GenFit

Brief Status

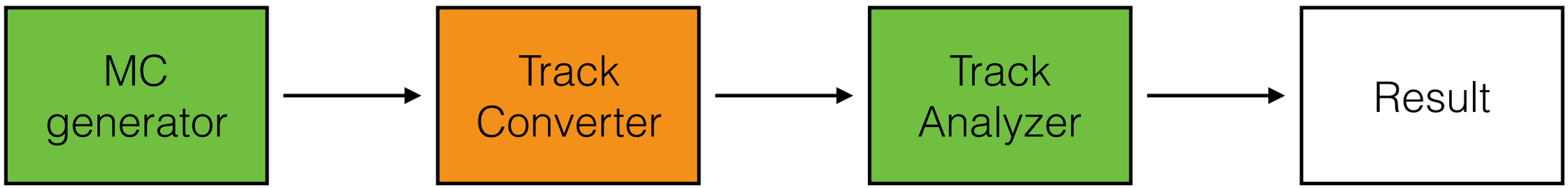
- 9/13일 아침까지 1 track fitting / 3 sec, 1000 track fitting / 3 hr 의 문제에 봉착해 있었음…
  - Variable magnetic field? (i.e. Field value I/O in a fitting)
  - Wrong assignment of hit point?

⇨ Not changed / No abnormal track

- Compare data setting/fitting codes with LAMPSroot : No definite difference.

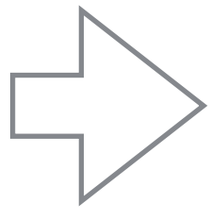
문제점 발견과 해결

Data flow



Track converter(MC 데이터를 Track 별로 모아 clustering 하는 program)에서 입자의 초기 위치와 운동량을 반대로 Assignment 하고 있었다…

InitMom = track->v
InitPos = track->p



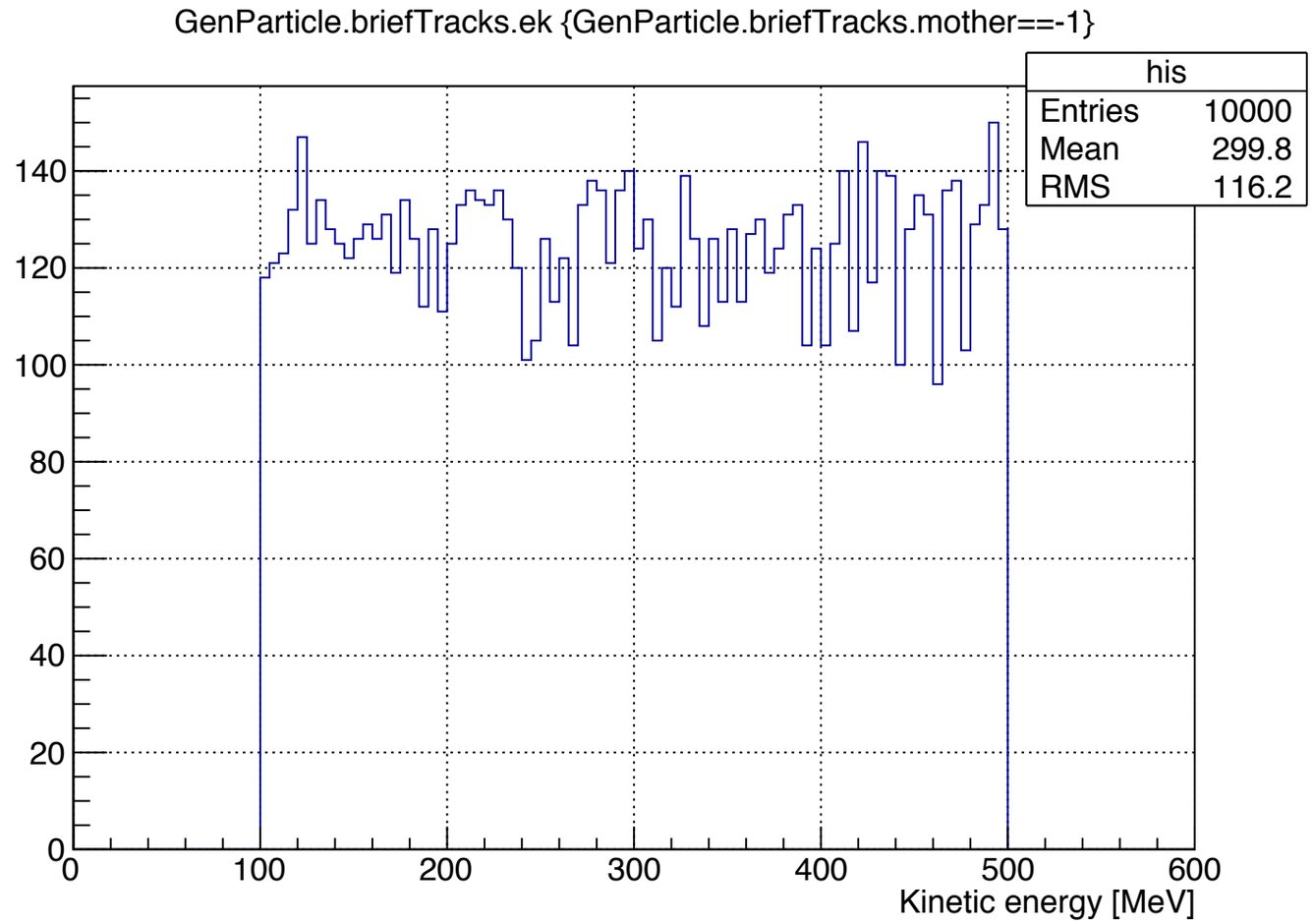
InitMom = track->p
InitPos = track->v

- Track fitting speed : 0.3 track/sec -> 400 track / sec.
- Fitting error(for proton event) : ~20% -> 0.

