CMS Internal Note

The content of this note is intended for CMS internal use and distribution only

September 14, 2004

A Kinematic fit and a decay chain reconstruction library

K.Prokofiev, Th. Speer

Physik-Institut Universität Zürich

Abstract

A kinematic fitting and chain reconstruction library was developed and implemented in the framework of the CMS reconstruction program. Vertex reconstruction is done using Least Mean Squares minimization with Lagrange multipliers and Kalman filter techniques. Constrained refit of tracks and vertices is performed via Least Mean Squares minimization with Lagrange multipliers.

In this note the architecture and the functional components of the library are described. The reconstruction of the decay $B_s \to J/\Psi\Phi \to \mu^+\mu^- K^+K^-$ is used to illustrate the various techniques and their results. The reconstruction of the full decay tree is performed in three different ways with and without reconstruction of intermediate neutral particles.

1 Introduction

Kinematic fitting is the application of physical laws governing particle production or decay to improve experimental resolution of the measurements [1]. The kinematic fit package implemented in the CMS reconstruction program allows full decay chain reconstruction according to user-defined decay model. The full decay tree is reconstructed by fitting all tracks and vertices in current decay and application of user-defined constraints on their parameters. The B_s decay mentioned above will be used as an example through out this note. For instance, the set of constraints to apply during the reconstruction of the $B_s \to J/\Psi\Phi \to \mu^+\mu^- K^+K^-$ decay can be the following:

- All four final state tracks come from the B_s decay vertex, since intermediate J/Ψ and Φ mesons have insignificant lifetimes of the order of $\sim 10^{-22}$ and $\sim 10^{-21}$ s respectively [2].
- The invariant mass of two muons should be equal to the invariant mass of theJ/Ψ meson (3.096 GeV). This constraint is possible since the Breit-Wigner width of J/Ψ meson is small in comparison with its mass and momentum ($\Gamma_{J/\Psi} = 87$ keV [2], $p_\mu^{Tr} = 2 \div 4$ GeV).
- The B_s momentum vector reconstructed at the decay vertex should point towards the primary interaction vertex, since we can assume that the B_s meson was produced there.

2 Decay chain reconstruction

During the kinematic fit a decay is reconstructed from "bottom" to "top" i.e. from the final state (for example, tracks, reconstructed in the tracker) to mother state (the B_s meson in our example). In principle, any 4-vector-like object can serve as an input yo the kinematic fit. In the current implementation, only *RecTracks* can be used as input, however, interfaces for jets and muons are under development.

Since the CMS detector has no particle identification system (for example kaon and pion trajectories are indistinguishable in the tracker, muons can however be identified using muon chambers), a mass hypothesis should be assigned to the final state tracks before making any kinematic analysis. In case different competing hypotheses exist for the desired decay channel in an event (e.g. combinatorial problem is present), the total χ^2 of the fit can be used as a selection criteria.

The decay tree is described by the three following classes: *KinematicParticle*, *KinematicVertex* and *KinematicTree*. The first two store the information about particles and vertices, and the latter one stores the actual position of every particle and vertex in the decay and provides a navigation mechanism between these. Below we put a short description of the functionality and the *UML* diagrams showing actual structures of the classes are given in appendix A.

2.1 KinematicTree

The *KinematicTree* class describes a decay chain hypothesis. The class contains pointers to all particles and vertices forming the decay chain and allows the user to navigate between them.

Several different trees can be merged by constraining their top particles to the same vertex. A new tree object, containing all available information will then be created. Diagrams illustrating the *KinematicTree*, its concept and functionalities are given in appendix B, along with a brief example of usage.

2.2 KinematicParticle

This class represents a particle in a decay chain. The class is designed to store the parameters describing the particle (reconstructed trajectory state and assigned or reconstructed mass), the corresponding covariance matrix, the charge, the χ^2 and the number of degrees of freedom assigned to the current particle during previous stages of the reconstruction. Every object of the class also stores the last constraint applied and the state of the particle before that constraint had been applied.

A *KinematicParticle* can be created by assigning a mass hypothesis to a reconstructed trajectory using the provided *KinematicParticleFactory*. It is also created by the kinematic particle vertex fitters when fitting several trajectories to the same vertex and calculating the decayed state.

2.2.1 Note about particle parametrization and *KinematicState* class

In the present version of the library, a particle state is described by 7 parameters (quasi-cartesian parametrization): a reference position in the global frame (x, y, z) , the momentum at this point (p_x, p_y, p_z) and the mass of the particle m (calculated as a result of previous fits or hypothesis assigned to the final state track). This parametrization should be exclusively used when deriving constraint equations and implementing the corresponding classes. The vector of 7 parameters and their joint (7×7) covariance matrix are cached in the *KinematicState* class.

In case the particle is created by a *KinematicParticleFactory*, a negligible error should be assigned to the mass hypothesis in order to fill the corresponding diagonal element of the covariance matrix and avoid singularities.

Every *KinematicParticle* contains a set of references to different *KinematicStates*. These can be accessed via corresponding public methods (see appendix A for reference). An extended helix "perigee" parametrization is also provided. The first five parameters $(\rho, \theta, \phi, \varepsilon, z_{tr})$ are identical to the usual perigee parameters [3] and the sixth parameter is the particle mass. The vector of parameters and the corresponding covariance matrix are stored in the *PerigeeKinematicState* class. In the present version, this parametrization can not be used for the fitting itself (constraint equations etc), it may however be used in a subsequent analysis. The conversion between the different parametrizations is done using the *KinematicPerigeeConversions* class.

Both parametrizations can be used for either neutral or charged particles. For neutral particles the trajectory is a straight line. In the extended perigee parametrization, the transverse curvature is replaced with the inverse of the transverse momentum of the particle $(1/p_T)$.

The next version of the fit will be parametrization-independent, such that any parametrization can be used to describe the trajectory and write the constraint equations. Indeed, the analytical solution of a Least Mean Squared minimization does not depend on the parametrization used to describe the input data [1], [5], [6] (see section 3).

2.3 KinematicVertex

This class stores the vertex information needed for the kinematic fit: position, covariance matrix, χ^2 and number of degrees of freedom. As the *KinematicParticle* class, *KinematicVertex* contains references to its previous states. A *KinematicVertex* is always produced as a result of a vertex fit of the set of *KinematicParticles*.

3 Kinematic fitting

Kinematic fitting deals with introducing constraints (physics laws) into minimization problems in order to improve experimental measurements, test hypothesis and to find unknown parameters. The mathematical approach based on Least Mean Squared minimization (LMS) is presently the most widely used technique for constrained fitting. Sequential methods such as the Kalman filter, and global methods (global minimization with Lagrange multipliers, global minimization with penalty functions, etc.) are the most common methods in high energy physics experiments.

Sequential algorithms are in general more flexible, allowing step by step update of estimated quantities, and are significantly faster than global ones, reducing CPU time by factorizing the inversion of covariance matrices [7]. However, only a limited number of problems can be solved using this approach, as it requires the initial data to be uncorrelated (joint measurement covariance matrix being block diagonal), and the analytical solution is different for each different constraint. In event reconstruction, sequential methods are widely used for track and vertex fitting.

The mathematical technique most widely used in kinematic fitting is the Least Means Squared minimization (LMS) with Lagrange multipliers. This method allows to constrain certain parameters to precise values ("hard" constraint: particle has a given mass, given momentum vector, etc...). Soft, or inequality constraints (particle mass lies in a certain region, is distributed with a given width, etc...), can be imposed using other methods, such as LMS with penalty functions [6].

One of the advantages of the Lagrange multipliers method is that,when the constraint equations are linear, the minimization problem can be solved analytically. In addition, most non-linear constraints can be linearized (first order Taylor expansion, for example). The minimization algorithm becomes then independent of the particular constraint equations and can be implemented as a standalone software package, where the choice of the constraint equations and linearization points is left to the user.

