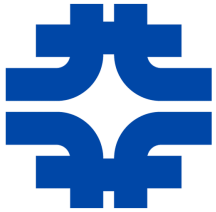


FNAL Software School
Day 3

Matt Herndon,

University of Wisconsin – Madison

Today's Activities

- Introductory slides
- Review Day 2
  - Investigate candidate finding
  - Demonstrate the positive effect of a better resolution model on the track fit
- Lecture
  - The tracking reconstruction problem: Candidate finding
  - Organization of a complex set of algorithms
- Daily project:
  - Candidate finding Algorithm

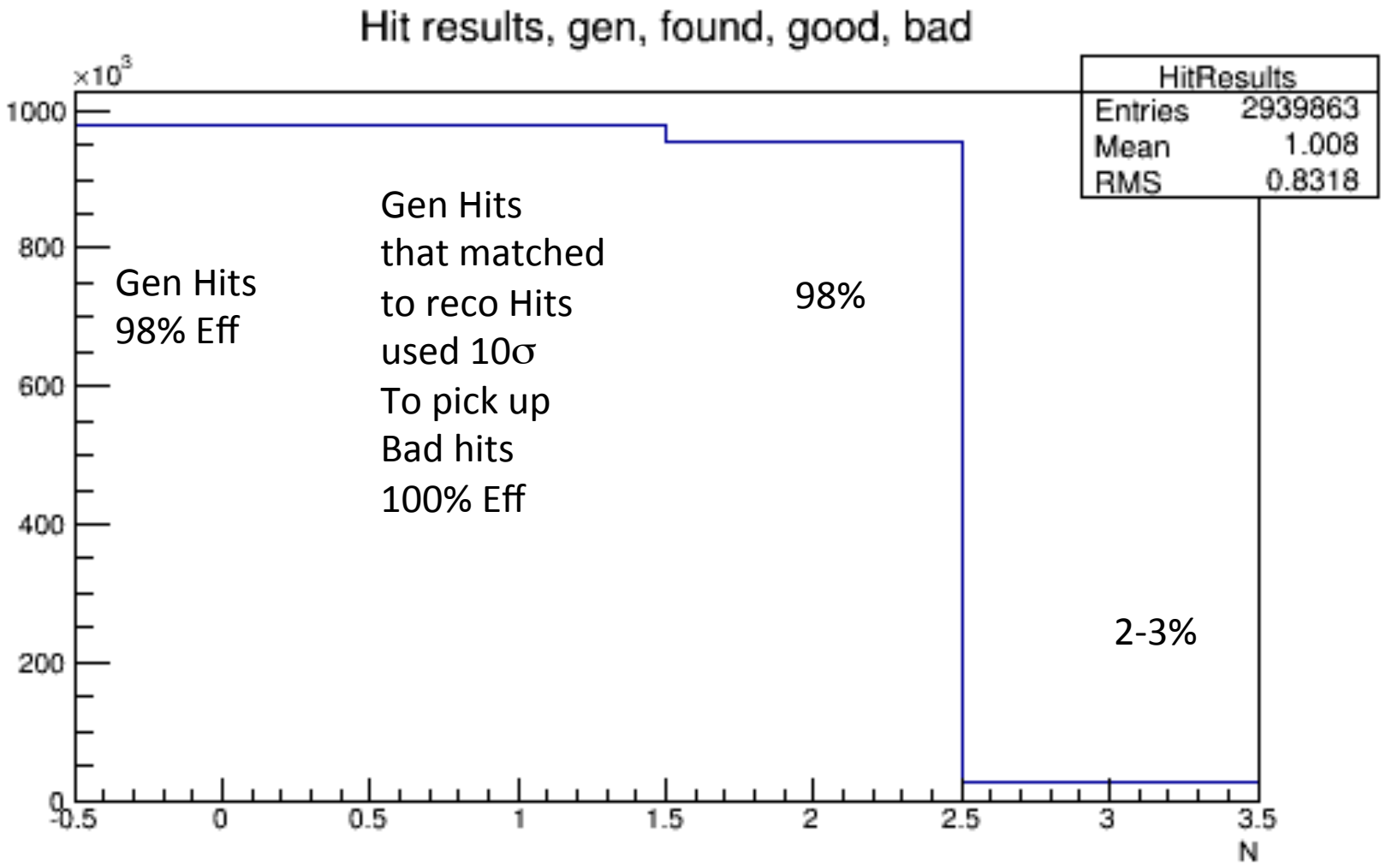
Today's Activities

- New coding project schedule
  - Detailed instructions for project after lecture(10:30)
  - Simultaneous
    - Remedial work to tie up loose ends of previous days project with instructor
    - Start new project with instructor
  - Lunch, please finish by 1PM
  - Meet with TAs to assess progress on project: 1PM
  - Programming in consultation with TAs
  - Primarily to allow the TAs time to do their work

