8.882 LHC Physics

Experimental Methods and Measurements

Charge Multiplicity Measurement

[Lecture 5, February 18, 2009]

Organizational Issues

Recitation session this week

- will be over video
  - format slightly different than last time
- prepare questions for the session

Charge Multiplicity Assignment

- is already available on the web
- start early to get a feel for how much work it is
- pointer:
  http://www.cmsaf.mit.edu/twiki/bin/view/Class8882/NChargedDescription

Physics

Colloquium Series

'09

Spring

The Physics Colloquium Series

Thursday, February 19 at ***4:15 pm*** *in room* ***34-101***

Matthias Burkardt

New Mexico State University / Jefferson Lab

"Nucleon Spin Physics"

For a full listing of this semester's colloquia,
please visit our website at **web.mit.edu/physics**

Lecture Outline

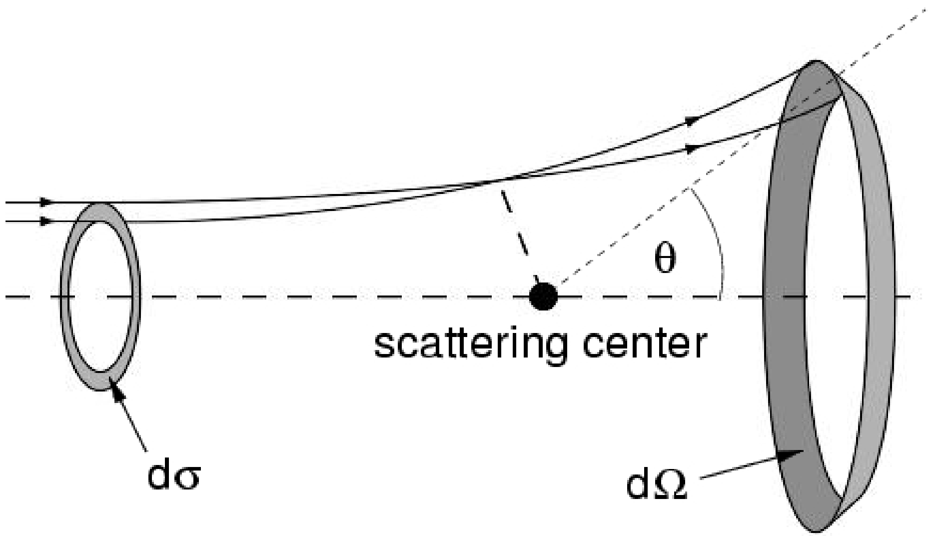
Charge multiplicity measurements

- observables and experimental status
- CDF data and how they are organized
  - trigger conditions and information
  - contents of the “ntuple”
- prototype analysis
- main components for full analysis
  - pile up events
  - secondary interactions

General Scattering Considerations

Why is the charge multiplicity interesting?

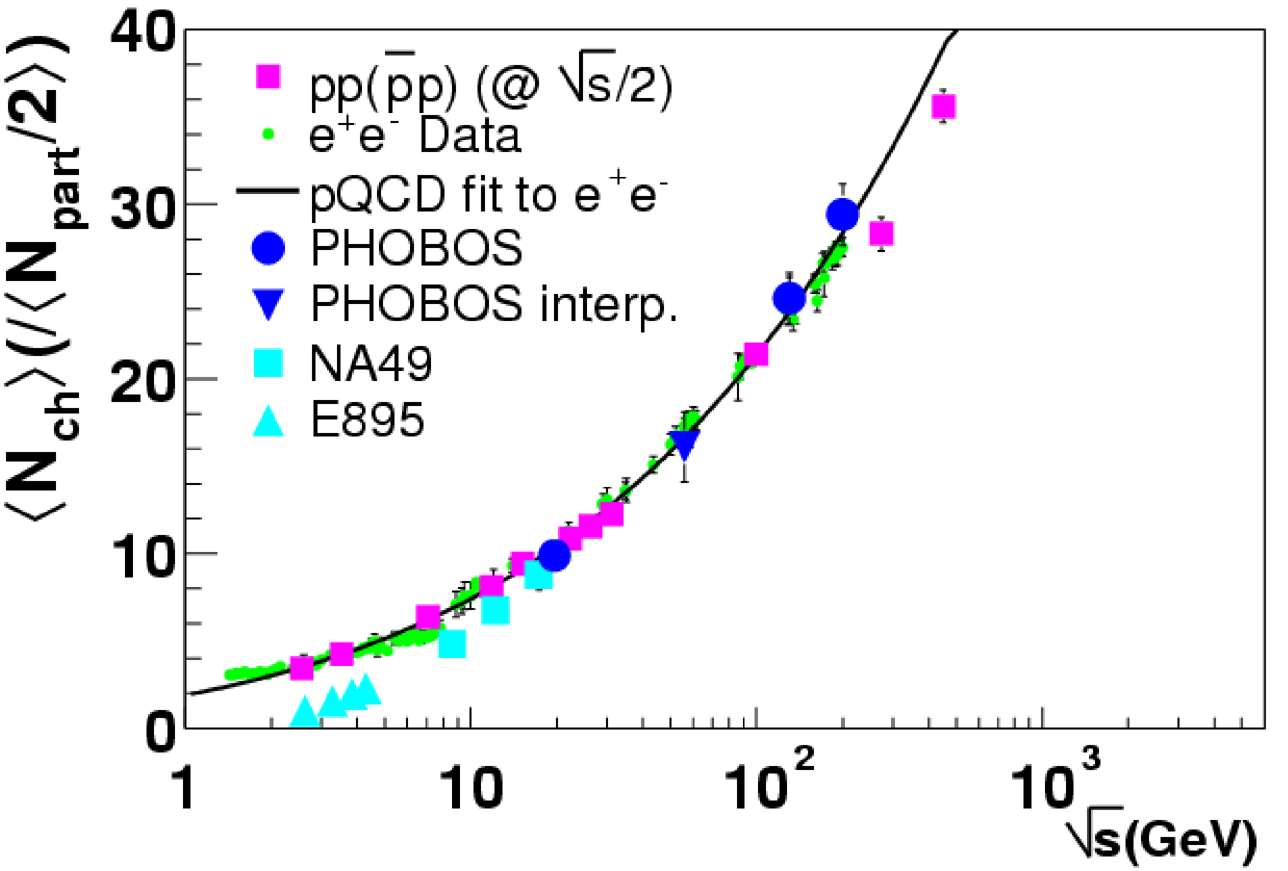
- in classical collision: no new particles get created
- one particle in, one particle out at angle θ
- for particle physics new particle creation main purpose
  - inelastic collision
  - creation of new particles with available energy
- some properties of newly created particles
  - how many?
  - type or mass
  - detailed kinematics
- difficult to predict
  - low momentum regime
  - non-perturbative QCD
  - no high-profile measurement in HEP



