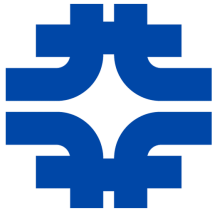


FNAL Software School
Day 5

Matt Herndon,

University of Wisconsin – Madison

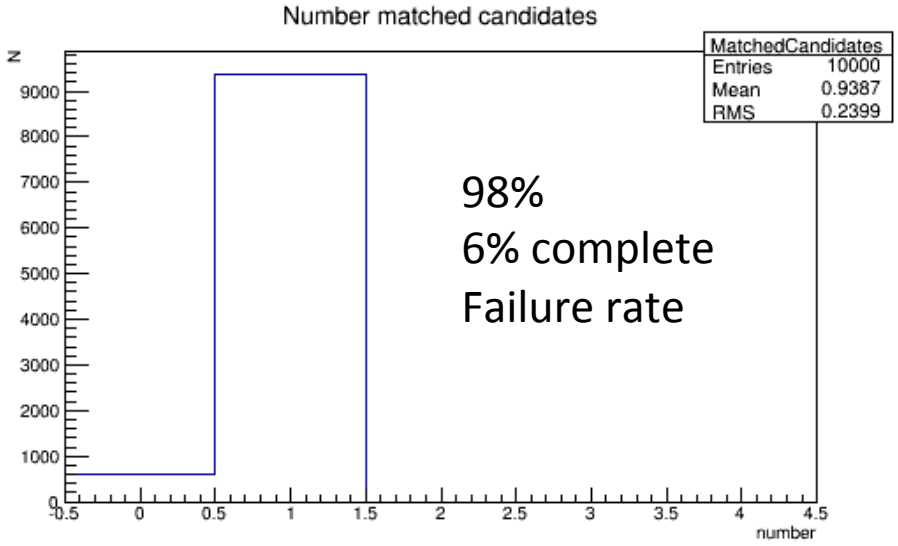
Today's Activities

- Review Day 4 exercise
  - Performance of TrackCandidateStrategy2X1SASML
    - ML searches multiple combinations of layers
- Lecture
  - Tracking performance
  - Review of software engineering principles
- Daily project:
  - Time to run track reconstruction!
  - Profiler demonstration at 1PM
- Should not be a long day

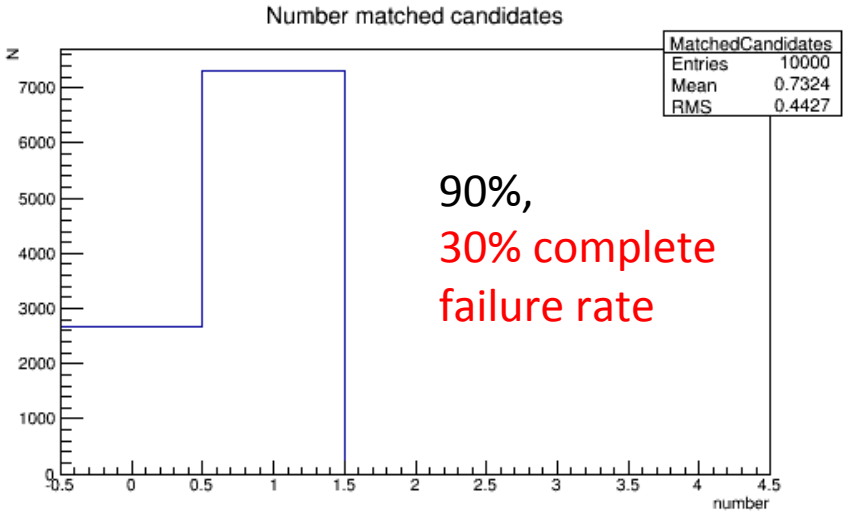
Track Candidates

- ML searches multiple combinations of layers
  - Add redundancy so that at least one candidate is found for each real track
  - However, 4 candidates can be found for each real track. They will be eliminated later during track reconstruction.

