Simulation & Reconstruction

Outlook

- Overview of the problem
- Monte Carlo Methods
- Detector Simulation
  - GEANT4
- Calorimetric reconstruction
  - Cluster
  - Jets
- Reconstruction for trackers
  - Global method
  - Local methods
  - Geometrical fitting
- Kinematical Fits

December 2013

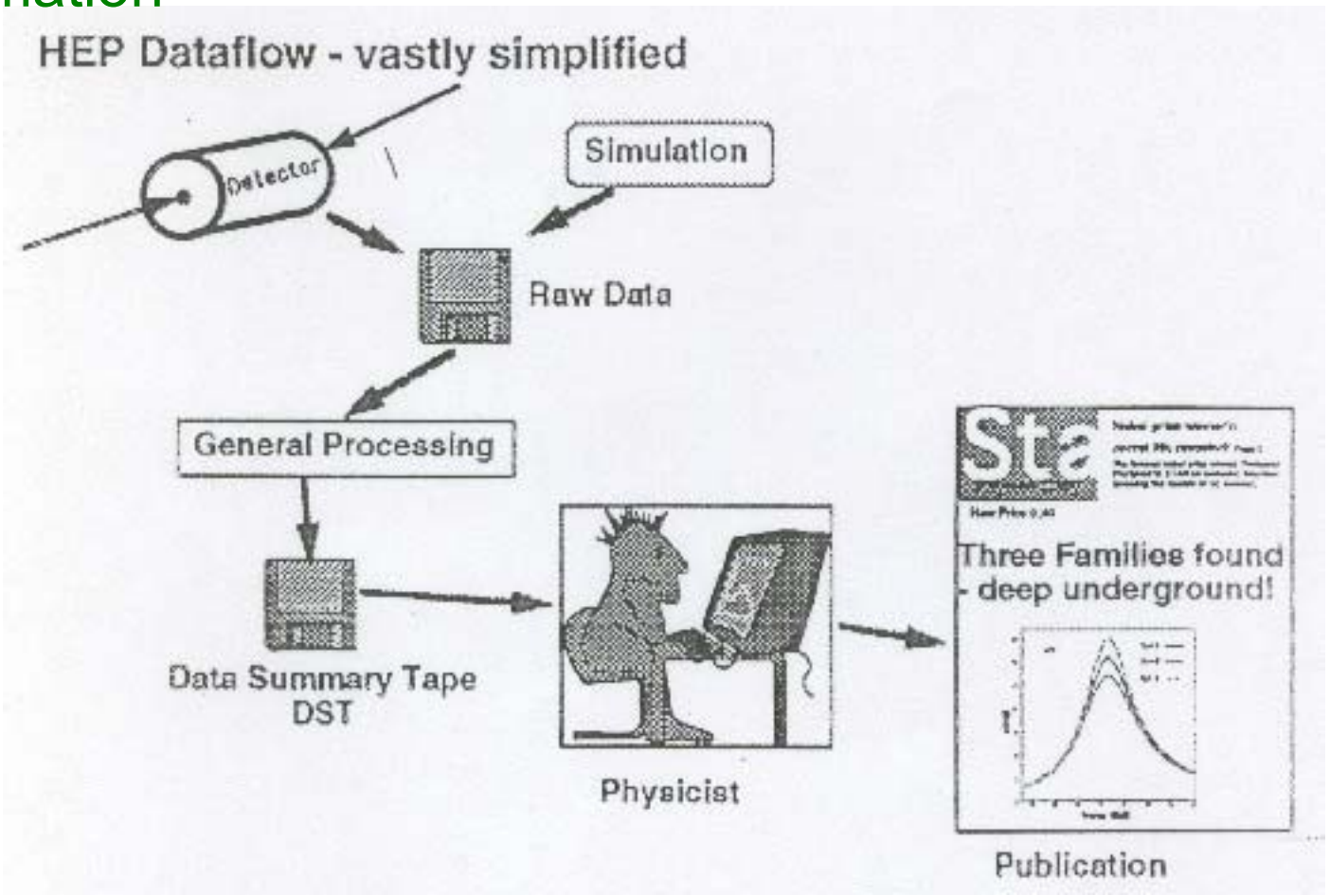
Sunanda Banerjee

Preface

- Reference:
  - Introduction to Experimental High Energy Physics: Richard Fernow (Cambridge University Press)
  - Data Analysis Technique for High Energy Physics: R.K.Bock, H.Grote, D.Notz, M.Regler (Cambridge University Press)
  - Experimental Technique in High Energy Nuclear and Particle Physics: T.Ferbel ed. (World Scientific)
  - http://geant4.cern.ch/
- Apology:
  - Nothing on event generators
  - Only hand-on can give the right impression of this field

Dream of an Experimentalist

Input: Detector with its > 100 Million channels pouring in information



Output: Paper in a reputed Physics journal
May be a discovery

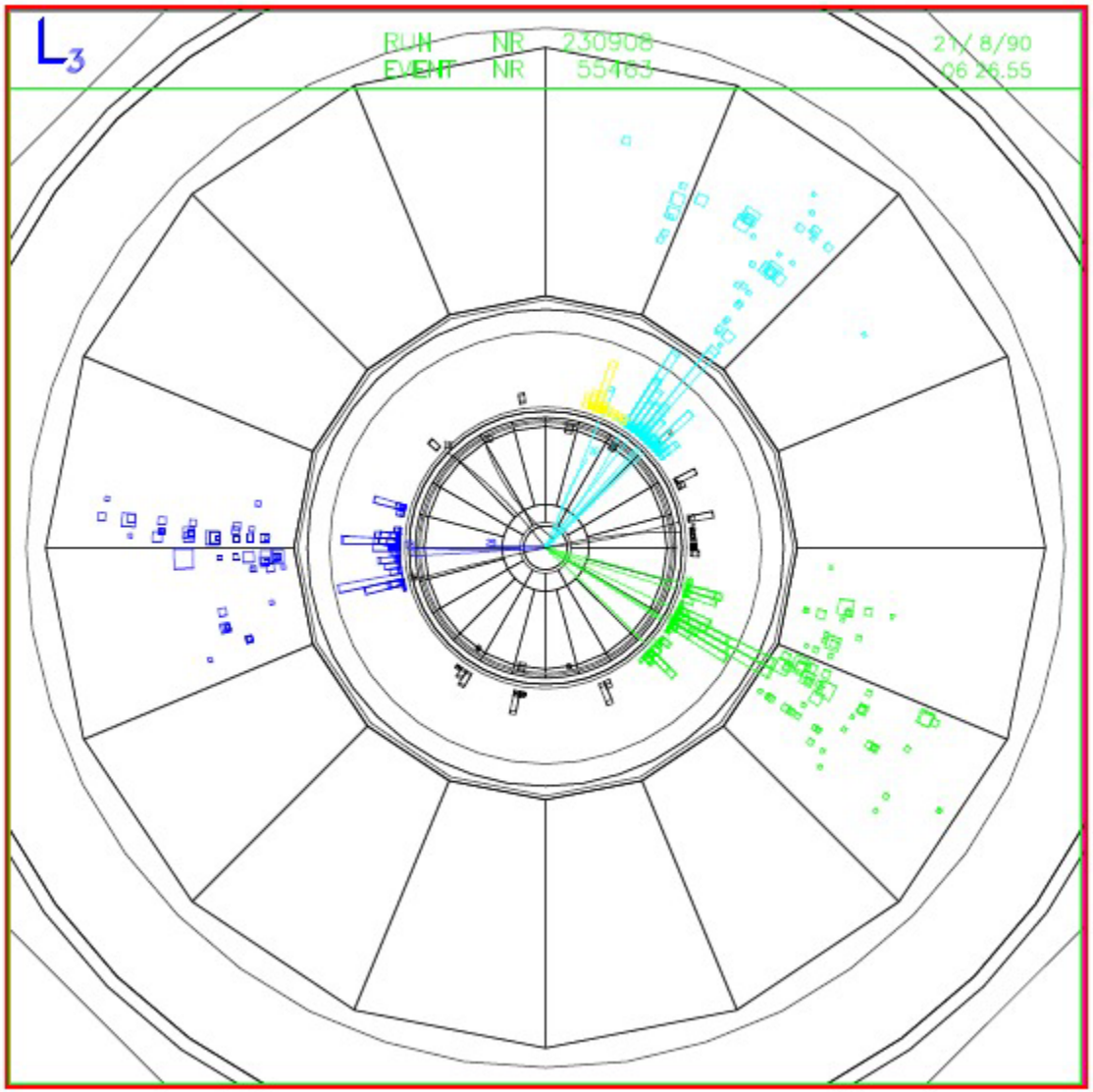
Detector Data

DAQ System: Stores string of bits in a storage device

```
0000000  123  cdef  8070  8070  4321  abcd  8061  8061
0000010    0  1fa4     0     0     0     8     0     1
0000020    0    31     0     2  4640  e400     0  9344
0000030    0     0     0     0     0     0     0     0
0000040    0     0     0     0     0     0     0    27
0000050    0     2     b  cb25  ffff  ff9d     0     0
0000060    0     0     0     0    20  5a38     0     0
0000070    0     0     0     0     0     0     0     0
*
00000e0    0     0     0     0     0     0     0   9c3
00000f0    0     2  4640  e400     0  9344     0     0
0000100    0     0     0     0     0     0     0     2
0000110    0   990    17  fe61     0    27     0     2
0000120    b  cb25     0     2    5f  f2f4  2d5c  2d8b
0000130    0     0    20  5a38     3   d40  1853   600
0000140    0    10     0     0     0   180     0     0
0000150    2   180     0     0     0     0     0     0
0000160    0     0     0     0     0     4     0     0
0000170    0     4     0     0     0     0     0     0
```

There could be a pattern as implemented in the DAQ system. But what could be source of this information?

Interpretation

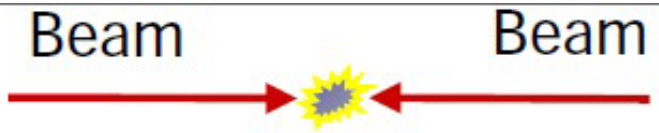


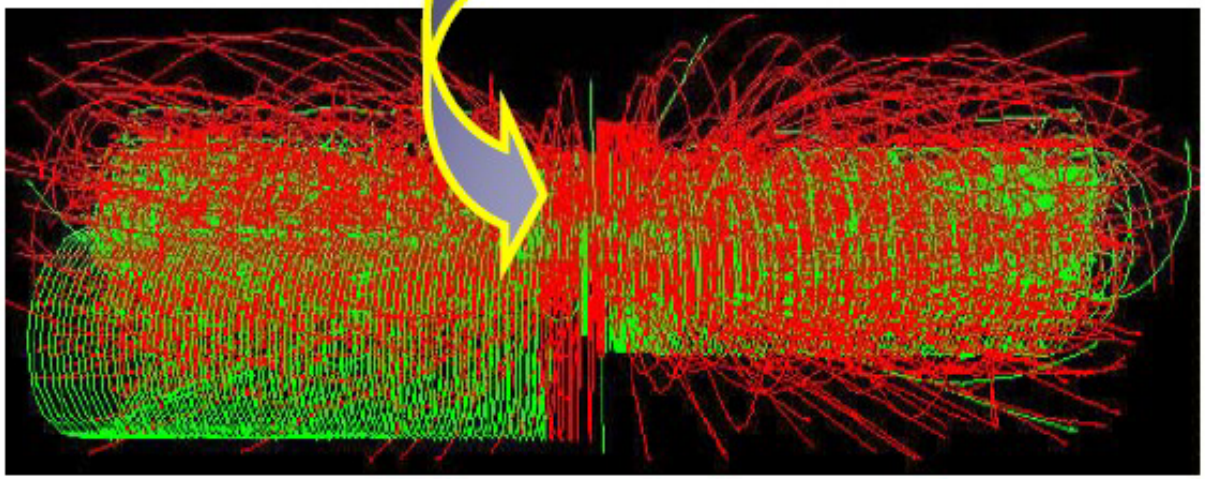
- The data could have been an event like this.
- But how to make sure that the interpretation has any touch of reality

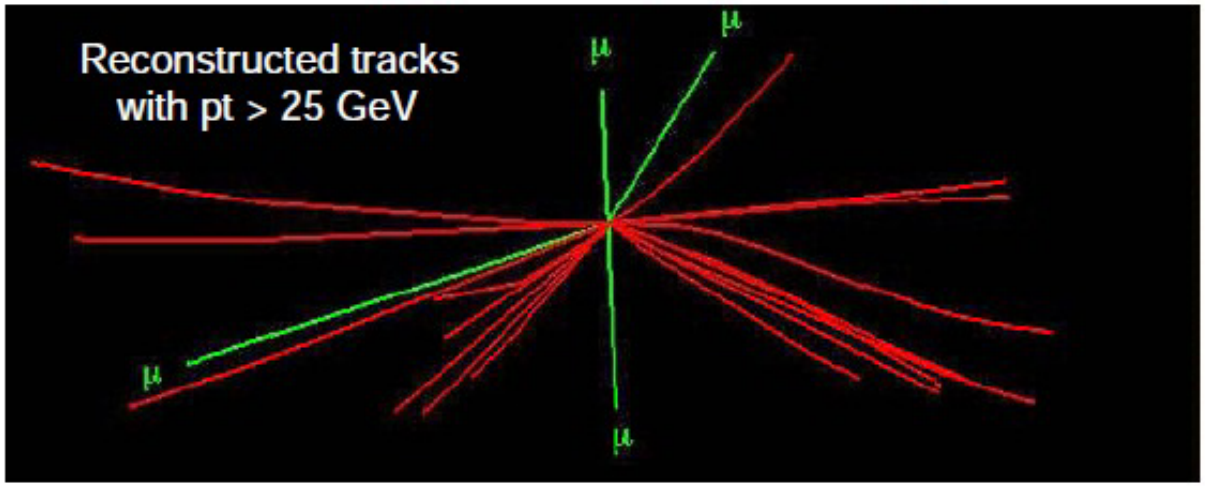
Present Day HEP Experiments

- A modern high energy physics experiment demands
  - Go to higher and higher energies to probe at shorter distances
    - Large number of particles are produced in a given interaction
  - Look for rarer processes → work at very high luminosities
    - Multiple interactions in a single event
- To cope such complex requirement
  - Use a very large and composite detector system
    - Tracking devices, calorimeters, …
  - To separate interactions and also particles coming out from a given interaction make finer readout
    - Large number of readout channels
- How one can rely on event analysis with signals coming from millions of channels
  - Dedicate time to understand how the detector performs or what the reconstruction/analysis software does to data

Complexity

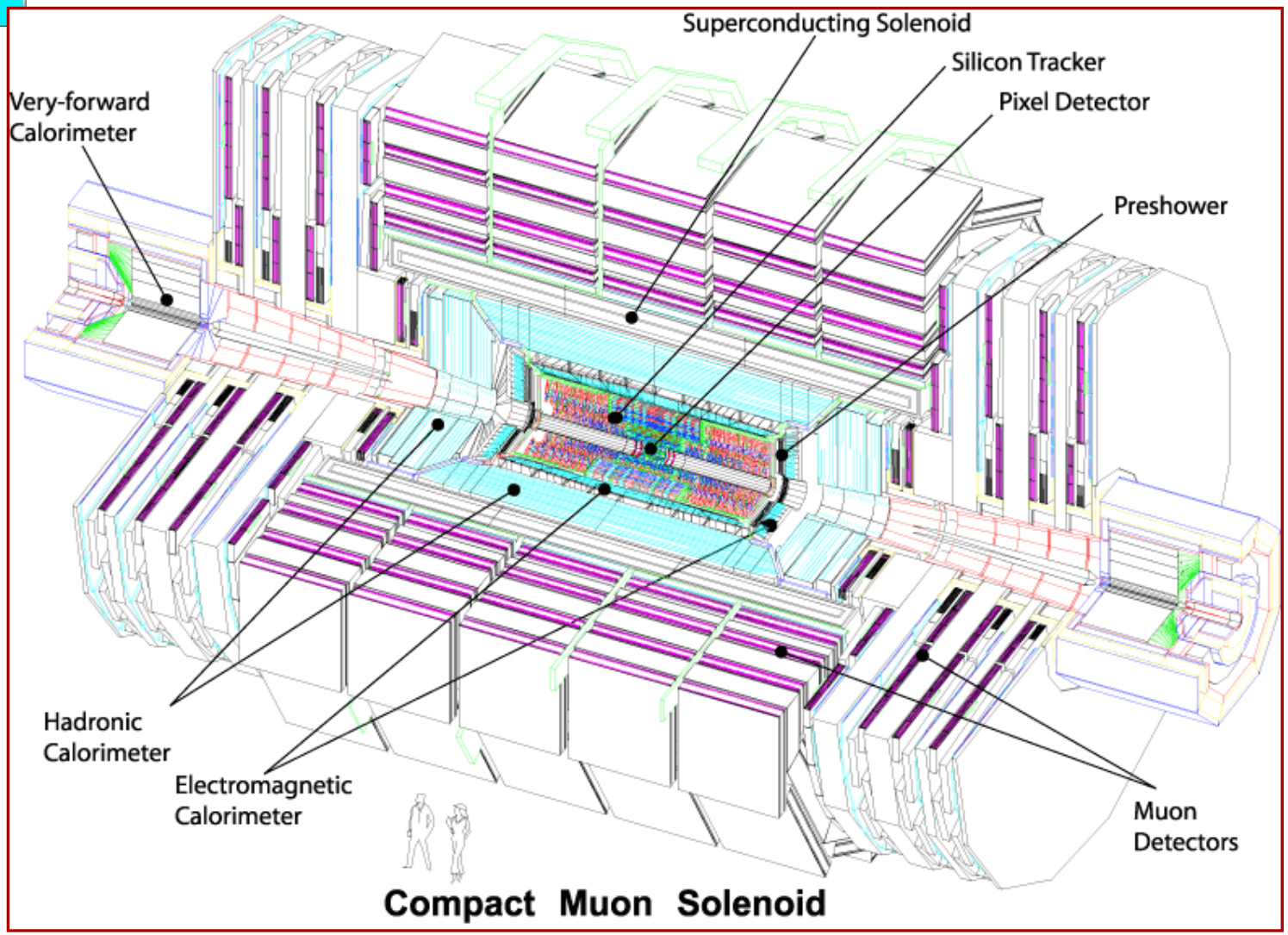






- High energy, high luminosity, high magnetic field are all required in modern HEP experiments
- But each of these gives rise to a high degree of complexity in the data coming out of the DAQ system of the detectors

A Modern Detector

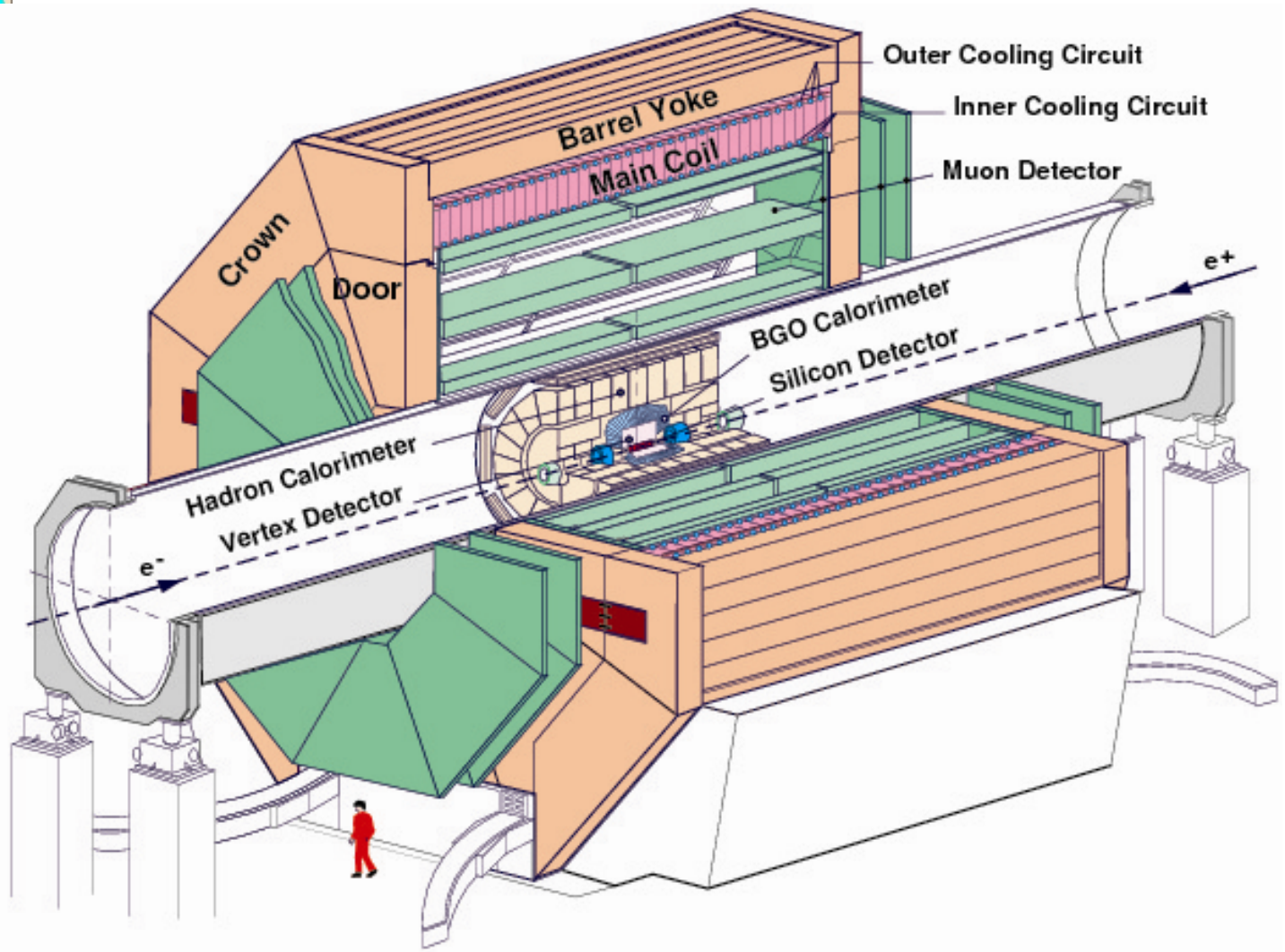


Large as well as complex detector system

Features of the detector

- Number of channels: >100 Million
- Size: 15m in diameter and 21.6m long (also extends to ~140 m)
- Precision of readout: 15-20 μm
- Alignment precision: 5-10 μm
- Dynamic range: few 10's of MeV to several TeV
- Bunch crossing time: 25-50 ns
- Data volume per event: ~1.5 MB
- Data recording rate: ~200 Hz
- Amount of data acquired: few peta byte

A detector of earlier generation



- Sizes were similar or a bit smaller. Precision was very demanding like the current one
- But event rate, number of channels were orders of magnitude smaller

Steps Taken

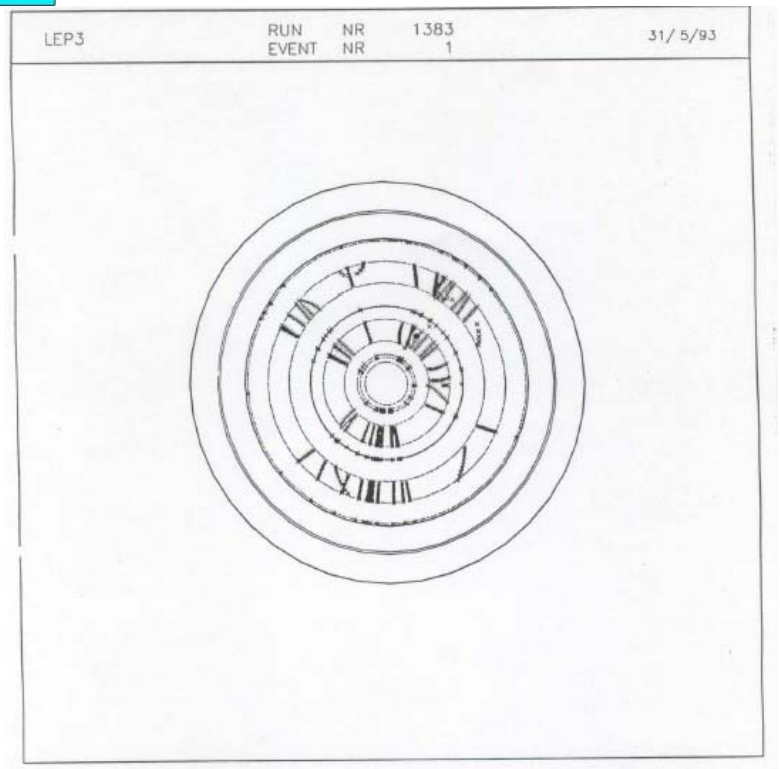
- To understand the performance of a complex detector:
  - Make a model of the detector: geometrical as well as physical
  - Start from a known input (a Physics model giving rise to a set of known particles with a complete specification of their four momenta and points of origin)
  - See the effect of passage of these charged and neutral particles through the detector media (may be in the presence of some electromagnetic field)
  - Convert energy deposits in sensitive part of the detector into detector signals building in all features of trigger and DAQ
  - Convert the detector signals into spatial coordinates and energy deposits in cells through calibration process
  - String together the space points to map with patterns followed by particle trajectories
  - Fit the trajectories to extract kinematic quantities
  - Have information of the same observable as measured and as expected
  - Compare the two estimations to determine detector effect, analysis bias, …..