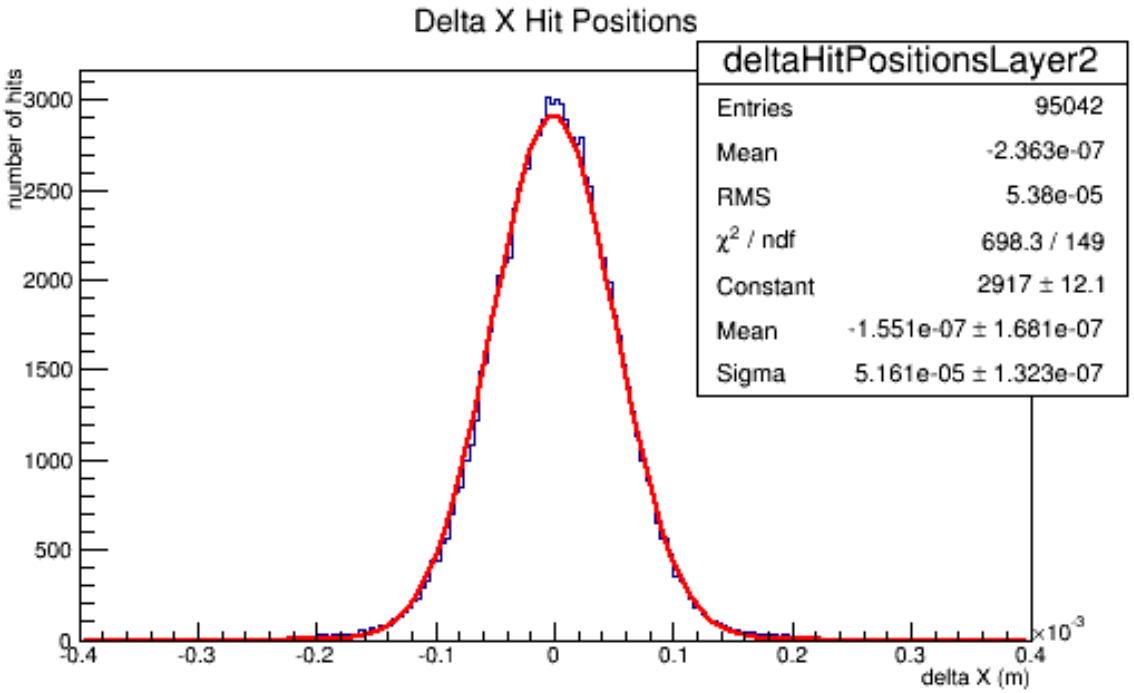
Day 2 exercise review

- Day 2 exercise
  - Evaluating hit finding performance
  - Applying the results to tracking reconstruction
- Project goals
  - Does the hit reconstruction code efficiently reconstruct hits
    - Should be no inefficiency. A detector inefficiency is programmed into the simulation, but is applied before GenHits are written.
  - Were the hits reconstructed accurately and with well understood resolutions
    - If cases where it was not, are they understood, and if so apply the results to the resolution model used for track fitting.
  - Does the track fit perform well after this work?

Hit Efficiency

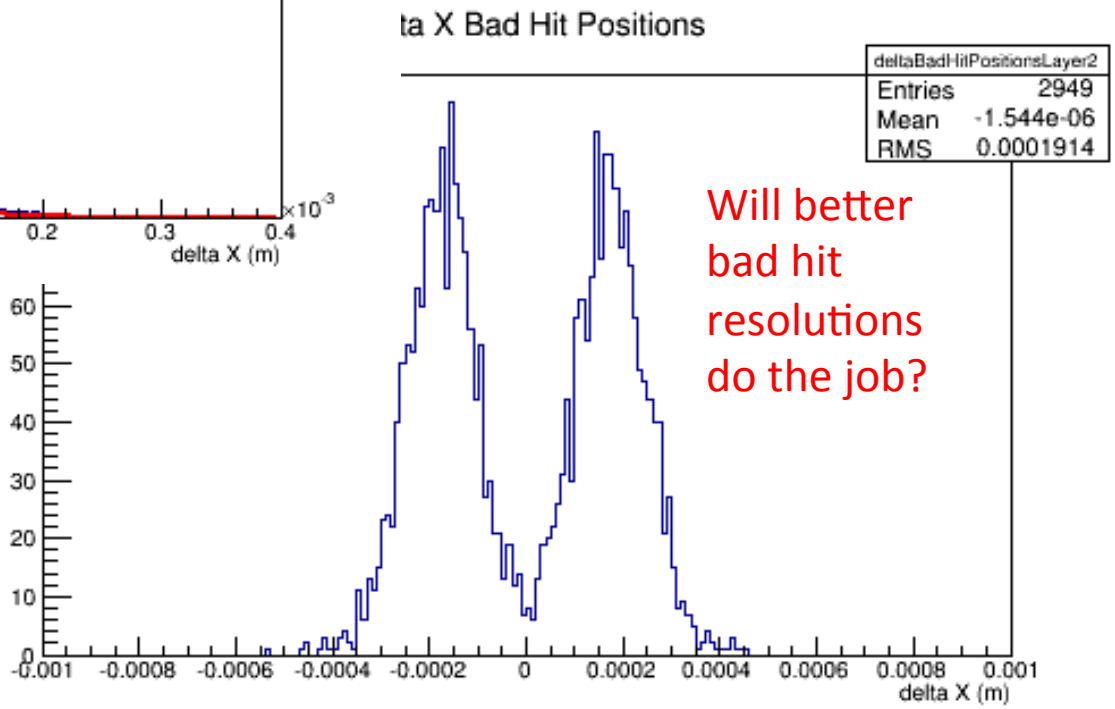


Hit Resolution



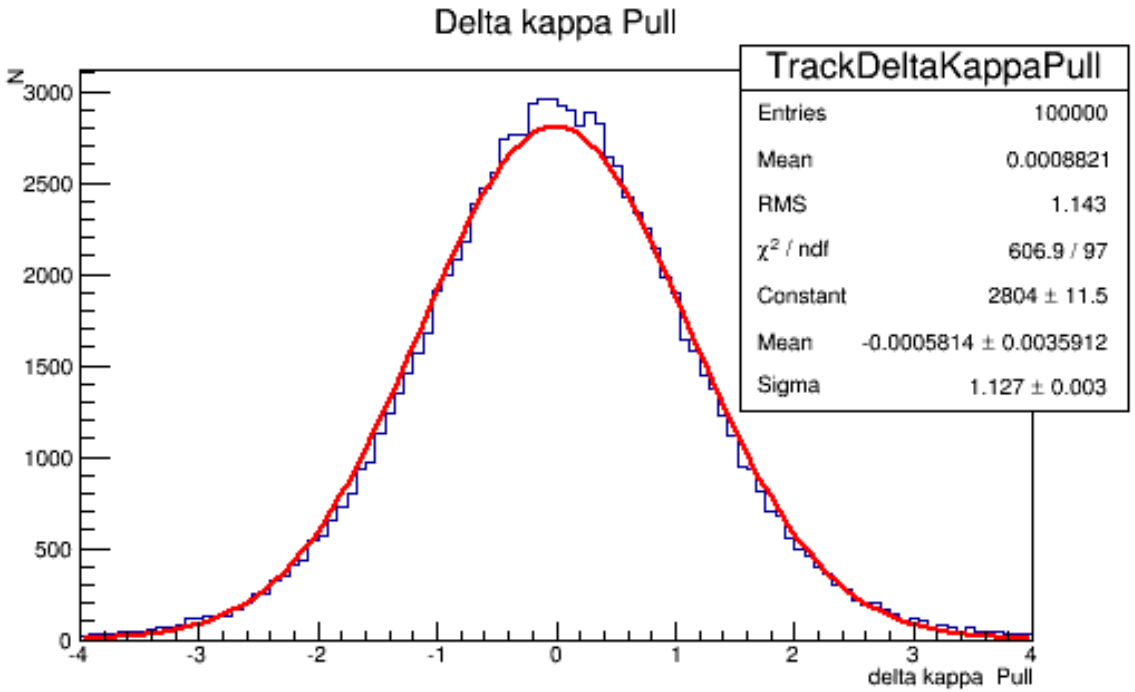
Expected ~50um
There is digitization noise. Should adjust

