

Part 6

vertex fitting

decay tree fitting

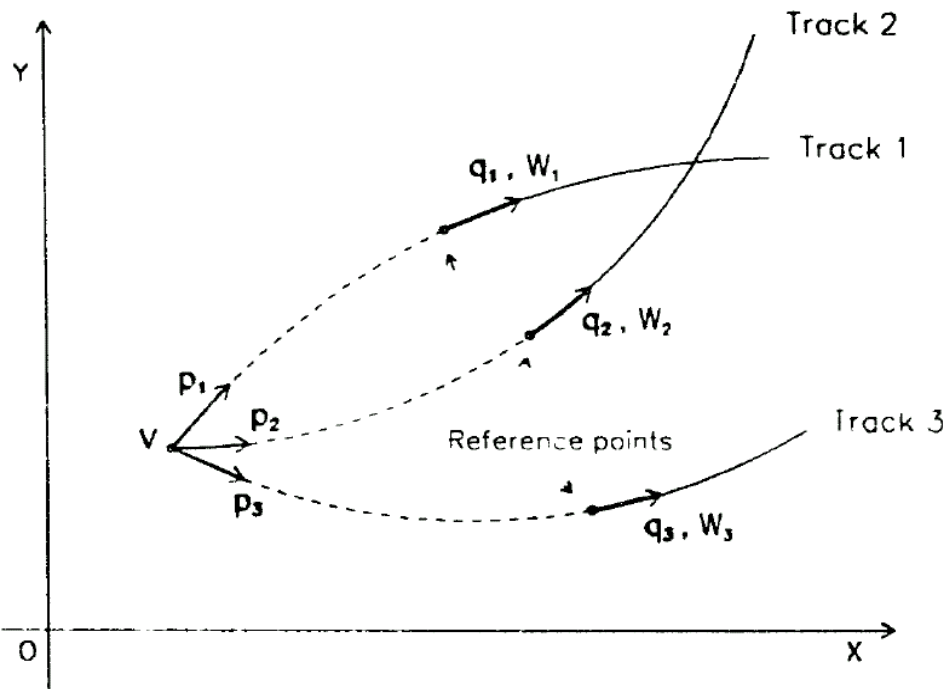
reconstructing decays

- most HEP analysis concern reconstruction of decaying particles
 - charged decay products are reconstructed as tracks
 - neutral decay products as clusters in a calorimeter
- reconstruction of decay vertex is essential ingredient for
 - measurement of invariant mass
 - measurement of lifetime, time-dependent CPV
 - identification: list of particles with macroscopic decay length is short

vertex fitting

- combine N tracks under the constraint that they come from common point
- input: parameters + covariance for each track
- output: vertex position, track momentum vectors, covariance matrix
- momentum only becomes relevant if there are kinematic constraints or if the field is non-zero at the position of the vertex
- compared to track fitting
 - more parameters: $3 + 3 \times N$
 - inherently non-linear
 - no equivalent of process noise

model of a decay



- model: 1 vertex + N momentum vectors
- data: N track parameter vectors with covariance matrix

- measured tracks are independent, so total chi-square is

$$\chi^2 = \sum_{\text{tracks } i} (q_i - h(x, p_i))^T V_i^{-1} (q_i - h(x, p_i))$$

measured track parameters,
e.g. 5-parameter helix

vertex position

momentum vector

covariance matrix for
measured track parameters,

track measurement model in vertex fit

- of course, this depends on how you have parameterized the tracks
- trivial in the forward-geometry parameterization in zero field

$$h(x, p) = \begin{pmatrix} x_0 \\ y_0 \\ t_x \\ t_y \\ p \end{pmatrix} = \begin{pmatrix} x - zp_x/p_z \\ x - zp_y/p_z \\ p_x/p_z \\ p_y/p_z \\ p \end{pmatrix}$$