3.1 LMS with Lagrange multipliers

The principle of fitting data with given constraints using the LMS with Lagrange multipliers technique is relatively simple. Consider a set of n experimental points $y = y_1, y_2, \ldots, y_n$ with their associated error matrices $V_y = (V_1, \ldots, V_n)$, subject to the set of k constraint equations $H(y) = 0$ (of the form $H_1(y) = 0, \ldots, H_k(y) = 0$). In the general case, these equations are nonlinear. If the LMS minimization method is selected to find the set of refitted parameters y^{ref}, the following minimization procedure is performed:

$$\chi^2 = (y^{ref} - y)V_y^{-1}(y^{ref} - y)^T \rightarrow min, \tag{1}$$

with the constraint equations:

$$H(y^{ref}) = 0. \tag{2}$$

In order to obtain the analytical solution for the problem, we linearize (a first-order Taylor expansion) our constraint equations around some given point y_{exp} and the following linear form is obtained:

$$\frac{\partial H(y_{exp})}{\partial y}(y - y_{exp}) + H(y_{exp}) = D\delta y + d = 0. \tag{3}$$

Multiplying the linearized constraint equation by some unknown factor λ, which is the Lagrange multiplier [6], and adding both equations, the following function is to be minimized [4]:

$$\chi^2 = (y^{ref} - y)V_y^{-1}(y^{ref} - y)^T + 2\lambda^T(D\delta y + d) \rightarrow min. \tag{4}$$

By minimizing this function with respect to y^{ref} and λ, the refitted parameters and their joint covariance matrix are obtained. The problem has a unique solution (y^{ref}, λ) [1].

If the desired precision of the fit is not achieved (e.g constraint equation is still far from satisfaction: deviation of $H(y^{ref})$ from zero is more than desired value), further fitting iterations can be performed, taking each time the result of the previous step as the linearization point for the next one. Iterations will be performed until the maximum number of iterations is performed or the residuals of the constraint equations are below the certain value.

If the set of constraint equations is linear with respect to the parameters to refit, an exact solution, satisfying $H(y^{ref}) = 0$, is obtained after the first iteration. When the constraint equations are highly nonlinear, the choice of linearization point is crucial. Therefore, for the fit to be successful, the initial point should be chosen in the vicinity of the probable solution and the constraint equations should be well approximated with their linear version in this region.

It may be worth to note that some constraints may be nonlinear in one parametrization and become linear in another. It is also an interesting fact that the initial parametrization of the input data does not play any role in the formalism described above, The input data and the constraint equations must be defined in the same reference frame, but the user can choose the frame which is the most convenient for the problem at hand (e.g. the frame in which the constraint equations are linear). A brief example, illustrating this idea is presented in the next section.

In the current (**ORCA 8**-based) version of the Kinematic Fit package, all the constraints should be implemented in the 7-parameters quasi-cartesian frame (see above). The release of a parametrization-independent version of the package is however foreseen in the near future.

A more complete mathematical description of the methods used in our package can be found in [1], [4], [5], [6] and [7]. The actual algorithms for kinematic fitting are implemented in the *KinematicFit* library.

3.2 Kinematic fit strategies and factorization of constraints

Two different strategies can be used when reconstructing a decay chain using the Kinematic fit. The most intuitive way is to refit the input data with all the constraints and to find all unknown parameters in a single fit (*Global strategy*). For example, the parameters of the decayed B_s meson can be calculated after a single fit where the two muon and two kaon tracks are constrained to come from a single vertex, and requiring the invariant mass of the two muons to be equal to the mass of the J/Ψ and the sum of the momenta to point towards the primary vertex.

However, a different strategy can be used. The two final state muons can first be fit to a vertex, reconstructing thus the J/Ψ parameters at this vertex, then constraining the mass of this intermediary state to be equal to the mass of the J/Ψ. After that, the J/Ψ and the 2 kaons would be fit to a vertex, reconstructing the B_s parameters at this

vertex, and the pointing constraint would finally be applied on the B_s. Here the application of the set of constraints in a single fit is substituted with a series of individual fits, applying the constraints one by one on individual, reconstructed particles (*Sequential strategy*).

This approach is possible due to a remarkable property of the LMS-based algorithms: the result of simultaneous application of the set of constraints is mathematically equivalent to their sequential application. The result of a global fit with a set of constraints is therefore equal to the result of a series of sequential fits, where each constraint is applied individually [1], [11].

The usage of the sequential fits becomes especially important when working with unstable particles with significant lifetimes, were reconstructed state are to be propagated inside the detector. This method has nevertheless one drawback. As the fit proceeds from the final state to the initial state by applying different constraints on the different decays on the chain in succession, only the initial state (e.g the B_s in our example) will reflect all constraints applied throughout the fit. Propagating this knowledge down the decay chain to the final states and refitting these with the full set of constraints is not straightforward and has not been implemented yet.

In order to use the advantages of both the global and the sequential strategies, different mathematical algorithms are implemented. For the global fit, a vertex fit with simultaneous application of additional constraints is implemented. For the Sequential fit, where the constraints are applied to an already reconstructed particle, a particle refit (for single or multiple particles) without vertex reconstruction is done. The application of both strategies to the reconstruction of the B_s decay will be discussed later.

3.3 Particle parameters refit without vertexing

If no vertex reconstruction is required, the refit of a set of N^{track} particles is straightforward in the formalism presented above. As the quasi-cartesian parametrization is used, with 7 parameters per track $(x, y, z, p_x, p_y, p_z, m)$, the input vector of parameters will have dimension $7 \cdot N^{track}$. The covariance matrices of each track (of dimension (7×7) can come from previous fits or be created by a *KinematicParticleFactory*. The same formalism can be used to refit a single particle.

The particles to refit are supposed to be uncorrelated, and their joint $(7 \cdot N^{track} \times 7 \cdot N^{track})$ covariance matrix is supposed to be block diagonal. In principle, the mathematical approach presented above does not distinguish between correlated and uncorrelated measurements and the framework can easily be extended to work with correlated particles if needed.

Once refitted, all parameters of all tracks will be correlated, i.e. their joint covariance matrix will not be block-diagonal anymore. However, the current version of the framework only stores the individual track covariance matrices and does not keep the cross-correlation terms yet.

As it was briefly mentioned in the previous section, the global LMS minimization algorithm behaves differently when applying constraints of linear and nonlinear type.

3.3.1 Linear constraints

For example, constraining the transverse momentum of the single particle results in the following set of the linear equations (it has no physical meaning, but it is a good demonstration of the principle):

$$H_1(y) = p_x - p_x^{fix} = 0,$$
$$H_2(y) = p_y - p_y^{fix} = 0 \tag{5}$$

where p_x and p_y are respectively the fourth and fifth parameters of the vector.

The matrix of derivatives D and the vector of values d taken at some expansion point y^{exp} will have the following form:

$$D = \frac{\partial H(y^{exp})}{\partial y} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \tag{6}$$

$$d = H(y^{exp}) = \begin{pmatrix} H_1(y^{exp}) \\ H_2(y^{exp}) \end{pmatrix} = \begin{pmatrix} p_x^{exp} - p_x^{fix} \\ p_y^{exp} - p_y^{fix} \end{pmatrix} \tag{7}$$

Here, the D matrix has two lines, since there are two equations, and 7 columns, as the 7-parameter frame was chosen. The vector of values d consist of two elements.

For each set of constraint equations, the computation of the matrix of derivatives and the vector of values has to be implemented in a class derived from the *KinematicConstraint* class (see appendix A). For example, for refitting the transverse momentum, a *TransverseMomentumKinematicConstraint* class, inheriting from *KinematicConstraint*, has to be written. This class should return the calculated matrix of derivatives and vector of values at a given expansion point: 7-dimensional in case of single particle refit, or $7 \times N_{track}$- dimensional in case of multiple track refit (without simultaneous vertex reconstruction). Instead of the expansion point, the vector of *KinematicParticles* can be passed to the constraint class. The parameters of the particles will then be used as expansion point. The *numberOfEquations* method, which indicates the number of equations in the particular constraint, would return a value of two in this case. Examples of the of the implementation of *KinematicConstraint* classes, can be found in *Vertex/KinematicFitPrimitives/src* directory of ORCA.

Two linear constraints, to be used without vertex reconstruction, are now implemented: (*MassKinematicConstraint* and *MomentumKinematicConstraint*). Both of them are designed to refit only one particle at the time. In case of need, the multi particle version can be implemented by the user without much effort. The implementation of the kinematic constraints, refitting several particles at the time, without vertex reconstruction, such as collinearity or back-to-back constraints, is foreseen for in future.