MC data generation

- Generating protons at center of TPC (x,y,z=0) with flat Ek spectrum.
- Beam direction : z direction 고정.

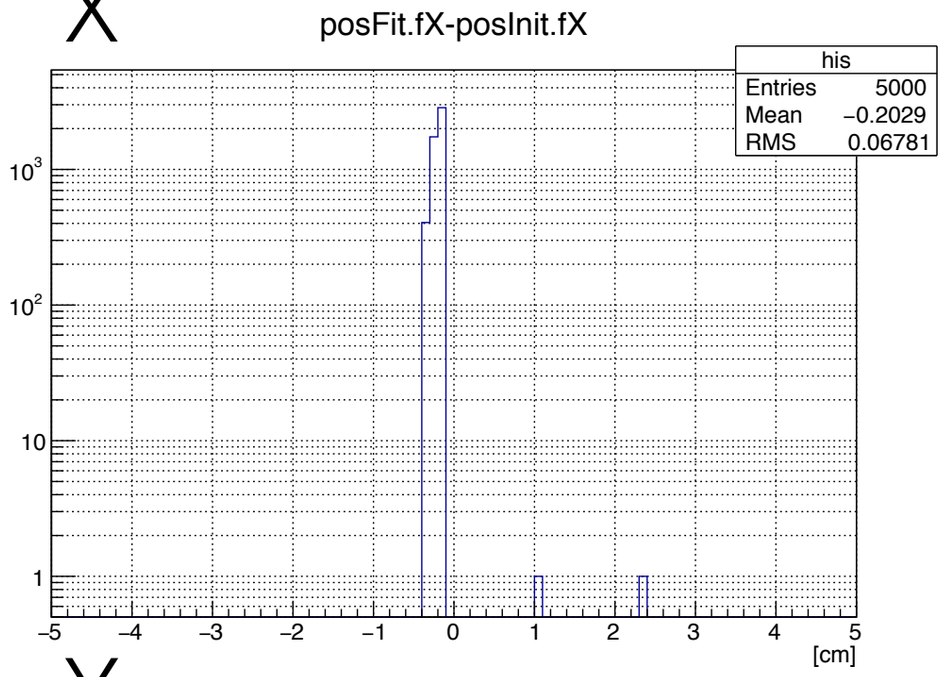
Ek distribution



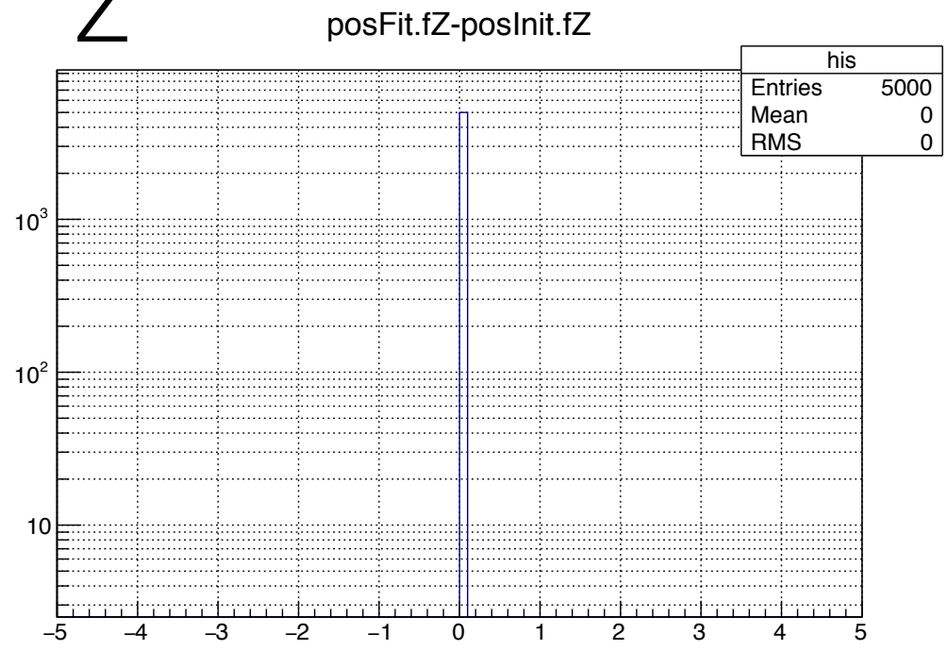