[sensorgeometry.txt]



Will better bad hit resolutions do the job?

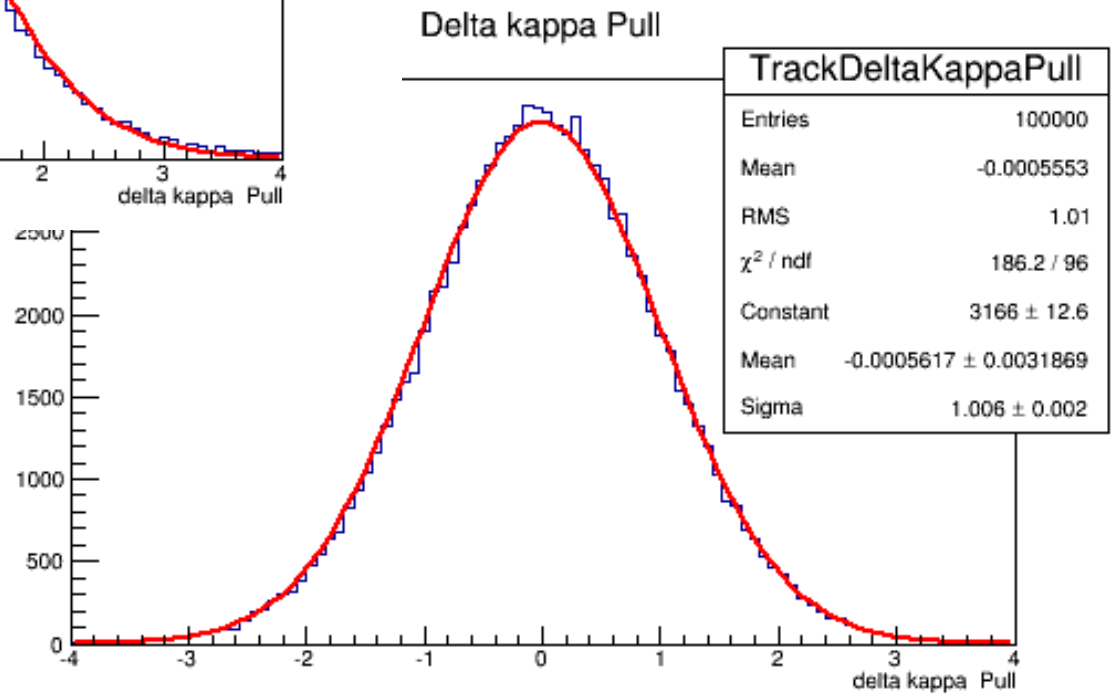
Perfect Tracking



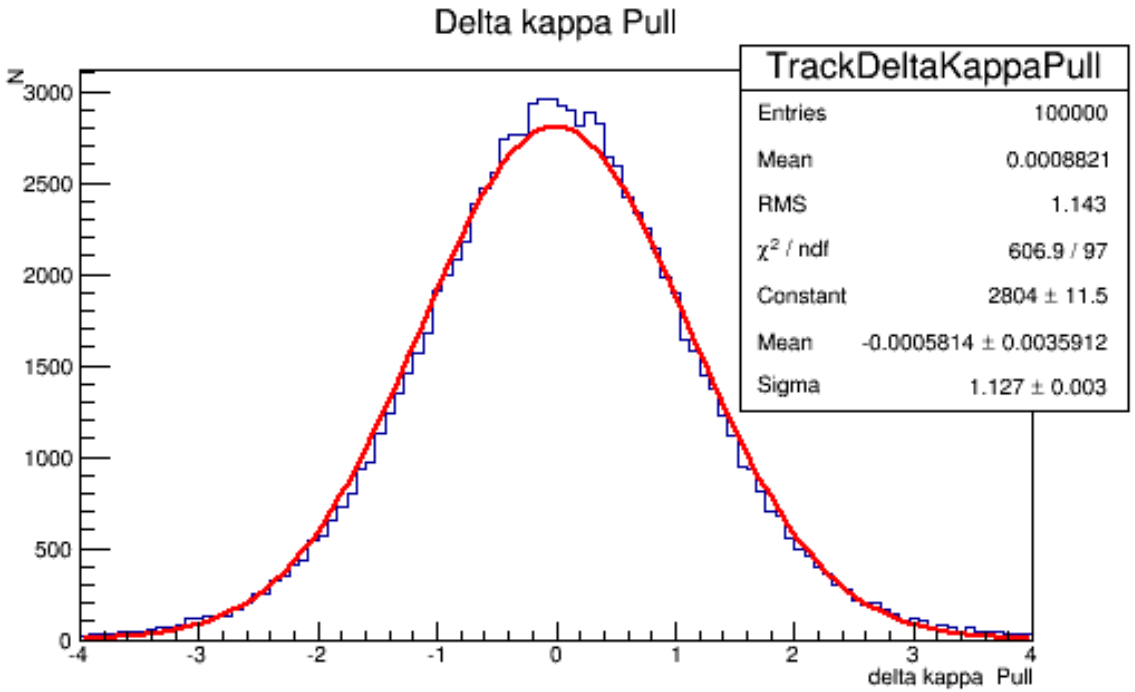
Results use perfect tracking. Matches reconstructed Hits to GenHits to find best set of reconstructed hits

Track parameters were misestimated by 12%-17% and slightly non Gaussian

After applying the new resolution the fit results are unbiased and well estimated.



