DDD workshop on reconstruction,

CERN, 8 July 2002

Muon - Lvl3 Tracks

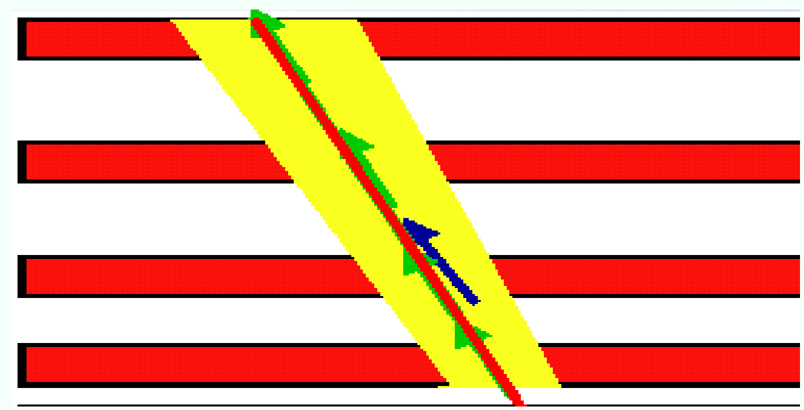
Stefano Lacaprara, INFN and Universitá di Padova

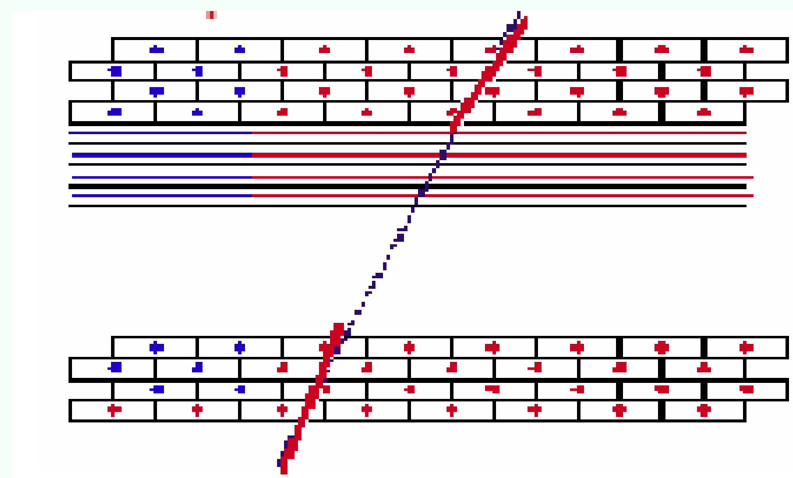
Outline:

- What we do,
- what do we need for local reconstruction
- and global one
- summary/conclusion

What is Muon reconstruction (L2 L3):

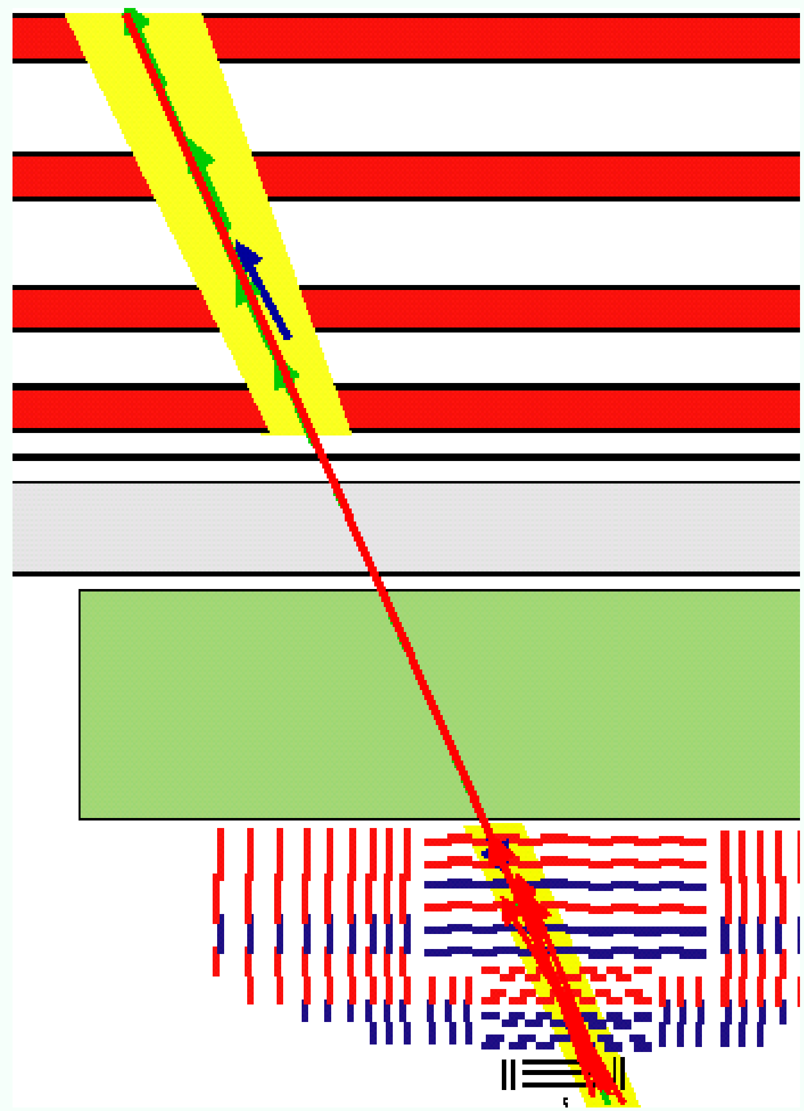
- Start from seed: L1 output (or internal);
- Define a region of interest around the seed;
- Start local reconstruction on chamber inside the region of interest (segments or hits as RecHits)
- Build trajectory (inside-out) using RecHits from the compatible chamber;
- Navigate to find more compatible chambers and RecHits;
- The backward kalman filtering outside-in;