Fitting results

Track starting position

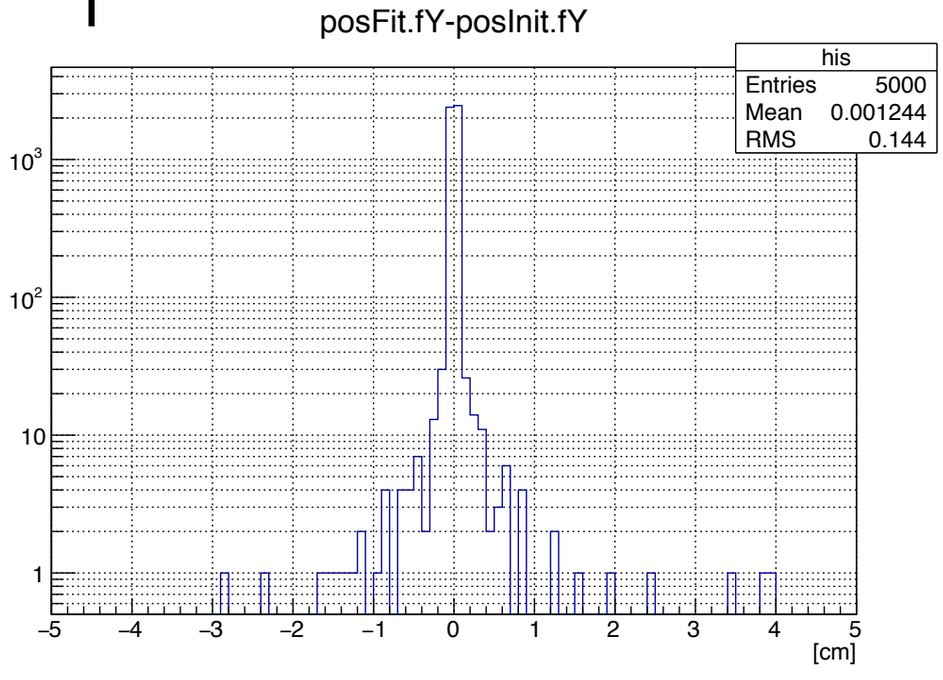
X



Z



Y

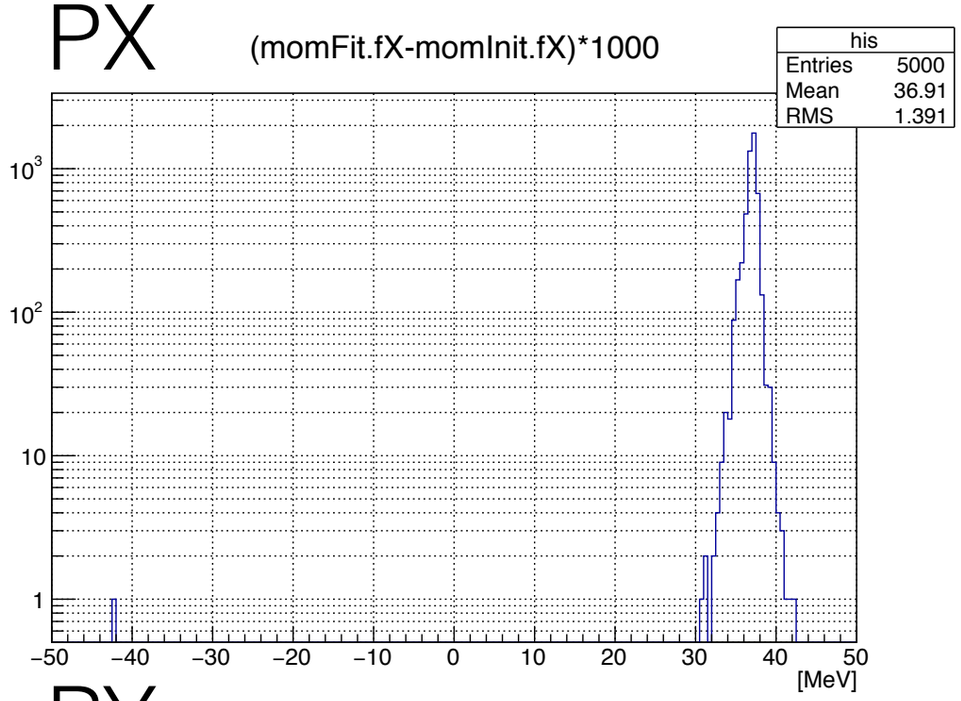


Measurement plane : Z = 0 고정

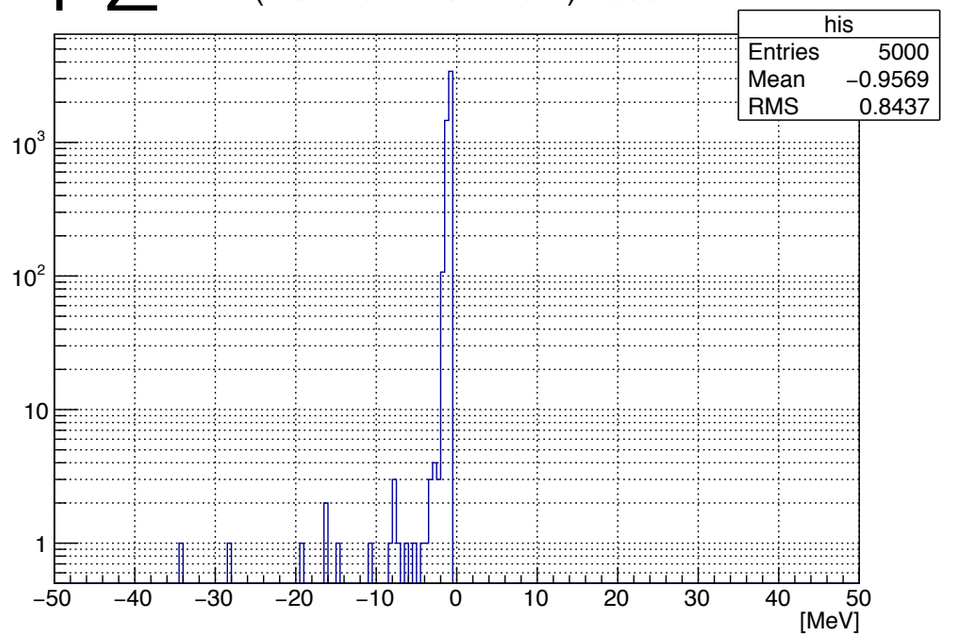