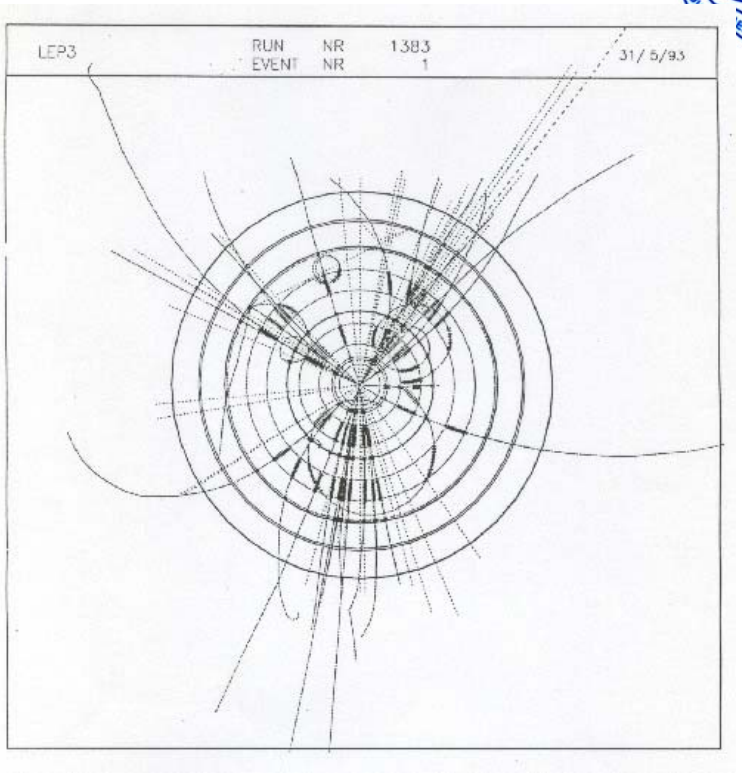
Simulation

Reconstruction

Analysis

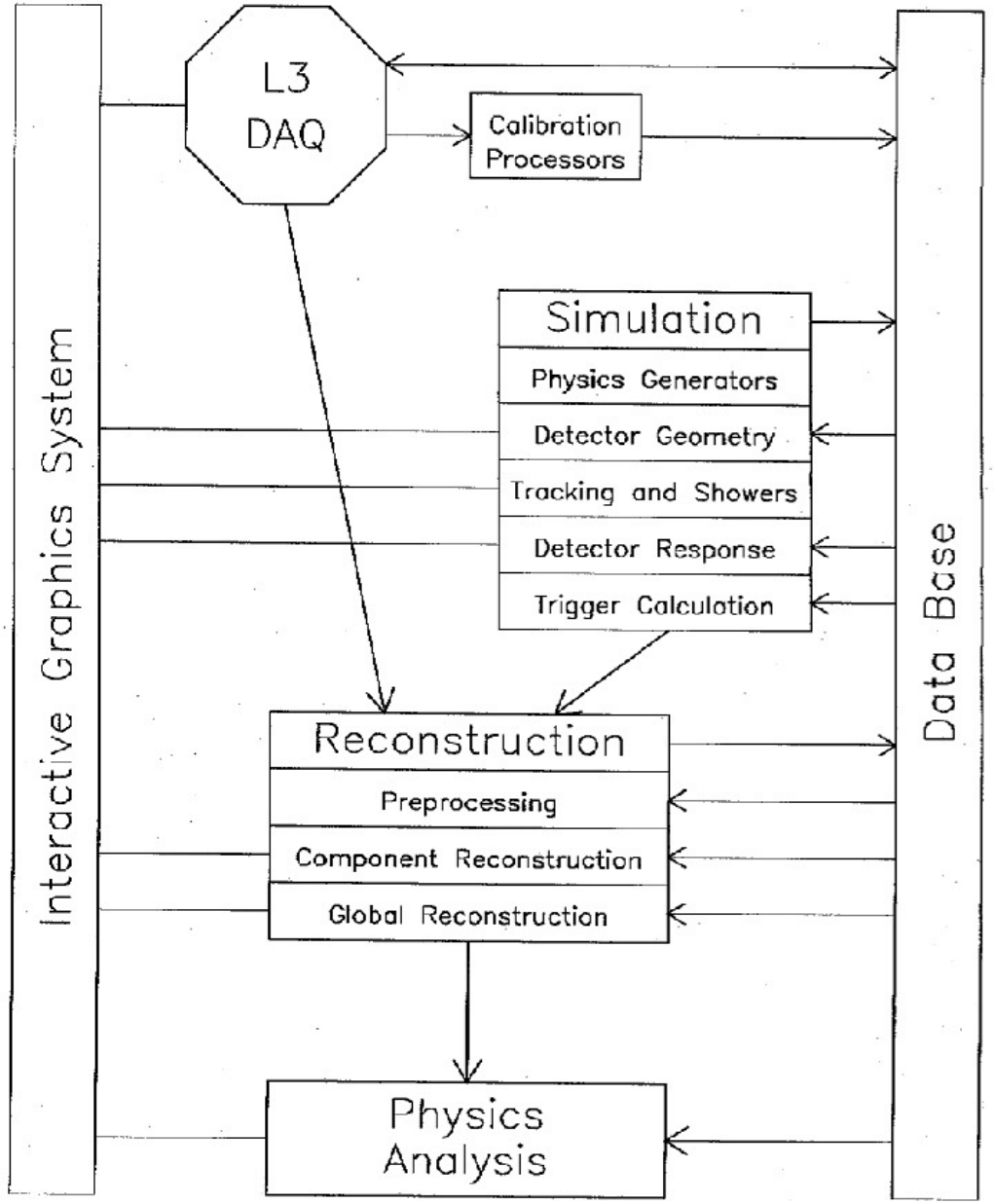
Seeing is believing





- Pattern recognition may be misleading in presence of materials inside a tracking device, non-uniform magnetic field, loopers, ….
- One can understand the performance only when tested with known inputs → simulation is a permanent tool in HEP experiments

Software Architecture of HEP Experiment



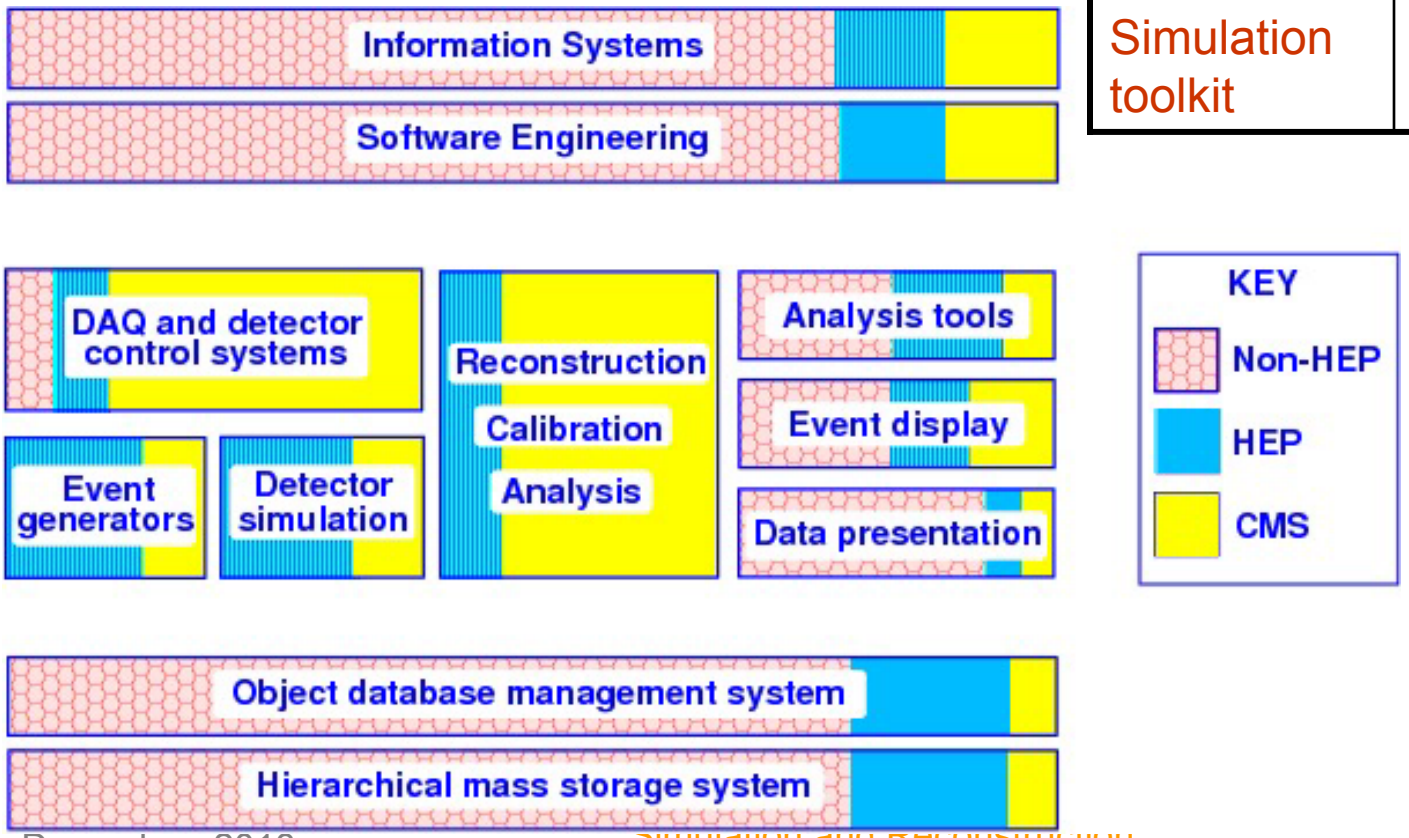
- Simulation and reconstruction are two major corner stones of a HEP experiment. But there are additional requirements:
  - Need visual verification tools to check if detector description is correct; to scan special events; to present results from statistical data analysis
  - Need asynchronous data to monitor the detector; monitor the accelerator performance; monitor data production; monitor physics decisions

ction

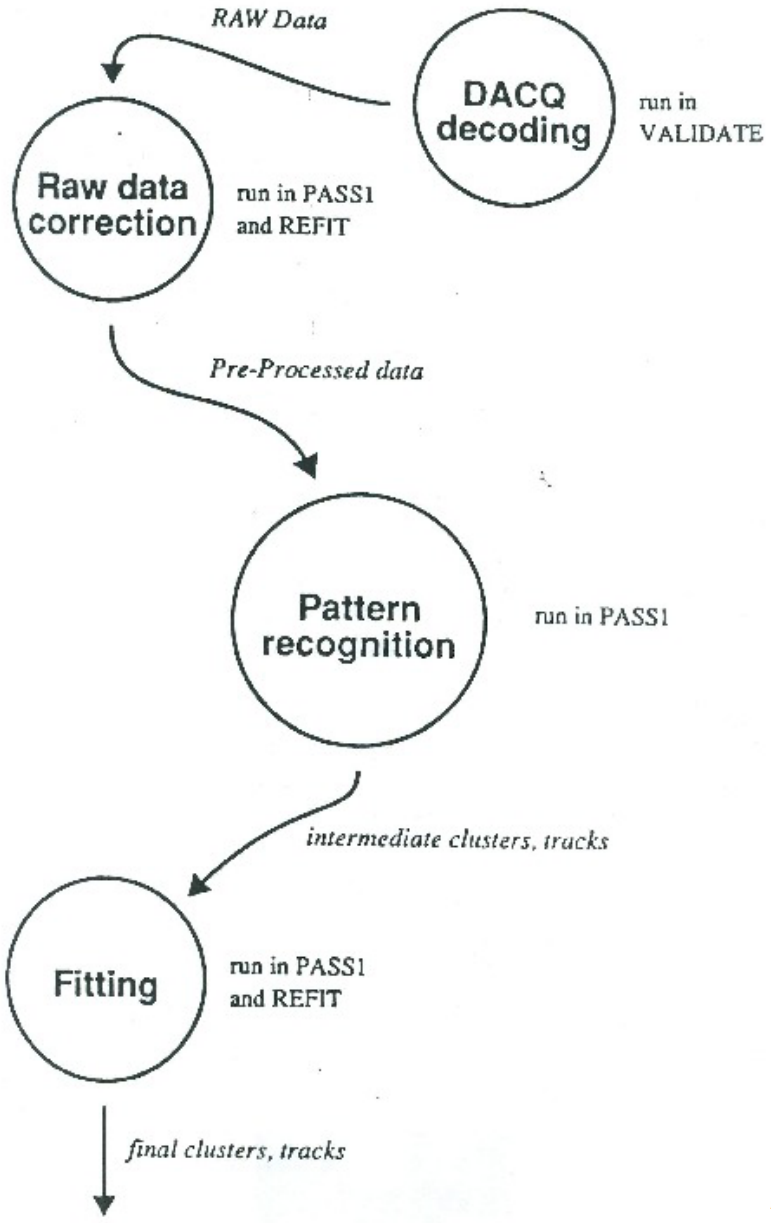
Current Wisdom

- To benefit from the experience of others:
  - Use as much common code as possible
    - ❖Same algorithms
    - ❖Same data bases
    - ❖Same physics models

	LEP era	LHC era
Common algorithms	CERNLIB	ROOT
Simulation toolkit	Geant3	Geant4

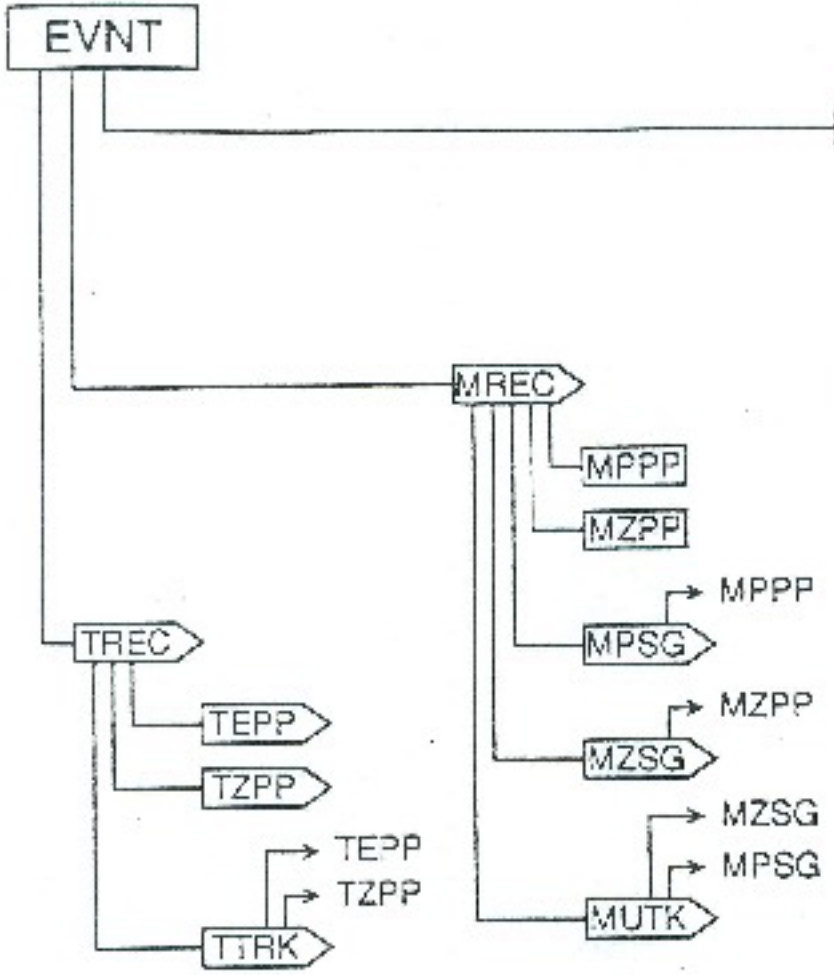


Data Flow (Earlier Generation)



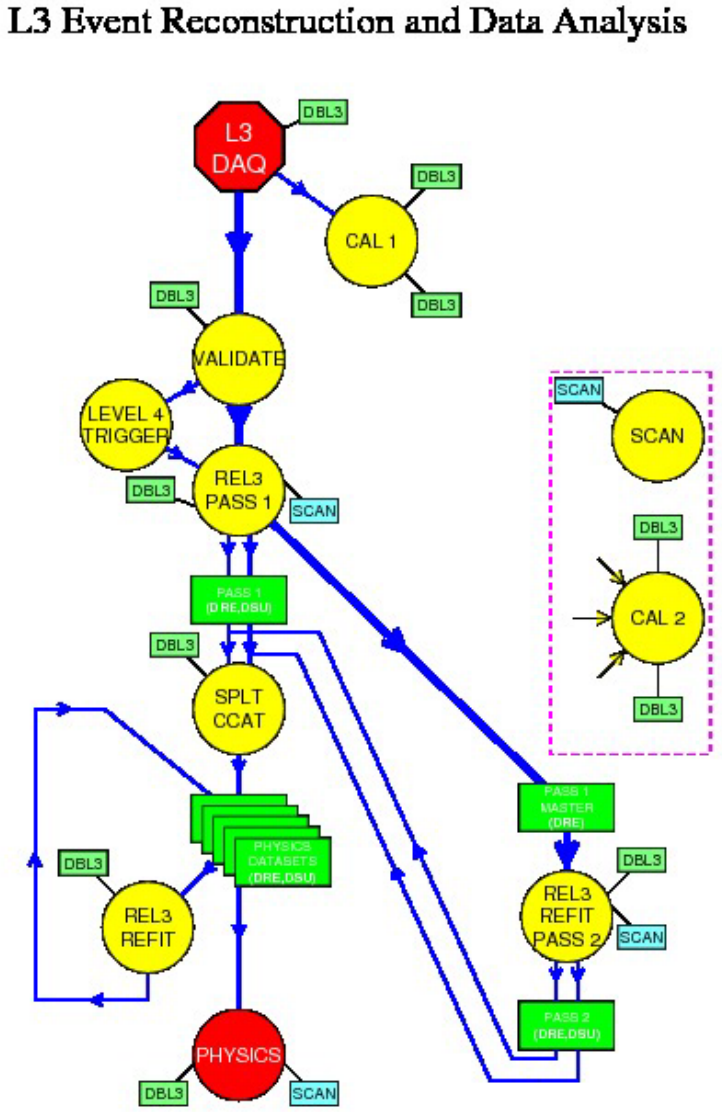
- Reconstruction was a 4 step process:
  - Validation: checksum, range of identifier and readout
  - Calibration: correct raw data and convert to more physical quantities
  - Pattern recognition: find tracks, clusters, …
  - Fitting: quality check on track, cluster; derive momentum, energy, …
- Do as much calibration, pattern recognition, fitting as possible locally in individual sub-detector
- Piece these information together to form global objects
- Make event as a collection of physical objects: particles, jets, …

Program Control (Earlier Generation)



- Communication among program modules via a well designed data structure
- As one proceeds through reconstruction phase, new objects are created referring older objects
- But handling data requires reduction of data volume per event
  - Successive steps in throwing away information – lose flexibility: full-, mini-, macro-, nan-DST's

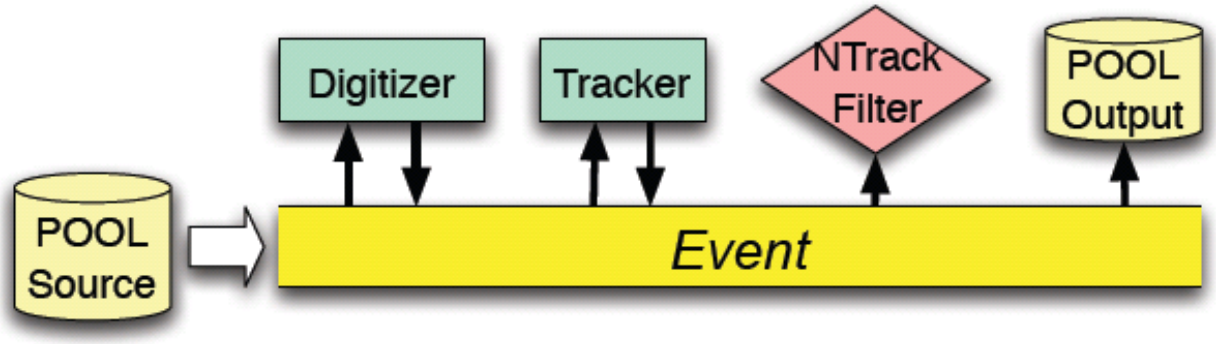
Complete Data Flow Diagram (L3)



- Improve accuracy by redoing reconstruction
  - Improved calibration
  - Improved algorithm
- Offline is testing ground for online (trigger algorithms)
- Book keeping is a must
  - Do not miss events (miss discoveries)
  - Do not repeat the same data (invent discoveries)

Data Flow (Today)

- All event data are stored in a single container – the "Event"
- Algorithms are implemented in component "modules"
  - Modules communicate only through the event
  - Execution is scheduled explicitly
  - Scheduling is done in the job configuration
  - Required modules are dynamically loaded at the beginning of the job



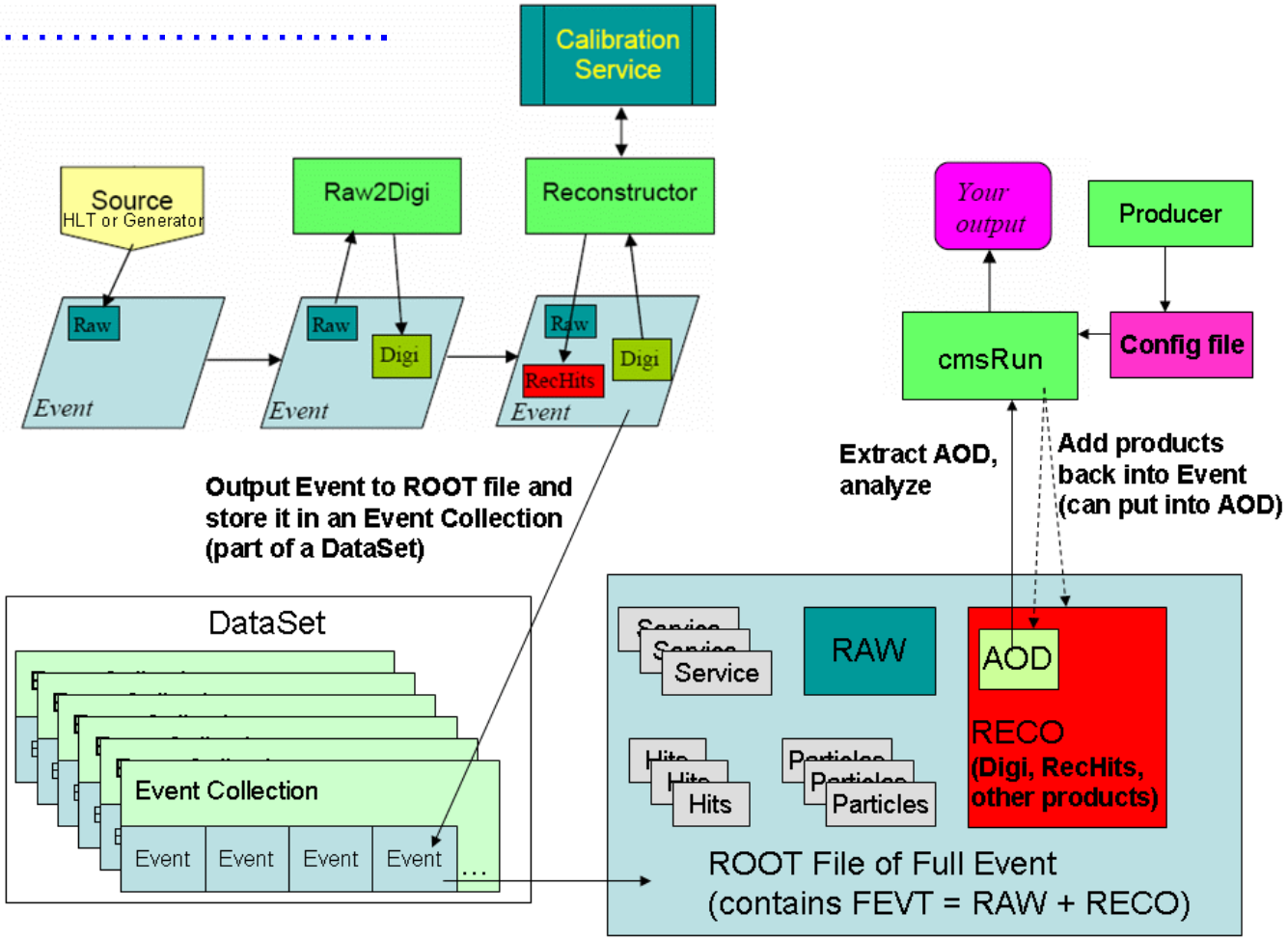
Intermediate architecture:

- Objects obtained through RecCollection<T>
- Any operation on the collection triggers reconstruction on-demand

Event & FrameWork

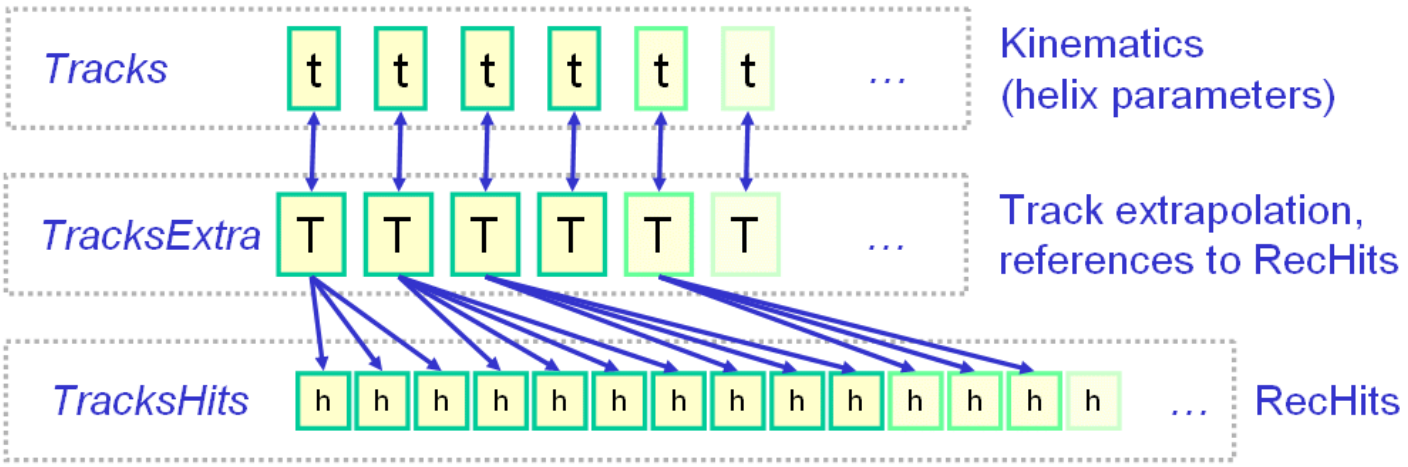
- Event is a collection of containers each containing products of a given type.
- There are several levels of completeness of the Event
  - FEVT: Full event
  - AOD: Analysis object data

.....