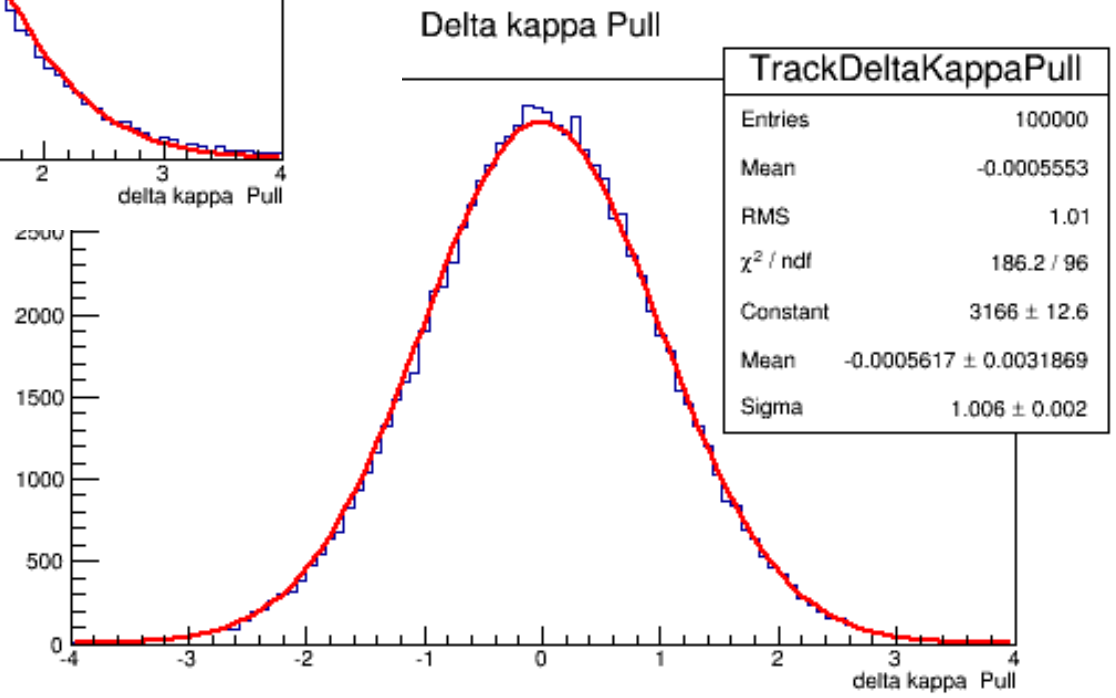
Perfect Tracking



Results use perfect tracking. Matches reconstructed Hits to GenHits to find best set of reconstructed hits

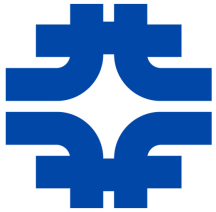
Track parameters were misestimated by 12%-17% and slightly non Gaussian

After applying the new resolution the fit results are unbiased and well estimated.



Is the the best solution?

- Track reconstruction often involves interactively searching through all the layers.
  - The search is performed in windows based intersection of the helix with the next layer and the estimated uncertainties on that intersection location.
  - Bad hits will lead to larger windows and biased positions
  - A solution is to record that you found a hit so you can keep track of whether you have found all the expected hits, but to not add the hits to track fit.
  - The Hits has a goodHit flag so it is easy to implement this solution.



FNAL Software School: Lecture 3 Track Candidate Finding

Matt Herndon,

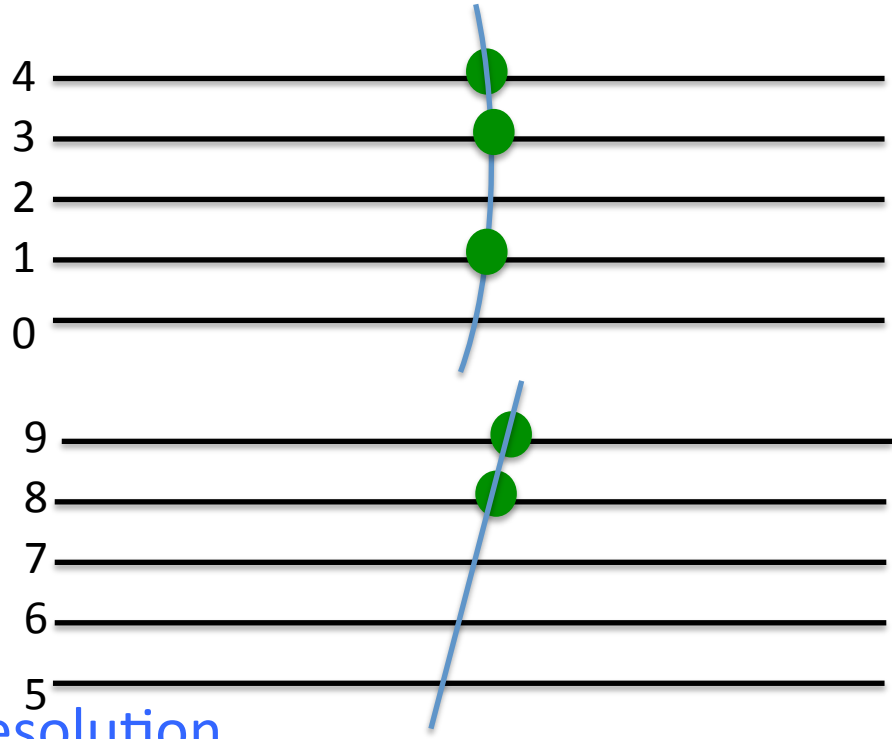
University of Wisconsin – Madison

Track reconstruction Problem

- Typical track reconstruction
  - Find an initial candidate track helix
    - Three points define a circle in the plane orthogonal to the magnetic field: r-phi (x-y) view
    - Two points define a line in the r-z view
    - In our case five Hits 3X and 2 Z or SAS
    - Minimal set of information needed to define the helix trajectory
  - Use that helix to define the locations to look for hits in the other layers
  - Iterate through all the layers.
  - Fit all hits to form the final track