Fitting results

momentum difference

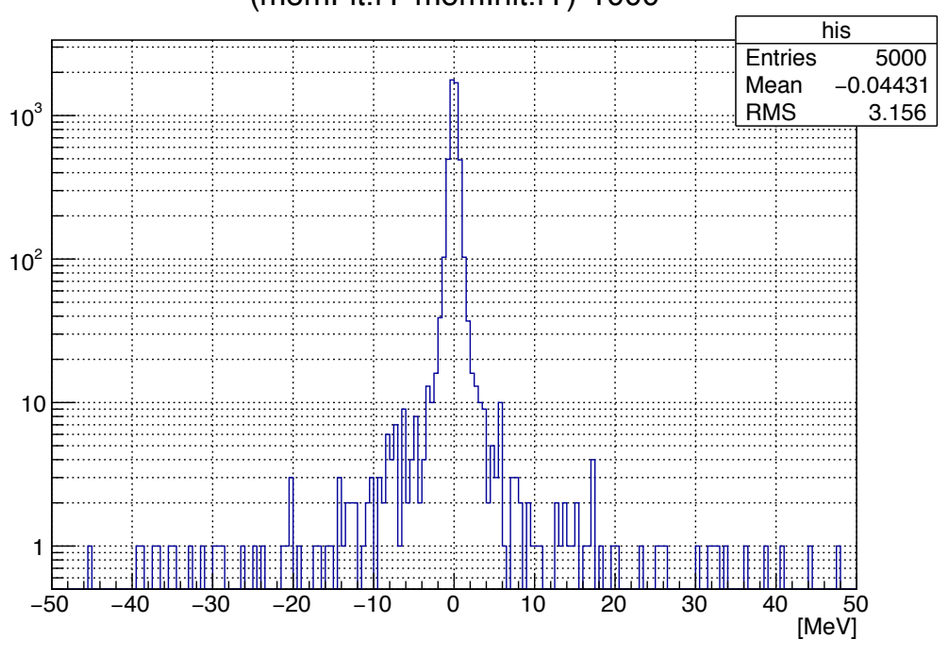
PX



PZ



PY



PX :

initial px == 0,

shifted ~40MeV after fitting

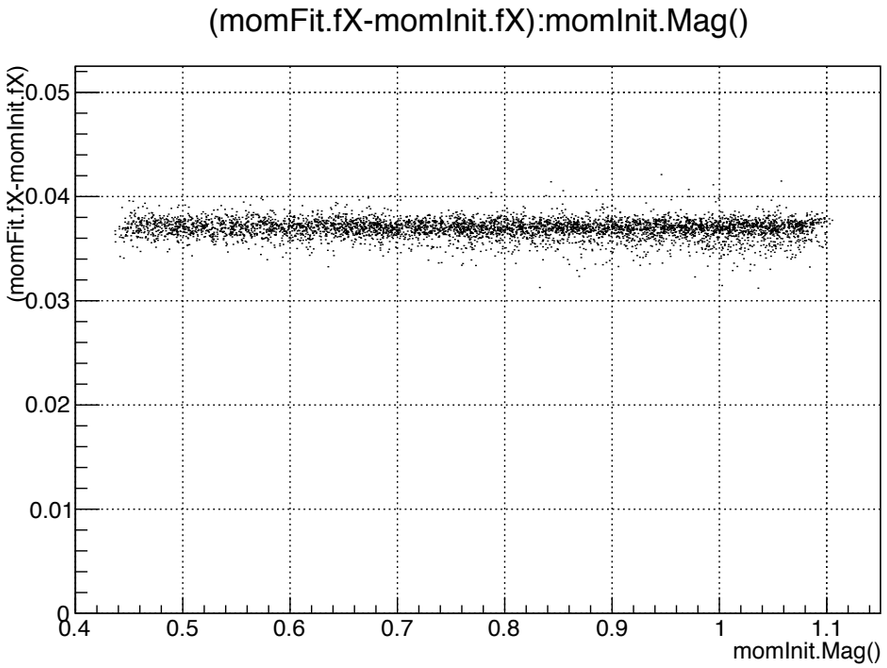
PZ :