Event Data

- Different data layers exist using different data formats and an application can use any layer(s).
- Branches map one to one with event data objects and are loaded or dropped on demand,
- Event data are identified by
  - C++ class type (like PCaloHitContainer)
  - Label assigned to the module that created the data (g4SimHits)
  - Product instance label assigned to the object within the module (HcalHits)
  - Process name that creates the data



Monte Carlo Technique

- Particle scattering and absorption
  - involves random processes
  - uses complicated multi-dimensional integrations
- Use Monte Carlo technique to do the multi-dimensional integration
  - Solution of a problem as a parameter of a hypothetical population and constructing a sample of the population to obtain estimates of the parameter → use random numbers to construct the sample

We may need to carry out the integration

$$I = \int_0^1 dx_n \cdots \int_0^1 dx_1 F(x_1, \cdots x_n)$$

Then with a set of random numbers in the range 0-1 determine F and this F will be an unbiased estimate of I

Repeat this estimate for a large number of times and the mean value F will converge to the value of I

Random Variables

- It can have more than one value (generally any value within a range)
- One cannot predict in advance which value it will take
- The distribution of the variable may be well known

Distribution of a random variable → probability of having a specific value

Probability distribution function is given by

$$g(u)du = P[u < u' < u + du]$$

Integrated distribution function:

$$G(u) = \int_{-\infty}^{u} g(u')du'$$

$$g(u) = \frac{dG(u)}{du}$$

G(u) increases monotonically with u. Normalization of g is determined by

$$\int_{-\infty}^{\infty} g(u)du = 1$$

Classification of Random Variables

- Truly random variables
  - Sequence of truly random numbers is completely unpredictable and hence irreproducible
  - Can be generated only through physical processes (timing of radioactive decays, arrival time of Cosmic Ray particles, ….)
- Quasi random numbers
  - Sequence do not appear random (high degree of correlation) but give right answers to Monte Carlo integration
  - There are strict mathematical formula and provide fast convergence of integration. They are of limited use
- Pseudo random numbers
  - Sequence generated with a strict mathematical formula, but indistinguishable from a sequence generated truly randomly
  - Most simulation programs use pseudo random numbers. The heart of the simulation process is generation of "Uniform Deviates": random numbers which lie within a specific range (0 to 1) with any number just as likely as the other

Pseudo Random Number

- Early Method: Start with a number of r digits. The first random number is the middle r/2 digits. Square this number and again take the middle r/2 digits for the next random number and so on. This procedure is machine dependent, has large correlation and also has small period.
- Multiplicative Linear Congruental Generator: This is the most common random number generator which generates a sequence of integers between 0 and m-1 (a large number) using the recurrence relation

$$r_i = \mathrm{mod}(a \cdot r_{i-1} + b, m)$$

where a is the multiplier; b is an additive constant and r^o is the starting value and m is the multiplier

This is very fast and is transportable with a proper choice of a, b and m. But it is not free from sequential correlation and has a short period (at most can have a period of m)

Lower order bits are much less random rather than higher order bits

Pseudo Random Number

So for a random number in the range of 1-10, it is better to use $1 + \mathrm{int}(10. * \mathrm{r_i})$ rather than $1 + \mathrm{mod}(\mathrm{int}(100000000 * \mathrm{r_i}), 10)$

The correlation can be improved by first making a table of random numbers generated using MLCG and then picking randomly from the table

- Subtraction Method: Subtract two randomized numbers provide transportable random numbers of rather large period:
  - Initialize a table in a slightly random order with numbers that are not strictly random
  - Randomize them by subtracting a number not especially random
  - Take the difference between two numbers in the table which are apart
  - Update the table position with this number
  - Go to the next sequence of the table

GEANT uses a generator based on subtraction method: RANECU

Example of a Generator

```
double RanecuEngine::flat() {
    const int index = seq;
    long seed1 = table[index][0];
    long seed2 = table[index][1];

    int k1 = (int)(seed1/ecuyer_b);
    int k2 = (int)(seed2/ecuyer_e);
    seed1 = ecuyer_a*(seed1-k1*ecuyer_b)-k1*ecuyer_c;
    if (seed1 < 0) seed1 += shift1;
    seed2 = ecuyer_d*(seed2-k2*ecuyer_e)-k2*ecuyer_f;
    if (seed2 < 0) seed2 += shift2;

    table[index][0] = seed1;
    table[index][1] = seed2;

    long diff = seed1-seed2;
    if (diff <= 0) diff += (shift1-1);
    return (double)(diff*prec);
}
```

prec=4.6566128E-10

shift1=2147483563

shift2=2147483399

ecuyer_a=40014; ecuyer_b=53668; ecuyer_c=12211;

ecuyer_d=40692; ecuyer_e=52774; ecuyer_f=3791;

Arbitary probability density

- To generate random numbers according to an arbitrary normalized probability density function F(x) one needs to choose the most efficient approach among the following methods:

Transformation Method

Suppose one has a random number generating function producing x uniformly in the range 0:1

→ Probability of generating a number between x and x+dx:

$$\begin{aligned} g(x)dx &= dx \quad \text{for } 0 \leq x \leq 1 \\ &= 0 \quad \text{for } 0 > x \text{ or } x > 1 \end{aligned}$$

and probability density function is normalized:

$$\int_{-\infty}^{\infty} g(x)dx = 1$$

Now we would like to generate random numbers y with a probability density function f(y)

Transformation Method

$$\int f(y)dy = 1$$

Using the fundamental transformation law of probability:

$$|f(y)dy| = |g(x)dx|$$
$$\text{or } f(y) = g(x)\left|\frac{dx}{dy}\right|$$

So one gets

$$dx = f(y)dy$$
$$\Rightarrow x = \int f(y)dy + c$$

This is to be inverted to get

$$y = y(x)$$

Let us generate an exponential distribution as an example

$$f(y)dy = a \cdot \exp(-a \cdot y)dy$$
$$\text{with } y \geq 0 \text{ and } a > 0$$

Transformation Method

This satisfies

$$\int_0^{\infty} f(y)dy = 1$$

$$\Rightarrow x = -\exp(-a \cdot y) + c$$

Imposing boundary conditions, namely x = 0 at y = 0 and x = 1 at y = ∞, one obtains c = 1.

Thus

$$y = -\frac{1}{a}\ln(1-x)$$

The transformation method can be generalized to more than one dimension:

x's are random deviates with joint probability distribution

$$g(x_1, x_2, \cdots)\, dx_1\, dx_2 \cdots$$

Each of the y's is a function of all x's (the number of variables in both x- and y- space is the same)

Transformation Method

$$\Rightarrow g(y_1, y_2, \cdots)dy_1 dy_2 \cdots = g(x_1, x_2, \cdots)\left|\frac{\partial(x_1, \cdots x_n)}{\partial(y_1, \cdots y_n)}\right| dx_1 dx_2 \cdots$$

$\left|\frac{\partial(x_1, \cdots x_n)}{\partial(y_1, \cdots y_n)}\right|$ is the Jacobian determinant of x's with respect to y's

Let us have two uniform deviates x's and the variables y's are defined as

$$y_1 = \sqrt{-2\ln x_1}\cos 2\pi x_2$$
$$y_2 = \sqrt{-2\ln x_1}\sin 2\pi x_2$$

Equivalently

$$x_1 = \exp\left[-\frac{1}{2}(y_1^2 + y_2^2)\right]$$
$$x_2 = \frac{1}{2\pi}\tan^{-1}\left(\frac{y_2}{y_1}\right)$$

The Jacobian determinant

$$\frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} = \begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix}$$
$$= -\left[\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{y_1^2}{2}\right)\right]\cdot\left[\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{y_2^2}{2}\right)\right]$$

Central Limit Theorem

Since this is a product of two functions; for the two y's, one gets two independent random deviates following Gaussian distribution of mean 0 and RMS 1

$$g(y)dy = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right) dy$$

Law of large numbers

- Concerns sum of large number of random variables
- Choose n numbers x's distributed randomly with uniform probability density in the interval a to b
- Evaluate f(x) for each value of x

$$\frac{1}{n}\sum_{i=1}^{n} f(x_i) \rightarrow \frac{1}{b-a}\int_a^b f(x)dx \qquad \text{for large n}$$

LHS is the consistent estimator of the integral which implies variance of f will be finite. Standard deviation of the estimator $\sim \sqrt{\frac{V(f)}{n}}$

This is the Central Limit theorem

Law of Large Numbers

Sum of large number of independent random variables is always normally distributed, no matter how the individual random numbers are distributed

Normal distribution is specified by its expectation a and variance σ^2

$$g(x)dx = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-a)^2}{2\sigma^2}\right) dx$$

If u_i are uniform deviates between 0 and 1 $\rightarrow g(u)du = du$

$$\langle u \rangle = \frac{\int_0^1 u \cdot du}{\int_0^1 du} = \frac{1}{2}$$

$$\text{variance } \sigma_u^2 = \frac{\int_0^1 (u - \frac{1}{2})^2 du}{\int_0^1 du} = \frac{1}{12}$$

So if one takes sum of k random variables: $u_{(k)} = \sum_{i=1}^{k} u_i$ then

$$\langle u_{(k)} \rangle = \frac{k}{2}$$

$$\text{variance } \sigma_{u_{(k)}}^2 = \frac{k}{12}$$

By choosing 12 random variables and computing $\sum_{i=1}^{12} u_i - 6$ one gets normally distributed variable of mean 0 and variance 1

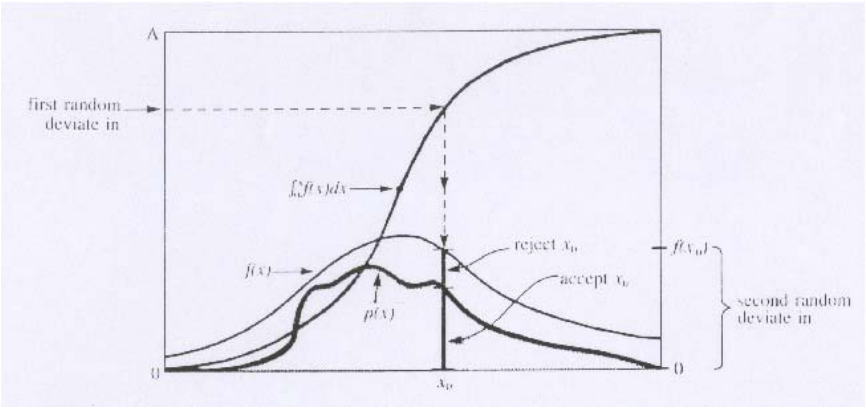
Arbitrary Distribution

Rejection Method

If probability distribution function is known and computable, this method can be applied. Here one does not require

- cumulative distribution function to be available
- distribution function to be inverted

For example, one needs to generate a random number in the interval a:b according to a probability distribution function proportional to p(x)



- Choose a function f(x) (comparison function) such that
  - the corresponding cumulative distribution function is computable and invertible
  - f(x) lies above p(x) every where between a and b
- Choose x using the transformation method applied to the function f(x)

Arbitrary Distribution

- Use a second deviate u uniformly distributed between 0 and 1. If $p(x)/f(x) \leq u$, then the value x is to be accepted; otherwise that vale of x is to be rejected and a fresh value has to be obtained using the transformation method.

General Method

- Express the probability density function as a sum of components:

$$F(x) = \sum_{i}^{n} \alpha_i \cdot f_i(x) \cdot g_i(x)$$

with α_i positive; $f_i(x)$ normalized probability density function which can be inverted; and $g_i(x)$ are computable functions satisfying $0 \leq g_i(x) \leq 1$

- Solution to the problem can be obtained as:
  - select an integer i randomly within $0 \leq i \leq n$ and probability of selecting $i \propto \alpha_i$
  - select a variable x´ from $f_i(x)$ using the transformation method
  - calculate $g_i(x)$ and accept $x = x'$ with selection probability $g_i(x)$
  - if rejected, go and select i once again and repeat

Example

- This is a good method for sampling x if
  - All sub-distribution function $f_i(x)$ can be easily sampled
  - The rejection functions $g_i(x)$ can be quickly computed
  - Mean number of tries $(\sim \sum \alpha_i)$not too large

Example (Pair production $\gamma\gamma^* \rightarrow e^+e^-$)

Let a photon of energy E produces an e^+e^- pair and the electron carries an energy fraction ε

$$\frac{d\sigma}{d\epsilon} = r_e^2 \frac{\alpha Z}{E^2} [Z + \xi(Z)] \left(\left[\epsilon^2 + (1-\epsilon)^2\right] [\phi_1(\delta) - F(Z)] + \frac{2}{3}\epsilon(1-\epsilon) [\phi_2(\delta) - F(Z)] \right)$$

$$\text{with } \delta = \text{screening variable} = \frac{136m}{Z^{\frac{1}{3}}E} \frac{1}{\epsilon(1-\epsilon)}$$

Example

Rewrite the probability density function for ϵ

$$\begin{array}{rcl} & & \alpha_i \quad \cdot \quad f_i \quad \cdot \quad g_i \\ F(\epsilon) & = & \frac{[\frac{1}{2}-\frac{m}{E}]^2}{3} F_1(\delta_{min}) \cdot \frac{3}{[\frac{1}{2}-\frac{m}{E}]^3}(\epsilon - \frac{1}{2})^2 \cdot \frac{F_1(\delta)}{F_1(\delta_{min})} \\ & + & \frac{1}{2} F_2(\delta_{min}) \quad \cdot \quad \frac{1}{[\frac{1}{2}-\frac{m}{E}]} \quad \cdot \frac{F_2(\delta)}{F_2(\delta_{min})} \end{array}$$

$$\text{with} \quad F_1(\delta) = 3\phi_1(\delta) - \phi_2(\delta) - 2F(Z)$$

$$F_2(\delta) = \frac{3}{2}\phi_1(\delta) + \frac{1}{2}\phi_2(\delta) - 2F(Z)$$

$$\delta_{min} = \frac{136m}{Z^{\frac{1}{3}}E} \cdot 4$$

Kinematic range for ϵ $\left(\frac{m}{E} \leq \epsilon \leq 1 - \frac{m}{E}\right)$

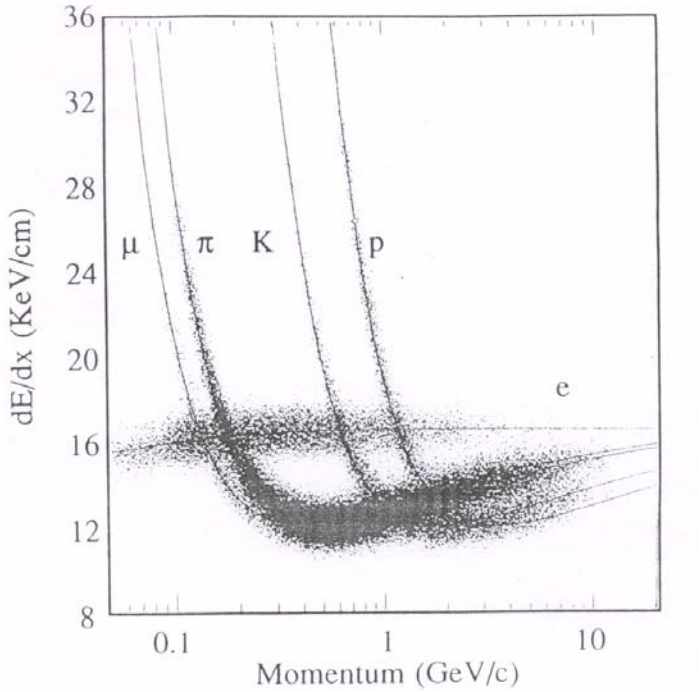
Using symmetry of σ under $\epsilon \leftrightarrow 1-\epsilon$ restrict $\left(\frac{m}{E} \leq \epsilon \leq \frac{1}{2}\right)$

Define a variable $B = \frac{\alpha_1}{\alpha_1+\alpha_2}$

- Generate r_0 with $0 \leq r_0 \leq 1$. If $r_0 < B$, choose $i = 1$, else $i = 2$
- Generate r_1 with $0 \leq r_1 \leq 1$. If $i = 1$, $\epsilon = \frac{1}{2}\left[1 - \frac{m}{2E}(1-r_1)^{\frac{1}{3}}\right]$. Otherwise estimate $\epsilon = \frac{m}{E} + \left(\frac{1}{2} - \frac{m}{E}\right) r_1$
- Generate r_2 with $0 \leq r_2 \leq 1$. If $r_2 \leq g_i(\epsilon)$, then accept this ϵ, else reject the current values of i, ϵ and try to find i again

Interaction of particles with matter

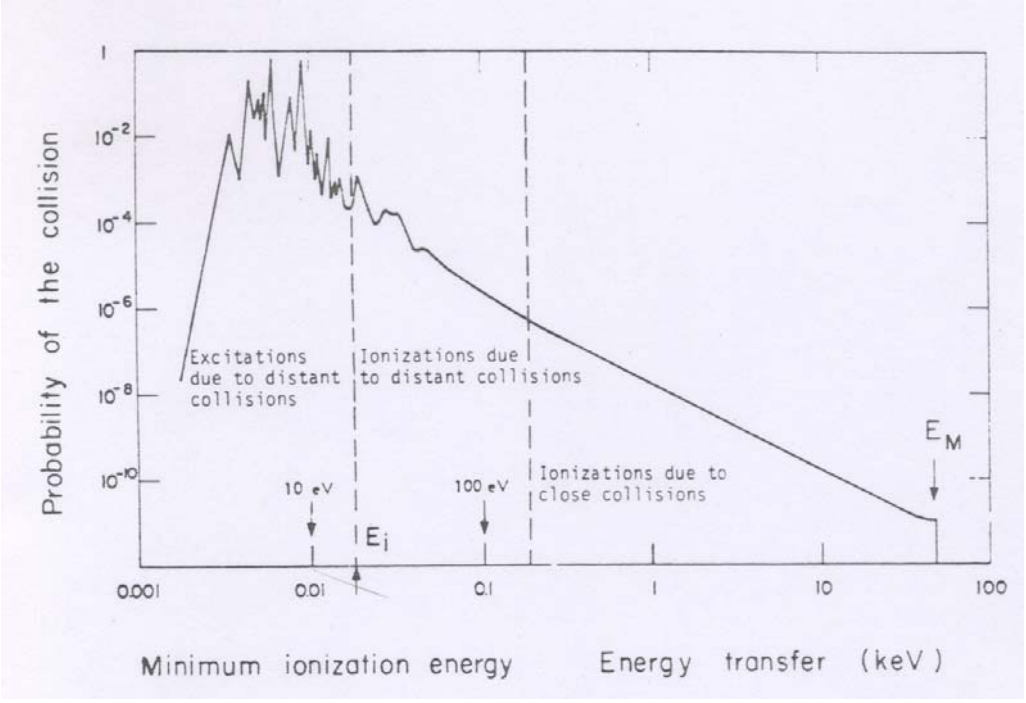
- Particles passing through matter interact with nuclei as well as with atomic electrons. The physical processes are broadly classified into two categories:
  - Discrete processes (bremsstrahlung, annihilation, elastic, …)
  - Continuous processes (energy loss, multiple scattering, …)
- Continuous energy loss (charged particles in matter)

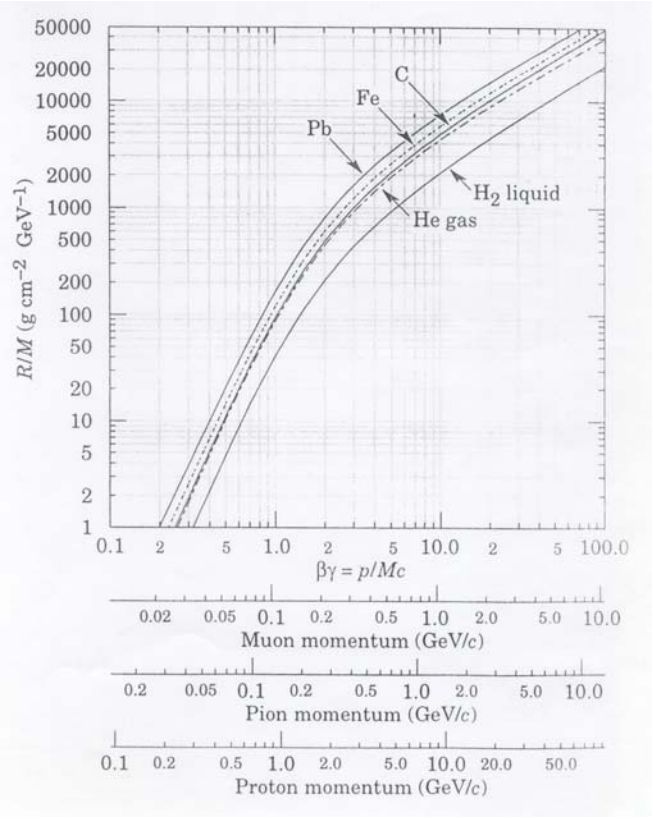


- At small β, -dE/dx decreases with momentum
- A minimum is reached at $\beta\gamma \approx 4$
- At large β, γ^2 term dominates → relativistic rise
- At very large $\beta\gamma$, saturation due to screening → density effect