Charged Particles in Heavy Ions

Very relevant

- simple observable to characterize the fireball
- identify state transition using charge multiplicity

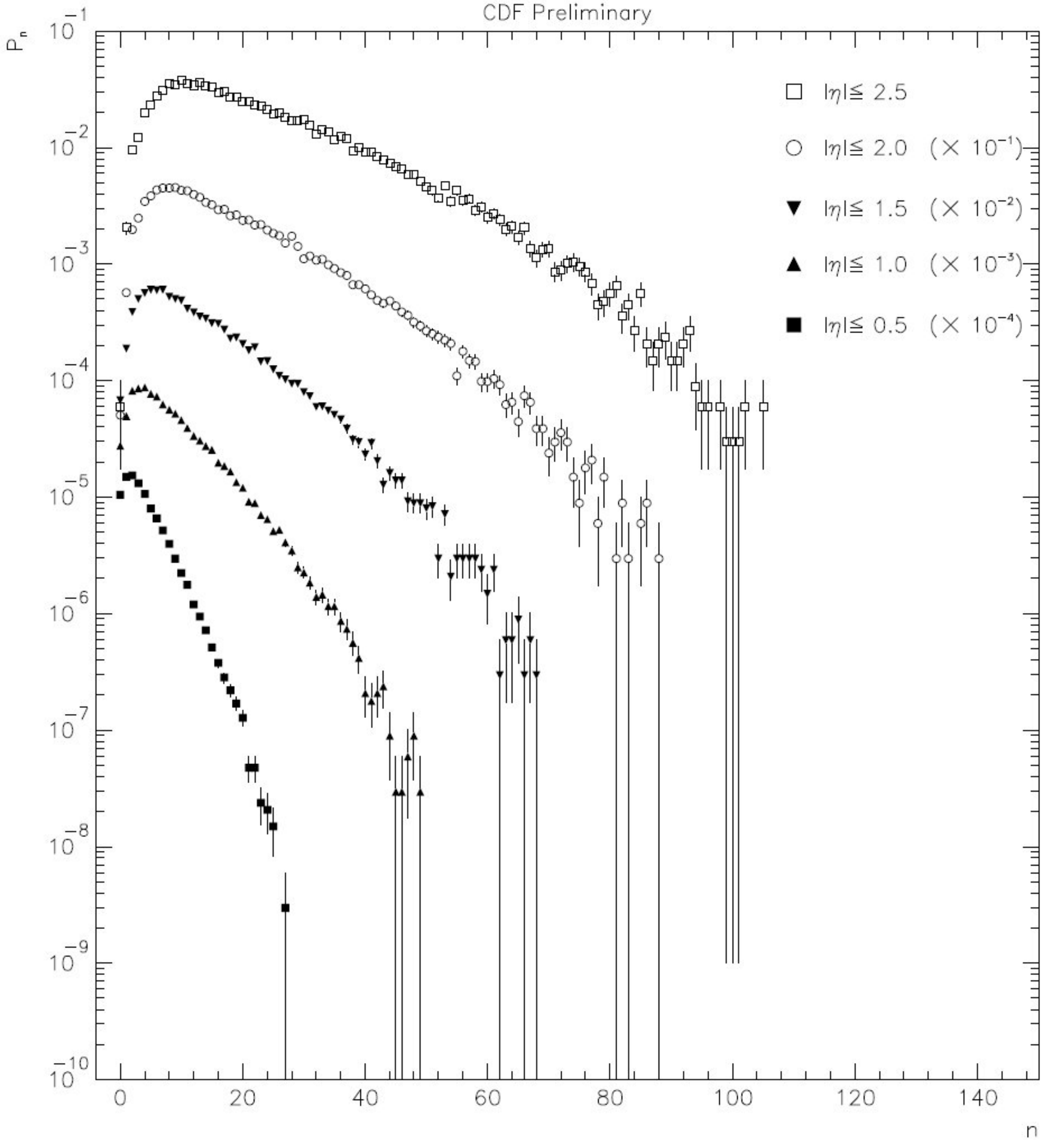


Compare interactions in HI collisions with proton-proton or proton-antiproton collisions → are HI interactions independent?