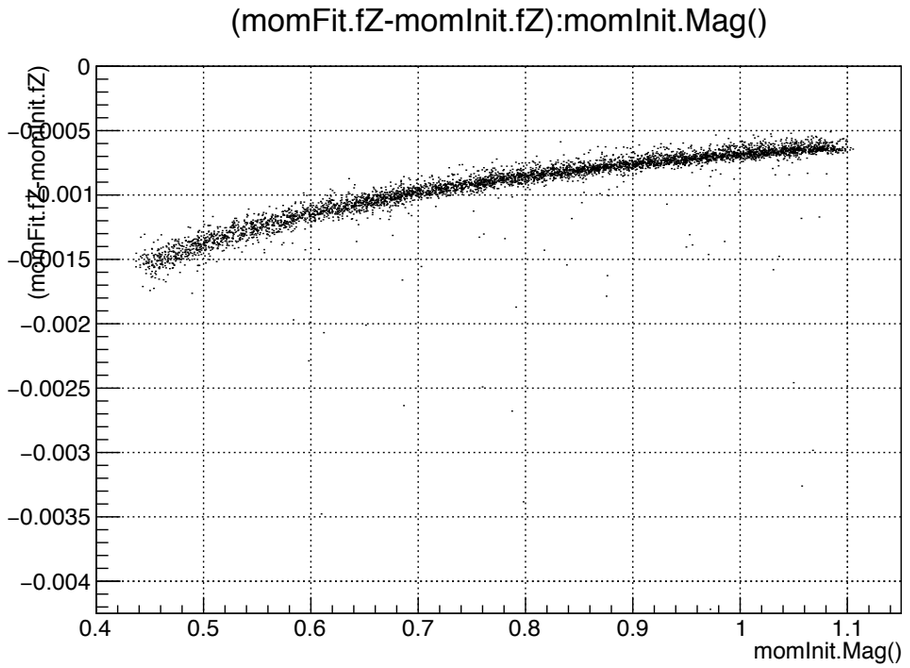
Always Fitted value < initial value

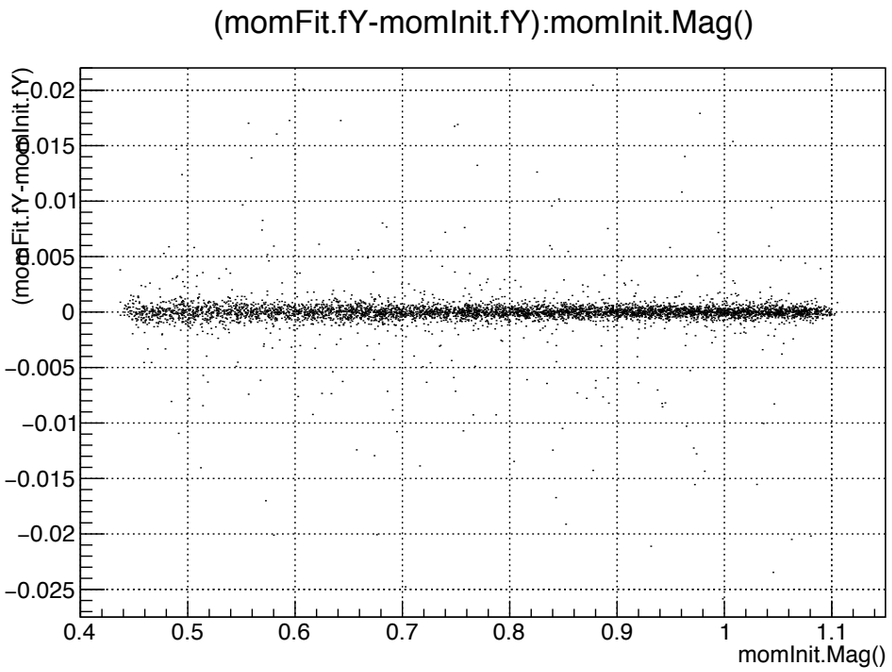
PY :

relatively large RMS/Error

기타


(momFit.fX-momInit.fX):momInit.Mag()
(momFit.fX-momInit.fX)
momInit.Mag()


(momFit.fZ-momInit.fZ):momInit.Mag()
(momFit.fZ-momInit.fZ)
momInit.Mag()

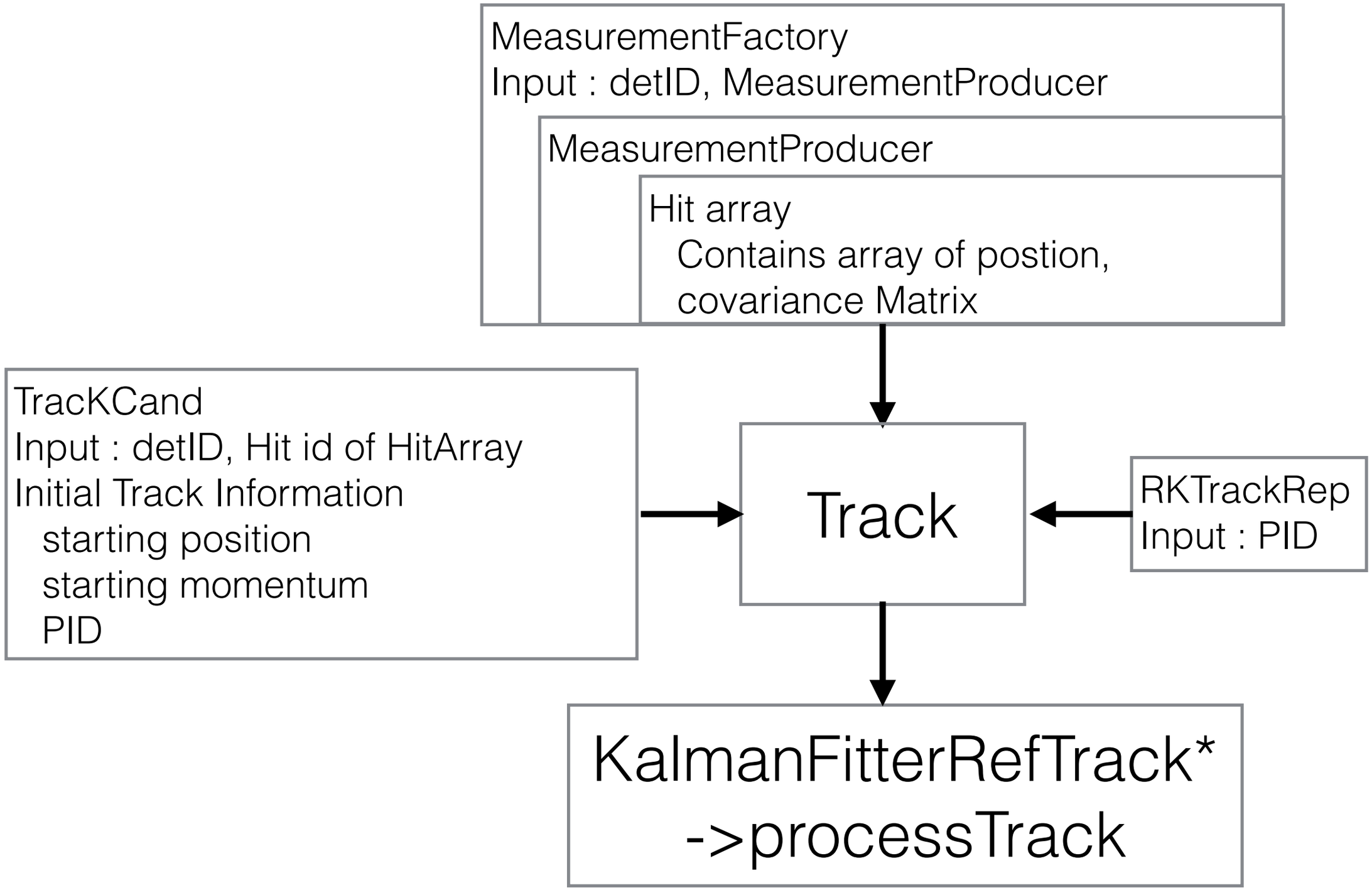

(momFit.fY-momInit.fY):momInit.Mag()
(momFit.fY-momInit.fY)
momInit.Mag()

Usage

Making Track

(Insert measured point to the Track)