- but quite a bit more complicated if dealing with helices

$$h(x, p) \equiv \begin{pmatrix} d_0 \\ \phi_0 \\ \omega \\ z_0 \\ \theta \end{pmatrix} = \begin{pmatrix} (p_{t0} - p_t)/aq \\ \text{atan2}(p_{y0}, p_{x0}) \\ qB_z/p_t \\ z - (\text{atan2}(p_y, p_x) - \phi_0)p_z/qB_z \\ \text{atan}(p_t/p_z) \end{pmatrix} \quad \begin{aligned} p_{x0} &= p_x + qB_z y \\ p_{y0} &= p_y - qB_z x \end{aligned}$$

- and now you still need the derivatives ... see [arXiv:physics/0503191](https://arxiv.org/abs/physics/0503191)

minimizing the chi-square

- 'naïve' global fit requires inversion of M-dimensional matrix, with $M=3+3N$
 - not great if number of outgoing tracks large
 - practical implementations don't do it that way
- two popular methods (closely related)
 - Billoir algorithm (Billoir, Fruhwirth, Regler (1985), Billoir, Qian (1992)
 - global fit, exploiting (empty) structure of $H^T V^{-1} H$
 - Kalman filter (Fruhwirth (1987), Luchsinger, Grab (1992), ..., Hulsbergen (2005))
- with both methods the new track momentum vectors can be calculated, but can also be omitted (which saves time)
- since measurement-model not linear, need iterations
- expressions not very illuminating, so we'll skip them

short intermezzo: exact constraints
(left over from yesterday morning)

measurement constraints

- up till now, contributions to chi-square looked like

$$\Delta\chi^2 = (m_i - h_i(x))^T V_i^{-1} (m_i - h_i(x))$$

- I'll call this type of contribution a ***measurement constraint***
- it can be more generally written as

$$\Delta\chi^2 = g_i(x)^T V_i^{-1} g_i(x)$$

- with the LSE we solve the over-constrained set of equations

$$\forall_i g_i(x) = 0$$

using the assigned inverse variance as a weight

- but now suppose that we have a relation between the parameters x that we want to be *exactly* satisfied?

exact constraints

- ***exact constraint*** expresses exact relation between the parameters x
 - for example: suppose x is a 4-vector with 4x4 covariance matrix and we want it to have exactly length m_B
- sometimes it is possible to simply eliminate 1 of the parameters
- more generic solution: add an extra term to the chi-square

$$\Delta\chi^2 = \lambda_j g_j(x)$$

- the parameter λ is a lagrange multiplier
- we now minimize the total chi-square wrt to λ and x simultaneously
- taking the derivative to lambda, you see how this imposes the constraint

$$0 = \frac{d\chi^2}{d\lambda_j} = g_j(x)$$

exact constraints in the progressive fit

- in the progressive fit, we can eliminate the lagrange multiplier

$$\chi_k^2 = (x - x_{k-1})^T C_{k-1}^{-1} (x - x_{k-1}) + 2\lambda_k^T g_k(x)$$

linearize around x_{k-1} : $g_k(x) = g_k(x_{k-1}) + G_k (x - x_{k-1})$

$$0 = \frac{1}{2} \frac{d\chi^2}{dx} = C_{k-1}^{-1} (x - x_{k-1}) + G_k^T \lambda_k$$

$$0 = \frac{d\chi^2}{d\lambda} = g_k(x_{k-1}) + G_k (x - x_{k-1})$$

solve, eliminate λ

$$x_k = x_{k-1} - K_k g_k(x_{k-1})$$

$$C_k = (1 - K_k G_k) C_{k-1} (1 - K_k G_k)^T$$

$$K_k = C_{k-1} G_k^T (G_k C_{k-1} G_k^T)^{-1}$$

- not surprising: expressions are identical to those for a measurement constraint with $V=0$!
- so, it is easy to include exact constraints in a progressive fit

mass constraints

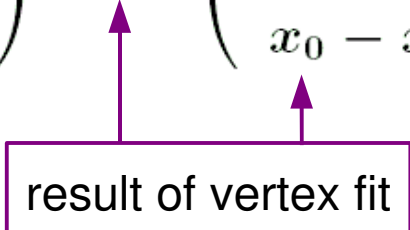
- as we shall see in a few minutes, it is sometimes useful to constrain the invariant mass of the decaying particle to a 'known' value
- use lagrange-multiplier technique

$$\chi^2_+ = \lambda \left[\sum (p_i^2 + m_i^2) - \left(\sum \vec{p}_i \right)^2 - m_{\text{pdg}}^2 \right]$$