CDF Charge Particle Multiplicity

Figure

- *x* axis: number of tracks in event, *n*
- *y* axis: number of events over total events, P_n
- various symmetric pseudorapidity intervals
- at 1800 GeV (Run I)
- careful, curves are artificially normalized to show them separately



Fit Charge Multiplicity Distribution

Fitting function, [2] TWiki

- negative binomial distr.

$$P(n; \langle n \rangle, k) =$$

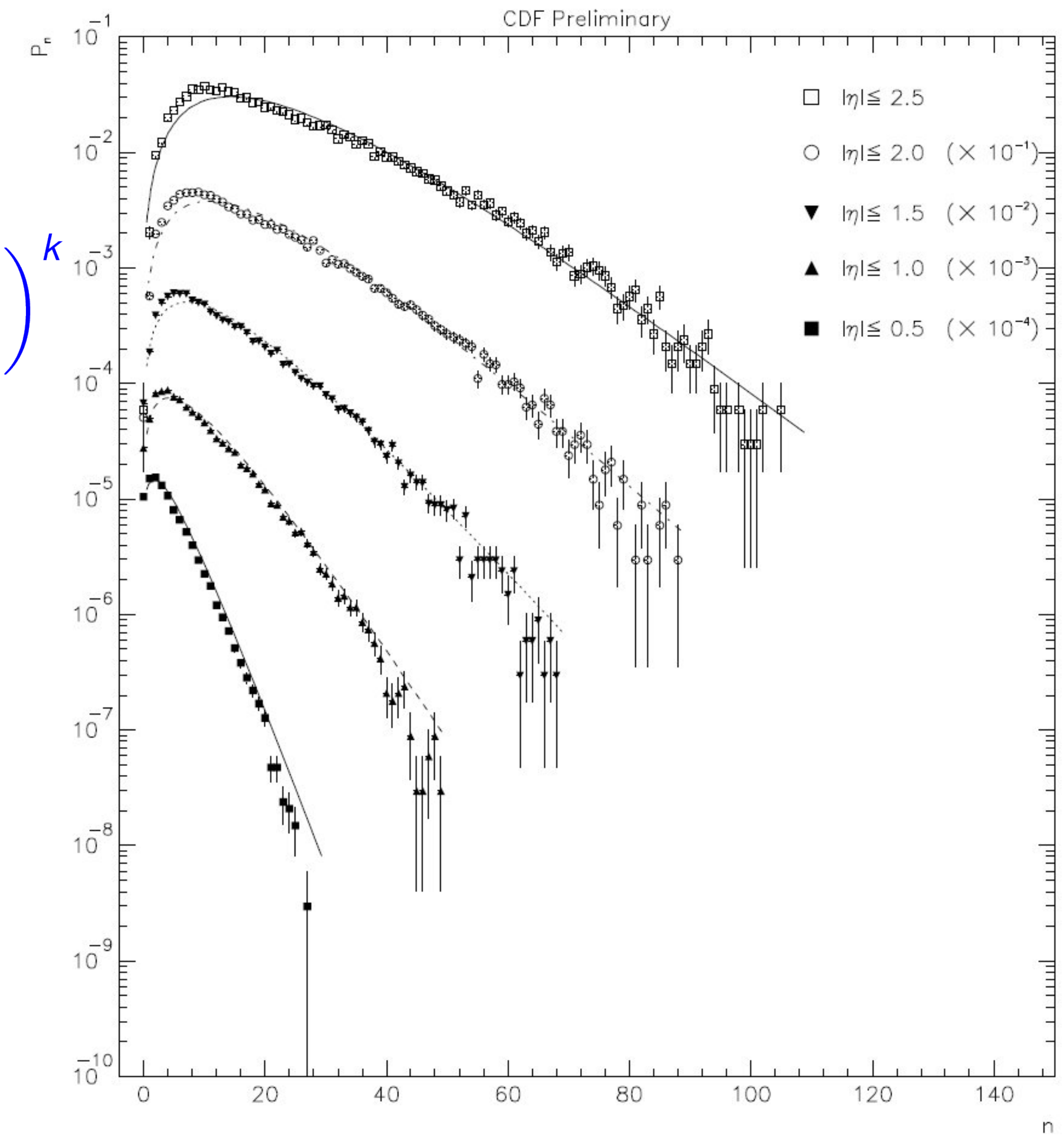
$$\left[\begin{array}{c} n-k+1 \\ k-1 \end{array} \right] \left[\frac{\langle n \rangle / k}{1 + \langle n \rangle / k} \right]^n \left(\frac{1}{1 + \langle n \rangle / k} \right)^k$$