Energy Loss

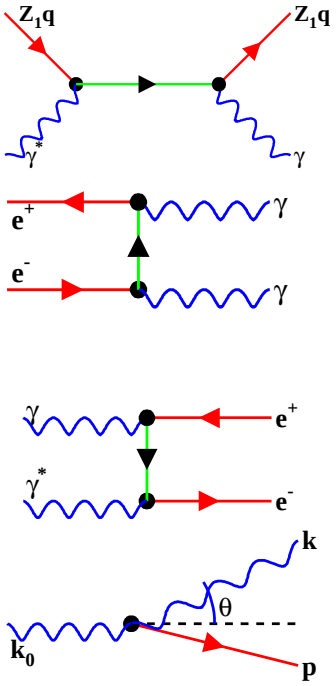
- Individual collisions are classified as
  - Distant collision: atoms react as a whole → excitation, ionization
  - Close collision: with atomic electrons → knock-on
  - Very close: with nuclei → radiation
- If no discrete process happens, particles eventually stops after losing all energies





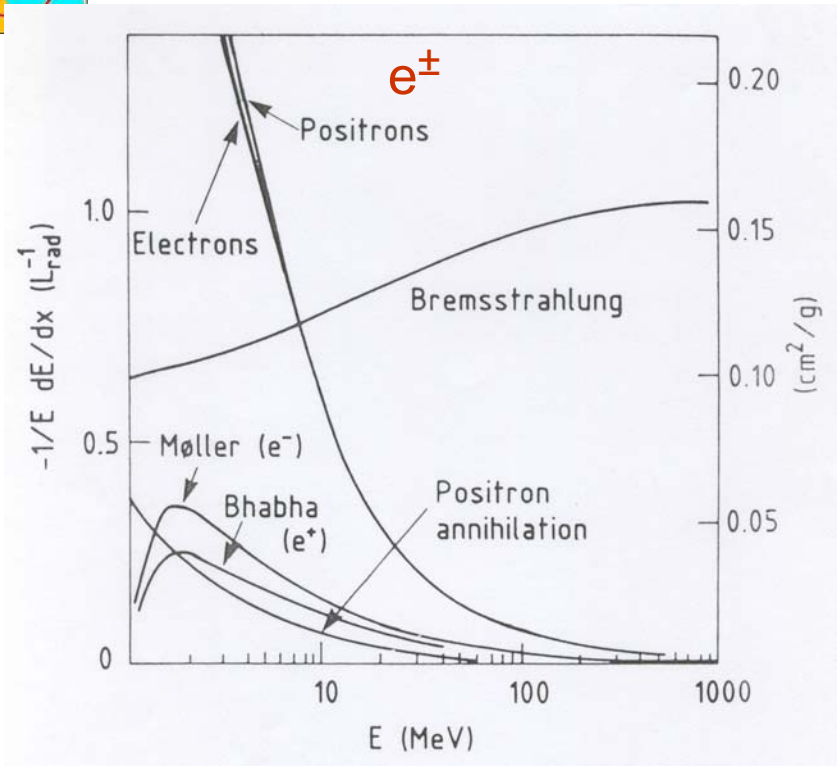
Discrete processes

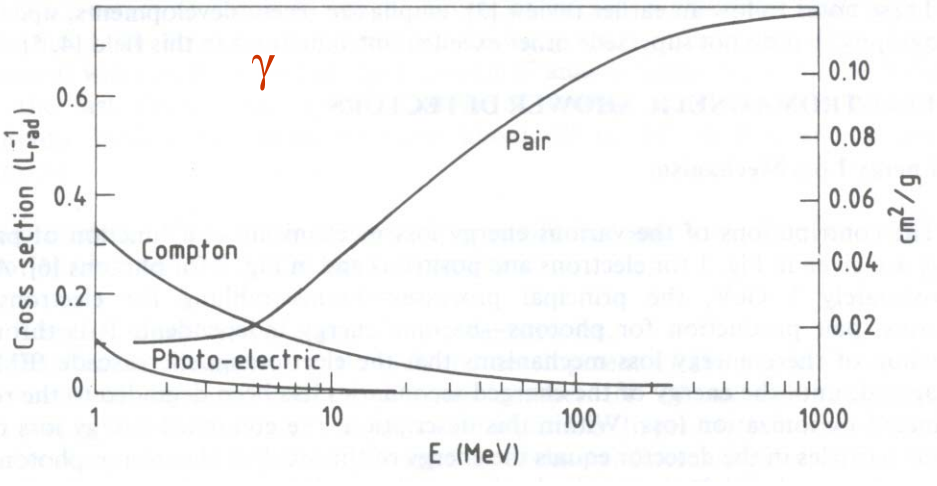
- Discrete processes:
  - Bremsstrahlung
  - Annihilation (positrons)
  - Elastic scatterings
  - Pair production
  - Compton scattering
  - Photo-electric effect
  - Decays of unstable particles (em/weak)
  - Strong interaction for hadrons



	Unchanged	Breaks
❖target	coherent	incoherent
❖projectile	elastic	inelastic

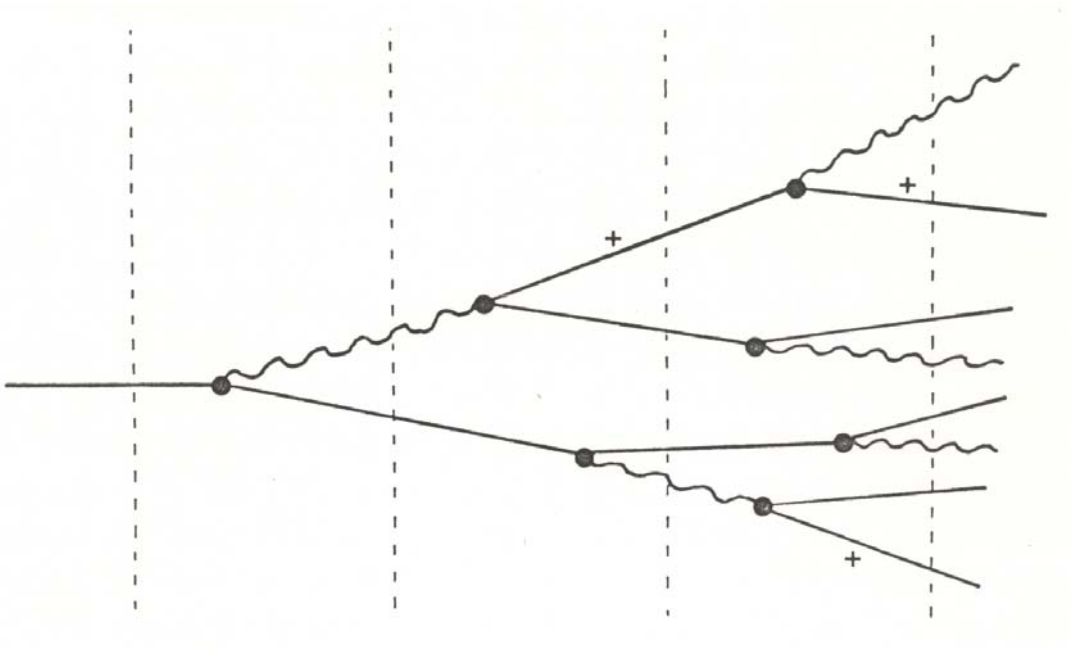
Electromagnetic Shower





- At energies above 100 MeV, $e^{\pm}$ loses energy mainly through bremsstrahlung emitting photons
- At similar energies, γ's interact with mainly through pair production generating $e^{\pm}$
- At high energies, $\sigma(E) \sim$ constant

EM Showers



- $e^+/e^-/\gamma$ cascade (degrading energy in each stage) mainly through successive bremsstrahlung and pair production
- Number of particles in the shower increases till the energies of the particles reach $E \rightarrow \varepsilon_c$, critical energy
- Beyond this energy, ionization/excitation takes over and the shower decays out

EM Shower Parameters

- Energy loss due to radiation is governed by L_R, radiation length of the material traversed. L_R in $g.cm^{-2}$

$$-\frac{dE}{dx} = \frac{E}{L_R} \qquad -\frac{1}{L_R} = 4\alpha N_A Z^2 r_e^2 \left[\ln(183 \cdot Z^{-\frac{1}{3}}) + \frac{1}{18}\right]$$

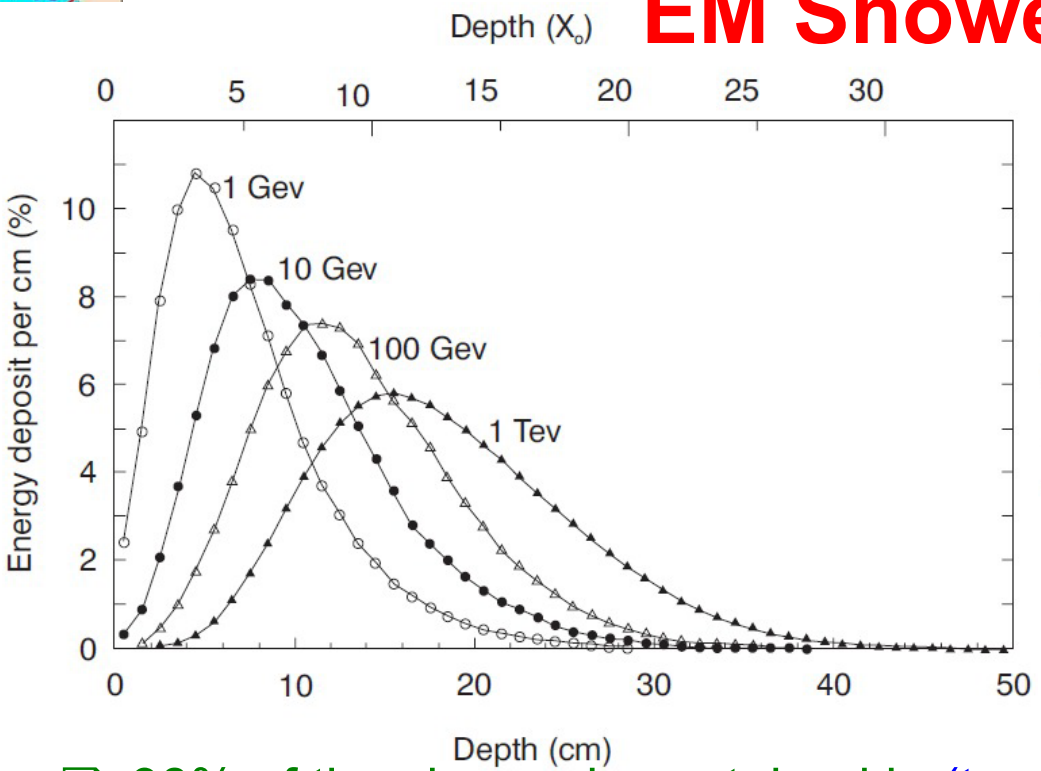
- Both bremsstrahlung and pair production are highly forward peaked. Lateral growth of the shower comes dominantly from multiple scattering at these energies
- Low energy end of a shower is generated through collision process

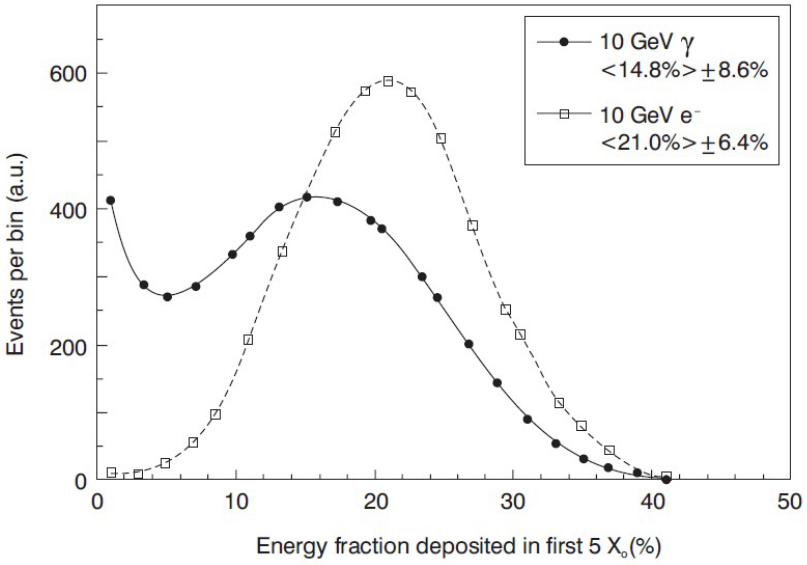
$$-\frac{\Delta E}{\Delta x}|_{collision} = -\frac{\epsilon_c}{L_R} \qquad \epsilon_c \simeq \frac{610}{Z + 1.24}\ (MeV)$$

- Beyond shower maximum. there is an exponential decay of the shower $[\exp(-t/\lambda_{Att})]$ $\lambda_{Att} \simeq (3.4 \pm 0.5) L_R$
- Angular distribution for Compton scattering, photo-electric effect is isotropic causing further increase in the lateral size of the shower
- Shower profile is determined by Moliere radius ρ_M. 95% of energy deposited is contained in a cylinder of radius $2\rho_M$.

$$\rho_M = 21\frac{L_R}{\epsilon_c} \simeq 7\frac{A}{Z}\ g \cdot cm^{-2}$$

EM Showers





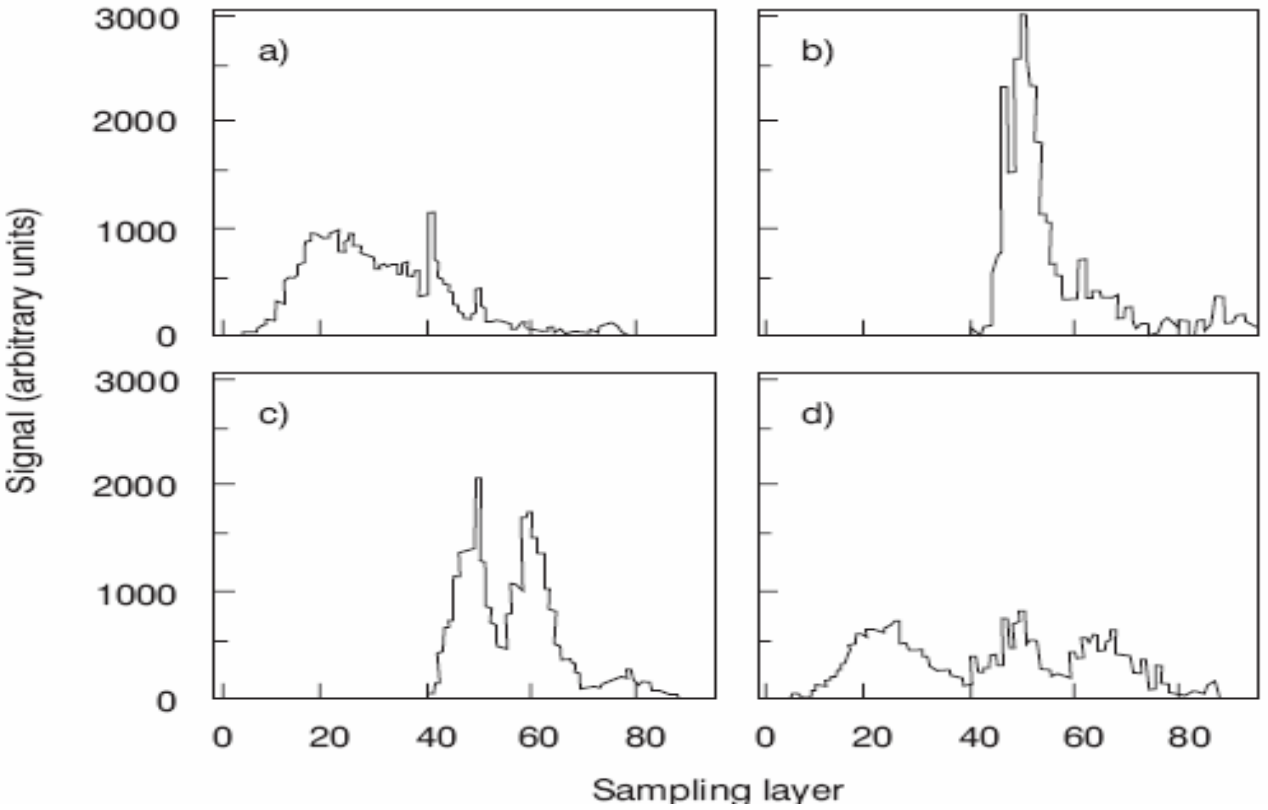
- 98% of the shower is contained in $(t_{max}+4\lambda_{att})$ where the position of shower maximum t_{max} increases only logarithmically with incident energy E.
- Lateral size of the shower changes with shower depths – broader at or beyond shower maximum.
- While radiation length (hence shower length) depends strongly on material, lateral size is roughly energy independent.
- Showers initiated by electrons and photons are different in the first few radiation lengths. For a fully absorbed shower the difference is reduced.

Hadronic Shower

- They are similar to electromagnetic shower, but with greater variety and complexity due to hadronic processes
- Strong interaction is responsible for
  - Production of hadronic shower particles, ~90% of these are pions. Neutral pions decay to 2 γ's which develop em showers
  - Interaction with nucleus – neutrons/protons are released from nucleus and the binding energy is lost from producing more shower particles
- EM showers produced by π°'s develop in the same way as those due to $e^{\pm}/\gamma$'s. Fraction of π° increases with energy. Typically EM energy fraction is ~30% at 10 GeV increasing to ~50% at 100 GeV.
- The remaining energy is carried by ionizing particles, neutrons and invisible component (lost in binding energies or carried by ν's from decays). In lead they are roughly in the ratio 56:10:34 and two-third of ionizing energy is due to protons.

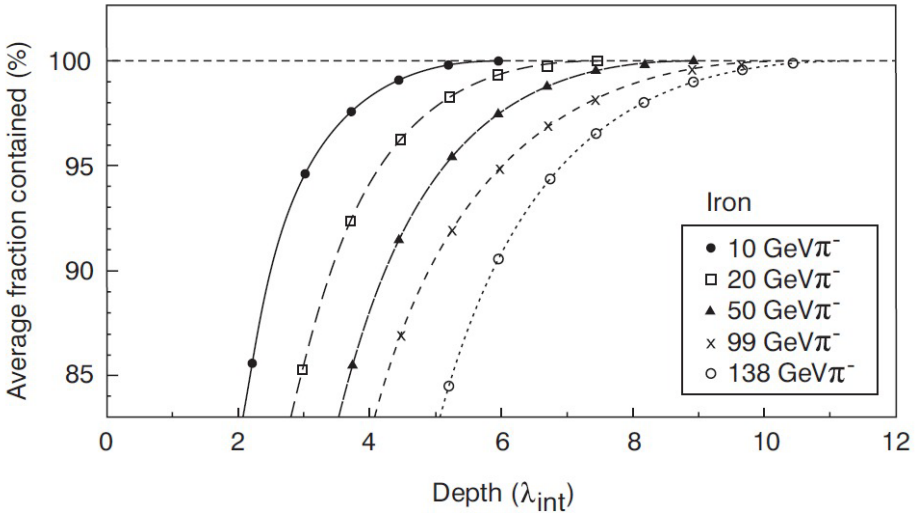
Fluctuations in Hadronic Showers

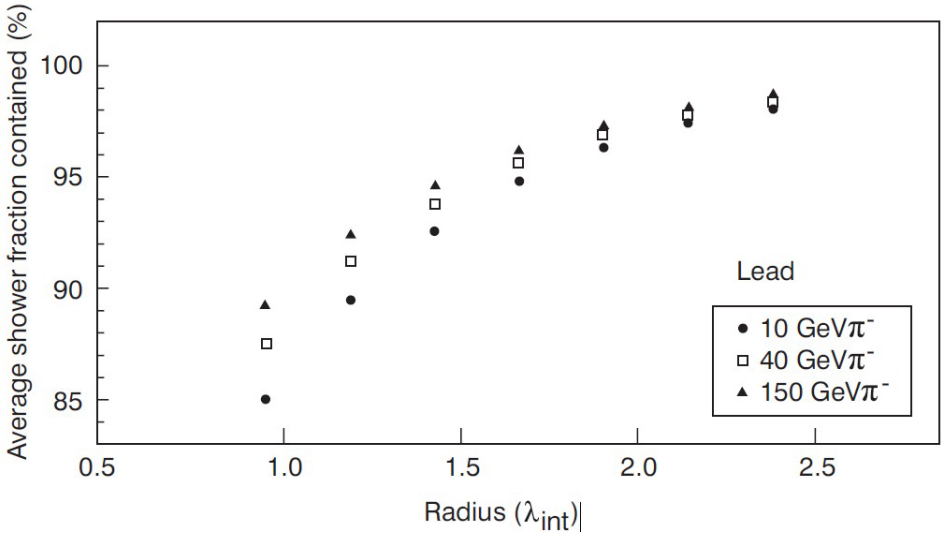
- There is a large variety of profiles in hadronic showers
- This depends on π° multiplicity in each step of interactions
- Leakage plays an important role even though the average containment is high



Hadronic Shower

- Typical scale is collision length $\lambda = \frac{A}{N_A \rho \sigma_{aA}}$
- Shower maximum occurs at $t_{max}(\lambda) \sim 0.2 \ln E + 0.7$
- Decay of shower is slower: power law ($\lambda E^{0.13}$) rather than logarithmic in E
- Transverse dimension is controlled by λ – laterally it takes less material to contain the shower at higher energies (larger fraction of EM energy)