- note
 - need to assign mass hypothesis to each track
 - this is a non-linear constraint
- as we have seen in 2nd lecture, an efficient way of dealing with such constraint is the 'progressive' method
 - first vertex without the constraint
 - then add the constraint
 - eventually iterate

adding a mass constraint

- with progressive method, chi-square looks like (in 1D, but easy to generalize)

$$\chi^2 = \begin{pmatrix} E_0 - E \\ p_0 - p \\ x_0 - x \end{pmatrix}^T C^{-1} \begin{pmatrix} E_0 - E \\ p_0 - p \\ x_0 - x \end{pmatrix} + \lambda [E^2 - p^2 - m_{\text{pdg}}^2]$$


result of vertex fit

- in the linear approximation, the minimum chi-square solution is

$$\begin{pmatrix} \hat{E} \\ \hat{p} \\ \hat{x} \end{pmatrix} = \begin{pmatrix} E_0 \\ p_0 \\ x_0 \end{pmatrix} - C G^T (G C G^T)^{-1} (E_0^2 - p_0^2 - m_{\text{pdg}}^2)$$

where the constraint derivative is $G^T = \begin{pmatrix} 2E_0 \\ 2p_0 \\ 0 \end{pmatrix}$

- since the constraint is non-linear, you would not need to iterate, using the technique discussed in the 2nd lecture
- can we do it simpler?

adding a mass constraint, faster method

- the faster method relies on coordinate transformation

$$\begin{pmatrix} E \\ p \end{pmatrix} \longrightarrow \begin{pmatrix} m^2 \\ p \end{pmatrix} \qquad C \longrightarrow FCF^T$$
$$F = \begin{pmatrix} 2E & 2p \\ 0 & 1 \end{pmatrix}$$

- the exact constraint does something very simple to m^2

$$\hat{m}^2 = m_{\text{pdg}}^2 \qquad \text{var}(\hat{m}^2) = 0$$

- now you propagate that information to the momentum

$$\hat{p} = p_0 - \frac{\text{cov}(m_0^2, p_0)}{\text{var}(m_0^2)} (m_0^2 - m_{\text{pdg}}^2)$$

$$\text{var}(\hat{p}) = \text{var}(p_0) - \frac{\text{cov}(m_0^2, p_0)^2}{\text{var}(m_0^2)}$$

- finally, you transform back to (E,p) coordinates
- using the same formulas, you can also propagate the information to the vertex position

why did I show you this?

- sometimes 'transformation' is good alternative to 'linearization'
 - it's not magic: it will not solve the problem that non-linear transformations of variance don't preserve confidence intervals
- the trick on the previous page allows you to add mass constraint to any p_4 with error
 - you can often do this at 'ntuple-level'
 - no need for complicated kinematic fits
- I wanted to introduce you to the concept of 'propagation' of information through covariance matrices

intermezzo: propagation formula

- suppose we have two observables (\mathbf{a}, \mathbf{b}) with covariance \mathbf{V}
- suppose we do something which makes that we know \mathbf{a} better

$$\mathbf{a} \longrightarrow \tilde{\mathbf{a}} \qquad \mathbf{V}_{aa} \longrightarrow \tilde{\mathbf{V}}_{aa}$$