Two parameter function

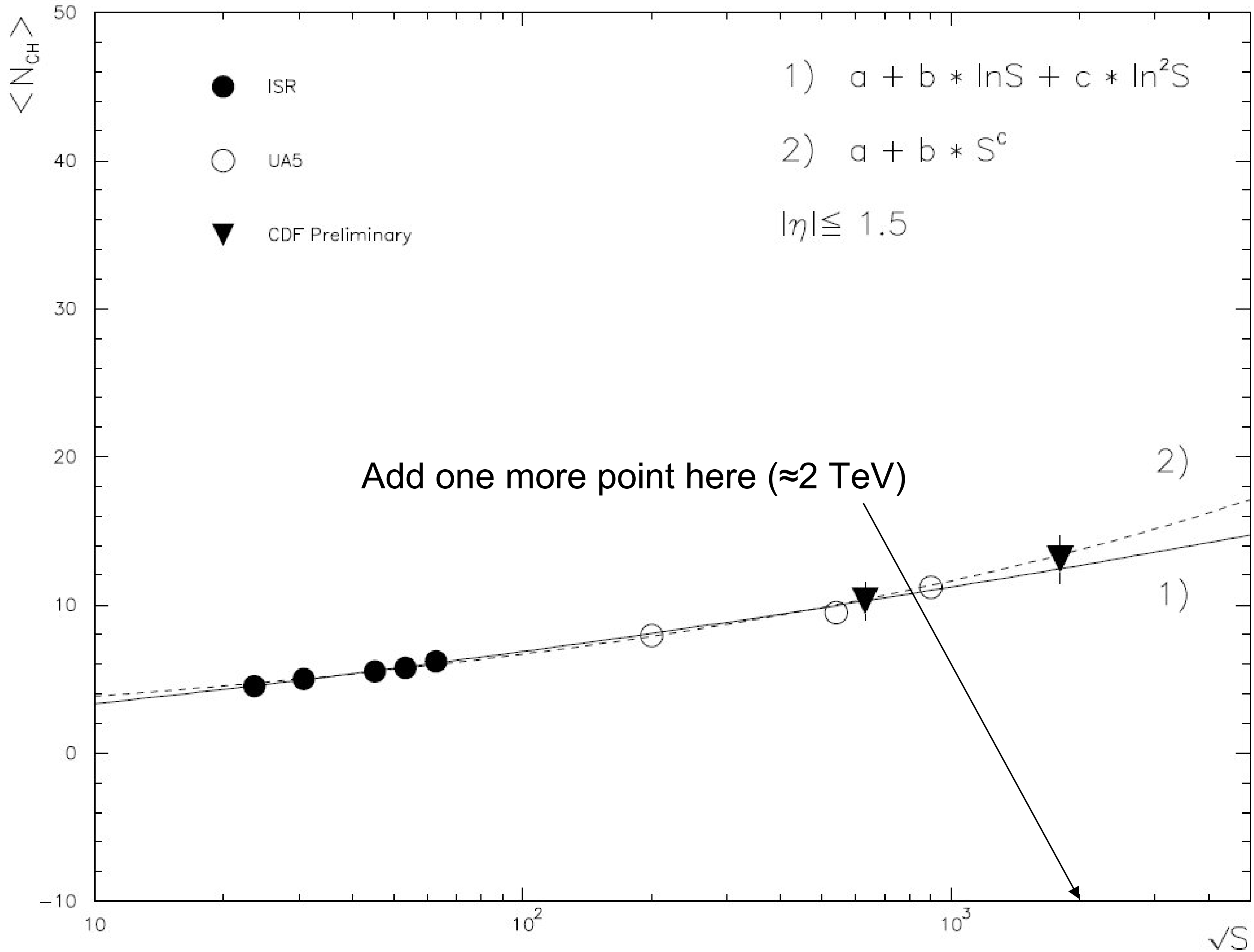
- <*n*> - average number of tracks
- *k* - shape parameter

Implementation in Ana/fit

- negativeBinomial.C
- assumes that you ran
  - Ana/script/runNTrks.C



Multiplicities versus Energy



Some Jargon You Should Know

Fill or store

- when operators fill machine with a fresh set of protons and antiprotons (or protons and protons at LHC)
- stores can take up to several days: luminosity drops exponentially in beginning, at some point one has to stop

Run

- at CDF: bunch of data with constant detector conditions
- ideally just one per store, often many per store

Luminosity section (LS)

- smallest unit of data for which luminosity is determined
- LS kept together in one file, consider asynchronous data taking

Hadron Collider Triggers

General

- trigger is crucial in hadron collider
- event rate: 3 (40 CMS) MHz, written to tape: 100-200 Hz
- trigger significantly shapes kinematic of output events
- problem: readout time of detector is long

Rough overview of CDF Trigger

- trigger system is pipelined and organized in levels
- pipelines allow to process in parallel
- levels allow to use increasing amount of information with increasing amount of decision time
- level 1: implemented in hardware, very fast (input 10^6)
- level 2: implemented in firmware, pretty fast (input 10^4)
- level 3: implemented in software, fast (input 10^3)

