Reconstruction of vertices and decayed particles with the AliKFParticle package

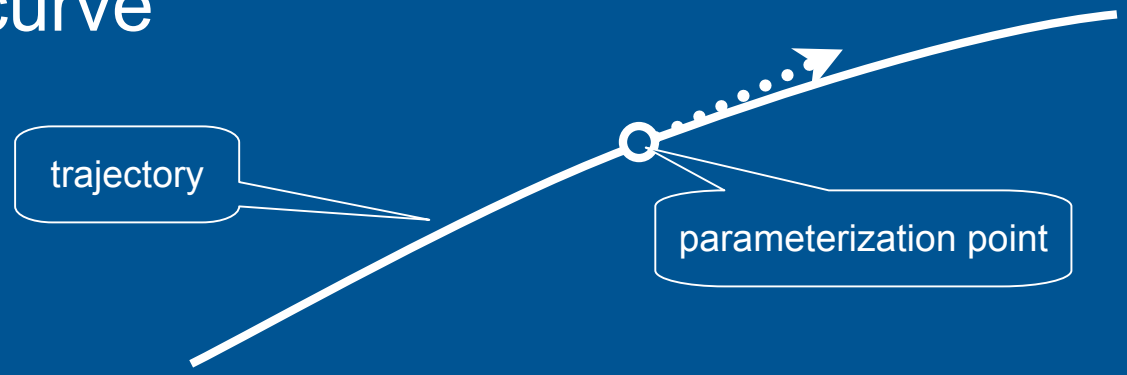
KIP

Sergey Gorbunov

GSI, November 25, 2008

KFParticle - a new object in analysis

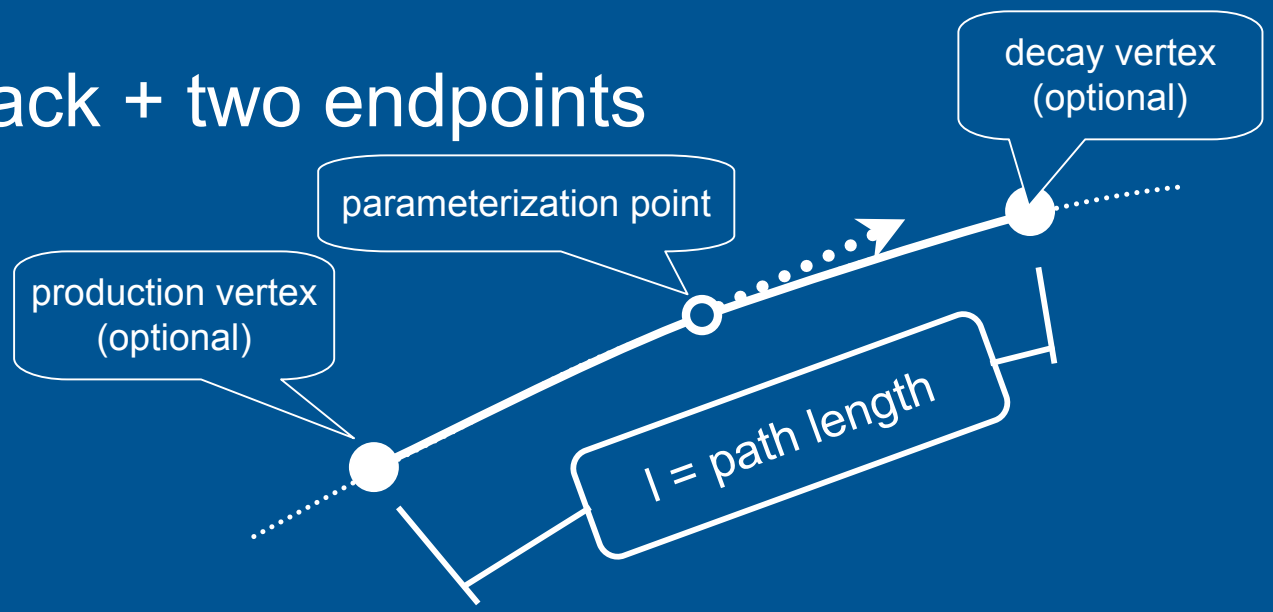
- Track: infinite curve



- KFParticle: track + two endpoints

KFParticle object

- KFParticle: track + two endpoints



Active parameters (state vector) : (XYZ, PxPyPz, E, s=l/p)