What is Muon reconstruction (L2 L3):

- Propagate trajectory from inner-most Muon station to Tracker region (outer layer or IP);
- Open windows and define tracker region of interest;
- Create one or more seeds for each L2 Muons;
- Reconstruct tracker track starting from these seeds
- propagate trajectory to muon station and use the muon RecHits to improve trajectory;



What do we need for Local Reconstruction:

- ⋆ The RecHits inside DT, CSC, and RCP chambers are build according to rather complex algorithms
- ⋆ DT hits:
  - ⇝ Hit position with *DistanceFromWire*=*DriftVelocity*$(\vec{B}, \theta, (t), \ldots)$×*DriftTime*
  - ⇝ Resolution=*Res*$(\vec{B}, \theta, (t), \ldots)$
- ⋆ CSC Hits:
  - ⇝ Fit of near-by strips signal with a proper charge distribution (Gatti): position and error from fit result
- ⋆ RPC Hits: clusterization of near strips.
- ⋆ In any case a lot of parameters are used in the hit reconstruction $\Longrightarrow$ suitable for storing in DDD

⋆ Several type of *"constant"*

- ⇝ *Global* constant, unique for all the chambers (now ∼all);
- ⇝ *Local* which may change from chamber to chamber: for the real CMS, eg. HV different for detectors with some problem, different gas pressure, etc...

  This can change the behavior of a chamber, and so the reconstruction algo (at least the algorithm parameters).
- ⇝ So it should be possible to access the parameters given a chamber ID
- ⇝ *formula parameters*: eg. in DT the DriftVelocity may be parametrized like (just to give a concrete example)

$$V = V_0 + (a + b \cdot B_\perp) + (c + d \cdot B_{||}) + (e + f \cdot \theta) + \ldots$$

- ⇝ Where should we put the $(V_0, a, b, c, ..., f, ...)$ constants?

- In the DDD: correct place for all the “numbers”, but the numbers are strictly related to a specific algo (i.e. formula), so formula in one place (code) and coefficient in an other (DDD): probably hard to maintain!
- In the code: numbers close to the code they are related to, so easy to maintain but hard-coded!

⋆ How to access to chamber specific parameters? Hierarchical access: first get the chamber, then the params of the chamber? how to deal with parameters which can be “local” but are shared by many chambers?

⋆ User customization: many numbers are now configurable via `.orcarc`, and the default value is in the code: it could be a good idea to allow for this kind of user configuration BUT taking the default value from the DDD.
An easy SimpleConfigurable↔DDD interface can be useful

- ⋆ Status of the detector, cells, CMS, . . .
- ⋆ Dead, noisy channel, defined according “run number”
- ⋆ Particular condition of the chambers/detectors (HV, gas, read-out, . . .)
- ⋆ All local reconstruction is done the detector reference frame, so, at this stage, no particular information about position of detector inside CMS is needed
- ⋆ Position of sub-detector parts in the detector reference frame, already available when the detectors are built (eg. SL to DT chamber frame transformation)
- ⋆ Alignment of sub-detector parts inside a detector: eg SuperLayer in a DT, layer in CSC, ...

What do we need for Global Reconstruction:

- Local reco is done in local frame, transformation to global is done by *DetUnit*, available once the DetUnit (i.e. the detector) is built;
- No explicit use of geometry related stuff is used during the global reco: all is delegated to Det, DetUnit, DetLayer.
- The navigation is performed using the DetLayer (collection of DetUnits): different strategies (navigation school or η based compatibility) but in any case the position of the DetUnit is taken from the DetUnit itself.
- The alignment information must be available for each DetUnit eventually with proper hierarchy: a whole wheel/disk is displaced, the a sector inside the wheel, then the chamber inside the sector, . . .
- Some number is used also in the global reco (now mostly hard-coded or SimpleConfigurable), less than in the local reco
- Many cuts of various type: χ^2 cuts, number of hits, ... NOT for DDD (or yes?)

- Most important point now: extrapolator in the Muon region is **GEANE**.
- This means that the GEANT3 geometry (namely iron position, magnetic field map, . . .) must be available to GEANE to proper extrapolate the trajectory
- A future substitute can be directly interfaced to DDD geometry, but it's unlikely that GEANE will ever be
- We must guarantee that the Geant3 geometry is exactly the same as the DDD one:
- not hard if the DDD is build from the cmsim tz, not so easy is the DDD input is an other.

Summary–conclusion

- DDD perfect place to store many parameters now scattered all over the code:
- Parameters may be specific for a specific detector or global;
- where to put algorithm formula parameters?
- User customization via `.orcarc`
- Geometric position to go from local to global frame;
- Alignment information;
- Detector status information;
- `GEANE`: until we use it, `GEANT3` geometry must be available.