- we can propagate this knowledge to \mathbf{b} using

$$\tilde{\mathbf{b}} = \mathbf{b} + \mathbf{V}_{ab} \mathbf{V}_{aa}^{-1} (\tilde{\mathbf{a}} - \mathbf{a})$$

$$\tilde{\mathbf{V}}_{bb} = \mathbf{V}_{bb} - \mathbf{V}_{ba} \mathbf{V}_{aa}^{-1} (\mathbf{V}_{aa} - \tilde{\mathbf{V}}_{aa}) \mathbf{V}_{aa}^{-1} \mathbf{V}_{ab}$$

$$\tilde{\mathbf{V}}_{ab} = \tilde{\mathbf{V}}_{aa} \mathbf{V}_{aa}^{-1} \mathbf{V}_{ab}$$

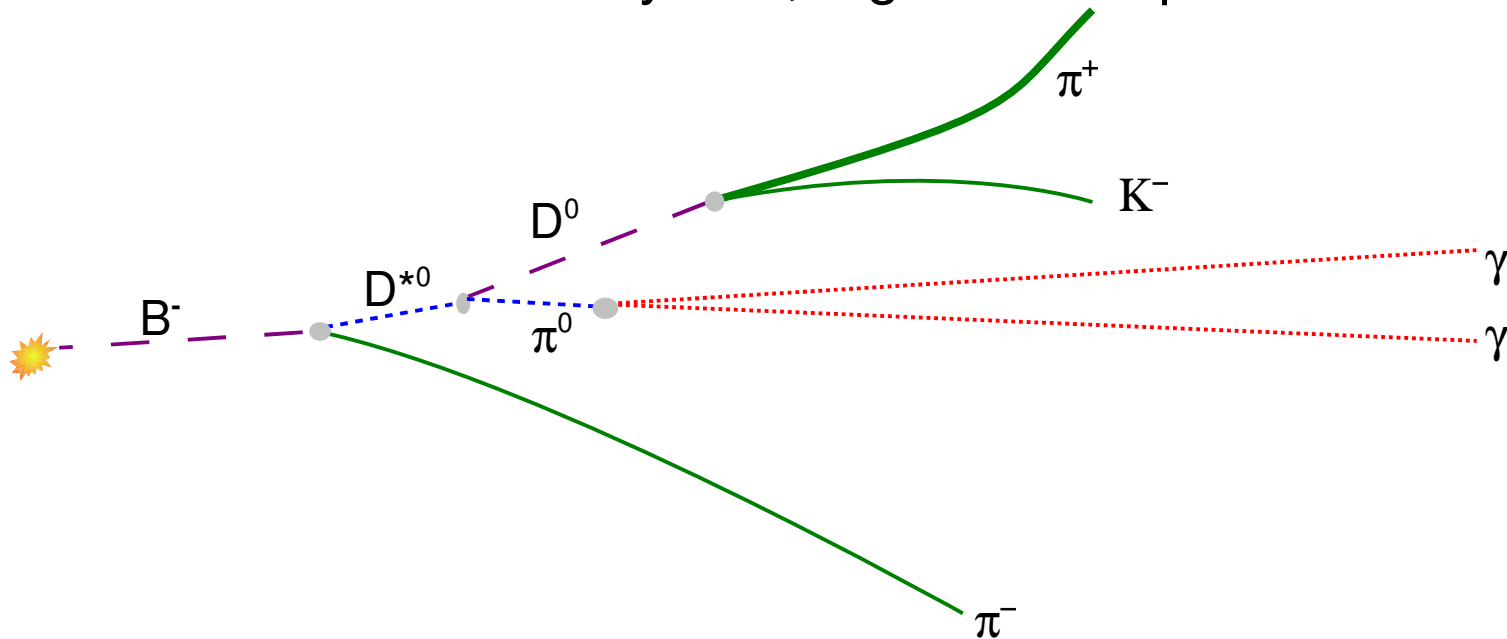
- you can derive this with the LSE. it is essentially just the progressive method again. formulas also work when \mathbf{a} and \mathbf{b} are vectors
- propagation is useful if you want to deal only with relevant subset of parameters when adding a constraint