3.3.2 Nonlinear constraints

As it was mentioned before, care should be taken when nonlinear constraints are to be implemented in the framework. We will illustrate this with the example of the "pointing" constraint. It was specially designed to include the topological information about B_s-meson production and decay into the analysis in order to improve the resolution of the position of the secondary vertex and of the momentum components. In this constraint, the vector pointing from the primary to the secondary vertex and the reconstructed momentum vector of the B_s are required to be parallel. This goal may be achieved via two different ways: either the secondary vertex is shifted, aligning the momentum vector with the direction to the primary vertex, or the momentum vector itself is rotated. If the constraint equations allow for both possibilities, the LMS based algorithm finds an optimum between the two, using partly each of them. The sketches on Fig. 1 and 2 illustrate this idea. Naming the angles under which the secondary

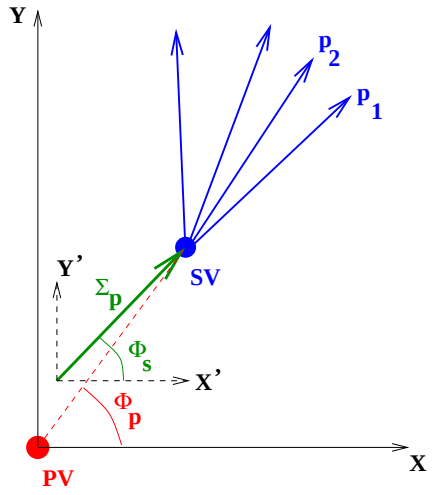

Figure 1: The B_s momentum and decay vertex reconstructed from decay products (transverse plane). The momentum vector is not aligned with primary-secondary vertex direction.

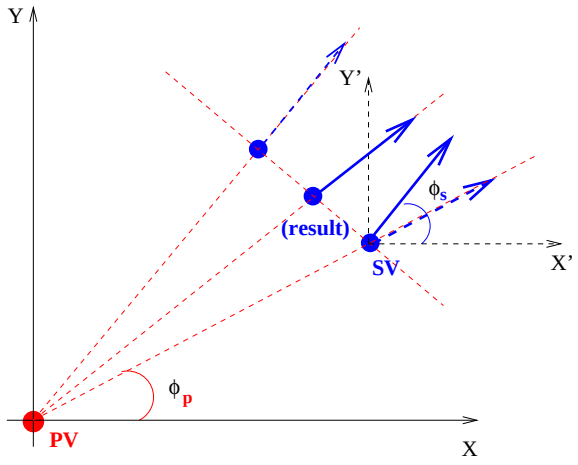

Figure 2: Possible solutions for the alignment problem. **Top:** vertex shift along direction perpendicular to the primary-secondary vertex vector. **Bottom:** rotation of the momentum vector. **Middle:** solution taken by LMS minimization, partly incorporating both possibilities.

vertex is seen from the primary vertex (ϕ_p, θ_p) and the angles of the B_s momentum at the secondary vertex (ϕ_s, θ_s) (all angles are defined in the global frame), the following (most probably not optimal) set of equations is proposed

to fulfill our requirements: [1)]

$$\tan\frac{\phi_p}{2}-\tan\frac{\phi_s}{2}=\frac{1-\cos\phi_p}{\sin\phi_p}-\frac{1-\cos\phi_s}{\sin\phi_s}=\frac{\sin\frac{\phi_p-\phi_s}{2}}{\cos\frac{\phi_p}{2}\cos\frac{\phi_s}{2}}=0 \tag{8}$$

This equation has a root at $\phi_p-\phi_s=0+2\pi\cdot k$ and singular points in ϕ_p or $\phi_s=\pi+2\pi\cdot k$, where k is a positive integer. This equation works well on the area of definition of the azimuthal angle, except for the singular points. When both ϕ_p and ϕ_s are close to π, the value of the equation is close to 0. The situations when one of the angles is equal to π and the other one is far away should be excluded. These events are very rare and represent no physical interest, and since the total χ^2 of this fit is very large, these events will be rejected by the χ^2 probability cut. Similar equation is obtained for the polar angle. In the quasi-cartesian frame, it is easy to rewrite our constraint equations in terms of the momenta of the particle and coordinates of the vertices:

$$H_\phi=\frac{1-\frac{dx}{\sqrt{dx^2+dy^2}}}{\frac{dy}{\sqrt{dx^2+dy^2}}}-\frac{1-\frac{p_x}{\sqrt{p_x^2+p_y^2}}}{\frac{p_y}{\sqrt{p_x^2+p_y^2}}}=0,$$

$$H_\theta=\frac{1-\frac{\sqrt{dx^2+dy^2}}{\sqrt{dx^2+dy^2+dz^2}}}{\frac{dz}{\sqrt{dx^2+dy^2+dz^2}}}-\frac{1-\frac{\sqrt{p_x^2+p_y^2}}{\sqrt{p_x^2+p_y^2+p_z^2}}}{\frac{p_z}{\sqrt{p_x^2+p_y^2+p_z^2}}}=0 \tag{9}$$

where $(x_s,y_s,z_s,p_x,p_y,p_z,m)$ are the parameters of the particle at the secondary vertex, (x_p,y_p,z_p) is the position of the primary vertex and (d_x,d_y,d_z) is the vector pointing from the primary to the secondary vertex ($dx=x_s-x_p$, $dy=y_s-y_p$, $dz=z_s-z_p$).

The matrix of derivatives is obtained by taking the partial derivatives of both the polar and the azimuthal equations with respect to the 7 parameters of the particle:

$$\frac{\partial H_\phi}{\partial x_s}=\frac{dx}{dy\sqrt{dx^2+dy^2}}-\frac{1}{dy} \qquad \frac{\partial H_\phi}{\partial y_s}=\frac{1}{\sqrt{dx^2+dy^2}}-\frac{\sqrt{dx^2+dy^2}}{dy^2}+\frac{dx}{dy^2}$$

$$\frac{\partial H_\phi}{\partial z_s}=0 \qquad \frac{\partial H_\phi}{\partial p_x}=-\left(\frac{p_x}{p_y\sqrt{p_x^2+p_y^2}}-\frac{1}{p_y}\right)$$

$$\frac{\partial H_\phi}{\partial p_y}=-\left(\frac{1}{\sqrt{p_x^2+p_y^2}}-\frac{\sqrt{p_x^2+p_y^2}}{p_y^2}+\frac{p_x}{p_y^2}\right) \qquad \frac{\partial H_\phi}{\partial p_z}=0$$

$$\frac{\partial H_\phi}{\partial m}=0 \qquad \frac{\partial H_\theta}{\partial x_s}=\frac{dx}{dz}\left(\frac{1}{\sqrt{dx^2+dy^2+dz^2}}-\frac{1}{\sqrt{dx^2+dy^2}}\right)$$

$$\frac{\partial H_\theta}{\partial y_s}=\frac{dy}{dz}\left(\frac{1}{\sqrt{dx^2+dy^2+dz^2}}-\frac{1}{\sqrt{dx^2+dy^2}}\right) \qquad \frac{\partial H_\theta}{\partial z_s}=\frac{\sqrt{dx^2+dy^2}-\sqrt{dx^2+dy^2+dz^2}}{dz^2}+\frac{1}{\sqrt{dx^2+dy^2+dz^2}}$$

$$\frac{\partial H_\theta}{\partial p_x}=-\frac{p_x}{p_z}\left(\frac{1}{\sqrt{p_x^2+p_y^2+p_z^2}}-\frac{1}{\sqrt{p_x^2+p_y^2}}\right) \qquad \frac{\partial H_\theta}{\partial p_y}=-\frac{p_y}{p_z}\left(\frac{1}{\sqrt{p_x^2+p_y^2+p_z^2}}-\frac{1}{\sqrt{p_x^2+p_y^2}}\right)$$

$$\frac{\partial H_\theta}{\partial p_z}=-\left(\frac{\sqrt{p_x^2+p_y^2}-\sqrt{p_x^2+p_y^2+p_z^2}}{p_z^2}+\frac{1}{\sqrt{p_x^2+p_y^2+p_z^2}}\right) \qquad \frac{\partial H_\theta}{\partial m}=0$$

The application of this constraint is discussed in the next section.