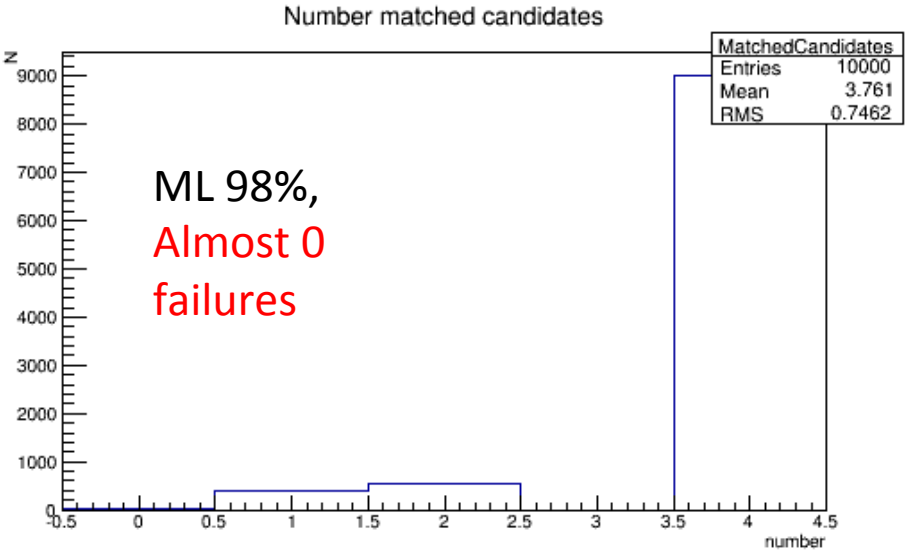
Track Candidates



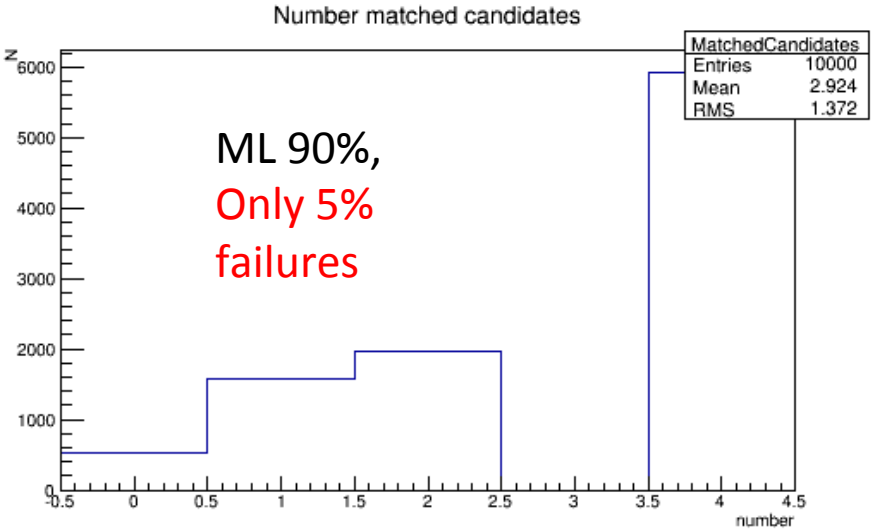
98%
6% complete
Failure rate



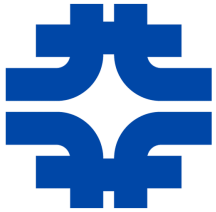
90%,
30% complete
failure rate



ML 98%,
Almost 0
failures



ML 90%,
Only 5%
failures



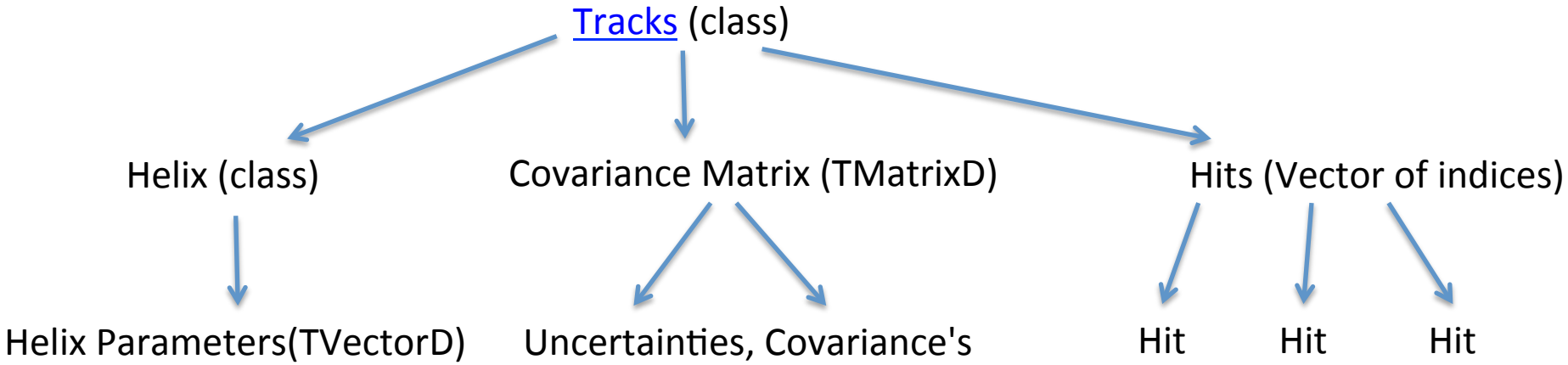
FNAL Software School: Lecture 5
Track Reconstruction and Software Engineering