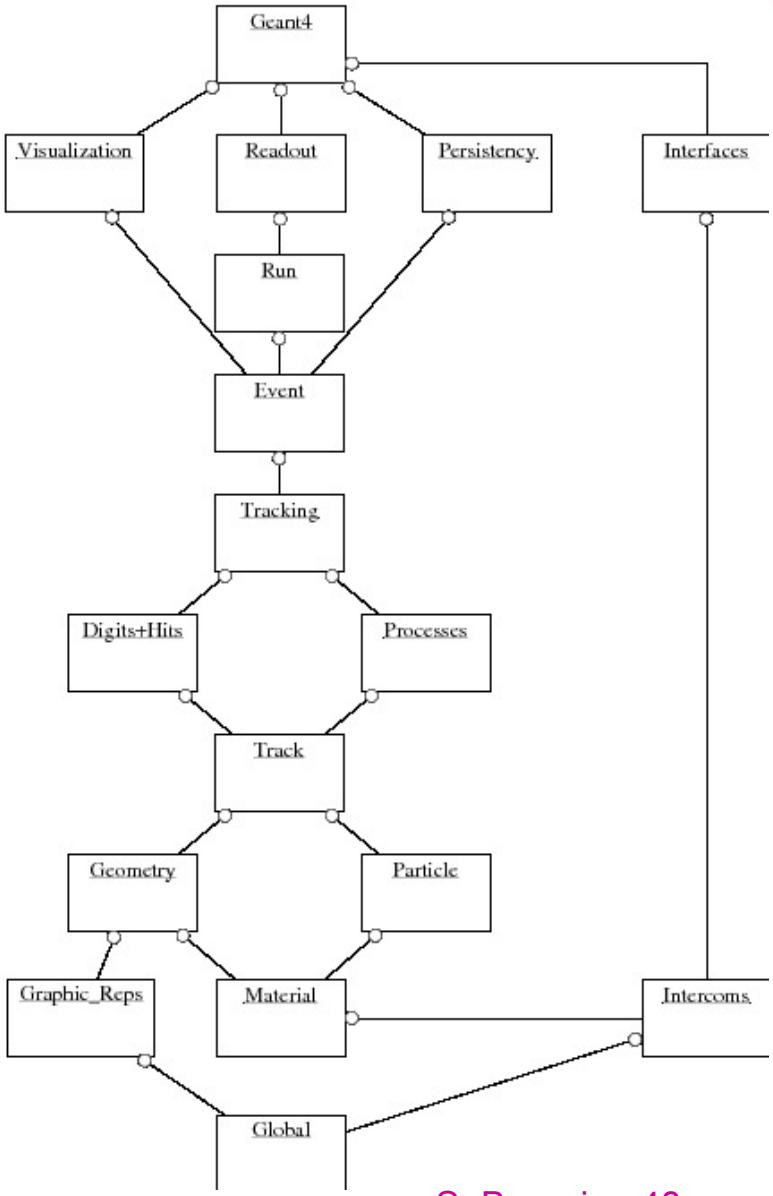


Detector Simulation

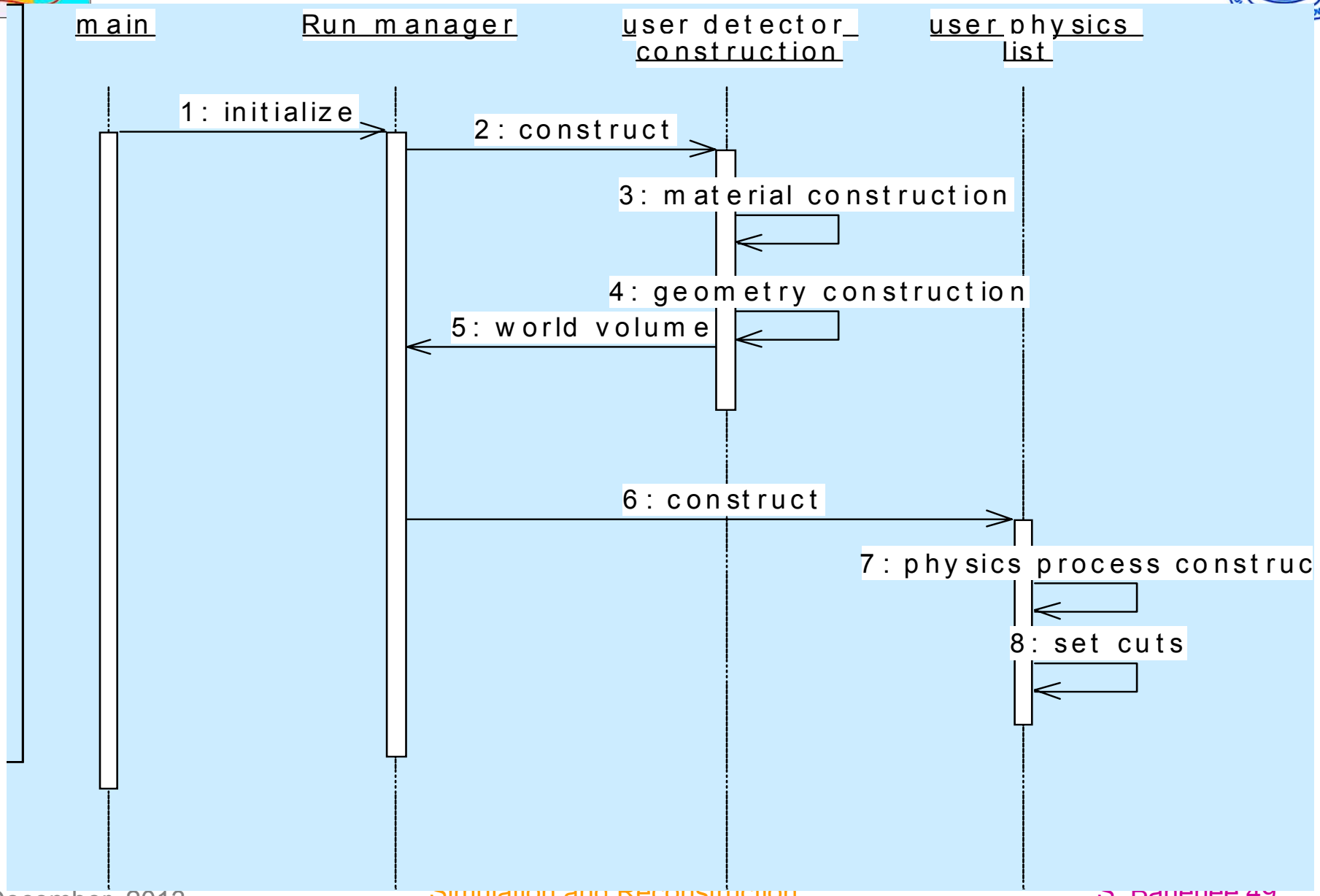
- Interaction of particles with matter + Monte Carlo techniques → Detector Simulation
- All modern experiments have a simulation code and they are usually based on some basic toolkit → one such example is Geant4
- Geant4 provides tools for particle transport and tools to model experimental environments
- The user needs to use Geant4 tools
  - to tell Geant4 kernel about the simulation configuration
  - to interact with Geant4 kernel itself
- The user must tell Geant4 what **he/she only** knows
  - The experimental scenario
    - Geometry, materials, sensitive and passive elements
    - Primary particles, radiation environment
  - What the user wants to happen during transport
    - which particles to be tracked
    - which physics processes would be of interest *(and which options for modeling are preferable)*
    - how precise the simulation is going to be

Geant4

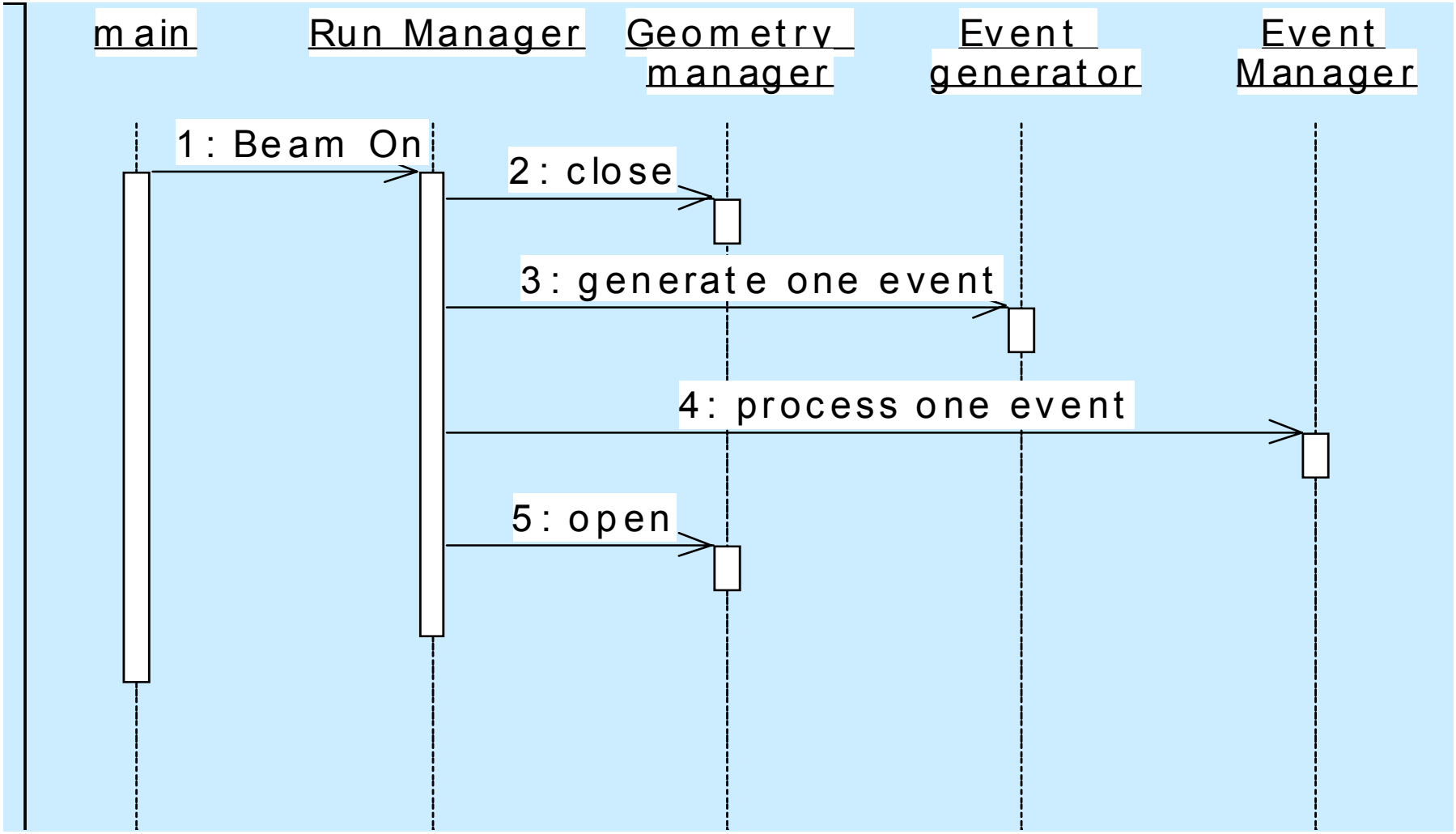
- Geant4 is a toolkit which is available for nearly 15 years
- It is based on object oriented technology. Many of its features are derived by a complete re-analysis and re-design of its predecessor Geant3
- It has modular structure divided into sub-domains linked with a uni-directional flow of dependencies
- It came from a collaboration of > 100 people from all over the world
- The first production version was released in 1999



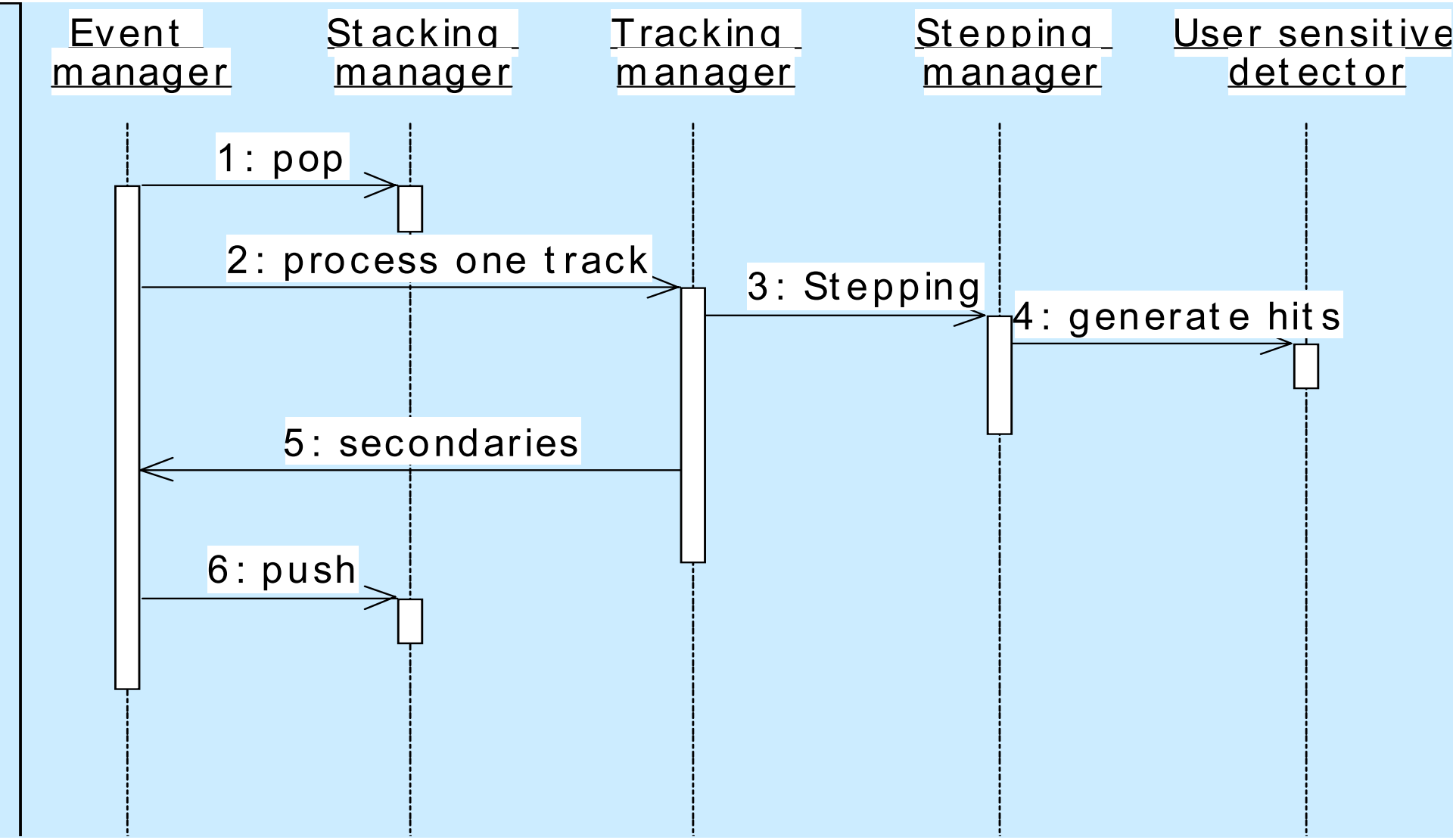
Initialization



Beam On

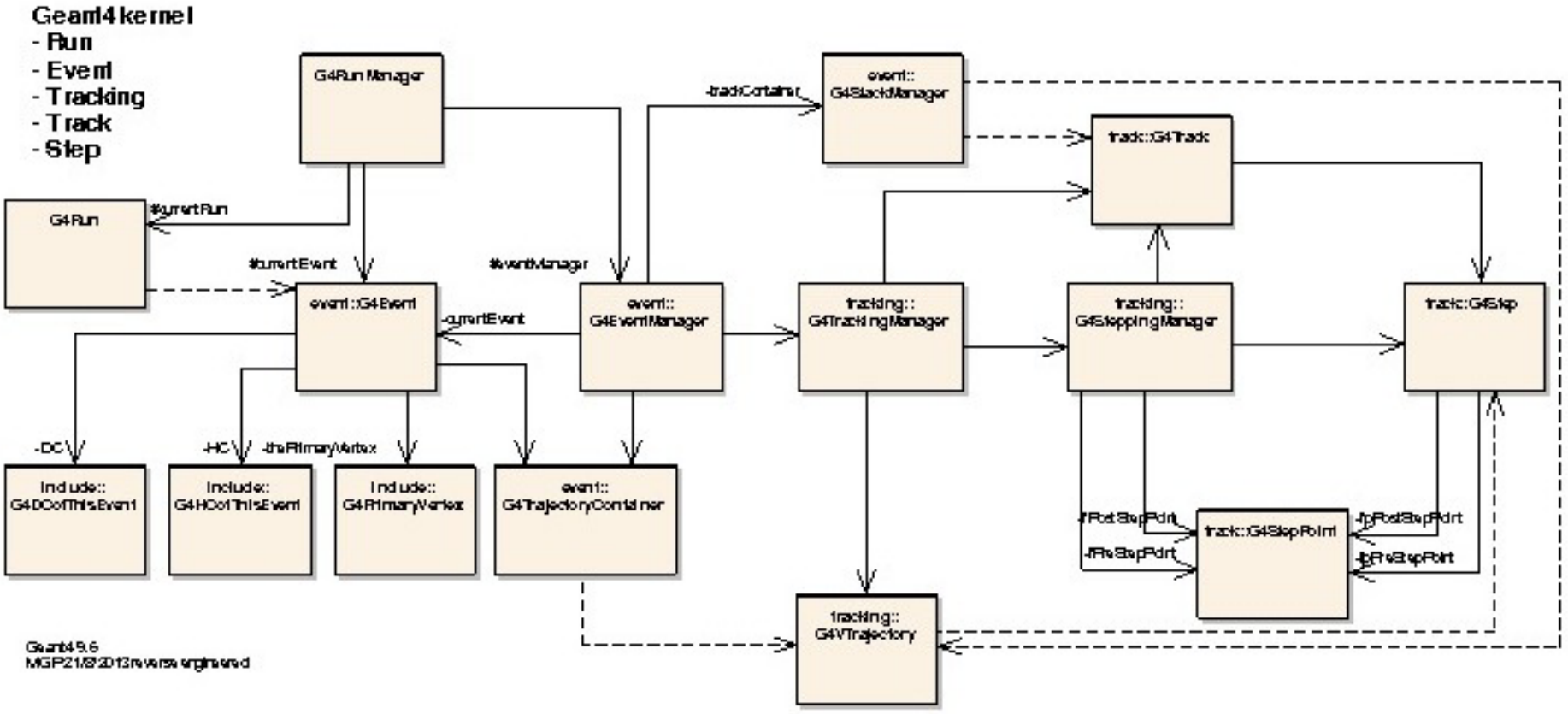

main
Run Manager
Geometry manager
Event generator
Event Manager
1: Beam On
2: close
3: generate one event
4: process one event
5: open

Event Processing



Some of the terminologies

- Run, Event, Track, Step, StepPoint
- Track ←→ trajectory, step ←→ trajectory point
- Process, Hit, ……



Run in Geant4

- As an analogy of the real experiment, a run of Geant4 starts with "Beam On".
- Within a run, the user cannot change
  - detector setup
  - settings of physics processes
- Conceptually, a run is a collection of events which share the same detector and physics conditions.
  - A run consists of one event loop.
- At the beginning of a run, geometry is optimized for navigation and cross-section tables are calculated according to materials appear in the geometry and the cut-off values defined.
- G4RunManager class manages processing a run, a run is represented by G4Run class or a user-defined class derived from G4Run.
  - A run class may have a summary results of the run.
- G4UserRunAction is the optional user hook.

Event in Geant4

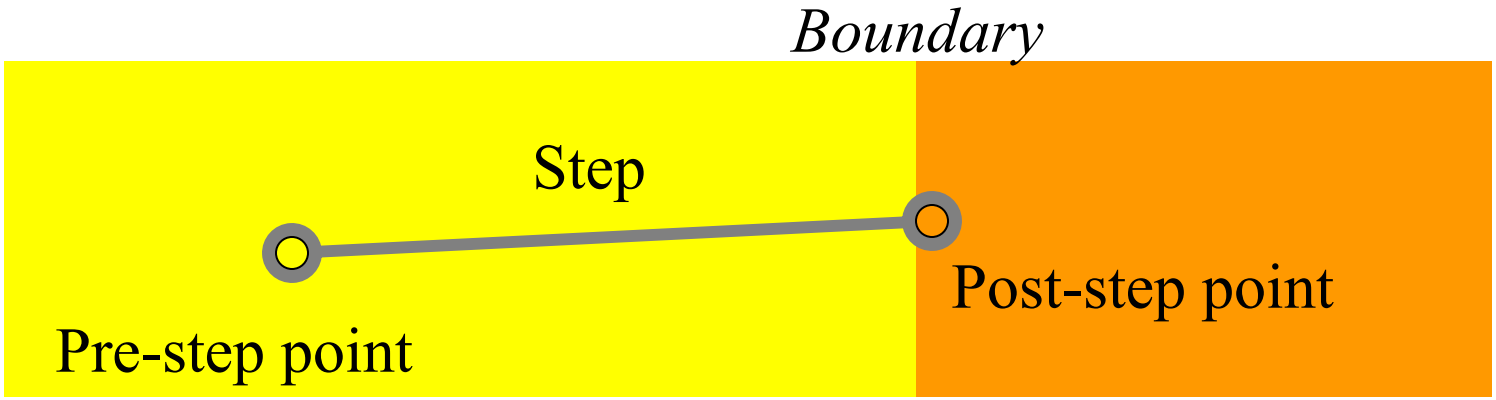
- An event is the basic unit of simulation in Geant4.
- At beginning of processing, primary tracks are generated. These primary tracks are pushed into a stack.
- A track is popped up from the stack one by one and "tracked". Resulting secondary tracks are pushed into the stack.
  - This "tracking" lasts as long as the stack has a track.
- When the stack becomes empty, processing of one event is over.
- G4Event class represents an event. It has following objects at the end of its (successful) processing.
  - List of primary vertices and particles (as input)
  - Hits and Trajectory collections (as output)
- G4EventManager class manages processing an event.
- G4UserEventAction is the optional user hook.

Track in Geant4

- Track is a snapshot of a particle.
  - It has physical quantities of current instance only. It does not record previous quantities.
  - Step is a “delta” information to a track. Track is not a collection of steps. Instead, a track is being updated by steps.
- Track object is deleted when
  - it goes out of the world volume,
  - it disappears (by e.g. decay, inelastic scattering),
  - it goes down to zero kinetic energy and no “AtRest” additional process is required, or
  - the user decides to kill it artificially.
- No track object persists at the end of event.
  - For the record of tracks, use trajectory class objects.
- G4TrackingManager manages processing a track, a track is represented by G4Track class.
- G4UserTrackingAction is the optional user hook.

Step in Geant4

- Step has two points and also “delta” information of a particle (energy loss on the step, time-of-flight spent by the step, etc.).
- Each point knows the volume (and material). In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume.
  - Because one step knows materials of two volumes, boundary processes such as transition radiation or refraction could be simulated.
- G4SteppingManager class manages processing a step, a step is represented by G4Step class.
- G4UserSteppingAction is the optional user hook.



Trajectory and Trajectory Point

- Track does not keep its trace. No track object persists at the end of event.
- G4Trajectory is the class which copies some of G4Track information.
  G4TrajectoryPoint is the class which copies some of G4Step information.
  - G4Trajectory has a vector of G4TrajectoryPoint.
  - At the end of event processing, G4Event has a collection of G4Trajectory objects.
    - /tracking/storeTrajectory must be set to 1.
- Keep in mind the distinction:
  - G4Track ←→ G4Trajectory, G4Step ←→ G4TrajectoryPoint
- Given G4Trajectory and G4TrajectoryPoint objects persist till the end of an event, one should be careful not to store too many trajectories:
  - e.g. avoid for high energy EM shower tracks.
- G4Trajectory and G4TrajectoryPoint store only the minimum information
  - One can create one's own trajectory / trajectory point classes to store the required information. G4VTrajectory and G4VTrajectoryPoint are the base classes.

Particle in Geant4

- A particle in Geant4 is represented by three layers of classes.
- G4Track
  - Position, geometrical information, etc.
  - This is a class representing a particle to be tracked.
- G4DynamicParticle
  - "Dynamic" physical properties of a particle, such as momentum, energy, spin, etc.
  - Each G4Track object has its own and unique G4DynamicParticle object.
  - This is a class representing an individual particle.
- G4ParticleDefinition
  - "Static" properties of a particle, such as charge, mass, life time, decay channels, etc.
  - G4ProcessManager which describes processes involving to the particle
  - All G4DynamicParticle objects of same kind of particle share the same G4ParticleDefinition.

Tracking and Process

- Geant4 tracking is general.
  - It is independent of
    - the particle type
    - the physics processes involving to a particle
  - It gives the chance to all processes
    - to contribute to determining the step length
    - to contribute any possible changes in physical quantities of the track
    - to generate secondary particles
    - to suggest changes in the state of the track
      - e.g. to suspend, postpone or kill it.

Process in Geant4

- In Geant4, particle transportation is a process as well, by which a particle interacts with geometrical volume boundaries and field of any kind.
  - Because of this, shower parameterization process can take over from the ordinary transportation without modifying the transportation process.
- Each particle has its own list of applicable processes. At each step, all processes listed are invoked to get proposed physical interaction lengths.
- The process which requires the shortest interaction length (in space-time) limits the step.
- Each process has one or combination of the following natures.
  - AtRest
    - e.g. muon decay at rest
  - AlongStep (a.k.a. continuous process)
    - e.g. Cerenkov process
  - PostStep (a.k.a. discrete process)
    - e.g. decay on flight

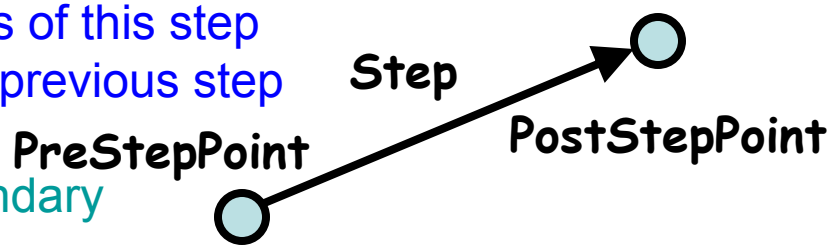
Track Status

At the end of each step, according to the processes involved, the state of a track may be changed.

- The user can also change the status in UserSteppingAction
- Status shown in brown are artificial, i.e. Geant4 kernel won't set them, but the user can set
  - fAlive
    - continue the tracking
  - fStopButAlive
    - the track has come to zero kinetic energy, but still AtRest process to occur
  - fStopAndKill
    - the track has lost its identity because it has decayed, interacted or gone beyond the world boundary
    - secondaries will be pushed to the stack
  - fKillTrackAndSecondaries
    - Kill the current track and also associated secondaries.
  - fSuspend
    - suspend processing of the current track and push it and its secondaries to the stack
  - fPostponeToNextEvent
    - postpone processing of the current track to the next event
    - secondaries are still being processed within the current event.