It seems to be obvious that the quasi-Cartesian parametrization is not the best choice for the derivation of an angular-dependent constraints. It would be more natural in this case to select a parametrization where the constraint equations become linear with respect to the angles (extended perigee, for instance) and do not have singular points. The development a frame-independent version of the library should simplify this task.

3.4 Vertex fitting and calculation of decayed particle parameters

The Kinematic fit allows the user to calculate the parameters of the decayed particle and their covariance matrix from the parameters and covariance matrices of decay products. This calculation is performed via the following

[1)] The equation 8 was chosen because of the simplicity of calculation of the derivatives in the quasi-cartesian frame. Taking partial derivatives of the equation $\phi_p-\phi_s=0$ in this frame would require a much more complex mathematical apparatus.

steps: reconstruction of the decay vertex from decay products, refit of the parameters of the decay products with the vertex constraint, calculation of the full refitted covariance matrix of all the refitted parameters, calculation of the parameters of the decayed particle and its covariance matrix. The calculation of the momentum of decayed particle is done by summing the momenta of refitted decay products at the vertex position. The covariance matrix of the new particle is calculated from the full particle-to-particle covariance matrix taking all correlations into account. A set of Jacobians is developed to calculate each block of the covariance matrix (position, momentum and position-momentum correlation) and the mathematical technique is in general identical to one used in [4].

There are two vertex reconstruction algorithms used so far in the Kinematic fit package: a Kalman filter and a global LMS minimization with Lagrange multipliers. The description of the Kalman filter can be found in [3]. The Kinematic fit package uses the advantage of the Kalman filter to be independent of the frame the input data is defined in and provides an easy way to access this technique via a *KinematicParticleVertexFitter* class. The starting point is the set of *KinematicParticles* (some of which can be the top particles of previously reconstructed trees) and a new *KinematicTree* is produced as a result of the fit. This tree contains the reconstructed particle, its decay vertex, all the refitted states of the decay products. The refitted states of subsequent decays are not refitted with this vertex constraint. The mathematical aspects of such a refit are still the subject of discussions.

The reconstruction of the vertex from a set of tracks is also possible via the LMS with Lagrange multipliers method. The number of parameters is then increased by three coordinates of the vertex position (x_v, y_v, z_v) and a set of special constraints has to be implemented. The total number of parameters becomes then $3 + 7 \cdot N^{track}$ and the joint covariance matrix of all the measurements has a dimension of $(3 + 7 \cdot N^{track}) \times (3 + 7 \cdot N^{track})$. As the coordinates of the vertex are not known apriori, a reasonable guess has to be taken. This can be the point of closest approach of the tracks, calculated by a *LinearizationPointFinder*. As the elements of the covariance matrix pertaining to the vertex coordinates are also unknown, it is natural to use very large values, as suggested in [6] and [7] and successfully implemented in [3]. As a nearly infinite error or negligible weight is assigned to the apriori knowledge, this technique is called the "Huge error method" [1].

The vertex constraint equations (2 per particle) force the trajectories of the particles to pass trough an unknown point. These equations can be derived by writing the trajectory parameters in helix parametrization (perigee [3] for example) and setting both transverse and impact parameters with respect to the unknown reference point to zero. The equations of motion of charged particle in the magnetic field should be used. A brief description of this procedure can be found in [1] and [8][2)].

As the resulting constraint equations are highly nonlinear, the iterative procedure described in section 3.1 should be applied. The LMS minimization with Lagrange multipliers mechanism for vertex fitting proposed in [1] is mathematically identical to the Kalman filter approach proposed in [7]. Both algorithms exist in **ORCA** and, as expected, show the same result in terms of primary and secondary vertex resolution. Despite the possibility of adding arbitrary constraint into the vertex fitting problem, LMS minimization with Lagrange multipliers has also some serious disadvantages, as will be outlined below.

3.4.1 Note about covariance matrix calculation

The calculation of the full covariance matrix for the formalism described in section 3.1 can be done using the standard error propagation method [1]:

$$V_y^{new} = V_y^{old} - V_y^{old} D^T (D V_y^{old} D^T)^{-1} D V_y^{old}, \tag{10}$$

where V^{old} is initial covariance matrix. The two matrices that are subtracted are of the same order of magnitude. When using the huge error method, briefly described in this section, the first three diagonal elements, representing the vertex coordinates, are typically in the order of 10^6 and the difference of the elements might be smaller than 10^{-10}. Due to the numerical precision, this can lead to significant errors in the calculations of the covariance matrix. Moreover, the precision of the algorithm becomes dependent on the value of the initial error. While the idea of the huge error method is to introduce an uncertainty on the first hypothesis of the unknown parameters as large as possible in order not to bias the fit, too large error terms might create numerical instabilities [3)].

[2)] For neutral particles, the derivation of the constraint equations is similar, as a straight line has to be used instead of a helix for the equations of motion of the particle. The parametrization has also to be changed to use the inverse transverse momentum $1/p_T$ instead of the transverse curvature ρ.

[3)] This problem is relevant for all the algorithms using the LMS minimization with Lagrange multipliers with the current mathematical approach. However, in the case of the refit of particles without vertex reconstruction the initial errors comes from the track reconstruction and are small enough not to suffer from this problem.

The Kalman filter [3] used when only the vertex fit is performed, does not suffer from this problem, as it is formulated as an information filter [7], where weight matrices are used instead of covariance matrices and the fit is performed iteratively. However, not every constrained minimization problem can be formulated in terms of the information filter, since it can not be derived for an arbitrary constraint.

Using the weight matrices instead of the covariance matrices, such that instead of subtracting errors, weights are added, is not straightforward. Indeed, combinations of V_y^{old} and of the D matrices, which are arbitrary due to the possibility of using different constraints, are not protected from being singular.

In order to avoid this problem, an adequate choice has to made for initial covariance matrix according, to the experimental conditions and the constraints imposed. For instance, in a pure vertex fit, initial diagonal elements of $\sigma^2_{init} = 10^4$ cm ^{2} do not seem to influence the fit in any way, and the error components returned are the same as for Kalman filter with $\sigma^2_{init} = 10^6$ cm ^{2}.

3.5 Application of constraints during vertex reconstruction

In a global fit, the application of the constraints during vertex reconstruction is possible using LMS with Lagrange multipliers by adding additional lines into the D matrix of derivatives. The vertex constraint introduces $2 \cdot N^{track}$ lines into the D matrix (two equations per track: longitudinal and transverse impact parameters), and N^{add} additional lines are added for the constraints. The size of vector of values changes accordingly. The fit is then identical to the procedure described in the section 3.4.

The fit is performed by the *KinematicConstrainedVertexFitter* class. The user should provide the pointer to the constraint object together with the input vector of particles. This constraint class should be inherited from the *MultiTrackKinematicConstraint* class, which is implemented in *KinematicFitPrimitives* library. An example of constraint and its implementation is given in the following section.

3.5.1 Two track mass constraint

This constraint requires the first two tracks form the set to have a given invariant mass. Both the parameters of the input particles and their vertex will be changed by the constraint. The actual constraint equation is derived using the equations of motion of charged particles in a magnetic field. This constraint can also be derived for neutral particles if needed.

Suppose we have two reconstructed particles with 7-parameters vectors

$$q_1 = (x_1, y_1, z_1, p_{1z}, p_{1y}, p_{1z}, m_1) \tag{11}$$

and

$$q_2 = (x_2, y_2, z_2, p_{2z}, p_{2y}, p_{2z}, m_2) \tag{12}$$

and charges q_1 and q_2, which we want to constraint to have a squared invariant mass M^2. Since these are charged particles in a magnetic field, all operations have to be performed at their common vertex (x^v, y^v, z^v).