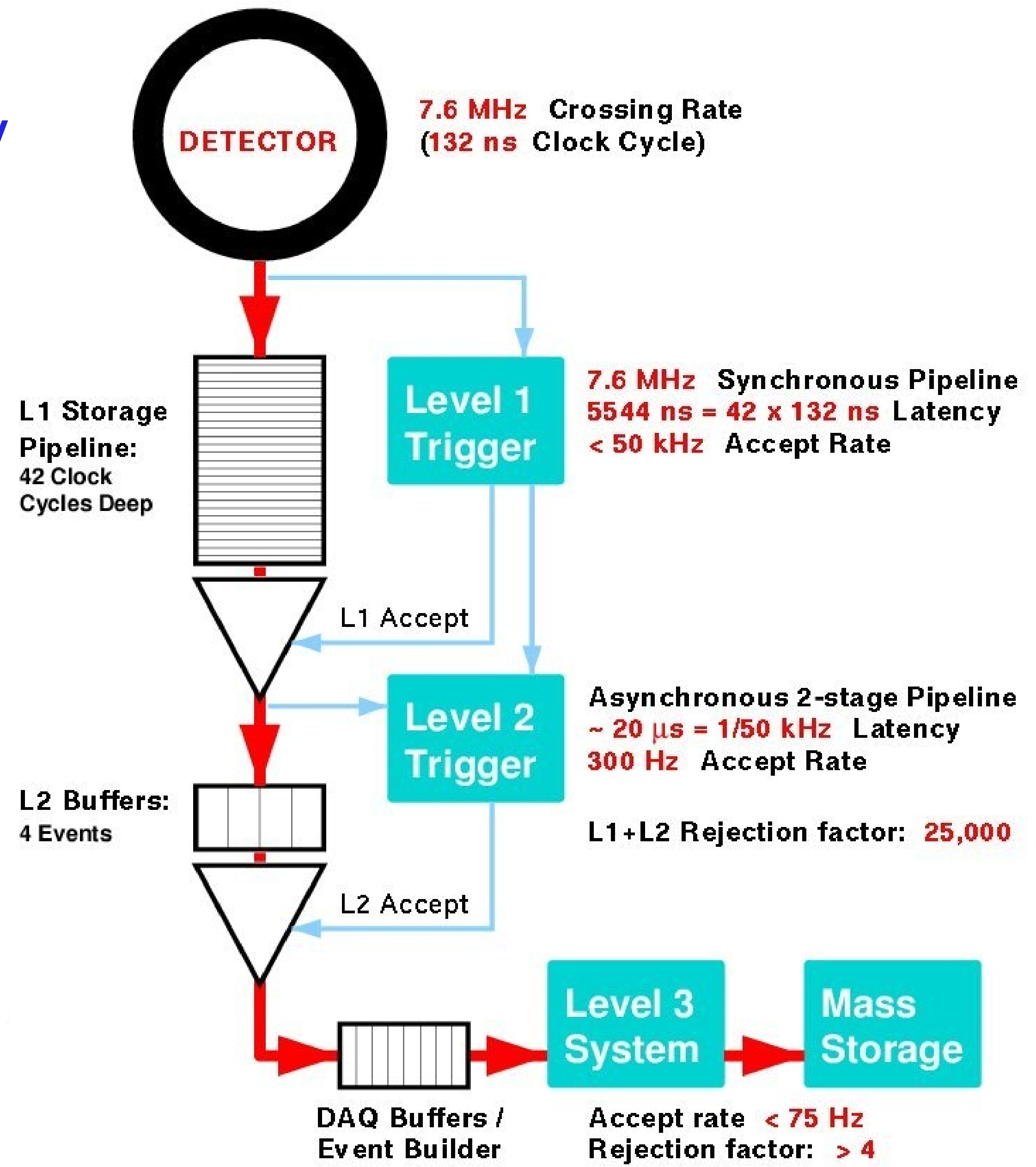
CDF Trigger System Overview

Properly join levels

- event needs to have fully defined level 1, 2, 3 sequence
- sequence called path
- path important for efficiency calculation

Our trigger

- minimum bias trigger
- looks into forward region to trigger on some minimal activity



Code Organization: BottomMods

General

- code sits in ~/8.882/614, please keep it there
- AC++ is general CDF reconstruction software, you should have nothing to do with AC++ (you cannot even run it)

BottomMods: a *B* reconstruction package

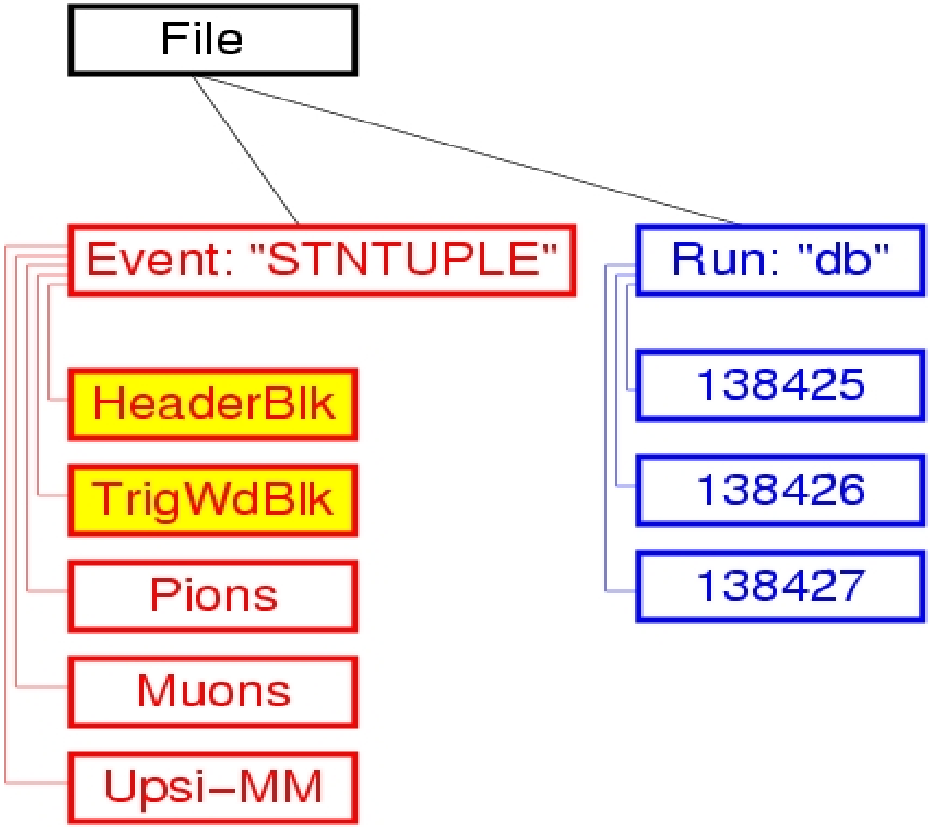
- cands – reconstruction of particles purely AC++
- mod – interface between AC++ and ntuple format
- stn – definition of the ntuple data blocks
- ana – general analysis modules and other helpers
- taggers – implementation of tagging algorithms
- tools – generic independent tools

Do not be afraid, just take a look at it!

Organization of CDF Data

Data in BStntuple format: a root TTree

- all files are split into two main TTrees
  - event based information (most): “STNTUPLE”
  - run based information: “db”
- run tree information
  - run number,
  - trigger table
  - beamline, *etc.*
- event tree information
  - header and trigger info
  - “Pions”: all tracks reconstructed as if they were pions
  - “Muons”: tracks reconstructed as if they were muons
  - *etc.*



The Header Block

General idea

- small datablock which contains very general event info
- used to very quickly/roughly select events, always loaded

TStnHeaderBlock

- int EventNumber () const;
- int RunNumber () const;
- int SectionNumber() const;
- int McFlag () const;
- *etc.*

The Trigger Word Block

General idea

- keep track of trigger decision at each level using the name of the trigger path

TStnTrigWdBlock

- int GetListOfPassedTriggers
  - (const TStnTriggerTable* Table,
  - const char* Name,
  - int Level,
  - TObjArray* List);
- TStnTriggerTable connects trigger name with bit index and is stored in the run dependent TTree “db”
- use analysis module: TPrereqFast which implements all of this for the analysis

Particle Block and Contents

TStnDataBlocks

- base class for a block of data
- block contains several instances of particular type of data