photons

- photons in final state
 - do not contribute to knowledge of vertex
 - however, needed when using mass constraint
- photon reconstructed as 'calorimeter cluster' with energy and position
 - enters X^2 just like reconstructed track
 - measurement model not completely trivial (see e.g. [arXiv:physics/0503191](https://arxiv.org/abs/physics/0503191))
 - 4 measurements, but only 3 constraints
- calorimeter clusters with other hypothesis can also be used, e.g. K_L
 - use only reconstructed position, not energy measurement
 - this only becomes a constraint when dealing with multi-level decay chains

decay trees

- now consider a multilevel decay tree, e.g. $B^- \rightarrow D^{*0} \pi^-$



- there are four types of objects, sometimes called 'particles'
 - reconstructed as track with mass hypothesis (e, μ , π ,K,p)
 - reconstructed as cluster with mass hypothesis (γ)
 - composite or virtual particles
 - with non-observable decay length ('resonances')
 - with macroscopic decay length ('non-resonances')

parameterizing a decay tree

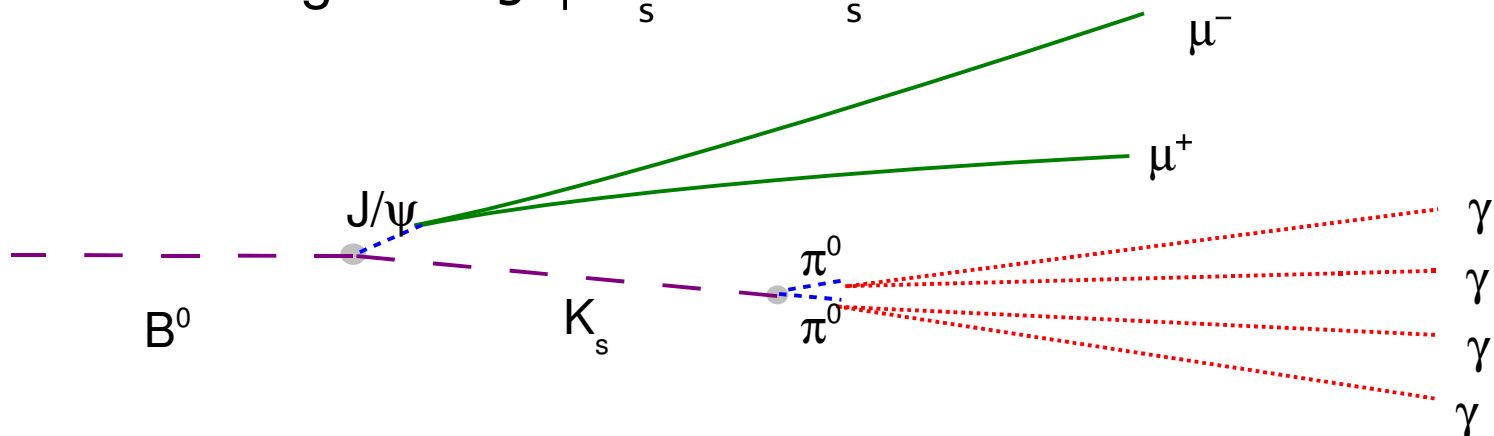
- there are many ways to do this, but this is most popular (with me, at least)
 - for each outgoing particle a momentum vector (mass is fixed)
 - for each 'composite' particles
 - a momentum vector, energy and decay vertex
 - if it is not at the 'head' of the decay tree and not a resonance, we add a decay length
 - if a resonance has a mother, we omit its decay vertex
- you can now count the number of parameters in the decay tree on the previous page: 9 momenta + 2 vertices + 1 decay length = 34
- how would you fit something like this?