All possible parameters with their estimated errors,
incl. position and direction at both vertices,
can be extracted from the state vector

- KFVertex == KFParticle (with no direction)

KFParticle object

- To get the particle parameters somewhere first use transport methods:
  - TransportToDecayVertex()
  - TransportToProductionVertex()

  - TransportToPoint(xyz)
  - TransportToParticle(AliKFParticle)
  - TransportToVertex(AliKFVertex)
  - …
- And then getters:
  - GetX()…, GetPx()… GetMass(), GetDecayLength()…
  - GetVertex();

KFParticle construction

- Construction from tracks

```
AliKFParticle PiPlus( ESDPiPlus, 211 ), KaMinus( ESDKaMinus, 321 ), D0;
```

PiPlus,KaMinus are constructed w/o decay and production vertices.

- Two fitting ‘brick's - AddDaughter and SetProductionVertex

• Construct the particle at its decay vertex using its daughters:

```
D0.AddDaughter( PiPlus );
D0.AddDaughter( KaMinus );
```

D0 is constructed and its decay vertex is set. The same via constructor D0(PiPlus, KaMinus);

• Set the production vertex :

```
D0.SetProductionVertex( PrimVtx );
```

D0 production vertex is set.
Important: its parameters at the decay vertex are also improved with the new information.

KFParticle construction

- Extra fitting routine - SubtractFromVertex

• Exclude the particle from the vertex w/o full refitting the vertex :

```
PiPlus.SubtractFromVertex( PrimVtx );
```

or

```
PrimVtx -= PiPlus;
```

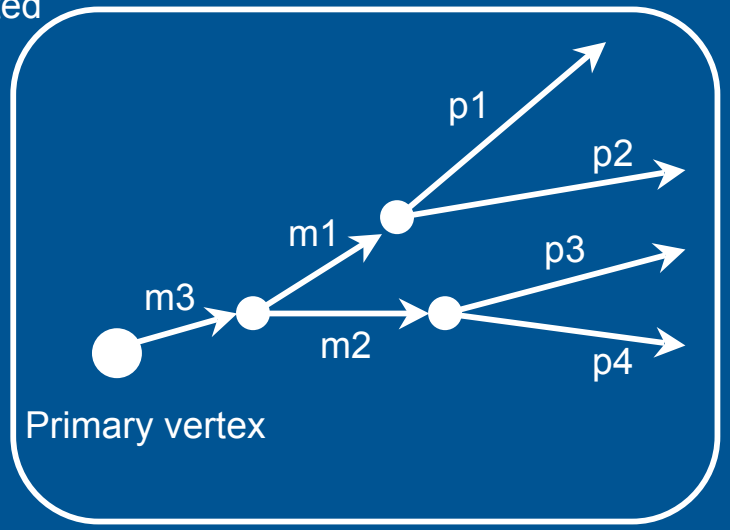
Important: excluding the secondary tracks from the primary vertex you improve the resolution and protect the constructed particles from the systematical bias in decay length.

Example of D0 construction

```
AliKFVertex PrimVtx( ESDPrimVtx);      // Primary vertex construction
AliKFParticle PiPlus( ESDPiPlus, 211 );        // Construction of daughters
AliKFParticle  KaMinus( ESDKaMinus, 321 ); //

AliKFParticle D0( PiPlus, KaMinus );     // Construction of D0 at its decay vertex

PrimVtx -= PiPlus;          // Subtract PiPlus, KaMinus from the primary vertex
PrimVtx -= KaMinus;     //    ( if they was was used in the ESDPrimVtx fit )

PrimVtx.AddDaughter( D0 );  // Improve the primary vertex. ( One can type “PrimVtx += D0;“ )

D0.SetProductionVertex( PrimVtx );  // D0 is now fully fitted

PiPlus.SetProductionVertex( D0 );      // PiPlus is now fully fitted
KaMinus.SetProductionVertex( D0 );  // KaMinus is now fully fitted
```