TStnStableBlock

- implementation for a Block of TStnStable

TStnStable

- object representing data stored for stable particle

TStnDecayBlock

- implementation for a block of TStnDecay

TStnDecay

- object representing data stored for decaying particle

Stable Particle Data

Internal information of TStnStable

- int   fHits;   // mostly hit informations
- int   fStat;   // various interesting things
- float fPhi0,fPhi0Err; // 5 track pars/uncertainties
- float fD0, fD0Err;
- float fPt, fPtErr;
- float fZ0, fZ0Err;
- float fCotT,fCotTErr;

Compactly packed information in fHits and fStat

- fHits has mostly hit information
- fStat contains mostly other status information

Rest are five helix parameters with uncertainties

Stable Particle Data, continued

Hit contents of TStnStable

- int SiHitMask() const; // hit per layer mask
- bool HasL00 () const; // has at least one L00 hit
- bool Has2L00 () const; // has 2 L00 hits
- int NumSiPhi() const; // number of silicon r-phi hits
- int NumSiZ () const; // number of silicon hits with z info
- int NumSiSA () const; // number of stereo angle hits
- int NumSi90 () const; // number of silicon 90° hits
- int NumCTAx() const; // number of COT axial hits
- int NumCTSt () const; // number of COT stereo hits

Stable Particle Data, continued

Status contents of TStnStable

- int StnTrkId () const; // Track id
- int MatchedXft () const; // matched to XFT?
- int MatchedXftJ () const; // matched to XFT J algorithm
- int MatchedXftS () const; // matched to XFT S algorithm
- int PvIndex () const; // index in PV block
- int Charge () const; // charge of track
- int TrigMode () const; // is a Track Trigger track?
- bool MatchedSvt () const; // is matched to SVT?
- bool Matched4H4L() const; // 4 hits in 4 different layers?

Decaying Particle Data

Decaying particle general

- result of a vertex fit, which assumes detailed decay
- if assumptions correct data becomes more precise

Internal information of TStnDecay

- int fNDcys; // number of decaying particles
- int fNStbs; // number of stable particles
- int fTrgWord; // trigger for candidate only
- float fProb; // vertex fit probability
- float fChi2; // chi squared of fit
- float fRphiChi2; // chi squared of fit only in r-phi
- DecayData fPart; // data of this decay particle
- DecayData fDcys[2]; // data of decay daughter particles
- StableData fStbs [5]; // data of stable daughter particles

Decaying Particle Data, continued

Decay data

- float Eta; // pseudo rapidity
- float Phi0; // phi direction at zero
- float Pt; // transverse momentum
- float Mass; // mass as determined in vertex fit
- float MassErr; // corresponding mass uncertainty
- float Lxy; // decay distance in *xy* plane
- float LxyErr; // corresponding uncertainty
- float Dxy; // impact parameter in *xy* plane
- float DxyErr; // corresponding uncertainty
- float Lz; // decay distance in *z* plane
- float LzErr; // corresponding uncertainty

Decaying Particle Data, continued

Stable data

- float Px; // momentum in x direction
- float Py; // momentum in y direction
- float Pz; // momentum in z direction

Remember: decay data are achieved in vertex fit which adjusts the track parameter according to decay hypothesis (details in later lectures).

Therefore each stable particle inside decaying particle has adjusted momentum.

Main Objects to Work with

Stable particles: “TStablePart”

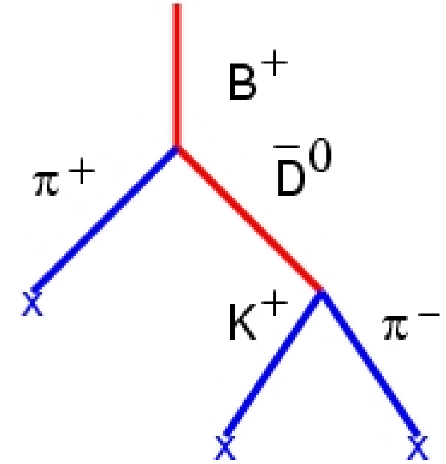
- charged particles: usually do not decay inside detector
- electrons, protons, muons, kaons, pions
- the last three decay, but usually not inside tracker

Decaying particles: “TDecayPart”

- any particle that decays to some charged particles
- *ex.* Y(upsilon) $\rightarrow \mu^+\mu^-$ or $B^+ \rightarrow \overline{D}^0 \pi^+$ with $\overline{D}^0 \rightarrow K^+\pi^-$
- recursive implementation as a tree

Optimal implementation

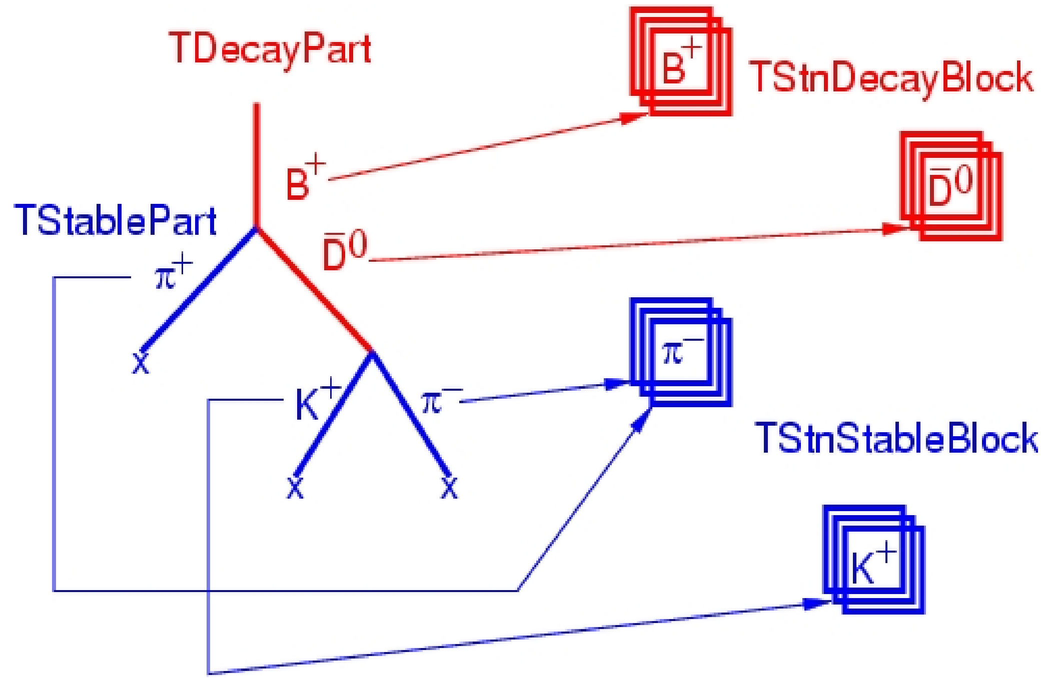
- these objects just point to entries in blocks
- structure created through links
- any info is stored only once