Step Status

- Step status is attached to G4StepPoint to indicate why that particular step was determined
  - Use "PostStepPoint" to get the status of this step
  - "PreStepPoint" has the status of the previous step
    - fWorldBoundary
      - step reached the world boundary
    - fGeomBoundary
      - step is limited by a volume boundary except the world
    - fAtRestDoItProc, fAlongStepDoItProc, fPostStepDoItProc
      - step is limited by a AtRest, AlongStep or PostStep process
    - fUserDefinedLimit
      - step is limited by the user Step limit
    - fExclusivelyForcedProc
      - step is limited by an exclusively forced (e.g. shower parameterization) process
    - fUndefined
      - Step not defined yet



- If the first step in a volume is to be identified, pick fGeomBoudary status in PreStepPoint
- If a step getting out of a volume is to be identified, pick fGeomBoundary status in PostStepPoint

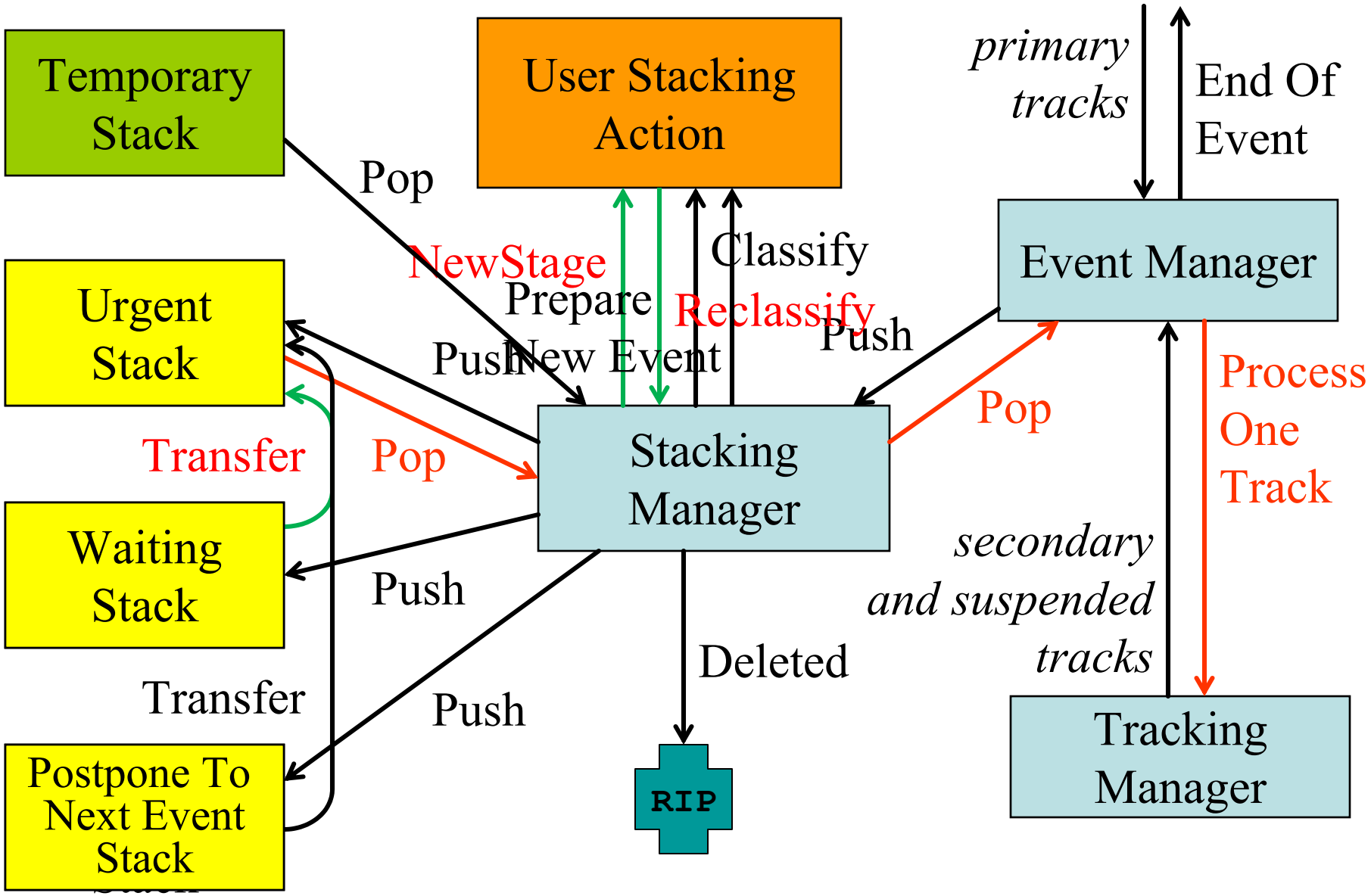
Extraction of useful information

- Given geometry, physics and primary track generation, Geant4 does proper physics simulation “silently”
  - the user has to add a bit of code to extract useful information
- There are two ways:
  - Use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)
    - the user has an access to almost all information
    - straight-forward, but do-it-yourself
  - Use Geant4 scoring functionality
    - assign G4VSensitiveDetector to a volume
    - Hits collection is automatically stored in G4Event object, and automatically accumulated if user-defined Run object is used
    - use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary

Track Stacks in Geant4

- By default, Geant4 has three track stacks:
  - "Urgent", "Waiting" and "PostponeToNextEvent“
  - Each stack is a simple "last-in-first-out" stack
  - User can arbitrary increase the number of stacks
- ClassifyNewTrack() method of UserStackingAction decides which stack each newly storing track to be stacked (or to be killed)
  - By default, all tracks go to Urgent stack
- A Track is popped up only from Urgent stack
- Once Urgent stack becomes empty, all tracks in Waiting stack are transferred to Urgent stack
  - And NewStage() method of UserStackingAction is invoked
- Utilizing more than one stacks, user can control the priorities of processing tracks without paying the overhead of "scanning the highest priority track“
  - Proper selection/abortion of tracks/events with well designed stack management provides significant efficiency increase of the entire simulation

Stacking Mechanism



Attaching user information

- Abstract classes:
  - The user can use his/her own class derived from the provided base class
  - G4Run, G4VHit, G4VDigit, G4VTrajectory, G4VTrajectoryPoint
- Concrete classes:
  - The user can attach a user information class object
    - G4Event - G4VUserEventInformation
    - G4Track - G4VUserTrackInformation
    - G4PrimaryVertex - G4VUserPrimaryVertexInformation
    - G4PrimaryParticle - G4VUserPrimaryParticleInformation
    - G4Region - G4VUserRegionInformation
  - User information class object is deleted when associated Geant4 class object is deleted

Generating Primary Particles

- Each Geant4 Event starts with generation of one or multiple primary particles
- It is up to the user to define primary particle properties
  - Particle type, e.g. electron, gamma, ion
  - Initial kinetics, e.g. energy, momentum, origin and direction
  - Additional properties, e.g. polarization
- These properties can be divided into a **primary vertex:** starting point in space and time
- **Primary particle:** initial momentum, polarization, PDG code, list of daughters for decay chains
- A primary particle can be a particle which can not usually be tracked by Geant4

Primary Generator Action

- Mandatory user action which **controls** the generation of primary particles
- **It should not generate primaries itself.** The primary generator does this.
- Implement your particle “shot”, “rail”, or machine gun here. It can also be a particle bomb if you like.
  - By using e.g. the G4ParticleGun
  - Repeatedly for a single event
  - Sampling particle type and direction randomly
  - Or using one of the other event generators
    - G4HEPEvtInterface
    - G4HEPMCInterface
    - G4GeneralParticleSource
    - G4ParticleGun

PrimaryGeneratorAction

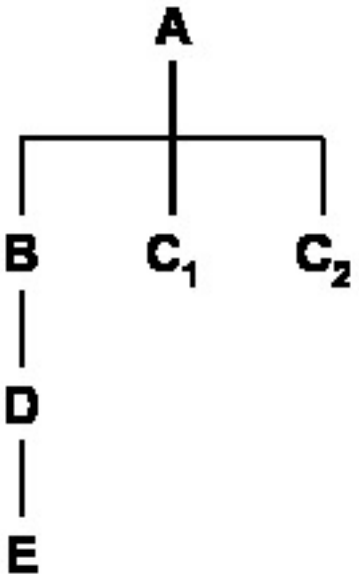
- Inherits from G4VUserPrimaryGeneratorAction
- User should override GeneratePrimaries for particle generation

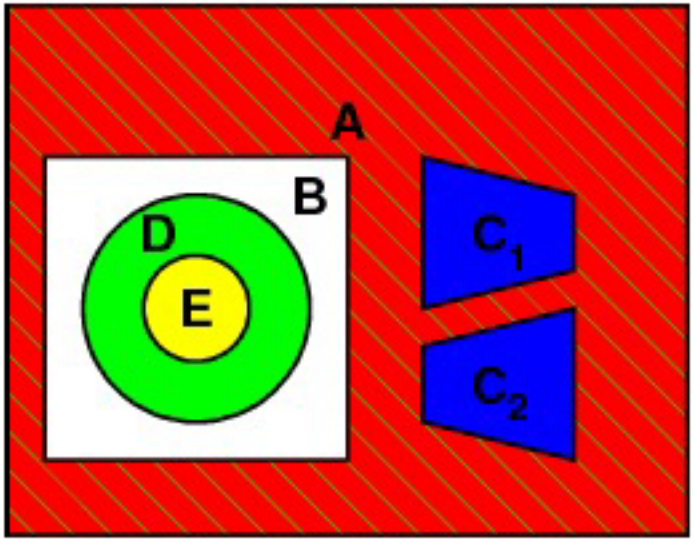
```
PrimaryGeneratorAction::PrimaryGeneratorAction(const G4String & parName,
G4double energy, G4ThreeVector pos, G4ThreeVector momDirection){

        const G4int nParticles = 1;
        fParticleGun = new G4ParticleGun(nParticles);
        G4ParticleTable* parTable = G4ParticleTable::GetParticleTable();
        G4ParticleDefinition* parDefinition = parTable->FindParticle(parName);
        fParticleGun->SetParticleDefinition(parDefinition);
        fParticleGun->SetParticleEnergy(energy);
        fParticleGun->SetParticlePosition(pos);
        fParticleGun->SetParticleMomentumDirection(momDirection);
}
```

```
PrimaryGeneratorAction::GeneratePrimaries(G4Event* evt){

        //some additional random sampling here

        fParticleGun->GeneratePrimaryVertex(evt);

}
```

Modeling of a Detector

- Detector is modeled by a geometrical shape and its material content → "Volume"
- Several volumes can describe different components of the detector system. Put them together in a hierarchical structure





Composite Volume ≡ Experimental Setup

Material

- Material has a Name, effective Atomic Number and Weight, Density, Radiation (L_R) and absorption (λ) length
- Can be defined by specifying these attributes
- If radiation and absorption lengths are not known but the chemical composition is known, one can furnish these information and Geant4 will compute the required attributes for the application
- One can also add the state, isotopic properties, …. Some of these are essential to study activation, ,,
  - Define pseudo-elements

    new G4Material (name, Z, A, density, state, temperature, pressure);
  - Define a mixture of elements in atomic or weighted proportion

    new G4Material (name, density, nComponents);

    ->AddElement (material, fraction);

    new G4Element (name, symbol, Z, A);

    ->AddElement (element, nAtom);
  - Build element from isotopes

    new G4Element (name, symbol, numIsotopes);

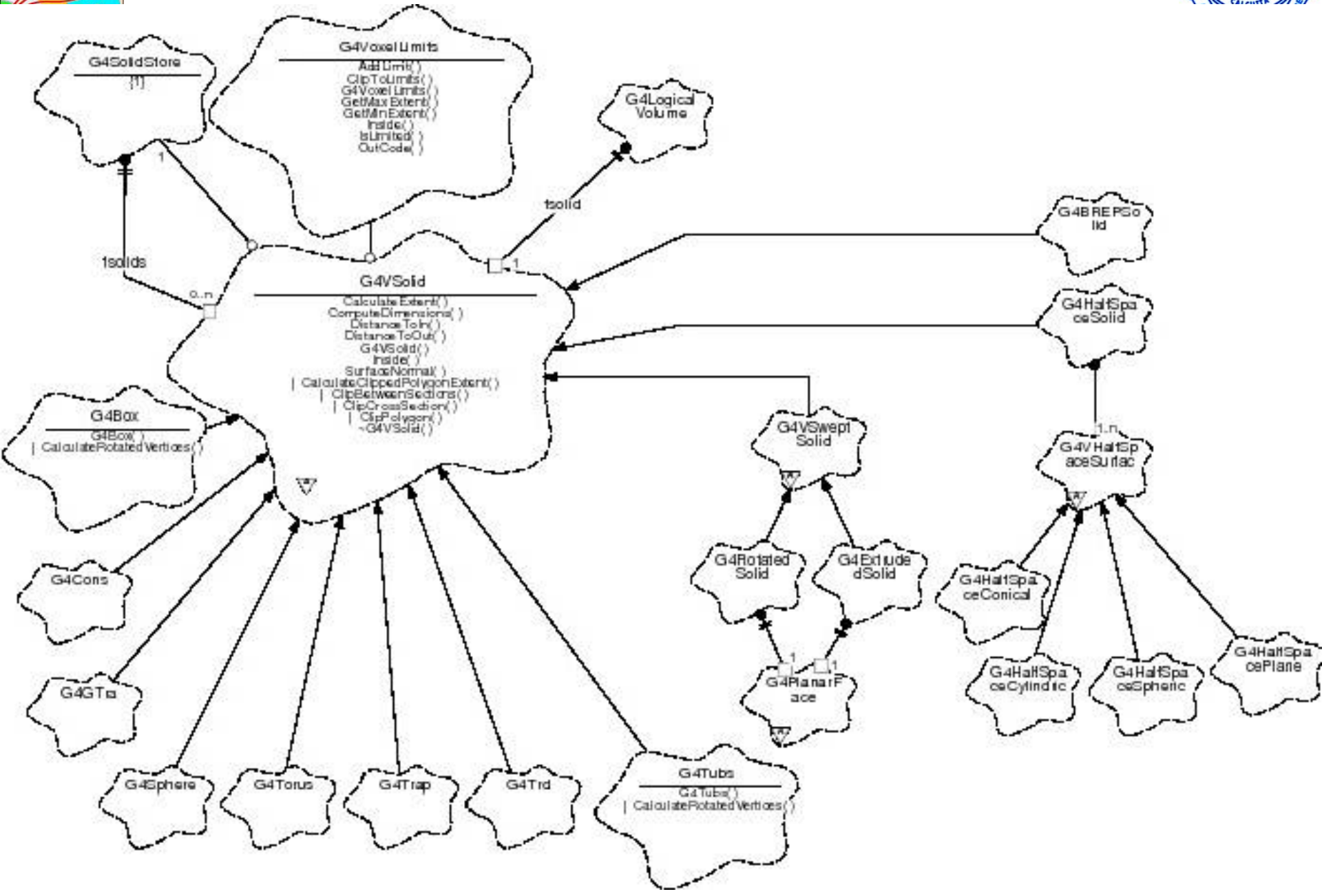
    new G4Isotope (name, Z, N);

    ->AddIsotope (isotope, abundance);

Volume

- A volume is defined by its shape, dimensional parameters and its material content. Shape with dimensional parameters is called a Solid and association of a Solid and Material is called a LogicalVolume.
- There are several ways of defining Solids:
  - Computed Solid Geometry (CSG): G4Box, G4Trd, G4Trap, G4Tubs, G4Cons, G4Sphere, G4PolyCone, …..
  - Boundary Representations (BREP): G4BrepSolidPcone, ….. (much slower navigation)
  - Boolean: Solids made out by adding, subtracting, intersecting several solids: G4RotateSolid, ….
  - STEP: Imported from the CAD system
- System of Units: Though internally, a convention is used for unit system, the recommendation is not to remember them and use the units explicitly:
  - double length = 5 * cm;
  - double angle = 30*deg;

Solids


G4SolidStore
{1}
G4VoxelLimits
AddLimit( )
ClipToLimits( )
G4VoxelLimits( )
GetMaxExtent( )
GetMinExtent( )
Inside( )
IsLimited( )
OutCode( )
G4LogicalVolume
1
1solids
0..n
tsolid
1
G4VSolid
CalculateExtent( )
ComputeDimensions( )
DistanceToIn( )
DistanceToOut( )
G4VSolid( )
Inside( )
SurfaceNormal( )
| CalculateClippedPolygonExtent( )
| ClipBetweenSections( )
| ClipCrossSection( )
| ClipPolygon( )
~G4VSolid( )
G4BREPSolid
G4HalfSpaceSolid
G4Box
G4Box( )
| CalculateRotatedVertices( )
G4VSweptSolid
1..n
G4VHalfSpaceSurface
G4Cons
G4RotatedSolid
G4ExtrudedSolid
G4HalfSpaceConical
G4GTra
1
1
G4PlanarFace
G4HalfSpaceCylindric
G4HalfSpaceSpheric
G4HalfSpacePlane
G4Sphere
G4Torus
G4Trap
G4Trd
G4Tubs
G4Tubs( )
| CalculateRotatedVertices( )

Define a volume

- First define a material, say Air consisting of 2 constituent elements Nitrogen and Oxygen in a given weight proportion.
  - First define Nitrogen and Oxygen with their appropriate Z, A values:

```
G4Element *eln = new G4Element("Nitrogen","N",Z=7,A=14.01*g/mole);
G4Element *elo = new G4Element("Oxygen","O",Z=8,A=16.00*g/mole);
```

  - Then define Air and add the two elements:

```
G4Material *air = new G4Material("Air",1.205E-03*g/cm3,2);
air->AddElement(eln,0.7);
air->AddElement(elo,0.3);
```

- Define a solid of a given name (say INOM) as a box of given half length, half width, half thickness:

```
G4Solid* inomSolid = new G4Box("INOM",1606*cm,706*cm,598*cm);
```

- Associate the solid with material to define the logical volume
  - `G4LogicalVolume* inomLog = new G4LogicalVolume(inomSolid,air,"INOML");`
- The reference frame is a right handed Cartesian coordinate system with the origin at the centre of the box

Define a Detector SetUp

To define a set up, one needs to

- Define a Master or World reference system
- Position the various components with respect to each other

Geant4 uses the concept of PhysicalVolume which is a LogicalVolume positioned in a Mother (PhysicalVolume or LogicalVolume) with a translation vector and rotation matrix (optional), For top level volume (defining the World reference system) the reference Mother Volume is a Null

- One useful was of defining daughter volume is by dividing an existing moher volume in equal n parts along a chosen axis (Cartesian, cylindrical or polar)
  - The creation and positioning is done in two separate step
- When a daughter is positioned inside a mother, the extent inside the mother occupied by the daughter gets filled with the material of the daughter volume

Can build up a tree like a Russian doll

Hooks for Positioning

```
G4PhysicalVolume* volume = new G4PVPlacemenet (rot,
                    G4ThreeVector(xpos*cm,ypos*cm,zpos*cm),
                    G4LogicalVolume* current,  Name,
                    G4LogicalVolume* mother,  false, copyNumber)
```

creates a PhysicalVolume volume by positioning a copy copyNumber of the LogicalVolume current inside the mother volume mother with a translation vector (G4ThreeVector) and a rotation matrix (G4RotationMatrix* rot)

If one needs to define a rotation matrix by specifying the angles (θ_i,φ_i) of the three axes (like in Geant3), one needs to follow the steps:

```
G4ThreeVector iAxis(sin(thetaI*deg)*cos(phiI*deg),
                    sin(thetaI*deg)*sin(phiI*deg), cos(thetaI*deg));
G4RotationMatrix* rot = new G4RotationMatrix();
rot->rotateAxis(xAxis,yAxis,zAxis);
rot->invert();
```

Hooks for Positioning (contd)

For dividing a parent volume, one needs to create the LogicalVolume using the standard steps (defining Solid, Material and LogicalVolume) and then position multiple replica through:

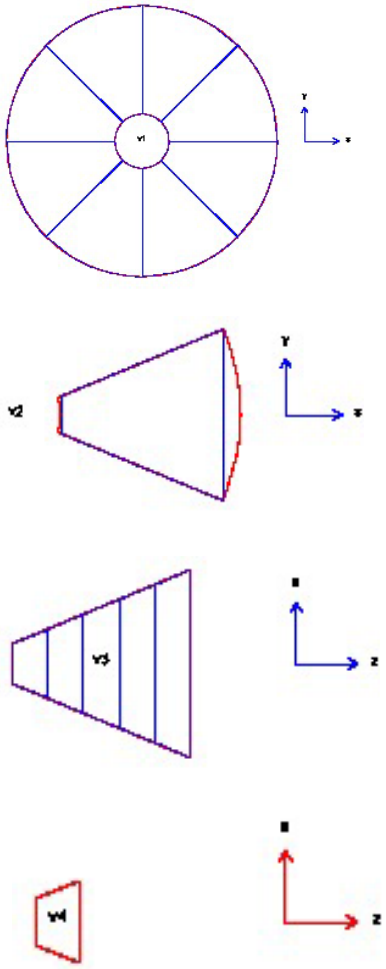
```
G4PhysicalVolume * volume = new G4PVReplica (NAME,
        G4LogicalVolume* current, G4LogicalVolume* mother,
        kAxis, nDivision, width, offset)
```

This needs more steps than in Geant3 (GSDVN) but is more general

The tree of physical volumes is instantiated at the time of tracking (G4VTouchable) and the will provide the unique identification of a volume

Example

- Construct geometry of a cylindrical drift chamber with 8 sectors each having 5 cells:



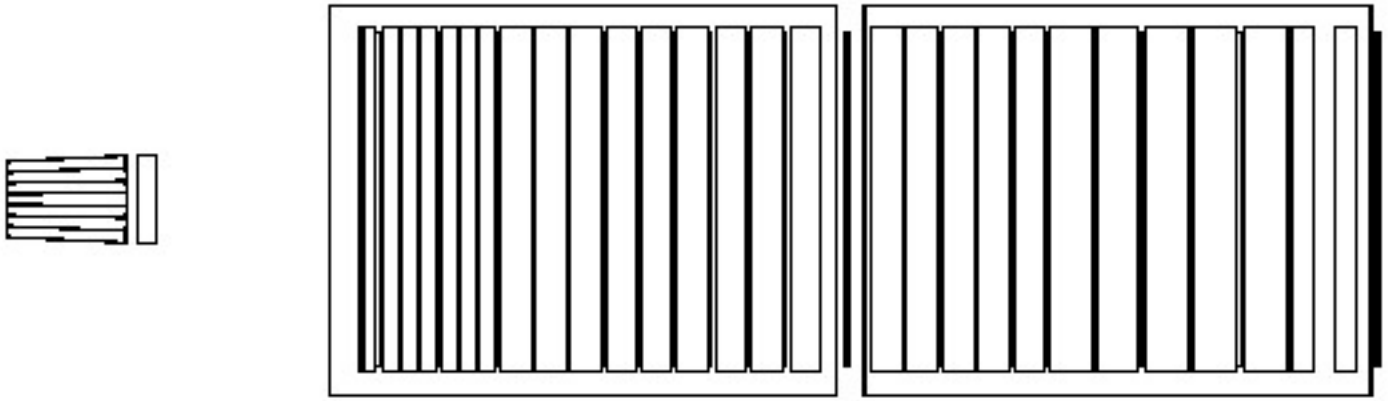
- Define volume VOL1 as a tube with inner and outer radii R_1, R_2 and half length L
- Divide the tube into 8 parts azimuthally and each section is called VOL2
- Define a trapezoid of half length L, width $[R_2\cos(\pi/8)-R_1]$ and two edges of dimension $2R_1\tan(\pi/8)$, 2R2tan(π/8). Position this volume VOL3 inside VOL2 with proper translation and rotation matrix
- Divide the trapezoid VOL3 into 5 parts along z-axis. Each part VOL4 will be a cell

