration, but still, Inclusion of SDD into tracking deteriorates its quality

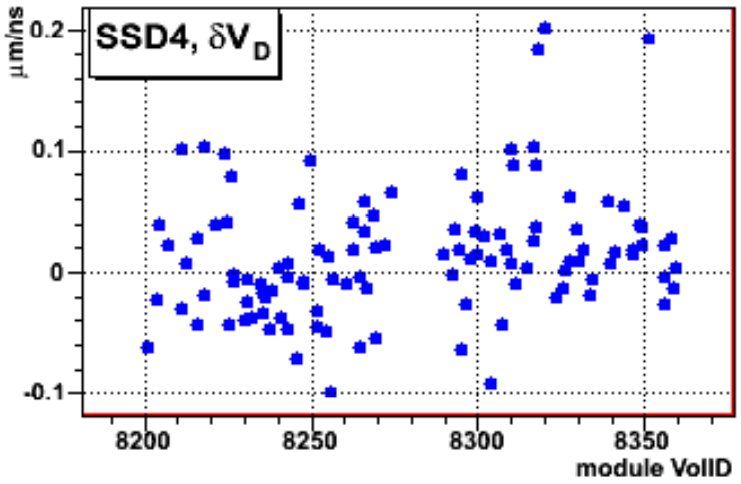


Obtained corrections to SDD calibration



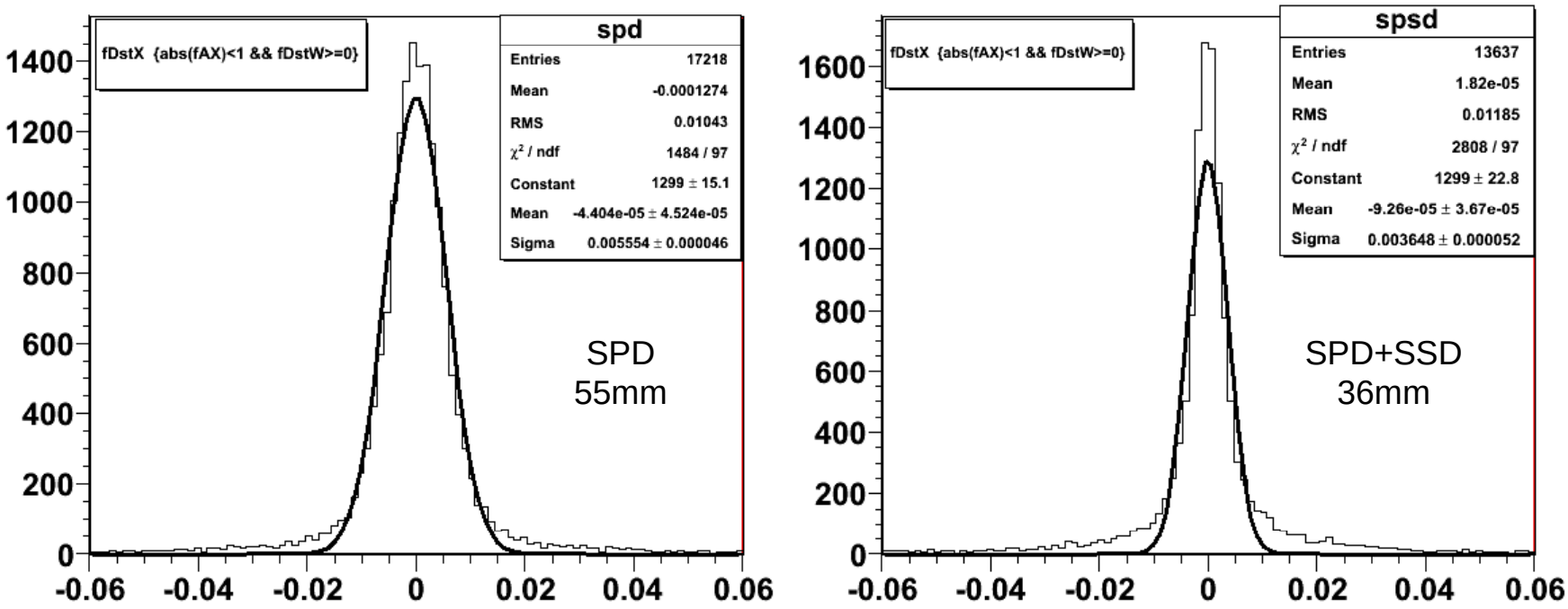






SPD and SSD layers were aligned simultaneously, providing global delta's for the full hierarchy of corresponding "alignable volumes".
Results are fully compatible with those obtained in Padova using MP1.