How the Decay Tree Works

TDecayPart tree

- links underlying components to the proper data blocks (index based)
- loop through all *B*$^+$ becomes trivial as links are automatically made when requiring highest level particle



Already taken care of

- possible trivial duplications are already excluded
- pions are all tracks which have been reconstructed
- other particles get pruned to avoid unused components

High Energy Analysis Code

General organization

- framework (Stntuple) processes data and allows user to define:
  - a sequence of (TStn)Modules
  - *ex.*: PrereqFast → SelectEvent → MakeHistogram
  - input data (often done in special input module)
  - output data (often done in special output modules)
- execution of the (Stntuple) analysis

High Energy Physics Analysis

(TStn)Modules provide standard entry points

- BeginJob(), called at the beginning of each processing
  - initialize some variables
  - book histograms or ntuples, *etc.*
- BeginRun(), called at the beginning of each new run
  - re-initialize run summaries or run dependent variables
- Event(int iEvent), called for each event
  - do analysis of the events you are looking at
- EndJob(), called at the end of the job
  - calculate summaries
  - maybe plot a picture on the screen

Analysis Prototype

Essential data components

- trigger: minimum bias
- interaction point
- tracks

Essential analysis procedures

- select correctly triggered events
- select proper tracks
- make P_n histogram
- apply fitter to the histograms

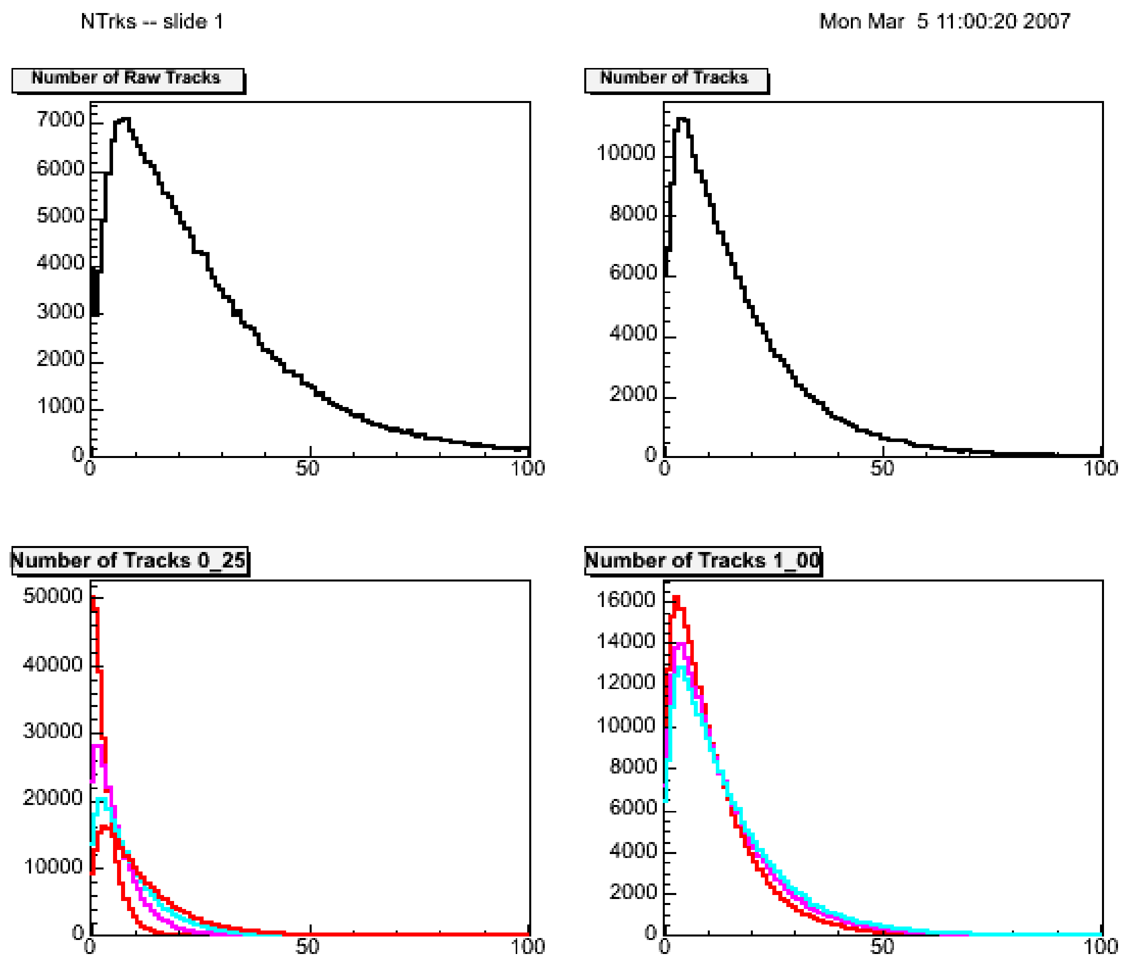
Prototyping the analysis: beginning to end, quickly

- ignore trigger, use all tracks, fit, no nice plots

Hands On: root command sequence

Create histograms for nTrks in |eta|< 0.25, 0.5, ...