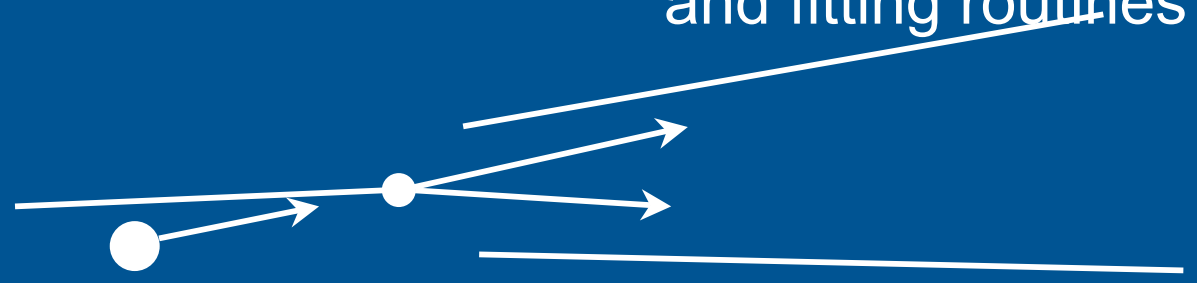
Example of fit of decay chain

```
AliKFVertex PrimVtx( ESDPrimVtx);                          // Set primary vertex
AliKFParticle p1( ESDp1, pdg1 ), p2( ESDp2, pdg2 ),        // Set daughters
              p3( ESDp3, pdg3 ), p4( ESDp4, pdg4 ) ;

AliKFParticle  m1( p1, p2 ), m2( p3, p4 ), m3( m1, m2 ); // Construct all mothers
PrimVtx += m3;  // Improve the primary vertex.
m3.SetProductionVertex( PrimVtx ); // m3 is fully fitted

m1.SetProductionVertex( m3 ); // m1 is fully fitted
m2.SetProductionVertex( m3 ); // m2 is fully fitted
p1.SetProductionVertex( m1 ); // p1 is fully fitted
p2.SetProductionVertex( m1 ); // p2 is fully fitted
p3.SetProductionVertex( m2 ); // p3 is fully fitted
p4.SetProductionVertex( m2 ); // p4 is fully fitted
```



Analysis with KFParticle

- Usual way of analysis: mess of tracks, vertices and fitting routines



Final objects:

- non-vertex fitted tacks (5 parameters)
- vertices (3)
- vertex fitted tracks (3)

Technical objects:

- secondary vertex(3)+mother(3)+daughters (3n)
  + mother at the primary vertex (3)
  + new primary vertex (3)
- decay chain: mess of daughters (3*n+3*n), mothers(3*n+3*n), secondary vertices(3n), primary vertex(3), set of mass constraints and the inheritance matrix.