Track Candidates

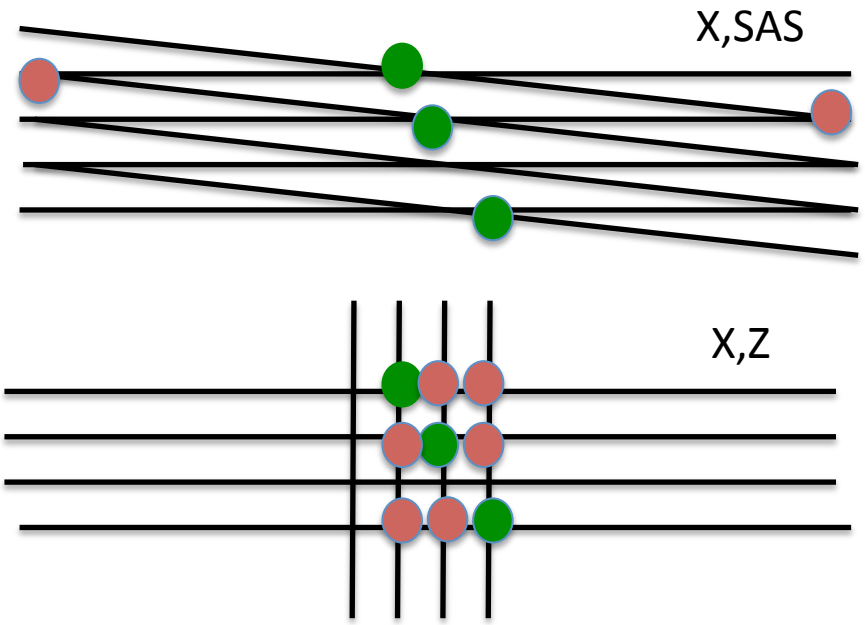
- X-Y view:
  - Looking down the Z axis, the axis of the magnetic Field
- Y-Z view:
  - Straight line
  - F = qv x B
  - No effect on z momentum



- Which layers to use for cands?:
  - 4, outer layer has best angular resolution
  - 0,1,3 equal, 2 is the worst
  - SAS 8,9 are have less ambiguity when combined with 4,3
  - Many combinations would have to be searched to allow for inefficiency on any one layer.

Detector design

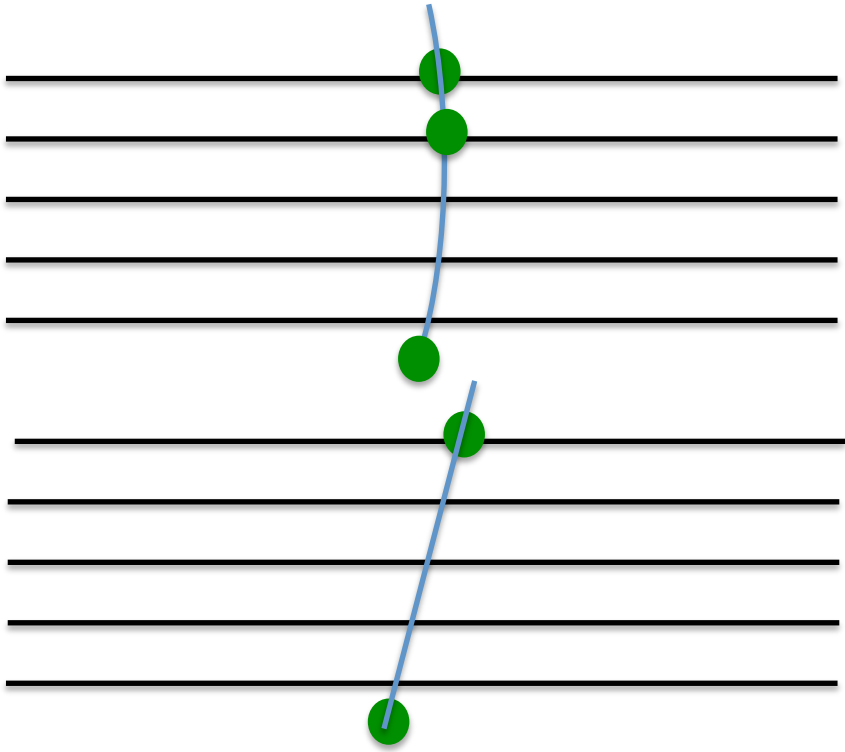
- Detector design and tracking
  - For this detector pairs of layers with SAS strips will drive the initial tracking phase
  - If there are hits on both layers of an X,SAS pair you can check whether the strips intersect.
  - At 1 degree you an SAS strip only overlaps 1.1% of the X strips



  - With 10 track all the pairing are likely to be unique. Even with more there wont be many ambiguous pairing unless you have 100
  - Without the overlap you would have to test 100 pairings instead of one or two.