fitting a decay tree: cascade method

- cascade method: fit most downstream vertices first, work your way upward
- this exploits that in the linear approximation all downstream information is contained in composite's parameters and covariance matrix
- in the example
 - first fit the $D \rightarrow K\pi$ decay
 - then fit the B, using the 'composite' D as input
 - once you have vertexed the D0, the K momentum is entirely irrelevant for the B vertex fit
- this method simple, fast and it almost always works
- it is also efficient if you e.g. want to use same D0 to reconstruct more than one B candidate

fitting a decay tree: global method

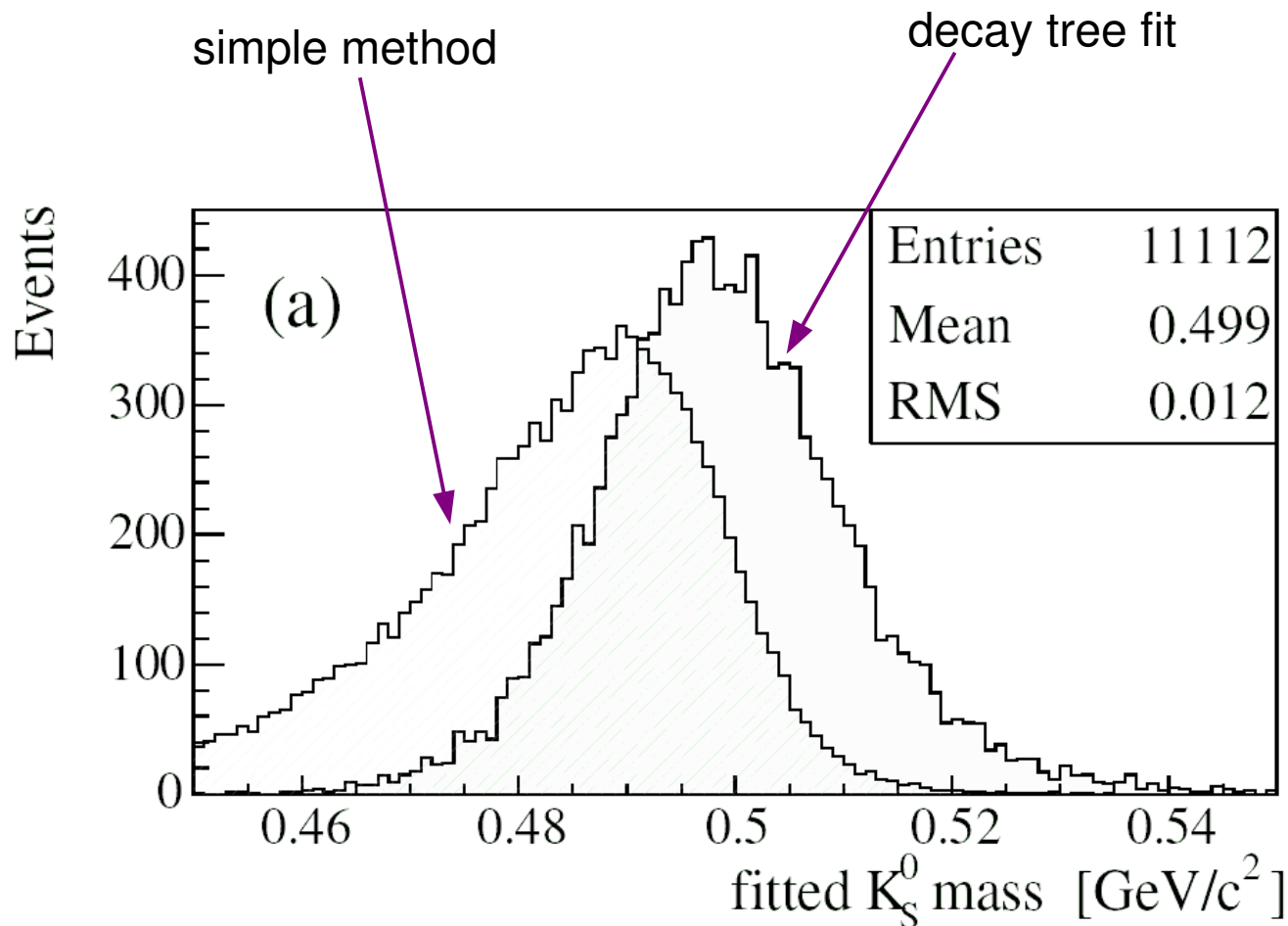
- global method: calculate complete tree in single fit
 - obtain covariance matrix for complete decay tree
 - in practical applications, a progressive fit works best (since covariance matrix is huge if number of particles large)
- global method has two advantages wrt to cascade method
 - better treatment of non-linearities
 - some decay trees cannot be fit with the cascade method, e.g.
'extreme vertexing: $B^0 \rightarrow J/\psi K_s$ with $K_s \rightarrow \pi^0 \pi^0$