Example

```
// define materials mat1, mat3 for VOL1, VOL3
G4VSolid* solid;
solid = new G4Tubs("VOL1", R1*cm, R2*cm, 0.5*L*cm, 0, twopi);
G4LogicalVolume *v1 = new G4LogicalVolume(solid, mat1, "VOL1");
// divide VOL1 along phi axis
solid = new G4Tubs("VOL2", R1*cm, R2*cm, 0.5*L*CM, -pi/8., pi/4.);
G4LogicalVolume *v2 = new G4LogicalVolume(solid, mat1, "VOL2");
new G4PVDivision ("VOL2", v2, v1, kPhi, 8, pi/4.);
// now the trapezoid
solid = new G4Trd ("VOL3", R1*tan(pi/8.)*cm, R2*tan(pi/8.)*cm, 0.5*L*cm, 0.5*L*CM, 0.5*(R2*cos(pi/8.)-
R1)*cm);
G4LogicalVolume* v3 = new G4LogicalVolume(solid, mat3, "VOL3");
// position VOL3 inside VOL2
G4ThreeVector xAxis(0,1.,0), yAxis(0,0,1.), zAxis(1.,0,0);
G4RotationMatrix* rot = new G4RotationMatrix();
rot->rotateAxis(xAxis, yAxis, zAxis);
rot->invert();
new G4PVPlacement (rot, G4ThreeVector(0.5*(R2*cos(pi/8.)+R1)*cm,0,0), "VOL3", v3, v2, false, 1);
// finally the cell
solid = new G4Trd ("VOL4", R1*tan(pi/8.)*cm, R2*tan(pi/8.)*cm, 0.5*L*cm, 0.5*L*CM, 0.5*(R2*cos(pi/8.)-
R1)*cm);
G4LogicalVolume* v4 = new G4LogicalVolume(solid, mat3, "VOL4");
new G4PVDivision ("VOL4", v4, v3, kZaxis, 5, 0.2*(R2*cos(pi/8.)-R1)*cm);
```

- This will create the volume tree:

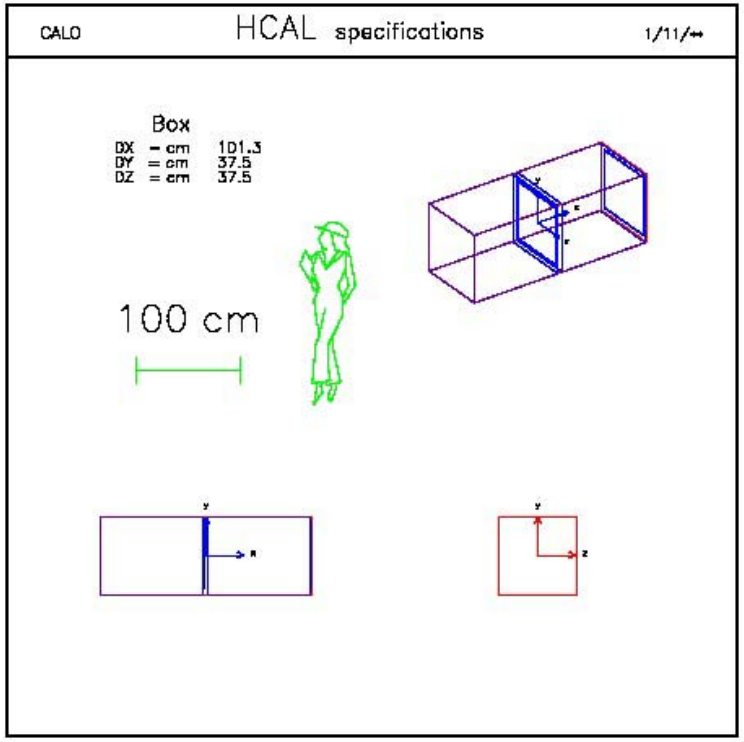
Practical Example

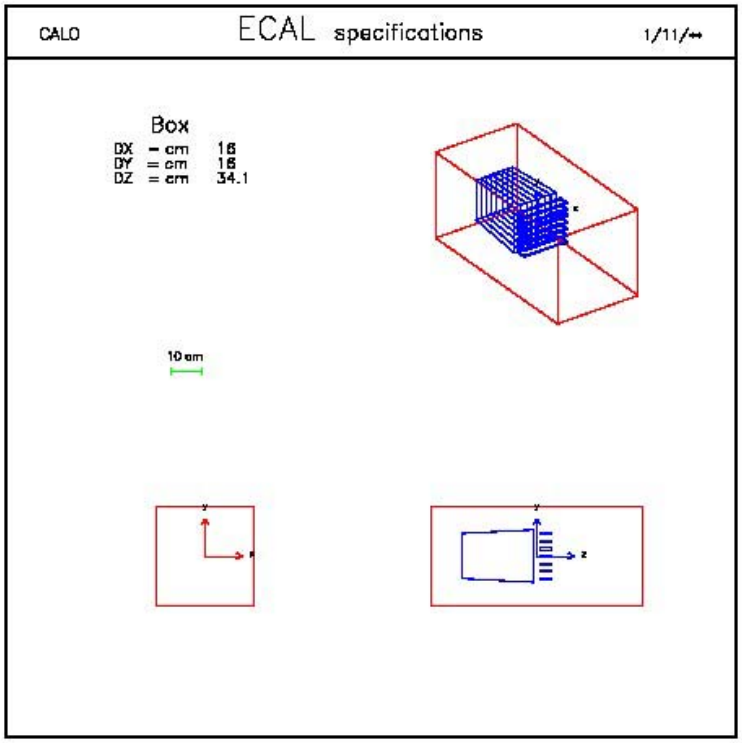


1996 CMS Test Beam Setup

- 7 x 7 crystal matrix made out of lead tungstate
- 28 layers of plastic scintillators interleaved with brass plates of varying thickness
- B-field along z-axis (perpendicular to the beam direction) with maximum field strength of 3 Tesla

CompositeCalorimeter

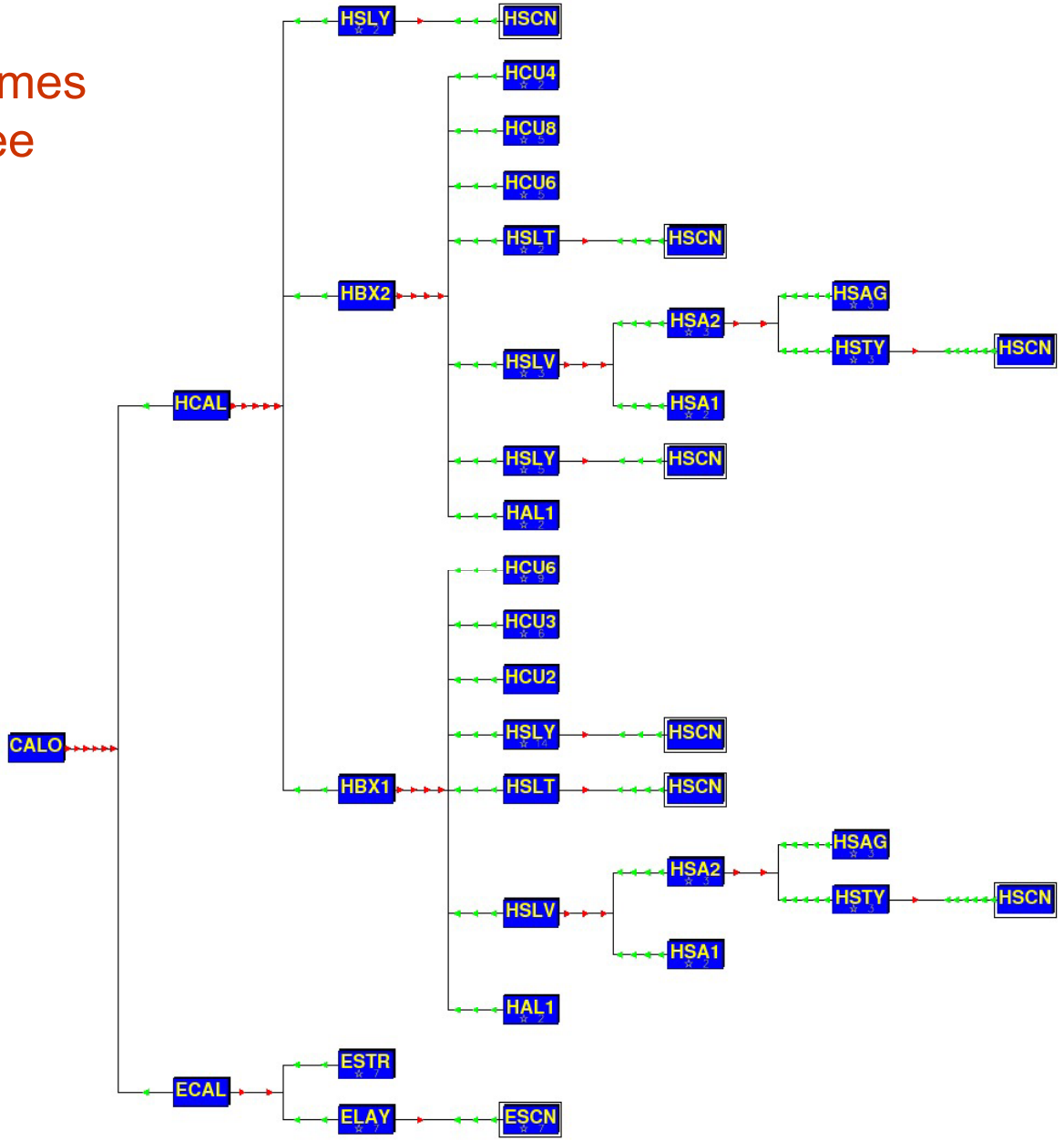




- HCAL contains two boxes made out of aluminum each housing absorber plates and scintillation layers
- ECAL contains the crystal matrix and some support structure
- Both ECAL and HCAL are placed in CALO which defines the world volume

Geometry Tree

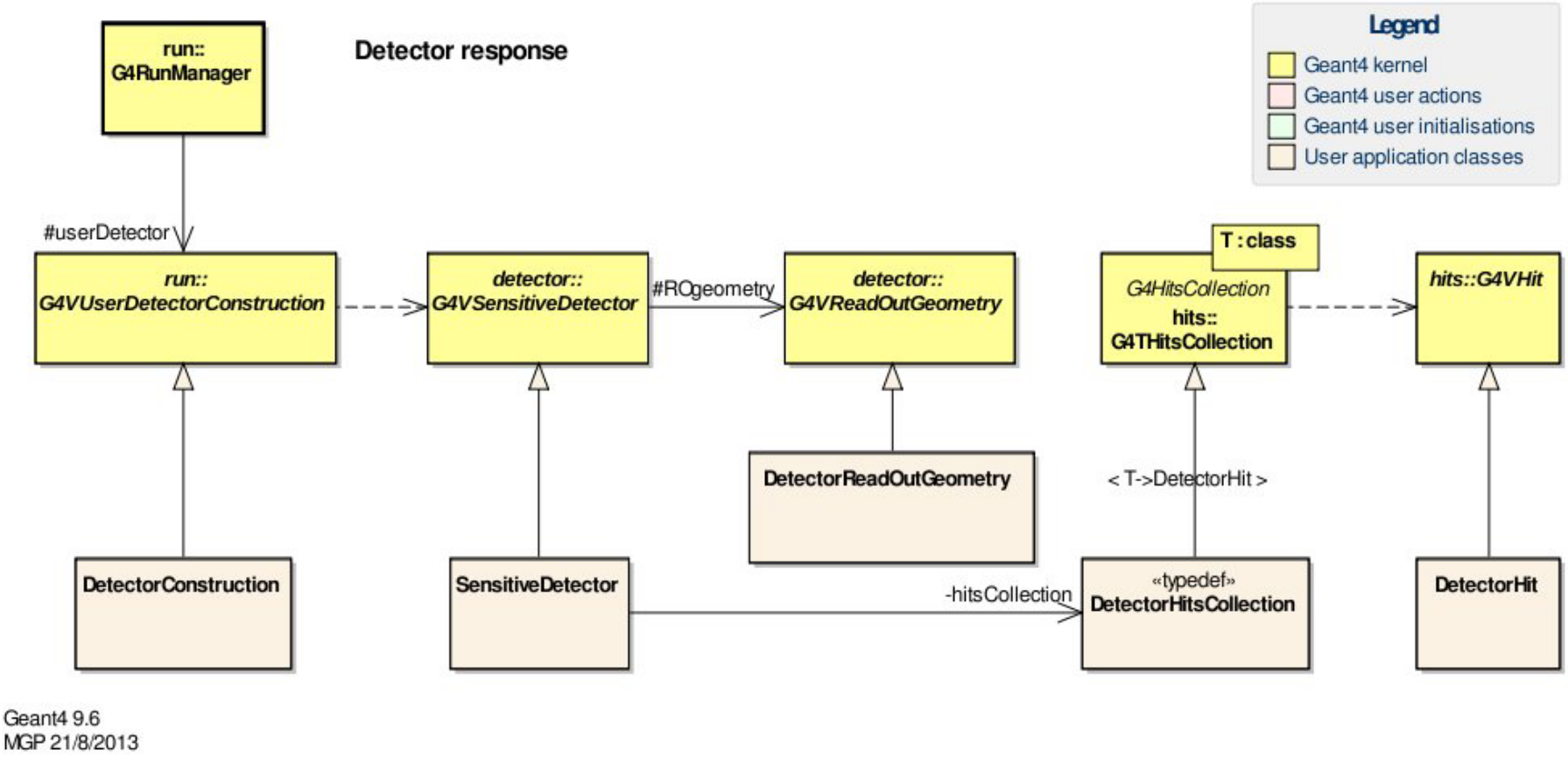
22 Logical Volumes
8 Levels in Tree



Extraction of useful information

- Given geometry, physics and primary track generation, Geant4 does proper physics simulation "silently"
  - The user needs to add a bit of code to extract useful information
- There are three ways:
  - Built-in scoring commands
    - Most commonly-used physics quantities are available
  - Use scorers in the tracking volume
    - Create scores for each event
    - Create own Run class to accumulate scores
  - Assign G4VSensitiveDetector to a volume to generate "hit"
    - Use user hooks (G4UserEventAction, G4UserRunAction) to get event / run summary
- The user may also use user hooks (G4UserTrackingAction, G4UserSteppingAction, etc.)
  - The user has full access to almost all information

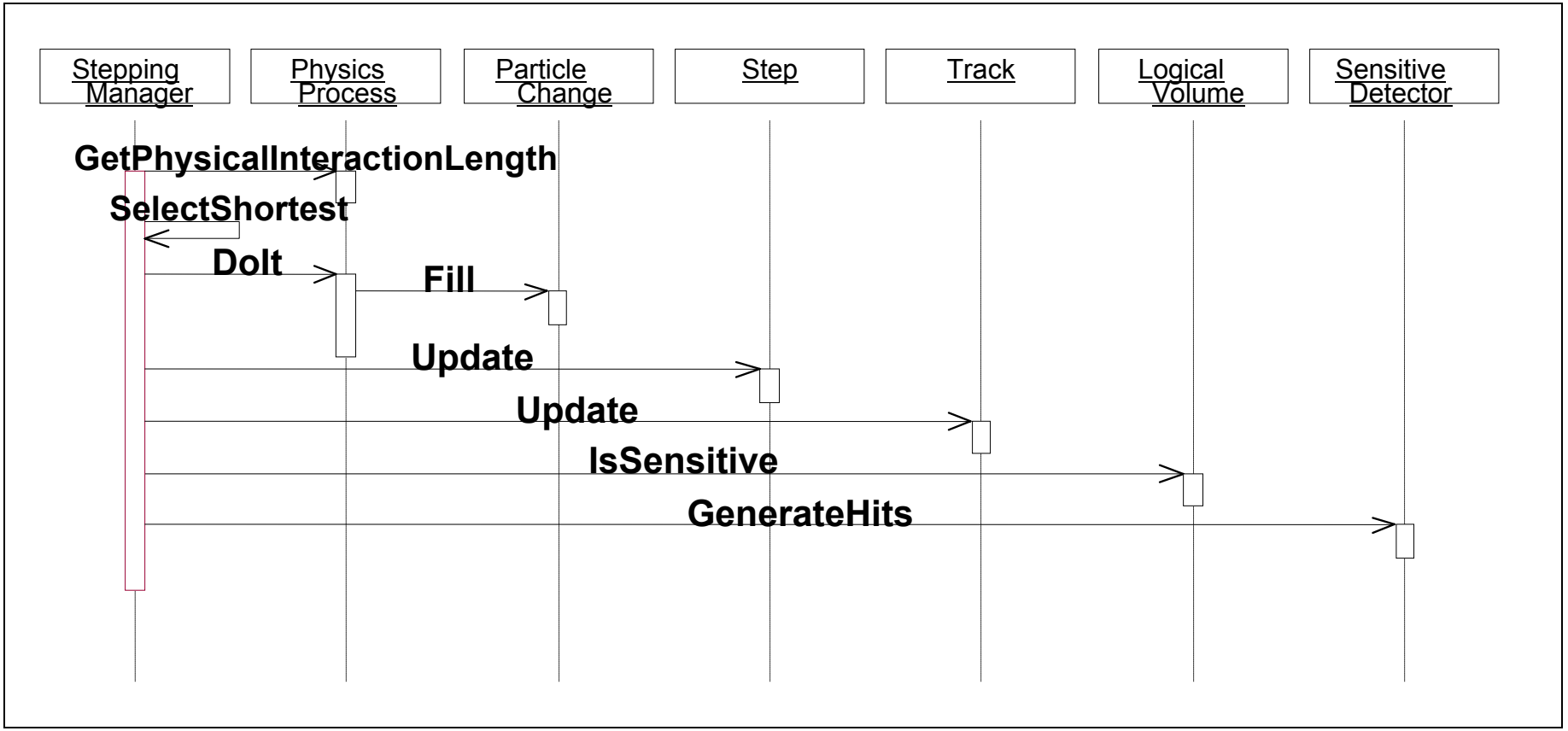
Detector Response



- This is done through sensitive detector which creates hit(s) using the information given in G4Step object. The user has to provide his/her own implementation of the detector response

Sensitive detector

- A G4VSensitiveDetector object can be assigned to a G4LogicalVolume
- In case a step takes place in a logical volume that has a G4VSensitiveDetector object, this G4VSensitiveDetector is invoked with the current G4Step object



Defining a sensitive detector

- The basic strategy

```
G4LogicalVolume* myLogCalor = ……;
G4VSensetiveDetector* pSensetivePart = new
  MyDetector("/mydet");
G4SDManager* SDMan = G4SDManager::GetSDMpointer();
SDMan->AddNewDetector(pSensitivePart);
myLogCalor->SetSensitiveDetector(pSensetivePart);
```

- Each detector object must have a unique name
  - Some logical volumes can share one detector object
  - More than one detector objects can be made from one detector class with different detector name
  - One logical volume cannot have more than one detector objects. But, one detector object can generate more than one kinds of hits
    - e.g. a double-sided silicon micro-strip detector can generate hits for each side separately

Hits collection, hits map

- Hit is a snapshot of the physical interaction of a track or an accumulation of interactions of tracks in the sensitive region of your detector
- G4VHitsCollection is the common abstract base class of both G4THitsCollection and G4THitsMap
- G4THitsCollection is a template vector class to store pointers of objects of one concrete hit class type
  - A hit class (deliverable of G4VHit abstract base class) should have its own identifier (e.g. cell ID)
  - G4THitsCollection requires the user to implement own hit class
- G4THitsMap is a template map class so that it stores keys (typically cell ID, i.e. copy number of the volume) with pointers of objects of one type
  - Objects may not be those of hit class
    - All of currently provided scorer classes use G4THitsMap with simple double
  - Since G4THitsMap is a template, it can be used by the sensitive detector class to store hits
- Hit objects are collected in a G4Event object at the end of an event

Hit Class

- Hit is a user-defined class derived from G4VHit
- The user can store various types information by implementing one's own concrete Hit class. For example:
  - Position and time of the step
  - Momentum and energy of the track
  - Energy deposition of the step
  - Geometrical information
  - or any combination of above
- Hit objects of a concrete hit class must be stored in a dedicated collection which is instantiated from G4THitsCollection template class
- The collection is associated to a G4Event object via G4HCofThisEvent
- Hits are accessible as collections:
  - through G4Event at the end of event
    - ❖to be used for analyzing an event
  - through G4SDManager during processing an event
    - ❖to be used for event filtering

Implementation of Hit class

```
#include "G4VHit.hh"
class MyHit : public G4VHit
{
  public:
     MyHit(some_arguments);
     virtual ~MyHit();
     virtual void Draw();
     virtual void Print();
  private:
     // some data members
  public:
     // some set/get methods
};

#include “G4THitsCollection.hh”
typedef G4THitsCollection<MyHit> MyHitsCollection;
```

Sensitive Detector class

- Sensitive detector is a user-defined class derived from G4VSensitiveDetector

```
#include "G4VSensitiveDetector.hh"
#include "MyHit.hh"
class G4Step;
class G4HCofThisEvent;
class MyDetector : public G4VSensitiveDetector
{
  public:
     MyDetector(G4String name);
     virtual ~MyDetector();
     virtual void Initialize(G4HCofThisEvent*HCE);
     virtual G4bool ProcessHits(G4Step*aStep,
                          G4TouchableHistory*ROhist);
     virtual void EndOfEvent(G4HCofThisEvent*HCE);
  private:
     MyHitsCollection * hitsCollection;
     G4int collectionID;
};
```

Sensitive Detector Types

- A tracker detector typically generates a hit for every single step of every single (charged) track
  - A tracker hit typically contains
    - Position and time
    - Energy deposition of the step
    - Track ID
- A calorimeter detector typically generates a hit for every cell, and accumulates energy deposition in each cell for all steps of all tracks
  - A calorimeter hit typically contains
    - Sum of deposited energy
    - Cell ID
- The user can instantiate more than one objects for one sensitive detector class. Each object should have its unique detector name
  - For example, each of two sets of detectors can have their dedicated sensitive detector objects. But, the functionalities of them are exactly the same to each other so that they can share the same class. See examples/extended/analysis/A01 as an example

Implementation of Sensitive Detector - 1

```
MyDetector::MyDetector(G4String detector_name)
            :G4VSensitiveDetector(detector_name),
             collectionID(-1)
{
  collectionName.insert("collection_name");
}
```

- In the constructor, the name of the hits collection which is handled by this sensitive detector is to be defined
- In case the sensitive detector generates more than one kinds of hits (e.g. anode and cathode hits separately), all collection names need to be defined

Implementation of Sensitive Detector - 2

```
void MyDetector::Initialize(G4HCofThisEvent*HCE)
{
  if(collectionID<0) collectionID = GetCollectionID(0);
  hitsCollection = new MyHitsCollection
       (SensitiveDetectorName,collectionName[0]);
  HCE->AddHitsCollection(collectionID,hitsCollection);
}
```

- Initialize() method is invoked at the beginning of each event.
- Get the unique ID number for this collection
  - GetCollectionID() is a heavy operation. It should not be used for every event
  - GetCollectionID() is available after this sensitive detector object is constructed and registered to G4SDManager. Thus, this method cannot be invoked in the constructor of this detector class
- The hits collection(s) are to be instantiated and then attached to the G4HCofThisEvent object given in the argument
- In case of calorimeter-type detector, hits for all calorimeter cells may be instantiated with zero energy depositions, and then inserted to the collection

Implementation of Sensitive Detector - 3

```
G4bool MyDetector::ProcessHits
  (G4Step*aStep,G4TouchableHistory*ROhist)
{
  MyHit* aHit = new MyHit();
  ...
  // some set methods
  ...
  hitsCollection->insert(aHit);
  return true;
}
```

- This ProcessHits() method is invoked for every steps in the volume(s) where this sensitive detector is assigned
- In this method, generate a hit corresponding to the current step (for tracking detector), or accumulate the energy deposition of the current step to the existing hit object where the current step belongs to (for calorimeter detector)
- geometry information is to collected (e.g. copy number) from "PreStepPoint"
- Currently, returning boolean value is not used.