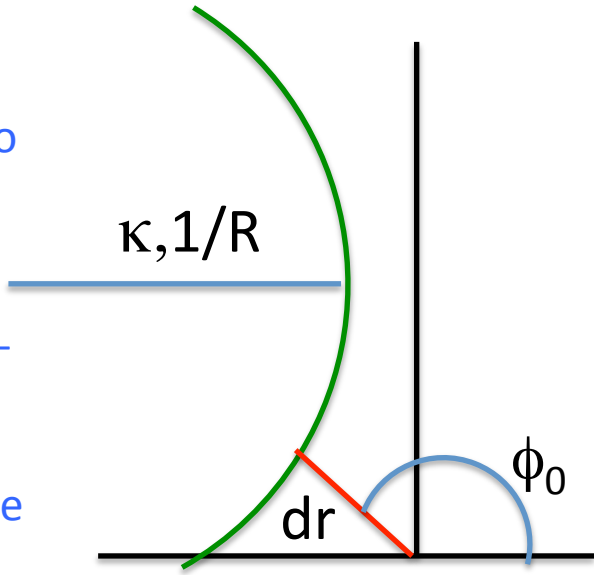
Track Candidates

- Another solution using 3 hits
- Using the primary vertex
  - Good: no inefficiency to find PV
  - Bad: what if the particles did not come from a primary vertex
    - Long lived particle decay products
    - No primary vertex – neutrino physics
- 2X1SAS algorithm
  - Use PV
  - Search 4, 3 and 9
  - Combination of 4+9 reduces conbinatorics and gives a Z position
  - Does not allow for inefficiency!
  - Should be interesting to compare and test for:
    - Inefficiency: overall and vs dR or dZ
    - execution speed



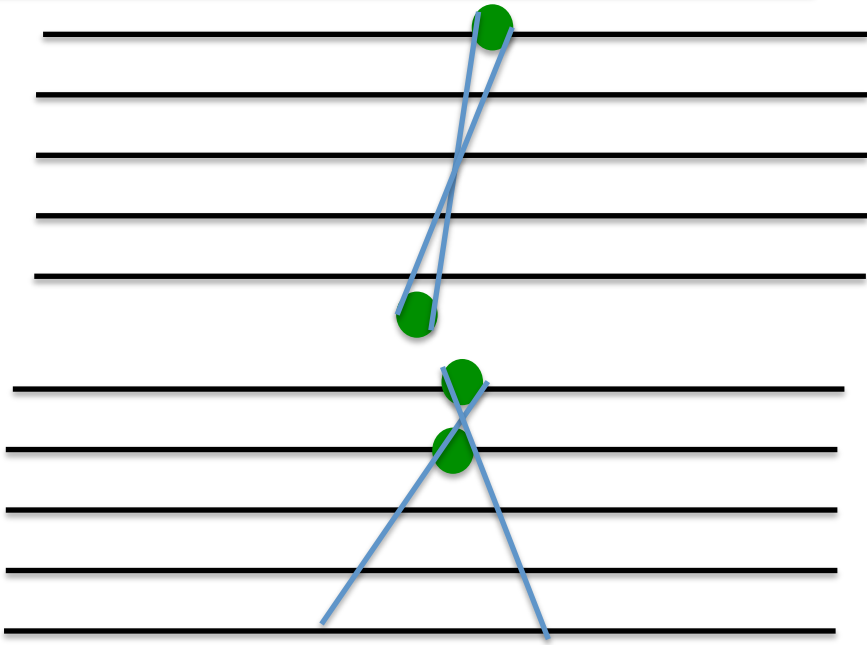
The Helix

- The track candidate will allow us to generate an initial Helix
- There are many ways to parameterize a helix
  - An initial starting point, the momentum, and the magnetic field.
  - We want a parameterization that will be optimal for track fitting
- Helix parameterization: follows physics/1305.7300
- Relative to a reference point a helical trajectory can be described by five parameters
  - dr, impact parameter in x-y. Distance of closest approach to the reference point (0,0,0) in x-y
  - Phi0, angle from the reference point to the point of closest approach in x.y
  - kappa, inverse of pT, singed to indicate handedness, + right-handed negative charge, - left-handed positive charge.
  - dz distance of closest approach to the reference point in z
  - tanLambda= cot(theta) pz/PT where theta is the angle to the positive z axis
  - These choices avoid several issues: Example: kappa goes to zero for very high pT tracks and changes sign smoothly



Track reconstruction and Hits

- Hit resolution is a critical issue in track reconstruction
  - Hit resolutions are needed to understand how big an area must be searched for hits on each layer.
  - Hit resolutions are needed to perform the final track fit