The constraint equation is simply an energy conservation law:

$$H_1 = E^2 - \vec{P}^2 - M^2 \tag{13}$$

where E is the sum of the energies ($E = e_1 + e_2 = \sqrt{\vec{p_1}^2 + m_1^2} + \sqrt{\vec{p_2}^2 + m_2^2}$) and $\vec{P}$ is the sum of the momenta at the vertex ($\vec{P} = \vec{p_{1v}} + \vec{p_{2v}}$). For the calculation of the total momentum the equations of motion in the solenoidal magnetic field are needed [8]:

$$\begin{aligned} p_x^v &= p_x - a(y_v - y) \\ p_y^v &= p_y + a(x_v - x) \\ p_z^v &= p_z \end{aligned} \tag{14}$$

where $a = -0.29979Bq$ is the factor relating the transverse momentum of the charged particle to the transverse curvature of its trajectory, B is the z-component of the magnetic field in Tesla, and q is the charge of the particle. Taking the partial derivatives of equation 13 with respect to the vertex position and the parameters of the particles, the line to add to the D matrix is obtained:

$$\frac{\partial H_1}{\partial x^v} = -2(p^v_{1y} + p^v_{2y})(a_1 + a_2) \qquad \frac{\partial H_1}{\partial y^v} = 2(p^v_{1x} + p^v_{2x})(a_1 + a_2)$$
$$\frac{\partial H_1}{\partial z^v} = 0 \qquad \frac{\partial H_1}{\partial x_1} = 2a_1(p^v_{2y} + p^v_{1y})$$
$$\frac{\partial H_1}{\partial y_1} = -2a_1(p^v_{1x} + p^v_{2x}) \qquad \frac{\partial H_1}{\partial z_1} = 0$$
$$\frac{\partial H_1}{\partial p_{1x}} = 2(1 + \frac{e_2}{e_1})p_{1x} - 2(p^v_{1x} + p^v_{2x}) \qquad \frac{\partial H_1}{\partial p_{1y}} = 2(1 + \frac{e_2}{e_1})p_{1y} - 2(p^v_{1y} + p^v_{2y})$$
$$\frac{\partial H_1}{\partial p_{1z}} = 2(1 + \frac{e_2}{e_1})p_{1z} - 2(p^v_{1z} + p^v_{2z}) \qquad \frac{\partial H_1}{m_1} = 2m_1(1 + \frac{e_2}{e_1})$$

and similarly for the second particle. Since the constraint applies only on the first two particles, all other elements of this line have 0 values. Only the first 17 elements are changed ($3 + 2 \cdot 7$). The vector of values d should be extended by 1 element. The $TwoTrackMassConstraint$ class from the ***KinematicFit*** library contains the implementation of this algorithm.

4 Example of B_s decay reconstruction

In this section we present possible strategies for the reconstruction of the decay $B^0_s \to J/\Psi\Phi \to \mu^+\mu^- K^+K^-$ using the Kinematic fit. The most important parameters for a further analysis are the mass of the reconstructed B_s and the position of its decay vertex. All the plots in this section are produced with **ORCA8.1.3** on a sample of 1000 signal events. On the generator level, the P_t of the final state muon and kaon tracks are required to be above 2.0 GeV/c and 0.5 GeV/c respectively. No further selection cuts are applied. Only signal tracks are used by associating the reconstructed tracks to the Monte Carlo particles. For the primary vertex, needed for the pointing constraint, the true position is taken from the Monte Carlo information.

Three strategies are compared:

- **Kalman filter:** The first strategy is the simplest: it includes only the reconstruction of the vertex using the Kalman filter and the parameters of the B_s are calculated at the secondary vertex from the refitted final state tracks. No further constraint is applied. The results of this algorithm will be used as a baseline for comparison throughout the rest of this section.
- **Global strategy:** The parameters of the decayed B_s meson are calculated after fitting the two muon and two kaon tracks to the same vertex, requiring the invariant mass of the two muons to be equal to the mass of the J/Ψ, in one global fit. The pointing constraint is then applied on the B_s.
- **Sequential strategy with one neutral particle propagation:** In the sequential fit the two final state muons are fit to a vertex, reconstructing thus the J/Ψ parameters at this vertex, and the mass of this intermediary state is constrained to be equal to the mass of the J/Ψ. The J/Ψ and the 2 kaons are then fitted to a vertex, and the B_s parameters at this vertex are calculated. Finally, the pointing constraint is applied on the B_s.

4.1 Kalman filter

On fig. 3 the distribution of the residuals of the invariant mass of the reconstructed B_s meson is shown ($M^{rec}_{B_s} - M^{PDG}_{B_s}$). This distribution is influenced by three factors only: the resolution of the detectors, the resolution of the track reconstruction algorithm and the resolution of the vertex reconstruction algorithm. When fitting this distribution with a Gaussian, the mean is approximately 15 MeV higher than the world-average mass of the B_s, and the standard deviation (which we quote as the resolution), is about 32 MeV.

It is believed that this effect arises from inhomogeneities of the magnetic field not properly taken into account during the track reconstruction. This effect is seen on the distribution of the residuals of the invariant mass of the J/Ψ. On fig. 4 and 5 the distribution of the residuals of the invariant mass of J/Ψ and Φ are presented. The mean of the distribution of the J/Ψ mass is some 10 MeV higher than the world-average, whereas the Φ mass distribution is well centered.

On fig. 6 the distribution of the pulls of the B_s mass ($\frac{M^{rec}_{B_s} - M^{PDG}_{B_s}}{\sigma^{rec}}$) is presented. The distribution is well modeled by a Gaussian with a standard deviation of 1.5, the uncertainty on the mass of the reconstructed B_s being underestimated [7]. As can be seen on fig. 7, the uncertainties on the transverse curvature of the reconstructed muons, estimated by ***CombiantorialTrackFinder*** are underestimated. The distributions of the residuals and pulls of the longitudinal and transverse coordinates of the reconstructed decay vertex are presented on fig. 9 and 10. The resolutions of the transverse and longitudinal coordinates are of 47 and 62 μm respectively and the errors are reasonably well estimated (see the pulls distributions on fig. 11 and 12).

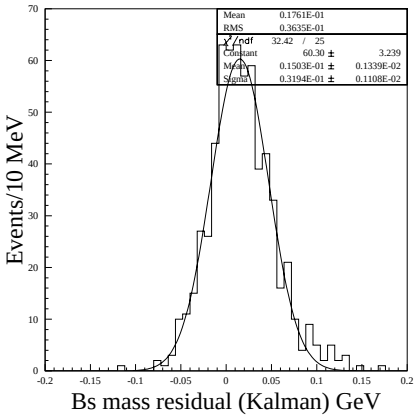

Figure 3: Distribution of the B_s mass residuals for the Kalman filter

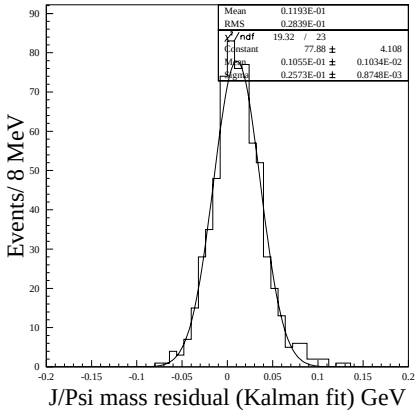

Figure 4: Distribution of the J/Ψ mass residuals for the Kalman filter.

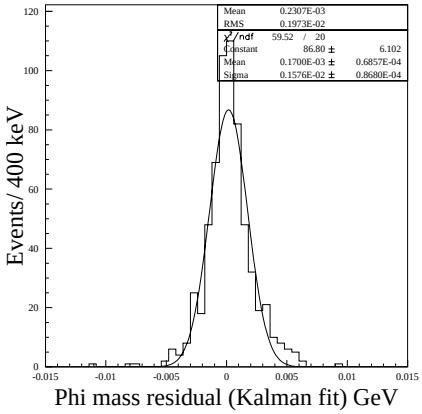

Figure 5: Distribution of the Φ mass residuals for the Kalman filter.

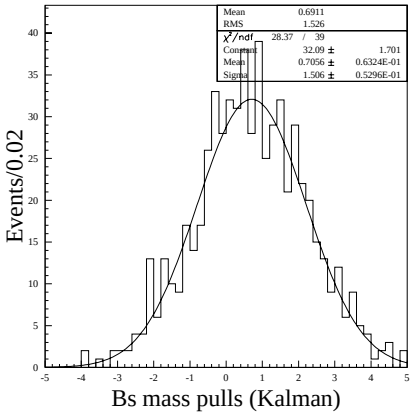

Figure 6: Distribution of the B_s mass pulls for the Kalman filter.