Flow Chart : Fit track



Flow Chart : Get Fitting Result

Get TrackPoint with FitterInfo from fitted track

```
TrackPoint* tp =
  fitTrack->getPointWithMeasurementAndFitterInfo(0,TrackRep)
0: initial point.
Get fitting information at the initial point
```

Get KalmanFittedState from TrackPoint

```
KalmanFittedSateOnPlane
kfsop(*(static_cast<KalmanFitterInfo*>
(tp->getFitterInfo(rep))->getBackWardUpdate()))
rep->extrapolateToPlane(kfsop, Original Plane)
```

Get state / Momentum, position, charge from state

```
TVectorD state = kfsop.getState()
kfsop.getPosMom( TVector3& pos, TVector3& mom)
```

Get Initial information

```
1.Get original state
MeasuredStateOnPlan
e stateRef( rep )

2.Set to constant State
const StateOnPlane
stateRefOrig(stateRef)

3.Get state /
momentum, position,
charge from state
TVectorD origState =
stateRefOrig.getState()
stateRefOrig.getPosMo
m(TVector3& pos,
TVector3& mom)
```

Classes

Track

- Collection of TrackPoint objects, AbsTrackRep objects and FitStatus objects

- Holds a number of AbsTrackRep objects, which correspond to the different particle hypotheses or track models which should be fitted. A 6D stateSeed and 6x6 covSeed should be provided as start values for fitting. When fitting the Track with a AbsFitter, a FitStatus object will be created, containing information about the fit. The fitted states will be stored in AbsFitterInfo objects in every TrackPoints.

- The fit will be performed for every AbsTrackRep, so after the fit there will be one AbsFitterInfo for each AbsTrackRep in every TrackPoint, as well as one FitStatus for every AbsTrackRep.

- vector<AbsTrackRep*> trackReps_
- vector<TrackPoint*> trackPoints_
- vector<TrackPoint*> trackPointsWithMeasurement_
- map<AbsTrackRep*, FitStatus*> fitStatuses_
- int mcTrackId_
- double timeSeed_
- TVectorD stateSeed_
- TMatrixDSym covSeed_

- Track()
- Track(TrackCand& trackCand, MeasurementFactory<AbsMeasurement>&, AbsTrackRep*=NULL)
- Track(AbsTrackRep*, TVectorD&)
- Track(AbsTrackRep*, TVector3&, TVector3&)
- Track(AbsTrackRep*,TVectorD&, TMatrixDSym&)
- Track(const Track&)
- Track& operator=(Track)

- double getTimeSeed()
- void setTimeSeed(double)
- TVectorD& getStateSeed()
- void setStateSeed(TVectorD&)
- void setStateSeed(TVector3&, TVector3&)
- TMatrixDSym& getCovSeed()
- void setCovSeed(TMatrixDSym&)
- void setMcTrackId(int)
- void insertPoint(TrackPoint*, int=-1)
- void insertPoint(std::vector<TrackPoint*>, int)
- void deletePoint(int)
- void insertMeasurement(AbsMeasurement*, int =-1)
- double getTrackLen(AbsTrackRep*=NULL, int=0,int=-1)
- double getTOF(AbsTrackRep*=NULL,int=0,int=-1)
- TrackCand* ConstructTrackCand()

- TrackPoint* getPoint(int)
- TrackPoint* getPointWithMeasurement(int)
- TrackPoint* getPointWithFitterInfo(int, AbsTrackRep*)
- TrackPoint* getPointWithMeasurementAndFitterInfo(int,AbsTrackRep)
- vector<TrackPoint*>& getPoints()
- vector<TrackPoint*>& getPointsWithMeasurement()
- uint getNumPoints()
- uint getNumPointWithMeasurement(int)

- AbsTrackRep* getTrackRep(int)
- vector<AbsTrackRep*> getTrackReps()
- uint getNumReps()
- int getIdForRep(AbsTrackRep*)
- AbsTrackRep* getCardinalRep()
- AbsTrackRep* getCardianlRepId()
- int getMcTrackId()
- bool hasFitStatus(AbsTrackRep*=NULL)
- FitStatus* getFitStatus(AbsTrackRep*=NULL)
- bool hasKalmanFitStatus(AbsTrackRep*=NULL)
- KalmanFitStatus* getKalmanFitStatus(AbsTrackRep*=NULL)
- void setFitStatus(FitStatus* fitStatus, AbsTrackRep*)

TrackPoint

- Object containing AbsMeasurement and AbsFitterInfo objects

- Track* track_
- vector<AbsMeasurement*> rawMeasurements_
- map<AbsTrackRep*, AbsFitterInfo*> fitterInfos_
- vector<AbsFitterInfo* > vFitterInfos_

- TrackPoint()
- TrackPoint(Track*)
- TrackPoint(vector<AbsMeasurement*>&, Track*)
- TrackPoint(AbsMeasurement*, Track*)
- TrackPoint(TrackPoint&, map<AbsTrackRep*>, AbsTrackRep*>&, vector<AbsTrackRep*>*=NULL)

- void addRawMeasurement(AbsMeasurement*)
- vector<AbsMeasurement*>& getRawMeasurements()
- vector<AbsFitterInfo*> getFitterInfos()
- void setFitterInfo(AbsFitterInfo* fitterInfo)
- AbsFitterInfo* getFitterInfo(AbsTrackRep*)
- KalmanFitterInfo* getKalmanFitterInfo(AbsTrackRep*)
- int getNumRawMeasurements()

TrackCand

- Track candidate

- To store a list of indices to cluster objects. Each cluster in the track is identified by it's detector ID and it's index in the corresponding TClonesArray.

- Also there is a ordering parameter to order hits.

- Optionally, plane indices for the hits can be stored(most importantly for fitting with the Daf).

- This information is used by the RecoHitFactory to automatically load RecoHits into a Track. Through his it is possible to define Tracks over an arbitrary number of different detectors.