Matt Herndon,

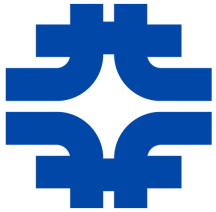
University of Wisconsin – Madison

Data and Algorithm Abstractions

- The DataObjects in our code represent a hierarchy of abstractions.



- A user just sees clearly named functions that get the various classes or values. As the developer you can then choose the underlying containers and classes and even change them
- Requires developers and users to follow best practices like using auto const & loops




Algorithm Abstraction

- Consider our clustering algorithm
- constructsHits
- (loop over layers)
  - findClustersOnLayer
  - (loop over strips)
    - test if adjacent
      - add to single cluster adc vector if adjacent
      - add single cluster adc vector to vector of clusters if not adjacent
    - buildHit
      - calculateStripPositionFromCluster
      - calculateLocalFromStripPosition
      - calculateGlobalFromLocalPosition
      - make the Hit using Hit constructor
  - (end loop over strips)
- (end loop over layers)
- Operations like those in buildHit are encapsulated and abstracted away from the user. buildHit then serves as the single correct way to make a hits.
- The top level of algorithm abstraction are the Modules which are designed for a single task.

Data Safety

- Consider the HitSet
  - Data encapsulation and const correctness in Hit and Hitset
    - All the member data is private
    - Access to values only through "const" get functions. const reference if a larger object – not a pointer!
    - An accidental "hit._layer = 3" rather than hit._layer == 3" is not possible.
    - Only necessary non const member is insertHit
  - Initializer syntax
    - All Hit private member initialized in constructor via the initialization list
    - The user can't forget to initialize something
    - No non const functions needed to set the data members. Data can not be changed after it is created.
    - Faster. The constructor will default initialize anything not in the list. If you initialize it in the body of the constructor or later it gets initialized twice.
- Enforced by the event framework which returns all event data objects as const
- Also use for algorithm code and module classes that produce and consume data.
  - In your own code you don't want to accidently alter the data objects your are creating and using

Integrated Design

- Performance assessment has to be built into the system from the beginning.
  - Required correct choices in data object design and a full set of performance assessment Modules designed to work with the reconstruction modules
- In reconstruction the elements of performance assessment are the core of your project goals
  - You want to achieve excellent efficiency, accuracy of both parameters and uncertainties, low fake rate, and fast execution speed with low memory use
- Also you have to design for issues like
  - Input and output of data
  - Calibration and application of calibrations
  - Automated testing of the code