- this decay tree is overconstrained if mass constraints are used for both π^0

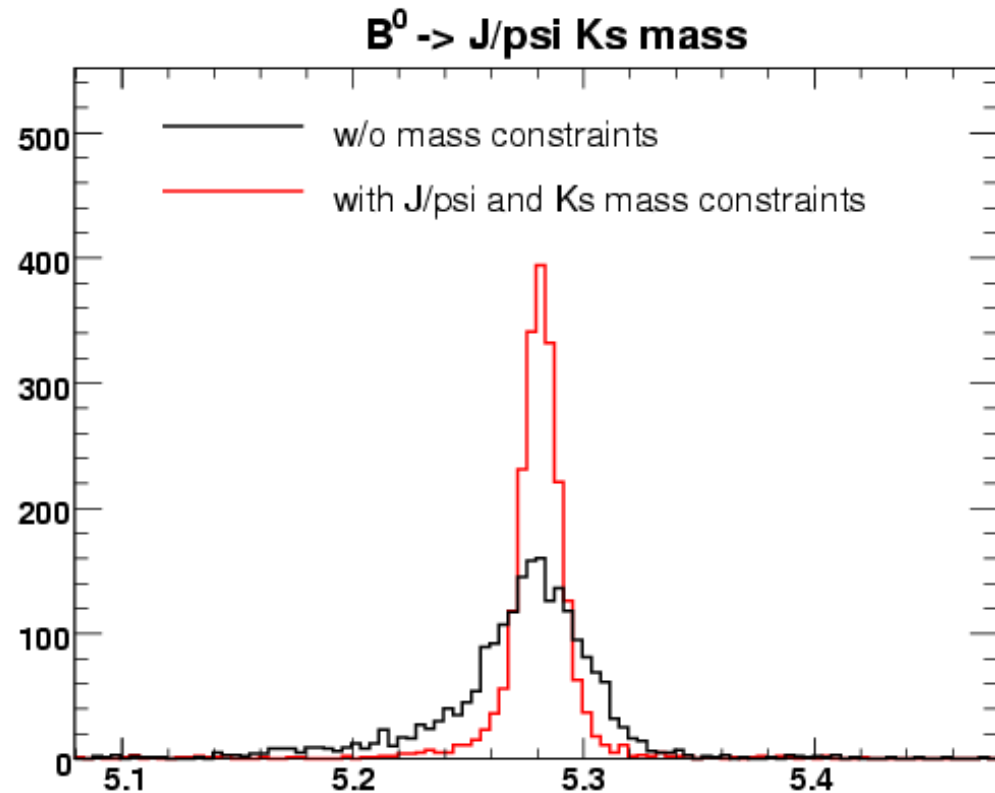
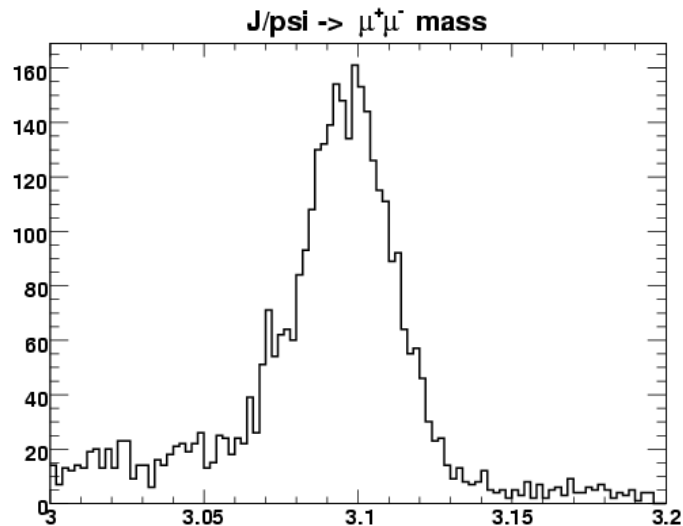
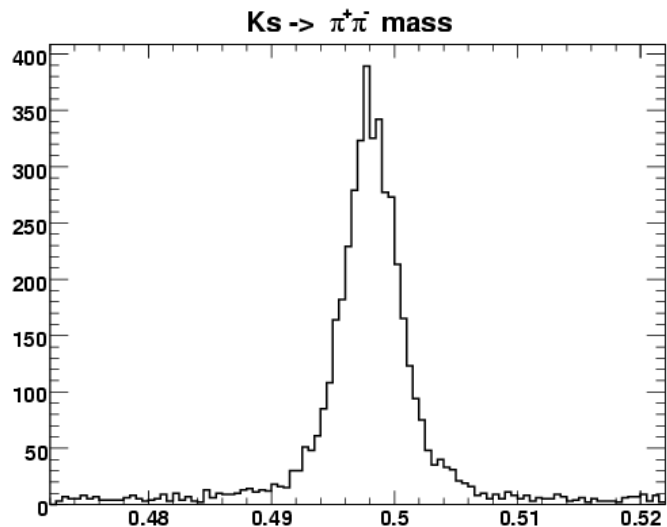
illustration of global fit