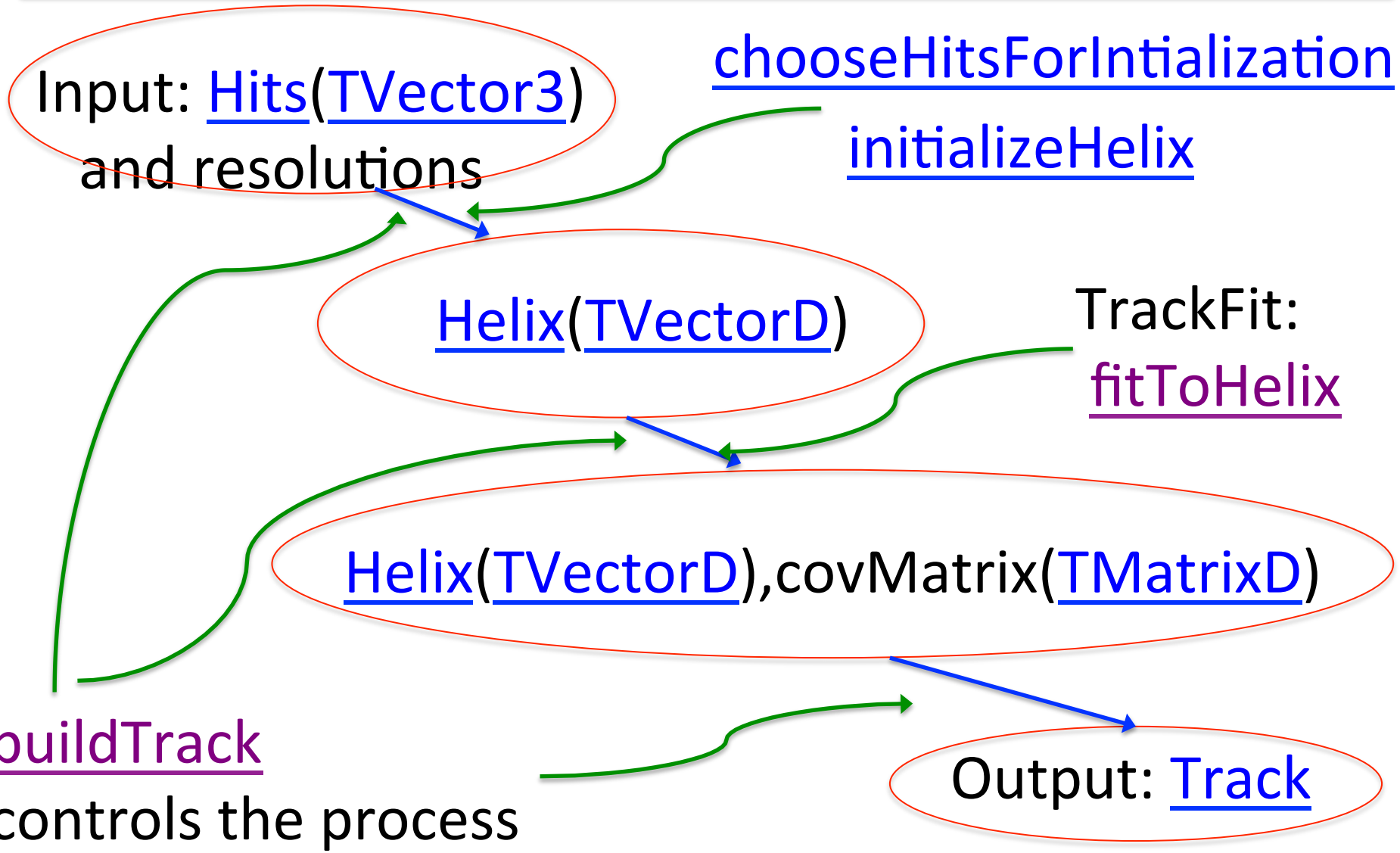


- With poorly understood resolutions track fit will calculate incorrect uncertainties
  - You will search for hits in an area that is too large or too small
  - Your physics analysis will not be reliable

The TrackFit

- A conceptually simple Chi2 fit
- Input: Hit position and hit resolutions (and an initial helix)
- The fit
  - Given a helix determine residuals between the helix and the hits on each layer
  - Determine a chi2 based on the resolution of the hits
  - Determine what changes in all 5 helix parameters change the chi2 by one unit. Essentially the derivatives of the helix parameters. An analytically solvable problem using matrix math.
  - Alter the all 5 track parameters to improve the chi2 and recalculate. The change to apply analytically calculable using the derivative matrices above.
  - Iterate progressively reducing the size of the alteration each time.
  - Typically converges in 10 iterations in our code!
- Output: A Track
  - Final helix parameters
  - Covariance matrix – diagonal elements give the 1σ uncertainties on the parameters based on the matrix calculations above
  - Chi2 and nDof
  - Also can calculate uncertainties at any point along the helix
- The input that directly effects the uncertainties is the hit resolutions

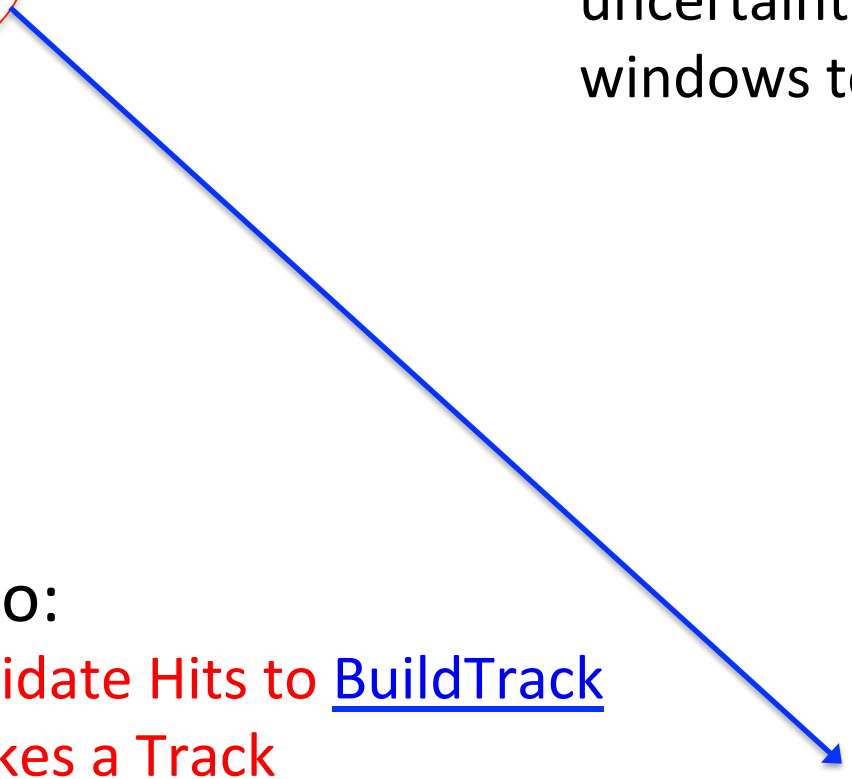
The Objects and Algorithms



Algorithm Abstraction ☺

Input: Hits

Full Track needed to calculate uncertainties and define windows to look for more hits