- In addition TrackCand offers members to store starting values for the fit.

- The starting values (seeds) for the fit are stored as a 6D state and its corresponding 6x6 covariance matrix. All seed getter and setter manipulate these two members but the user can chose using TVector3 or TMatrixD to get/set the seed state. However this information is not automatically used in genfit. But a pointer to a TrackCand can be passed to the a RKTrackRep constructor to make used of this information without manually extracting it from the TrackCand object.

- vector<TrackCandHit*> hits_
- int mcTrackId_
- int pdg_
- double time_
- TVectorD state6D_ // position & momentum
- TMatrixDSym cov6D_ //error?
- double q_ // charge

- TrackCand()
- void addHit(int,int, int = -1, double =0)
- void addHit(TrackCandHit* hit)

TrackCandHit

- Hit object for use in TrackCand. Provides IDs and sorting parameters

- int detId_
- int hitId_
- int planeId_
- double sortingParameter_

- TrackCandHit(int, int, int, double)

- void addHit(int,int, int = -1, double =0)

- void addHit(TrackCandHit* hit)

AbsTrackRep

- Abstract base class for a track representation

- Provides functionality to extrapolate a StateOnPlane to another Detplanae, to the POCA to a line or a point, or cylinder or sphere. Defines a set of parameters describing the track. StateOnPlane objects are always defined with a track parameterisation of a specific AbsTrackRep. The AbsTrackRep provides functionality to translate from the internal representation of a state into cartesian position and momentum (and covariance) and vice versa.

- int pdgCode_
- char propDir_
- uint debugLvl_

- AbsTrackRep()
- AbsTreackRep(int png, char propDir)
- AbsTrackRep* clone()

- double extrapolateToPlane(StateOnPlane&, SharedPlanePtr&, bool, bool)
- double extrapolateToLine(StateOnPlane&, TVector3&,TVector3& bool,bool)
- double extrapolateToLine(StateOnPlane&, TVector3&, TVector3&, TVector3&, TVector3&, TVector3&, bool, bool)
- double extrapolateToPoint(StateOnPlane&, TVector3&, bool, bool)
- double extrapolateToPoint(StateOnPlane&, TVector3&, TMatrixDSym&, bool, bool)
- double extrapolateToCylinder(StateOnPlane&, double, TVector3&, TVector3&, bool, bool)
- double extrapolateToCone()//
- double extrapolateToShere()
- double extrapolateBy() //
- double extrapolateToMeasurement(StateOnPlane*, AbsMeasurement*, bool, bool)
- uint getDim()

- TVector3 getPos(StateOnPlane&)
- TVector3 getMom(StateOnPlane&)
- TVector3 getDir(StateOnPlane&)
- void getPosMom(StateOnPlane&, TVector3&, TVector3&)
- void getPosDir(StateOnPlane&, TVector&, TVector3&)
- TVectorD get6DState(StateOnPlane&)
- TMatrixDSym get6DState(StateOnPlane&)
- void getPosMomCov(MeasuredStateOnPlane&, TVector3&, TVector3&, TMatrixDSym&)
- void get6DStateCov(MeasuredStateOnPlane&, TVectorD&, TVectorD&)
- double getMomMag(StateOnPlane&)
- double getMomVar(MeasuredStateOnPlane&)

- int getPDG()
- double getPDGCharge()
- double getCharge(StateOnPlane&)
- double getQop(StateOnPlane&)
- double getMass(StateOnPlane&)
- char getPropDir()
- double getRadiationLength()
- double getTime(StateOnPlane&)

- void setPosMom(StateOnPlane&, TVector3&, TVector3&)
- void setPosMom(StateOnPlane& TVectorD&)
- void setPosMomErr(MeasuredStateOnPlane&, TVector3&, TVector3&, TVector3&)
- void setPosMomCov(MeasuredStateOnPlane&, TVector3&, TVector3&, TMatrixDSym&)
- void setPosMomCov(MeasuredStateOnPlane&, TVectorD&, TMatrixDSym&)
- void setChargeSign(StateOnPlane& , double)
- void setQop(StateOnPlane&, double)
- void setTime(StateOnPlane&, double)

AbsMeasurement

- Contains the measurement and covariance in raw detector coordinates
- Detector and hit ids can be used to point back to the original detector hits (clusters etc.).

- TVectorD rawHitCoords_
- TMatrixDSym rawHitCov_
- int detId_
- int hitId_
- TrackPoint* trackPoint_

- AbsMeasurement()
- AbsMeasurement(int nDims)
- AbsMeasurement(const TVectorD&, TMatrixDSym&, int, int, TrackPoint*)

- TrackPoint* getTrackPoint()
- void setTrackPoint(TrackPoint*)
- TVectorD& getRawHitCoords()
- TVectorDSym& getRawHitCov()
- int getDetId()
- int getHitId()
- int getDim()

- void setRawHitCoords(TVectorD&)
- void setRawHitCov(TMatrixDSym&)
- void setDetId(int)
- void setHitId(int)
- SharedPlanePtr constructPlane(const StateOnPlane& state)
- vector<MeasurementOnPlane*> constructMeasurementsOnPlane(const StateOnPlane&)
- AbsHMatrix* constructHMatrix(AbsTrackRep*)

StateOnPlane

- A state with arbitrary dimension defined in a DetPlane.

- The dimension and meaning of the state vector are defined by the track parameterisation of the rep. sharedPlane is a shared_pointer, the ownership over that plane is shared between all StateOnPlane objects defined in that plane. The definition of the state is bound to the TrackRep rep. Therefore, the StateOnPlane contains a pointer to a AbsTrackRep. It will provide functionality to extrapolate it and translate the state it into cartesian coordinates. Shortcuts to all functions of the AbsTrackRep which use this StateOnPlane are also provided here.