- the alternative, used before we had decay tree fits: forget about K_S decay length and simply attach the photons to the B vertex
- but K_S flies several 10s of cm \Rightarrow totally wrong K_S mass (and B mass)



mass constraints in decay tree

- to illustrate what mass constraints can do for you, consider $B^0 \rightarrow J/\psi K_s$ but now with $K_s \rightarrow \pi^+\pi^-$ (this is the 'normal' decay)



large impact of mass constraints mostly due to 'recovery' of tail in J/ψ mass

other constraints

- depending on what your experiment looks like, you might have more information for your fit
- e.g. in the B factories
 - origin of initial particle
 - average interaction point is calibrated using $e^+e^- \rightarrow \mu^+\mu^-$ events
 - used in decay tree fits to constrain origin of B or D mesons
 - energy of initial particle
 - if your X comes from $e^+e^- \rightarrow X\bar{X}$, (like the B in the B-factories), then the CMS energy of X can be constrained to $\sqrt{s}/2$
- adding these constraints to your fit should be straightforward now!

concluding remark: 'garbage-in is garbage-out'

- if the errors of the input don't make sense, then the errors of the output don't make sense either
- for example, be careful with mass constraints
 - a tail in your mass distribution can bias the vertex position if you constrain the mass
 - there is not much you can do about this (unless you are adventurous and experiment with things like the Gaussian-Sum filter)
- often, result is compromise between maximum statistical power and minimum systematic errors
 - you apply the mass constraint if it helps
 - you use a control channel to make sure it doesn't hurt

what I skipped: vertex finding

- in experiments like Babar almost all vertexing is 'hypothesis-based'
 - we assume these tracks come from $B \rightarrow J/\psi K_s$. now fit it
- in LEP/Fermilab/LHC experiments vertex finding is more important
 - how many interactions were there in this event?
 - was there a B decay in this event?
- the techniques look similar to track finding, but combinatorics is less important limitation
- example: reconstruct a primary vertex
 - combine all tracks in one vertex
 - remove tracks with large chisquare contribution
 - refit if necessary