- Analysis with KFParticle: only KFParticles with two operations

KFParticle: functionality

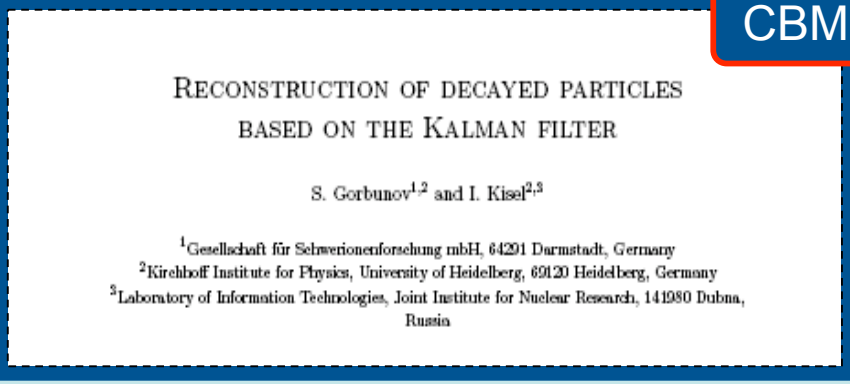
```
//* Initialisation from ALICE track, PID hypothesis can be provided
AliKFParticle( const AliExternalTrackParam &track, Int_t PID );
//* Construction of mother particle by its 2-3-4 daughters
AliKFParticle( const AliKFParticle &d1, const AliKFParticle &d2 );
AliKFParticle( const AliKFParticle &d1, const AliKFParticle &d2,
                                        const AliKFParticle &d3 );
AliKFParticle( const AliKFParticle &d1, const AliKFParticle &d2,
               const AliKFParticle &d3, const AliKFParticle &d4 );
//* Simple accessors
Double_t GetX    () const ; //* x of current position
Double_t GetY    () const ; //* y of current position
Double_t GetZ    () const ; //* z of current position
Double_t GetPx   () const ; //* x-compoment of 3-momentum
Double_t GetPy   () const ; //* y-compoment of 3-momentum
Double_t GetPz   () const ; //* z-compoment of 3-momentum
Double_t GetE    () const ; //* energy
Double_t GetS    () const ; //* decay length / momentum
Int_t    GetQ    () const ; //* charge
Double_t GetChi2 () const ; //* chi^2
Int_t    GetNDF  () const ; //* Number of Degrees of Freedom
Double_t GetParameter ( int i ) const ;
Double_t GetCovariance( int i ) const ;
Double_t GetCovariance( int i, int j ) const ;
//* Accessors with calculations( &value, &estimated sigma )
//* error flag returned (0 means no error during calculations)
Int_t GetMomentum    ( Double_t &P, Double_t &SigmaP ) const ;
Int_t GetMass        ( Double_t &M, Double_t &SigmaM ) const ;
Int_t GetDecayLength ( Double_t &L, Double_t &SigmaL ) const ;
Int_t GetLifeTime    ( Double_t &T, Double_t &SigmaT ) const ;
```

```
//* Add daughter to the particle
void AddDaughter( const AliKFParticle &Daughter );
//* Add daughter via += operator: ex.{ D0; D0+=Pion; D0+= Kaon; }
void operator +=( const AliKFParticle &Daughter );
//* Set production vertex
void SetProductionVertex( const AliKFParticle &Vtx );
//* Set hard mass constraint
void SetMassConstraint( Double_t Mass );
//* Everything in one go
void Construct( const AliKFParticle *vDaughters[], Int_t NDaughters,
                const AliKFParticle *ProdVtx=0, Double_t Mass=-1 );
//* Transport the particle to its decay vertex
void TransportToDecayVertex();
//* Transport the particle to its production vertex
void TransportToProductionVertex();
//* Transport the particle close to xyz[] point
void TransportToPoint( const Double_t xyz[] );
//* Transport the particle close to another particle p
void TransportToParticle( const AliKFParticle &p );
//* Calculate distance from another object [cm]
Double_t GetDistanceFromVertex( const Double_t vtx[] ) const ;
Double_t GetDistanceFromVertex( const AliKFParticle &Vtx ) const ;
Double_t GetDistanceFromVertex( const AliESDVertex &Vtx ) const ;
Double_t GetDistanceFromParticle( const AliKFParticle &p ) const ;
//* Calculate sqrt(Chi2/ndf) deviation from another object
Double_t GetDeviationFromVertex( const AliKFParticle &Vtx ) const ;
Double_t GetDeviationFromParticle( const AliKFParticle &p ) const ;
//* Subtract the particle from the vertex
void SubtractFromVertex( AliKFParticle &v ) const ;
```

Summary

- The Kalman filter mathematics was extended to perform direct estimation of decayed particle parameters
- The method provides physical parameters (momentum, energy and life time) of the particle in addition to its production and decay vertices.
- The method uses physical particle parameterization which is universal with respect to detector geometry
- Method has been implemented in CBM and ALICE experiments using user-friendly C++ interface.

CBM note



RECONSTRUCTION OF DECAYED PARTICLES
BASED ON THE KALMAN FILTER

S. Gorbunov[1,2] and I. Kisel[2,3]

[1]Gesellschaft für Schwerionenforschung mbH, 64291 Darmstadt, Germany
[2]Kirchhoff Institute for Physics, University of Heidelberg, 69120 Heidelberg, Germany
[3]Laboratory of Information Technologies, Joint Institute for Nuclear Research, 141980 Dubna, Russia