- TVectorD state_
- TVectorD auxInfo_ (charge, flight direction)
- SharedPlanePtr sharedPlane_
- AbsTrackRep* rep_

- StateOnPlane(AbsTrackRep*)
- StateOnPlane(TVectorD&, SharedPlanePtr&, AbsTrackRep*)
- StateOnPlane(TVectorD&, SharedPlanePtr&, AbsTrackRep*, TVectorD&)

- TVectorD& getState()
- TVectorD& getAuxInfo()
- SharedPlanePtr& getPlane()
- AbsTrackRep* getRep()

- void setState(TVectorD&)
- void setPlane(SharedPlanePtr&)
- void setStatePlane(TVectorD&, SharedPlanePtr&)
- void setAuxInfo(TVectorD&)
- void setRep(AbsTrackRep*)

- Other functions are adopted from AbsTrackRep.

MeasuredStateOnPlane

- StateOnPlane with additional covariance matrix

- TMatrixDSym cov_

- MeasuredStateOnPlane(AbsTrackRep* = NULL)

- MeasuredStateOnPlane(TVectorD& state, TMatrixDSym&, SharedPlanePtr&, AbsTrackRep*)

- MeasuredStateOnPlane(TVectorD& state, TMatrixDSym&, SharedPlanePtr&, AbsTrackRep*, TVectorD&)

- MeasuredStateOnPlane(StateOnPlane&, TMatrixDSym&)

- void setStateCov(TVectorD&, TMatrixDSym&)
- void setStateCovPlane(TVectorD&, TMatrixDSym&, SharedPlanePtr&)
- void setCov(TMatrixDSym&)
- TMatrixDSym& getCov()

DetPlane

- Detector plane

- A detector plane is the principle object to define coordinate systems for track fitting in genfit. Since a particle trajectory is a one-dimensional object (regardless of any specific parameterisation) positions with respect to the track are always measured in a plane.

- Which plane is chosen depends on the type of detector. Fixed plane detectors have their detector plane defined by their mechanical setup. While wire chambers or time projection chambers might want to define a detector plane more flexibly.

- This class parameterises plane in terms of an origin vector o and tow plane-spanning directions u and v

- TVector3 o_
- TVector3 u_
- TVector3 v_

- DetPlane(TVector3&, TVector3&, TVector3&)
- DetPlane(TVector3&, TVector3&)

SharedPlanePtr

- Shared pointer to a DetPlane
- ownership can be shared, e.g between multiple StateOnPlane objects.
- The DetPlane will automatically be deleted, if no owner remains.

- typedef boost::shared_ptr<DetPlane> SharedPlanePtr

DetPlane

- Detector plane

- A detector plane is the principle object to define coordinate systems for track fitting in genfit. Since a particle trajectory is a one-dimensional object (regardless of any specific parameterisation) positions with respect to the track are always measured in a plane.

- Which plane is chosen depends on the type of detector. Fixed plane detectors have their detector plane defined by their mechanical setup. While wire chambers or time projection chambers might want to define a detector plane more flexibly.

- This class parameterises plane in terms of an origin vector o and tow plane-spanning directions u and v

- TVector3 o_
- TVector3 u_
- TVector3 v_

- DetPlane(TVector3&, TVector3&, TVector3&)
- DetPlane(TVector3&, TVector3&)

AbsFitter

- Abstract base class for fitters.

- int debugLvl_

- AbsFitter()
- processTrack(Track*, bool)
  - Process all reps. Start with the cardianlRep and resort the hits if necessary (and supported by the fitter)
- processTrackWithRep(Track*, AbsTrackRep*, bool)
  - process Track with one AbsTrackRep of the Track. Optionally resort the hits if necessary (and supported by the fitter)

AbsFitterInfo

- This class collects all information needed and produced by a specific AbsFitter and is specific to one AbsTrackRep of the Track

- TrackPoint* trackPoint_
- AbsTrackRep* rep_
- SharedPlanePtr sharedPlane_

- AbsFitterInfo()
- AbsFitterInfo(TrackPoint*, AbsTrackRep*)

- MeasuredStateOnPlane& getFittedState(bool)
- MeasurementOnPlane getResidual(uint, bool, bool)
- TrackPoint* getTrackPoint()
- AbsTrackRep* getRep()
- void setTrackPoint(TrackPoint*)
- void setRep(AbsTrackRep*)
- SharedPlanePtr& getPlane()
- void setPlane(SharedPlanePtr&)

MeasurementFactory

- Factory object to create AbsMeasurement objects from digitised and clustered data

- The MeasurementFactory is used to automatically fill Track objects with hit data. For each detector type used on AbsMeasurement Producer has to be registered in the factory. The factory can then use the index information from a TrackCand object to load the indexed hits into the Track.

- map< int, AbsMeasurementProducer<measurement_T*>
hitProdMap_

- MeasurementFacotry()

- void addProducer(int, AbsMeasurementProducer<measurement_T>*)
- measurement_T* createOne(int, int, TrackCandHit*)
- vector<measurement_T*> createMany(TrackCand&)

MeasurementProducer

- Template class for a measurement producer module

- A MeasurementProducer module is used by MeasurementFactory to create Measurements for one specific detector type.

- It is assumed that each detector has as output of its digitisation / clustering some sort of hit or cluster class which stores all information that corresponds to a measured hit in that detector. The MeasurementProducer converts this information into a class that can be handled by gent. This class is realised as a Measurement (a class inheriting from AbsMeasurement).

- In order to use the MeasurementProducer facility, a Measurement has to implement a constructor which takes as an argument a pointer to the cluster class and a TrackCandHit. This constructor serves as the initialising constructor for the Measurement.

- The MeasurementProducer will fetch the cluster objects from a TClonesArray and use the initialising constructor to build the corresponding Measurement.

- TClonesArray* hitArrayTClones_

- MeasurementProducer(TClonesArray*)
- AbsMeasurement* produce (int index, TrackCandHit*)