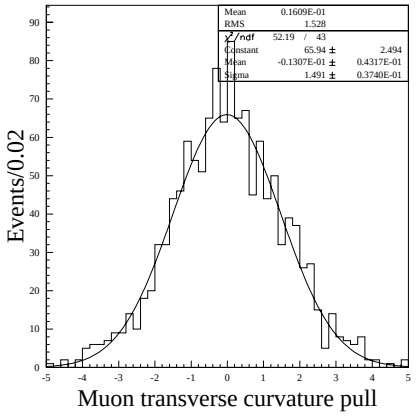

Figure 7: Reconstructed muons transverse curvature pulls

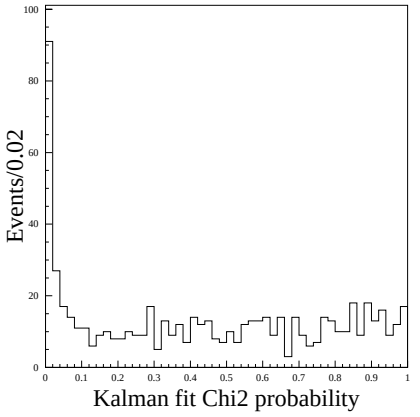

Figure 8: Distribution of the total χ^2 probability for the Kalman filter.

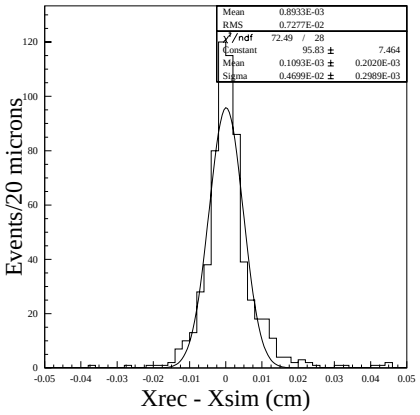

Figure 9: Distribution of the X vertex coordinate residuals for the Kalman filter.

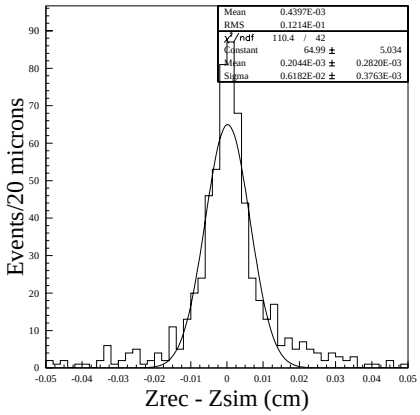

Figure 10: Distribution of the Z vertex coordinate residuals for the Kalman filter.

The total χ^2 probability distribution of the Kalman fit is presented on the fig. 8. The probability distribution is flat over the whole range with a peak at 0. This is due to the fact that the residuals of the track parameters are not exactly Gaussian-distributed and have significant tails. This problem is discussed in more details in [3] and [9].

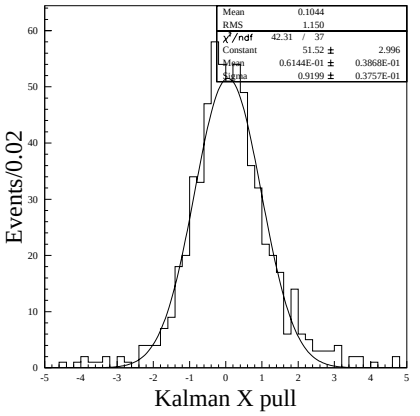

Figure 11: Distribution of the X vertex coordinate pulls for the Kalman filter.

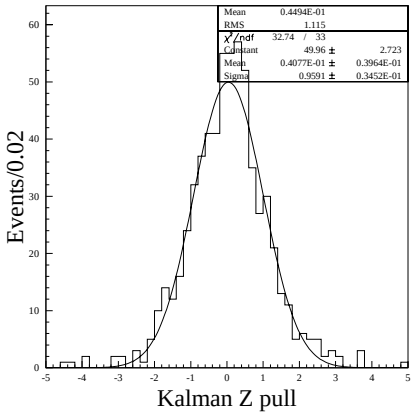

Figure 12: Distribution of Z vertex coordinate pulls for the Kalman filter.

4.2 The global strategy with the pointing constraint

The distribution of the residuals of the invariant mass of the B_s for the global strategy is presented on fig. 13. The resolution on the reconstructed mass is 13.5 MeV and the distribution is more centered (approximately 4 MeV shift) with comparison to the Kalman case. The distribution of the mass pulls has the same shape and width as in the Kalman filter case (see fig.14). The total χ^2 probability distribution has the same shape as in the Kalman case, as is expected, since both methods are LMS-based and both expect the measurements errors to be Gaussian-distributed (see fig.15).

The pointing constraint changes both the direction of the momentum and the position of the secondary vertex, which influences the invariant mass through the corresponding terms of the covariance matrix. This means that in the case of badly reconstructed events (where the B_s-meson momentum is very far from being aligned with the primary vertex), the mass can be significantly changed by the pointing constraint. Such events are very rare and their fit will have a large χ^2 , which can be used as a rejection criterion.

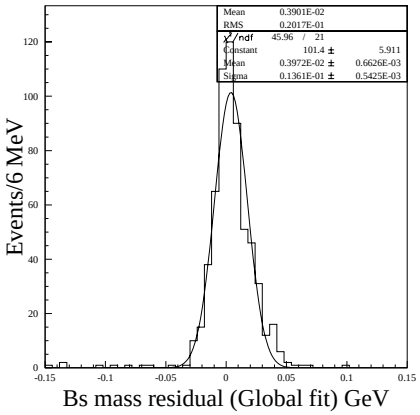

Figure 13: Distribution of the B_s mass residuals for the global fit.

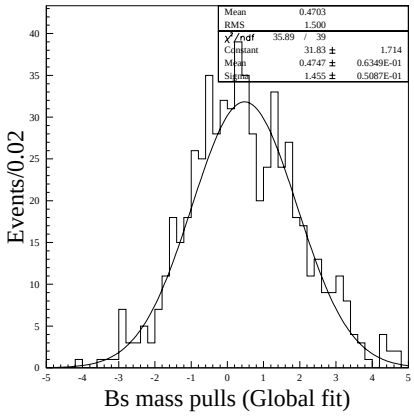

Figure 14: Distribution of the B_s mass pulls for the global fit.

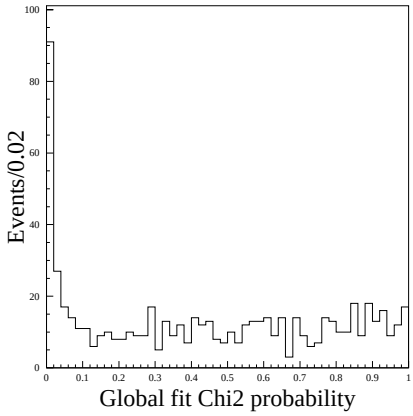

Figure 15: Distribution of the total χ^2 probability for the global fit.

The transverse coordinates resolution of the secondary vertex improve with respect to the pure Kalman case. The RMS becomes smaller and the width of the core, when fitted with a Gaussian, improves as well. The distribution of the residuals of the Z-components deviate significantly from a Gaussian distribution. While a much narrower core is formed, the RMS is similar to that obtained with the Kalman filter, indicating more tails. The pull distributions are still reasonably modeled by Gaussians, albeit with standard deviations somewhat larger than one. The distribution of vertex position residuals for X and Z coordinates are presented on fig. 16 and 17. The vertex position errors are reasonably well estimated: on the fig. 18 and 19 the distributions of the X and Z vertex coordinate pulls are presented.

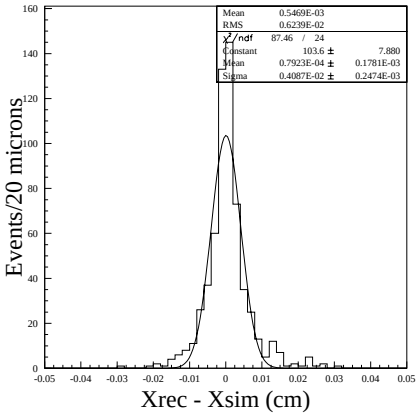

Figure 16: Distribution of the X vertex coordinate residuals for the global fit.