Why do all this

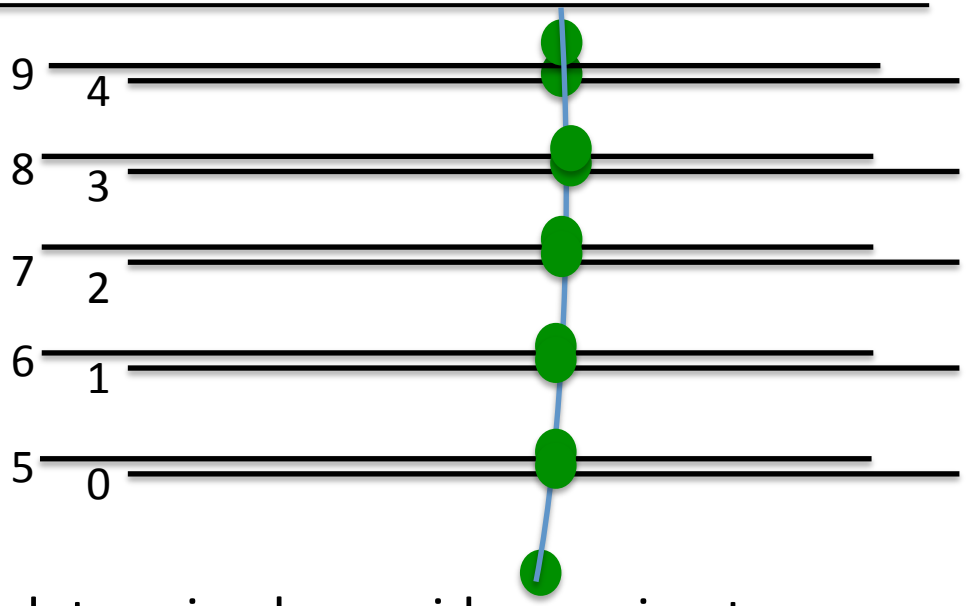
- const correctness
  - Data safety
- Initializer constructor calls
  - A user can't incorrectly construct the object neglecting to fill in certain information
  - Data safety
  - Faster
- Const references for access to all Objects instead of pointers
  - Data safety
  - Avoid segmentation faults and memory leaks
- Algorithm abstraction
  - Better organized and readable code
  - Maintainability: Allows upgrades to underlying methods without effecting users
- Integrated design
  - Plan and design in advance so that your software includes methods both meet and to test/demonstrate the project goals are met. Typically requires infrastructure that spans the dataObjects and algorithm code

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
    - 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

- We now have an initial seed track
  - Iterate searching for hits in each layers
    - The track parameters are used to predict where to look for hits
    - The estimated uncertainties determine how wide a region to search
  - Refit adding each hit
    - Improves the estimation of parameters when looking for the next hit
  - Apply quality criteria at each step to remove fake tracks
  - Compare all tracks at the end to select the best ones



Tracking Hierarchy

- Track reconstruction: TrackRecoModule
  - Module level since we need to evaluate the success of this task

TrackRecoModule

TrackRecoStrategy2X1SAS (recoTracks)

1 LayerFinder (findTracks)

2 contentionTrackSetFilte

a findTrack

b simpleTrackSetFilter

once

i findHits

c duplicateTrackSetFilter

a-c For 10 layers

i-iv For n tracks

ii buildTracks

iii goodTrack (Selector)

iv bestTracks (Filter)

Controlling Tracking

- Several places you can tune the tracking to improve performance
  - Order that layers are called
    - TrackRecoStrategy2X1SAS (findTracks)
  - Whether to call the contentionFilter after each or specific layers
    - TrackRecoStrategy2X1SAS (findTracks)
  - chi2NDof and minPt cuts used in goodTrack selectors
    - configurereco.txt

Controlling Tracking

- Some general ideas
  - Order of layer search is important
    - Once the layers used in Candidate construction are searched the duplicateFilter will remove most of the the multiple patterns searched.
    - Search windows are determined by using the helix parameter uncertainties to calculate an uncertainty window on the next layer. You should avoid extrapolating over many layers.
    - The resolutions of candidates are different in X and Z. Large uncertainties will mean large search windows. You could search the layers with small uncertainties first.
  - Cutting very tightly on chi2NDof or using the contentionFilter early can reduce efficiency, but may boost performance

Performance Assessment

- Assess
  - Efficiency
  - Accuracy of parameters and uncertainties
  - Fake object rate
  - Execution speed and memory usage
- A General principal
- Need a mechanism of unambiguously associating reconstructed objects to generated ones to assess efficiency, resolution and fake rate.
- Helix matching works well with reconstructed tracks.
  - Δ(gen-reco)/σ for all five track parameters
- Efficiency and fake rate
  - Select on Δ(gen-reco)/σ for all 5 track parameters
  - Eff: = (matched Reco)/(Gen)
  - Fake = (unmatched Reco)/Reco

Todays Project

- Performance assessment of tracking
- Update youe FNALComp project
- Edit TrackCompareWithGenModule
  - Insert counter to keep track of, genTracks, recoTracks, recoTracks matched to gen
  - Print out efficiency and fake rate in endjob
  - Consider the tolerance used for matching tracks. Remember you are matching 5 parameters, so 3sigma would have a noticeable effect when repeated 5 times.
- Performance assessment
  - Determine execution speed – run by typing: time ./trackReco > logfile
  - Determine how many 10 track events you can run in order 1 minute (~100 on my computer)
  - Measure eff and fake rate
  - Look at histograms from TrackCompareWithGenModule
  - Start adjusting the tracking parameters. You should be able to get a factor of 2 in speed. Eff, fake rate, speed should be your three metrics