- What you do:
  - Give candidate Hits to BuildTrack and it makes a Track

Output: Track

Design Choices

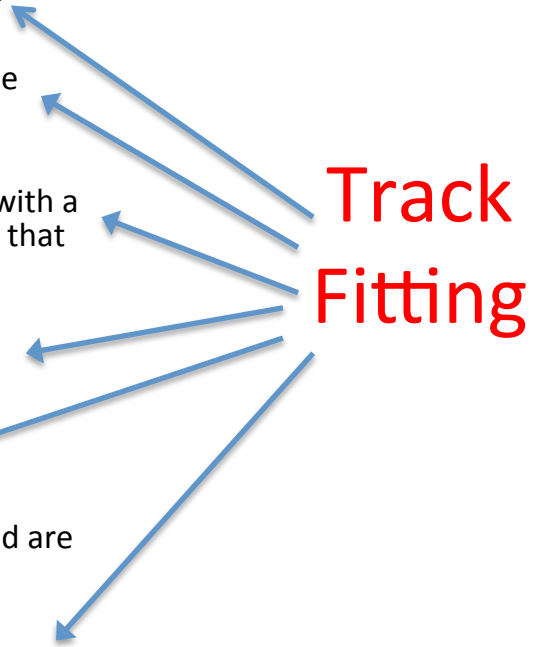
- Design choices
  - Use of TVector and TMatrix allows a large part of the math be reduced to Matrix multiplication
  - The internal functions of the objects perform the multiplication, transpose and inversion operations
  - The track fit follows physics/1305.7300
  - The article explains the fit in terms of matrices that must be constructed and then transposed, inverted and multiplied
  - The code reads exactly that way
- Algorithm abstraction
  - Allows the code to structured to follow the logical steps outlined in the article

Design Choices

- Design choicesL Free functions
  - Most of the parts of the trackFit are free functions
  - InitializeHelix can by used to make a helix for the TrackCandidateStrategy
  - Derivative calculations are used to find uncertainties on the intersection with a layer
  - intersectWithPlane used to place hits in the simulation Modules
  - Two above used to calculate residuals between the track and hit on layers and test that track parameter uncertainties are well measured.
  - Coordinate transformations are used in many places.
  - Free functions are used extensively where even compact calculation has to done that has utility in many places.
  - The std library is primarily composed of free functions

Course Goal Revisited

- Goal
  - Learn how to write well designed and effective reconstruction software that integrates well into a large scale computing project
- What does that mean?
  - Follows best practices
    - Many of the best practices are there to facilitate the elements of the goal.
  - Easy to read
    - A user or other developer can read and understand quickly what your code does.
  - Easy to maintain
    - Need to improve something? Well designed code will often let you do so with a change at a single point without effecting any of the classes and functions that use the code you've changed.
  - Simple
    - The simplest solution is used when various solutions are equally effective.
  - Safe
    - Data elements are safe from being altered when they should not be.
  - Fast uses minimal memory, Not really
    - A fact of particle physics computing is that we deal with large data sets and are CPU and memory limited.
  - Effective
    - Defined in terms of the project goal. In reconstruction typically, efficient, accurate, and low fake rate (reconstruction of Hits, Tracks that don't exist!) reconstruction of objects.



Tracking

- A complex reconstruction task
- Design considerations
  - Identify the primary logical steps
    - Simplifies our understanding of the problems
    - These will likely be at the Module level
    - Allows assessment of the performance of each step since we can write the results to objects and use them in a histogramming module
  - Identify subtasks
    - Is the problem complex enough that we need a hierarchy of task(yes)
    - Are task unique? One member function.
    - Are they performed in several places. Free functions.
    - Do they have similar structure.

Tracking Hierarchy

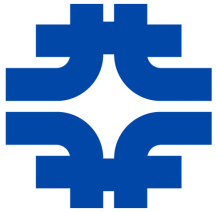
- Candidate reconstruction: TrackCandidateModule
  - Find the initial helix used to search for hits.
  - Module level since we need to evaluate the success of this task, without the candidate we can't proceed to find the track.
  - Specific candidate strategies: TrackCandidateStrategy
    - A class used in TrackCandidateModule since it is complex
    - Probably several versions, but all with the same general structure.
    - Find the hits to build the candidate: findHitCandidates
      - Calls goodCandidate that evaluates the quality: free function
    - Build track given the Hits
      - Needed to estimate uncertainties for the next step.
      - Calls goodCandidate since more information is available to evaluate the candidate.

Candidate Finding

- Module: TrackCandidateModule
  - Input: reconstructed HitSet
  - Ouput: TrackSet of track candidates
    - A full TrackSet so we can examine it to access performance
- Algorithm
  - Class: TrackCanidateStrategy2X1SAS
  - findHitCanidates
    - Loops over layers over the HitSet and forms all combinations of Hits on layers 4,3,9
    - calls intersectStrips to determine if an SAS hit is consistent with an X Hit
      - The overlap is 1.1% of the detector. Removes many candidates
    - Calls initializHelix to make a Helix
    - Calls goodCandidate based on a TrackingSelector to decide whether to keep the candidate
  - Builds candidates using buildTrack

Project Specifications

- Track Candidate Finding
- Project Goal
  - 1) Improve TrackCandidateStragegy2X1SAS to search additional layers in case of detector inefficiency.
- Input: HitSet
- Output: TrackSet of track candidates
- Using
  - TrackCandidateStrategy2X1SAS??
    - Make a copy of the other strategy renamed
    - Account for hit finding inefficiency. i.e. make candidates using at least one more layer 4,3 , 4,1 and 3,1. Try more than one SAS layers as well 9 and 8 as appropriate. Make all combinations that include an X, SAS pair.
- Code
  - You will produce a findHitCanidates function with appropriate use of
    - intersectStrips, initializHelix, goodCandidate
- Simple performance evaluations
  - Size of the candidate list (in printout)
  - Run gprof to profile and get execution time of various modules. Compare TrackCandidateStrategy2X1SAS and new strategy




Backup

Tracking Hierarchy

- Track reconstruction: TrackRecoModule
  - Module level since we need to evaluate the success of this task
  - Search for hits on each layer in turn: Layerfinder
    - A class because of the complexity.
    - Find all candidate hits for one track on one layer: findHits
    - Build the tracks: buildTracks
    - Call goodCandidate to evaluate quality.
    - Call contentionFilter that chooses best possibilities
  - Call goodCandidate to evaluate quality
  - Call contentionFiler that chooses best posibillities.