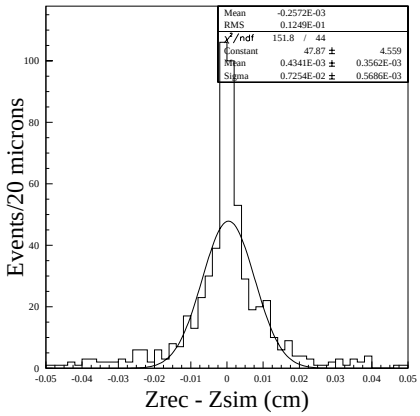

Figure 17: Distribution of the Z vertex coordinate residuals for the global fit.

4.3 Sequential strategy

On the fig. 20 the B_s mass residual for the sequential strategy is shown. The distribution has the same shape as in the global case, but the resolution is slightly worse. The distribution of mass pulls and the total χ^2 probability of the fit are presented on fig. 21 and 22. The X and Z vertex components resolutions and pulls are also very similar to the Global fit result.

Several other sequential strategies were also studied, e.g. individual reconstruction of both the J/Ψ and Φ mesons, and fitting them to the same vertex, which entails the propagation and the vertex fit of two neutral tracks etc. In all cases, the resolution on the B_s-meson mass was found to be very similar (within 3 MeV) once all constraints are applied, as is predicted by the factorization of constraints theorem. Overall, the global strategy yields somewhat

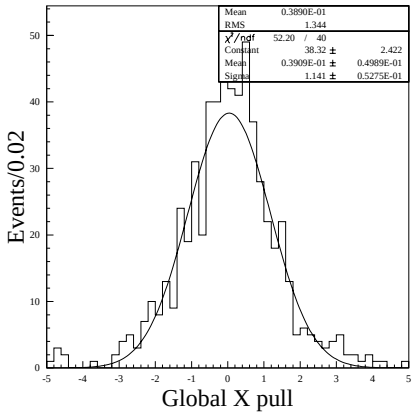

Figure 18: Distribution of the X coordinate pulls for the global fit.

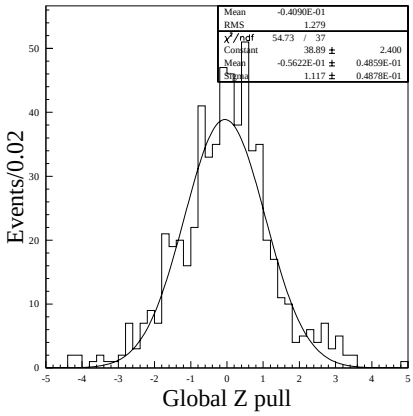

Figure 19: Distribution of the Z vertex coordinate pulls for the global fit.

better results.

The discrepancies observed between the different approaches come from the precision of the calculations. The global method involves less calculations than all other methods: only one vertex fit and no intermediary propagations are done.

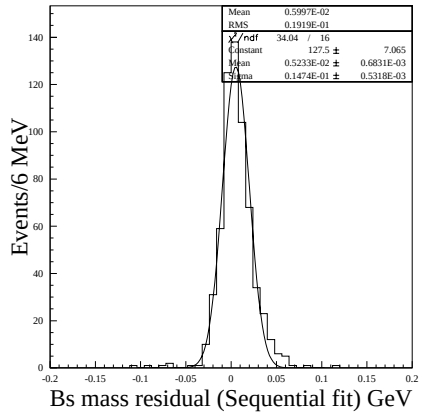

Figure 20: Distribution of the B_s mass residuals for the sequential fit.

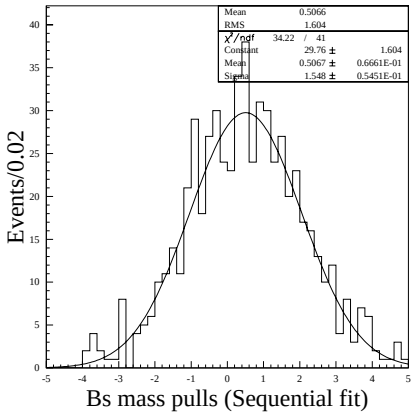

Figure 21: Distribution of the B_s mass pulls for the sequential fit.

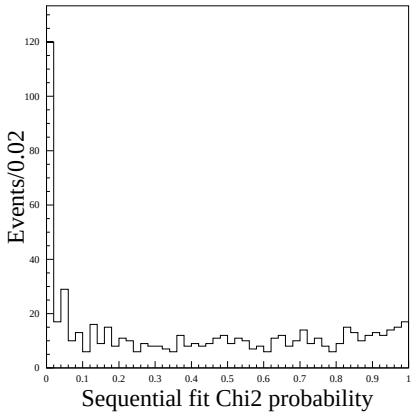

Figure 22: Distribution of the total χ^2 probability for the sequential fit.

5 Conclusion

A Kinematic Fit package using LMS with Lagrange multipliers method for the CMS reconstruction framework was implemented and several kinematic constraints were developed. A mechanism to model decay chains by trees was created. These trees can also be created and used independently of a kinematic fit.

The package was tested on the reconstruction of the decay $B_s \rightarrow J/\Psi\Phi$. A resolution of 13.5 MeV is achieved for the reconstructed B_s mass. The resolution on B_s decay vertex position is also improved with respect to the standard Kalman filter method. Several reconstruction strategies (global and sequential, with and without neutral state propagations) were studied, and their results were found to be very similar within computing precision.

In order to use the advantage of algorithms being independent of parametrization, software architecture should be changed. The possibility to construct Kinematic Particles from any reconstructed objects should be included in the nearest future.

The unflatness of the total fit χ^2 distribution, first noticed with the Kalman filter formalism for vertex reconstruction [3] and arising from non-Gaussian character of measurement errors was also observed in global LMS minimization with and without additional constraints. The possible solution is to try to use algorithms accounting for non-Gaussian structure of data, like Gaussian Sum Filter (GSF) [9] or non-LMS based algorithms. However, the presented framework is general enough to incorporate with minimal changes any kind of vertex fitting algorithms, presently existing in ORCA.

Only "value-like" constraining mechanisms were implemented in the framework so far. In order to incorporate "distribution-like" [6] constraints, the LMS minimization with penalty functions or similar formalism should be implemented. This can be done without change to the framework.

References

[1] P.Avery "Applied Fitting Theory VI: Formulas for Kinematic Fitting" CBX 98-37, June 9, 1998.

[2] European Physical Journal C 15, 1-878(200)

[3] T.Speer and K.Prokofiev "Vertex fitting with the Kalman filter formalism in the ORCA reconstruction program" CMS IN 2003/008

[4] P.Avery "Applied Fitting Theory VII: Building Virtual Particles" CBX 98-38, June 8, 1998.

[5] R.Frühwirth et al. "Vertex reconstruction and track bundling at the LEP collider using robust algorithms" Computer Physics Communications 96 (1996) 189-208

[6] A.G.Frodsen et al. "Probability and statistics in particle physics" Universitetsforlaget, 1979

[7] R.Frühwirth et al. "Data analysis techniques for high-energy physics" Cambridge University Press 2000

[8] P.Avery "Applied Fitting Theory IV: Formulas for Track Fitting" CBX 92-45, August 20, 1992

[9] R.Frühwirth, T.Speer "Vertex A Gaussian Sum Filter for Vertex Reconstruction" CMS IN AN 2003/006

[10] P.Avery "Kinematic Fitting Algorithms and Lessons Learned from KWFIT" http://www.phys.ufl.edu/ avery/fitting.html

[11] P.Avery "General Least Squares Fitting Theory" CBX 91-72, October 18, 1991

A The *KinematicFitPrimitives* library structure and class dependencies

On the fig. 23 the collaboration diagram of the *KinematicFitPrimitives* library is presented. The KinematicTree class stores the set of *KinematicParticles* and *KinematicVertices*. Every particle or vertex contains the reference to its previous state and last constraint applied. The lists of most important user methods of the *KinematicParticle*, *KinematicVertex* and *KinematicConstraint* classes are presented on fig. 24, 25 and 26.