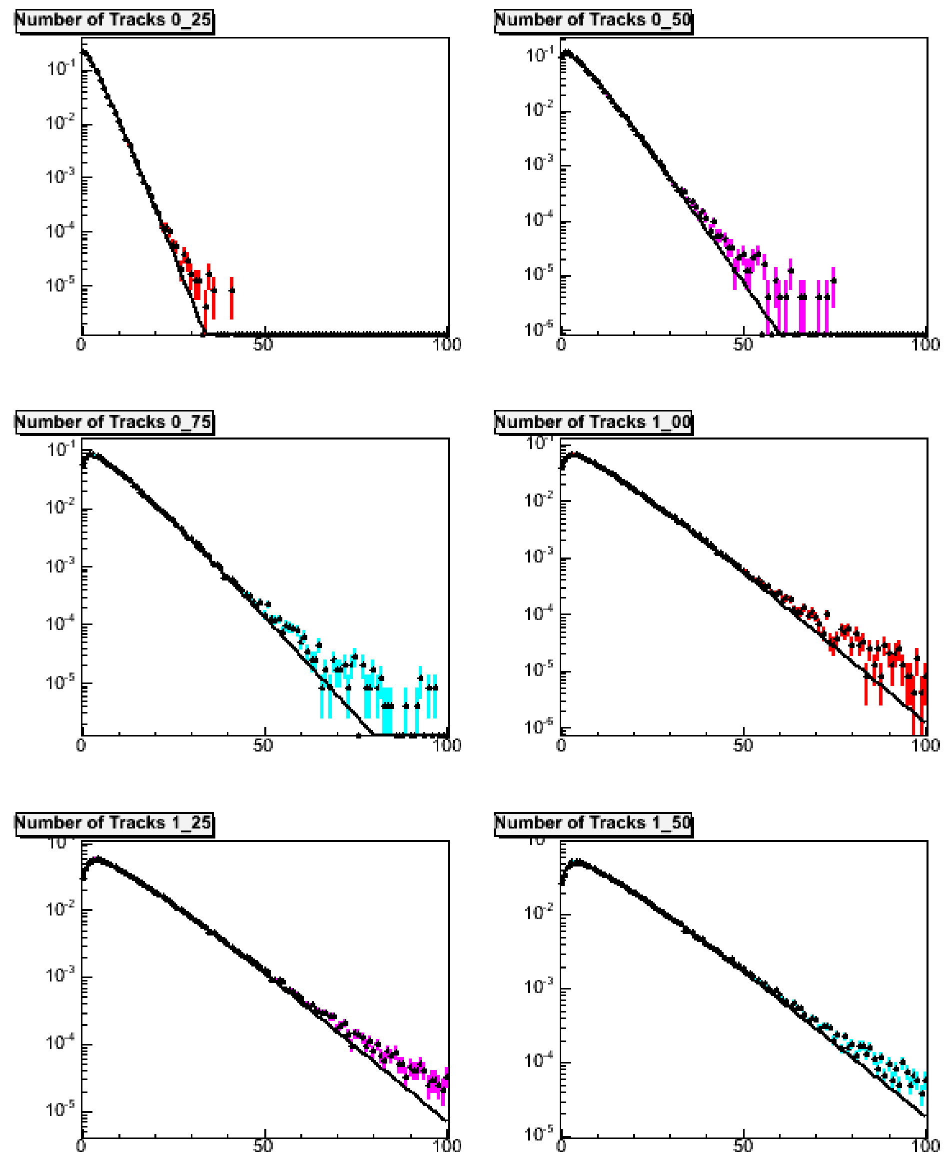
- source ~/8.882/614/INIT; cd ~/8.882/root;
- root -l ../614/Ana/scripts/runNTrks.C



Hands On: root command sequence

Perform the base fits

- root -l \
- ../614/Ana/fit/\
- negativeBinomial.C



Let's Clean Up the Mess

Counting tracks: what can go wrong?

- we want events from one interaction only
  - depending on instantaneous luminosity there could be several: call them pile-up
  - interactions are Gaussian distributed along beamline width 30 cm
  - reject events which have ambiguous track origin in *z* direction
- did we really get all tracks?
  - tracker becomes inefficient into the forward direction
  - central region should be pretty good
  - interaction point position makes difference: tracker about 3m long
  - event well inside the center are safe
  - efficiency can be checked with Monte Carlo
  - does Monte Carlo describe the data?

Let's Clean Up the Mess

Counting tracks: what can go wrong?

- do the tracks really come from the interaction point
  - interested in tracks directly from proton-antiproton interaction
  - particle might have nuclear interaction in beampipe or silicon detector
  - additional track would be created and move our number up
  - our pile-up rejection should help already
  - tracks would not point at the beamline: large impact parameter
- maybe you can think of more problems?

Most importantly get a feel for what happens if!

Conclusion

Number of charged tracks

- essential Heavy Ion physics variable
- indicator for phase transition
- proton-(anti)proton number essential calibration

HEP analysis framework

- input, module sequence (path), output
- modules provide standard user entry points:
  - BeginJob, BeginRun, Event, ...

CDF data organized in Stable/Decaying particles

- tree structure for more advanced applications
- Pions – are all tracks and so far all we need

Prototype analysis discussed: your task is to refine

Next Lecture

Data Analysis Strategies and Essential

- motherhood and apple pie
- proper work style
  - preparation of setup
  - manuals and tutorials
  - prototyping
- design of an analysis
  - data processing
  - histograms and ntuples
- rule of thumb for coding