DX @ Y=0, $|X_0|<1$



Check of the alignment stability in time (since the time is not available in the AliTrackPoints, instead the entry number of the track in the tree is used)

Check of alignment stability with time
DX @ Y=0 vs “time” (event number)

SPD +SSD, at least 1 hit/layer

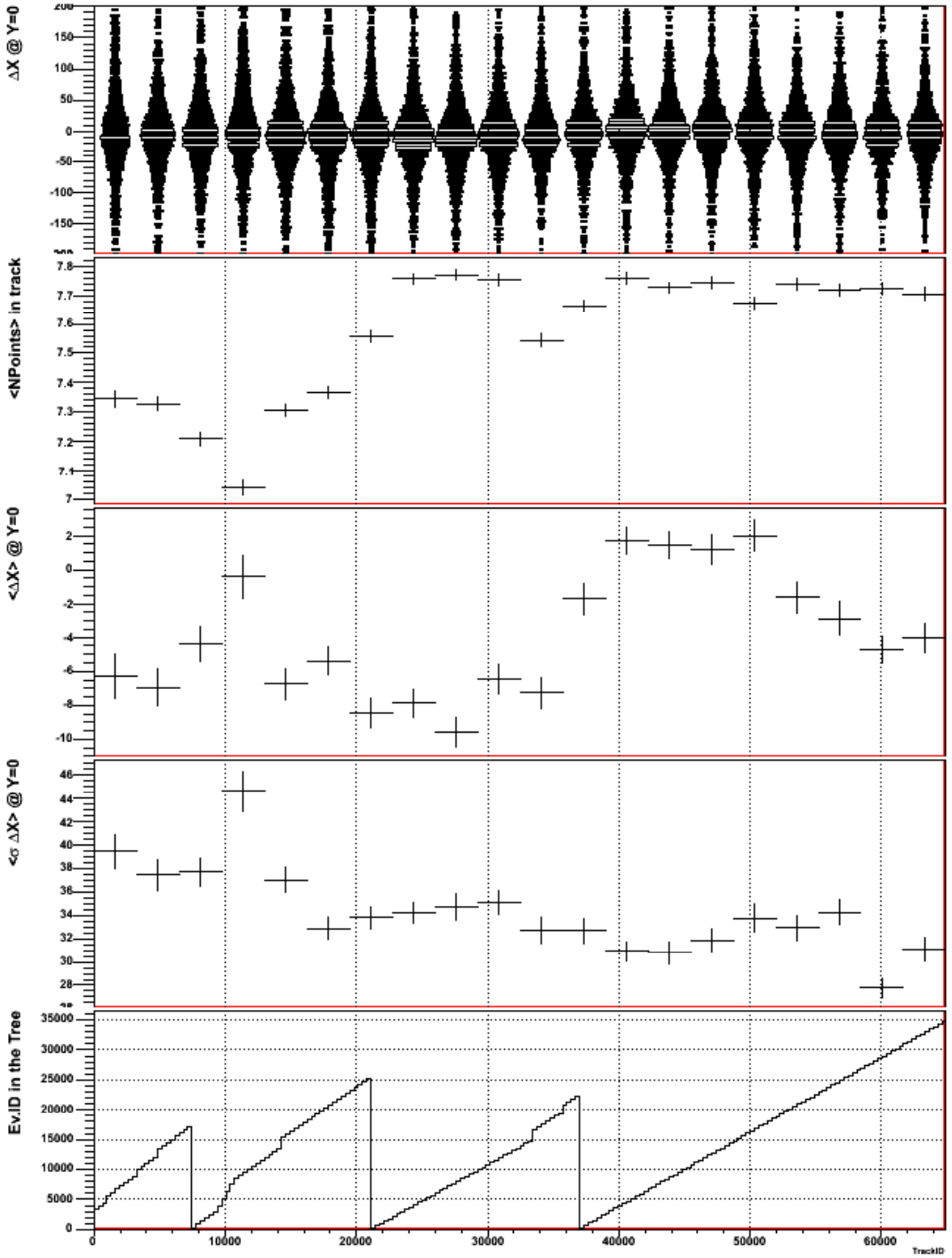
.

<N> points per track

Mean DX @ Y=0
(from FitSliceY)

s(DX) @ Y=0
(from FitSliceY)

Event entry in f 4 AliTrackPoint files (part1 – part4)



SPD only, at least 4 hits

DX @ Y=0 vs “time”.

<N> points per track

Mean DX @ Y=0
(from FitSliceY)

s(DX) @ Y=0
(from FitSliceY)