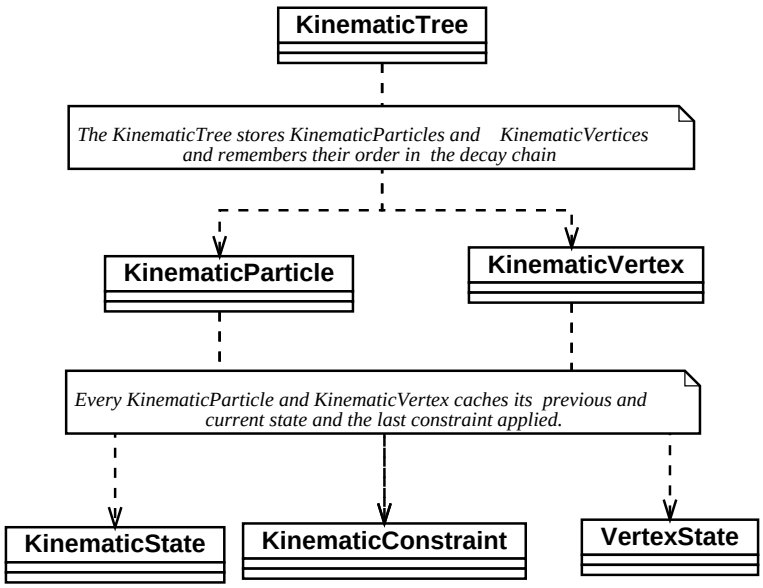

Figure 23: The *KinematicFitPrimitives* library structure

KinematicParticle

```
+initialState(): KinematicState
+currentState(): KinematicState
+stateAtPoint(): KinematicState
+lastConstraint(): KinematicConstraint *
+correspondingTree(): KinematicTree *
+chiSquared(): float
+degreesOfFreedom(): float
```

Figure 24: The *KinematicParticle* class user interface

B The *KinematicTree* functionality

On the fig. 27 the possible structure of the reconstructed *KinematicTree* for the $B_s \rightarrow J/\Psi\Phi \rightarrow \mu\mu KK$ decay is shown. The tree is reconstructed from final state tracks to the mother particle. Every tree component caches pointers to its previous states and constraints applied, as well as the pointer to the current tree the component belongs to. The actual graph structure of the *KinematicTree* class is hidden from the user. The set of public methods which allow to navigate in the tree is provided.

The tree is based on the graph *DDD* library. A schematic illustration of the navigation principle is shown on the fig. 28. The graph is a private data member of the *KinematicTree* class. It has *RefCountedKinematicParticles* as edges and *RefCountedKinematicVertices* as nodes. These are accessible through the graph walker, which is also a private date member of the *KinematicTree* class.

On the fig. 29, the UML diagram showing the most important access and navigation method is presented. For the code example of the tree navigation and use see the analysis examples at */ORCA/Vertex/KinematicFitTest/test*.

KinematicVertex
+vertexIsValid(): bool +correspondingTree(): KinematicTree* +vertexBeforeConstraint(): RefCountedKinematicVertex +vertexState(): VertexState +position(): GlobalPoint +error(): GlobalError +chiSquared(): float +degreesOfFreedom(): float

Figure 25: The *KinematicVertex* class user interface

KinematicConstraint
+value(exPoint:const AlgebraicVector &): pair<AlgebraicVector, AlgebraicVector> +derivative(exPoint:const AlgebraicVector &): pair<AlgebraicMatrix, AlgebraicVector> +value(par:vector<const RefCountedKinematicParticle>): pair<AlgebraicVector, AlgebraicVector> +derivative(par:vector<RefCountedKinematicParticle>): pair<AlgebraicMatrix, AlgebraicVector> +numberOfEquations(): int

Figure 26: The *KinematicConstraint* class user interface

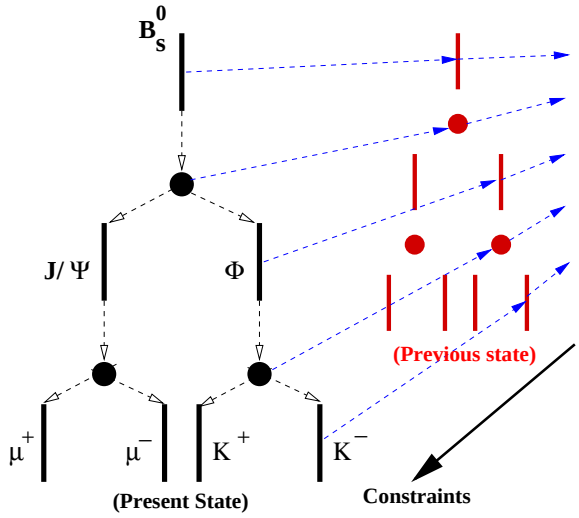

Figure 27: A possible reconstructed tree of $B_s^0 \rightarrow J/\Psi\Phi \rightarrow \mu^+\mu^- K^+ K^-$ decay. Every tree component "remembers" its state before the last refit.

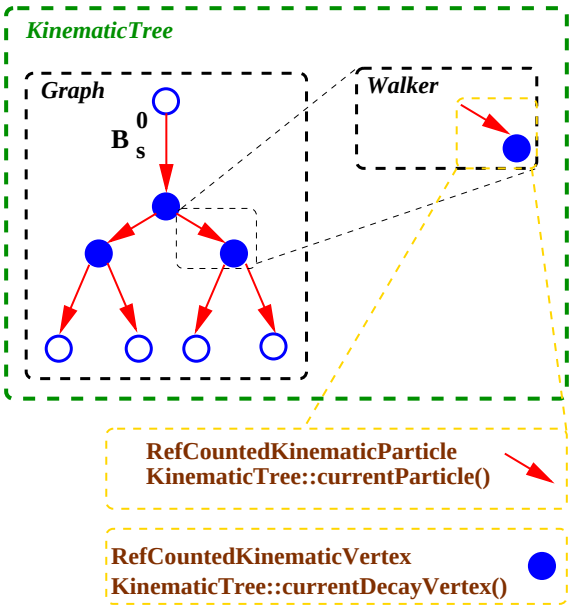

Figure 28: Internal graph-based structure of the *KinematicTree*. User accesses it via the simple navigation mechanism

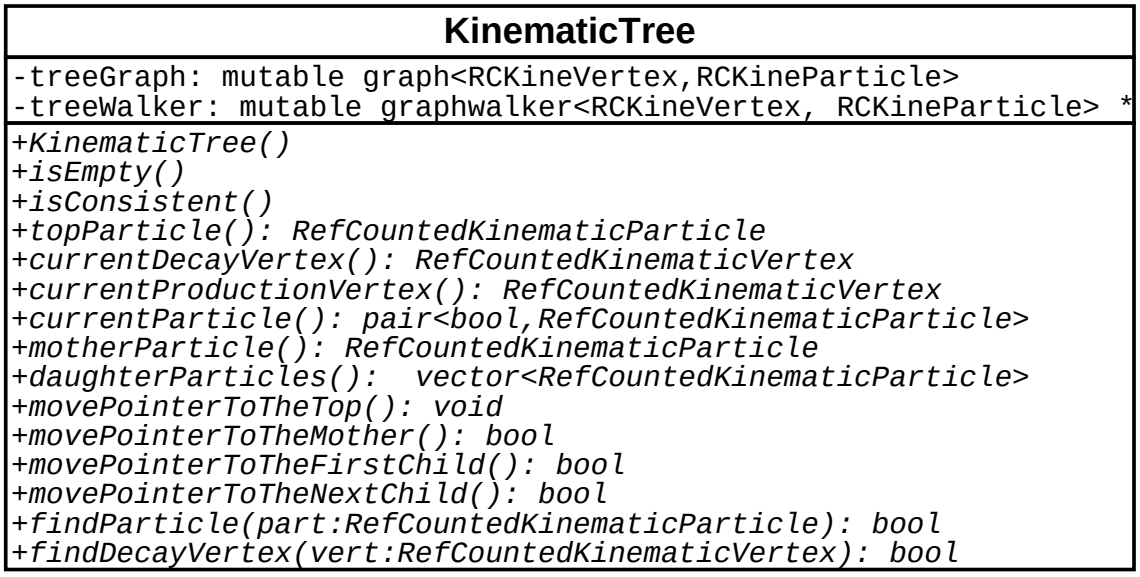

Figure 29: The *KinematicTree* class users interface.