Implementation of Sensitive Detector - 4

```
void MyDetector::EndOfEvent(G4HCofThisEvent*HCE)  {;}
```

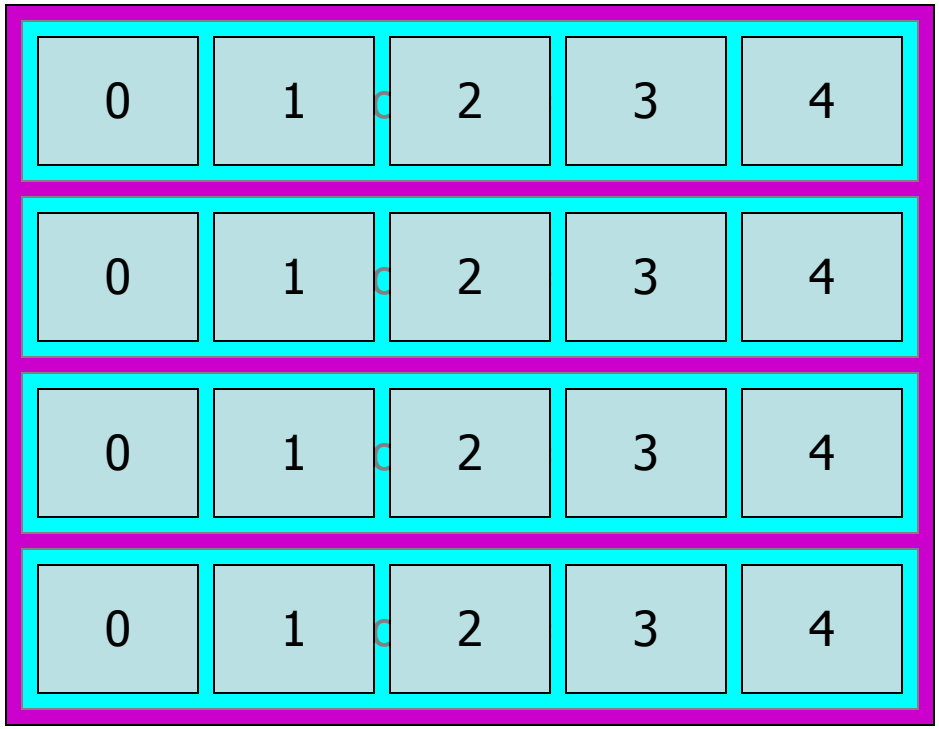
- This method is invoked at the end of processing an event.
  - It is invoked even if the event is aborted.
  - It is invoked before UserEndOfEventAction.

Step point and touchable

- As mentioned already, G4Step has two G4StepPoint objects as its starting and ending points. All the geometrical information of the particular step should be taken from "PreStepPoint"
  - Geometrical information associated with G4Track is identical to "PostStepPoint"
- Each G4StepPoint object has
  - Position in world coordinate system
  - Global and local time
  - Material
  - G4TouchableHistory for geometrical information
- G4TouchableHistory object is a vector of information for each geometrical hierarchy
  - copy number
  - transformation / rotation to its mother
- Since release 4.0, *handles* (or *smart-pointers*) to touchables are intrinsically used. Touchables are reference counted

Copy number

- Suppose a calorimeter is made of 4x5 cells
  - and it is implemented by two levels of replica
- In reality, there is only one physical volume object for each level. Its position is parameterized by its copy number
- To get the copy number of each level, suppose what happens if a step belongs to two cells



  - geometrical information in G4Track is identical to that in "PostStepPoint"
  - User cannot get the correct copy number for "PreStepPoint" if one directly accesses to the physical volume
- touchable is to be used to get the proper copy number, transform matrix, etc.

Touchable

- G4TouchableHistory has information of geometrical hierarchy of the point.

```
G4Step* aStep;
G4StepPoint* preStepPoint = aStep->GetPreStepPoint();
G4TouchableHistory* theTouchable =
   (G4TouchableHistory*)(preStepPoint->GetTouchable());
G4int copyNo = theTouchable->GetVolume()->GetCopyNo();
G4int motherCopyNo
        = theTouchable->GetVolume(1)->GetCopyNo();
G4int grandMotherCopyNo
        = theTouchable->GetVolume(2)->GetCopyNo();
G4ThreeVector worldPos = preStepPoint->GetPosition();
G4ThreeVector localPos = theTouchable->GetHistory()
   ->GetTopTransform().TransformPoint(worldPos);
```

G4HCofThisEvent

- A G4Event object has a G4HCofThisEvent object at the end of (successful) event processing. G4HCofThisEvent object stores all hits collections made within the event.
  - Pointer(s) to the collections may be NULL if collections are not created in the particular event
  - Hits collections are stored by pointers of G4VHitsCollection base class. Thus, one has to cast them to types of individual concrete classes
  - The index number of a Hits collection is unique and unchanged for a run. The index number can be obtained by

```
G4SDManager::GetCollectionID("detName/colName");
```

    - The index table is also stored in G4Run

Usage of G4HCofThisEvent

```
void MyEventAction::EndOfEventAction(const G4Event* evt)  {
  static int CHCID = -1;
  If(CHCID<0) CHCID = G4SDManager::GetSDMpointer()
     ->GetCollectionID("myDet/collection1");
  G4HCofThisEvent* HCE = evt->GetHCofThisEvent();
  MyHitsCollection* CHC = 0;
  if (HCE) {
     CHC = (MyHitsCollection*)(HCE->GetHC(CHCID)); }
  if (CHC) {
    int n_hit = CHC->entries();
    G4cout<<“My detector has ”<<n_hit<<" hits."<<G4endl;
    for (int i1=0;i1<n_hit;i1++) {
      MyHit* aHit = (*CHC)[i1];
      aHit->Print();
    }
  }
}
```

When to invoke GetCollectionID()?

- Which is the better place to invoke G4SDManager::GetCollectionID() in a user event action class, in its constructor or in the BeginOfEventAction()?
- It actually depends on the user's application
  - Note that construction of sensitive detectors (and thus registration of their hits collections to SDManager) takes place when the user issues RunManager::Initialize(), and thus the user's geometry is constructed.
- In case user's EventAction class should be instantiated before Runmanager::Initialize() (or /run/initialize command), GetCollectionID() should not be in the constructor of EventAction.
- While, if the user has nothing to do to Geant4 before RunManager::Initialize(), this initialize method can be hard-coded in the main() before the instantiation of EventAction (e.g. exampleA01), so that GetCollectionID() could be in the constructor

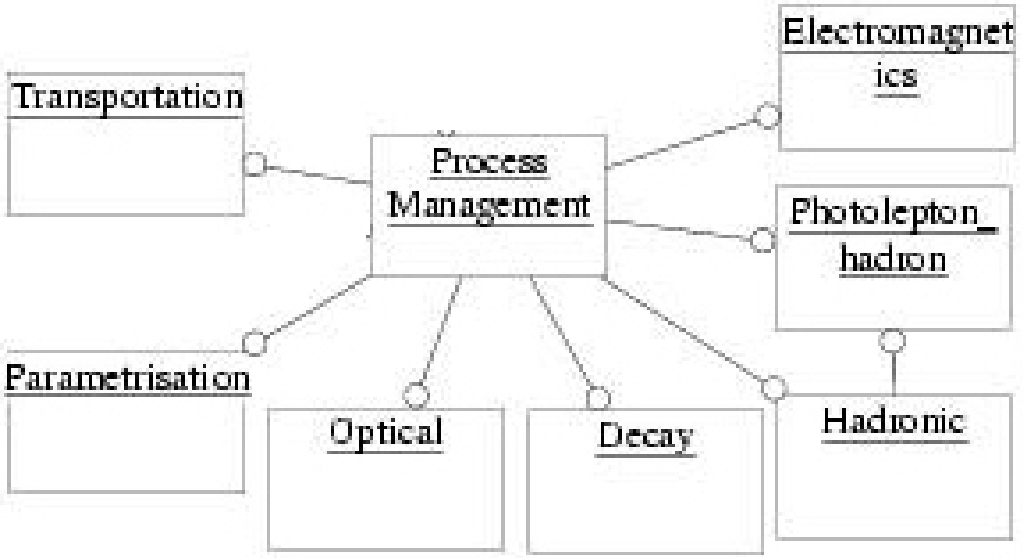
Physics in Geant4

- From the Minutes of LCB (LHCC Computing Board) meeting on 21/10/1997:

"It was noted that experiments have requirements for **independent, alternative physics models**. In Geant4 these models, differently from the concept of packages, allow the user to **understand** how the results are produced, and hence improve the **physics validation**. Geant4 is developed with a modular architecture and is the ideal framework where existing components are integrated and new models continue to be developed."

Processes in Geant4

- Processes describe how particles interact with material or with a volume
- Three basic types
  - **At rest** process
    (eg. decay at rest)
  - **Continuous** process
    (eg. ionisation)
  - **Discrete** process
    (eg. Compton scattering)



- Transportation is a process
  - interacting with volume boundary
- A process which requires the shortest interaction length limits the step

Electromagnetic Physics

- Applicable to
  - electrons and positrons
  - γ, X-ray and optical photons
  - muons
  - charged hadrons
  - Ions
- Several physics models are available. Standard EM physics is extended at low energies using many data driven techniques to improve the quality of simulation at low energies
- *All obeying to the same abstract Process interface*: transparent to tracking

- Models available for
  - Multiple scattering
  - Bremsstrahlung
  - Ionization
  - Annihilation
  - Photoelectric effect
  - Compton scattering
  - Pair production
  - Rayleigh scattering
  - γ conversion
  - Synchrotron radiation
  - Transition radiation
  - Reflection, refraction
  - Cherenkov radiation
  - Scintillation
  - …….

Models for Hadronic Interactions

- Data driven models: When sufficient data are available with sufficient coverage, data driven approach is optimal way
  - neutron transport, photon evaporation, absorption at rest, isotope production, inclusive cross section, ….
- Parameterized models: Extrapolation of cross sections and parameterizations of multiplicities and final state kinematics
  - adaptation of GHEISHA and now a newer version on the way
- Theory based models: Includes a set of different theoretical models describing hadronic interactions depending on the addressed energy range
  - diffractive string excitation, dual parton model or quark gluon string model at medium to high energies
  - intra-nuclear cascade models at medium to low energies
  - nuclear evaporation, fission models,`… at very low energies

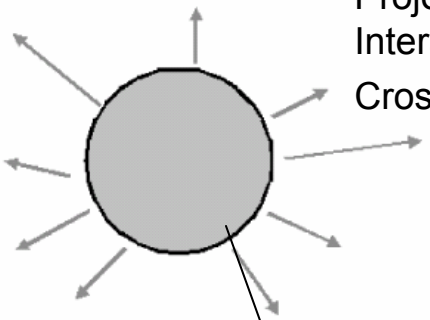
Hadronic Models

- There are a large number of such models to be validated by data
  - **Precompound:** Takes care of the nucleon-nucleus collision and nuclear de-excitation and valid for energies below 100 MeV
  - **LEP:** Low energy parameterized model derived from GHEISHA and is intended for incident energies below 25 GeV
  - **Binary Cascade:** Data driven intra-nuclear cascade model intended for incident energy between 100 MeV and 5 GeV
  - **Bertini Cascade:** Bertini intra-nuclear cascade model intended for incident energy between 100 MeV and 9 GeV
  - **CHIPS:** Quark level event generator based on Chiral Invariant phase space model above a few hundred MeV
  - **QGS:** Quark gluon string model and is intended for incident energy above 12 GeV
  - **FTF:** Fritiof model implementation intended for incident energy above 4 GeV
  - **HEP:** High energy parametrized model derived from GHEISHA and is intended for incident energies above 25 GeV
  - **G4QMD:** Ion-ion collision to overcome limitation of light ion binary cascade model

Introduction to Physics Lists

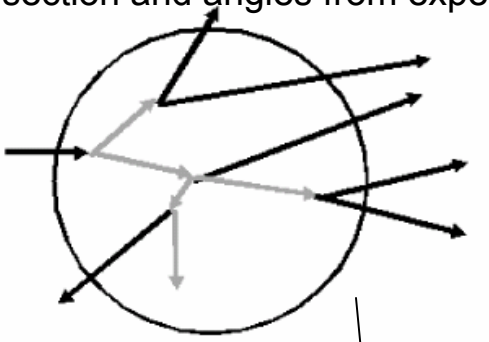
4

Nuclear deexcitation
Evaporation etc.

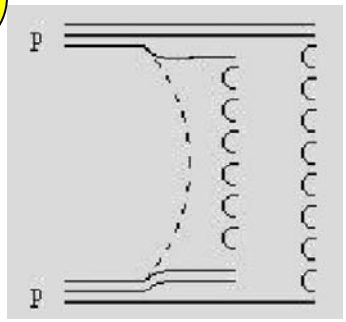


3

Bertini nucleon-nucelon cascade
step-like concentric nuclear potential in 3d
Projectile transported along straight-lines
Interaction according to free mean path
Cross-section and angles from experiment

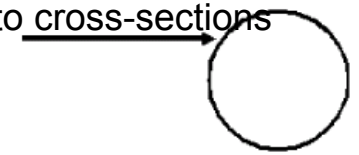


2



1

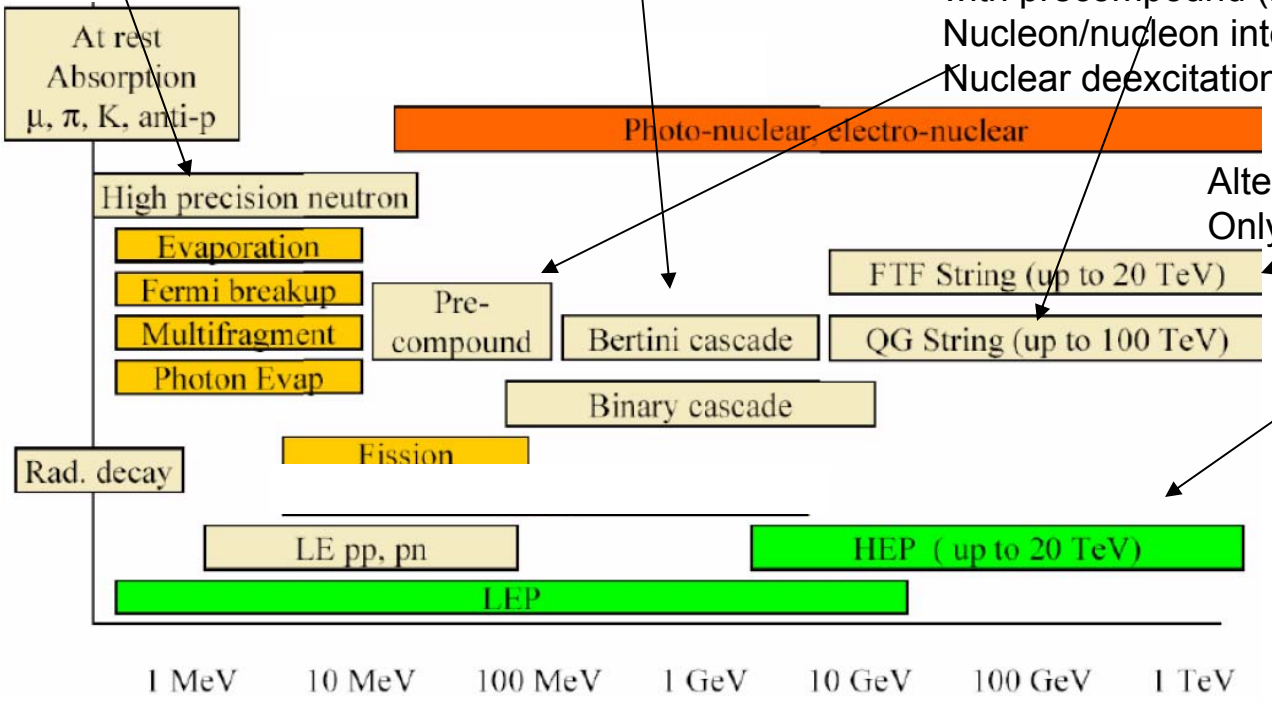
Particle nucleus
collision according
to cross-sections



QGS: Quark-Gluon String

Nucleon is split in quark di-quark
Strings are formed
String hadronisation (adding qqbar pair)
fragmentation of damaged nucleus
with precompound (P)
Nucleon/nucleon interaction+
Nuclear deexcitation

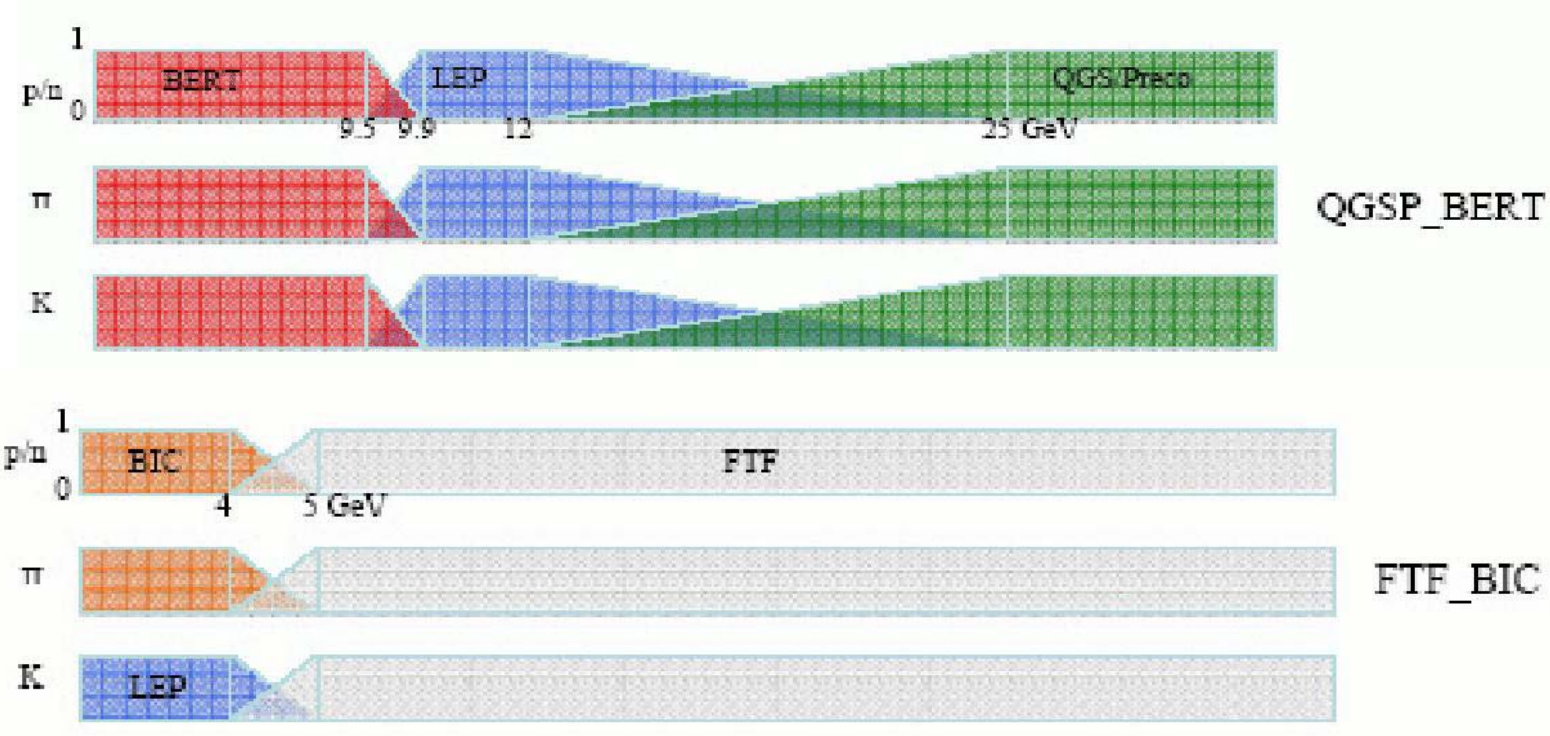
Fritiof:
Alternative string frag.
Only momentum exchanged



Parameterised
models
(as in old Gheisha)

Physics List

- Since none of the models could explain all physics processes, it is customary to register several physics processes in a list.
  - EM processes are usually valid over the entire energy domain but each discrete process is described separately, e.g., pair production, Compton scattering, …
  - Hadronic processes are valid over a finite energy domain. Two models may have validity over an overlapping energy region



Pre-processing

- What is recorded as a detector signal?
  - ADC or TDC information: digitized quantity corresponding to some integrated charge or timing
  - At best one can record some time profile of charge accumulation from this raw information
- Translate these information into primary measurements
  - Need calibration constants (as in any measuring device) to set correctly the scale of the measured quantities
- These primary measurements may have to be transformed into some physical quantities which are directly related to passage of particles through matter (e.g. measure drift time, relate to position of the ionization centres) → require another set of constants
- These constants apply to
  - Millions of read out channels
  - Different components of the detectors

Calibration

- These numbers are not strictly constants
  - Depend on atmospheric condition (temperature, pressure, …)
  - Depend on other environmental variables (gas composition, voltages, movement of the support structures, …)
- Keep on calibrating the detector
  - Test beam runs before integrating all the sub-detectors
  - Employ dedicated calibration runs parallel to data taking runs
  - Use collision data themselves
- In addition one needs to know
  - Current status of the detector (if all the channels are active and efficient)
  - Relative position of the detector elements
- Keep on monitoring detectors during data taking. Also have dedicated measurement systems to measure relative detector elements

Data Processing

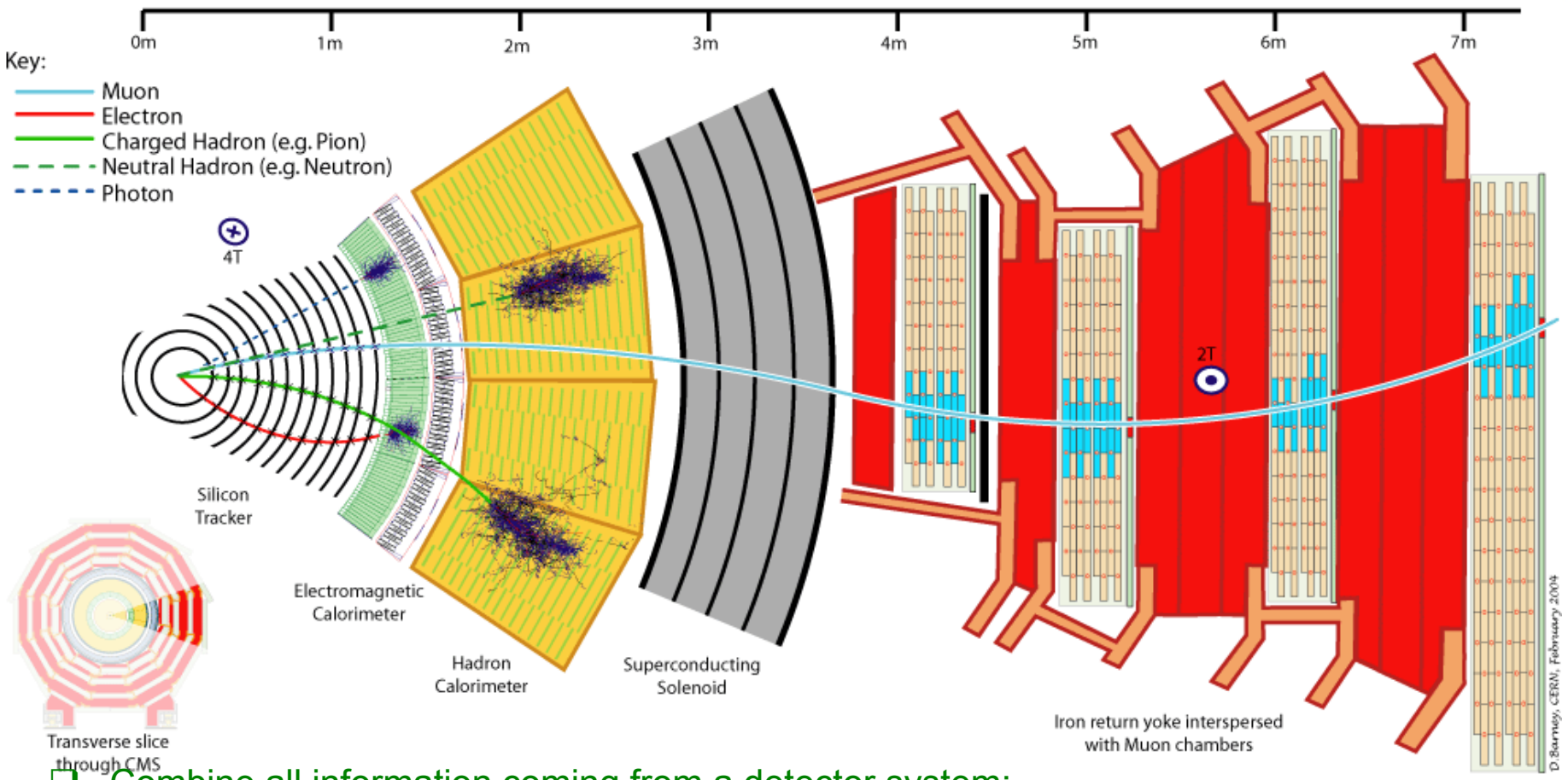
- This gives rise to several data streams
  - Data collected during bunch crossing (possibly due to interactions) Synchronous data
  - Data from all calibration runs, monitoring tasks, alignment devices, …….. Asynchronous data
- Using data themselves for calibration purpose causes maximum constraint on data processing – cannot have a single pass reconstruction of the collected data
- Standard production schedule
  - Pass 1 in pseudo real time (within few hours from data taking)
  - Several re-reconstruction passes subsequently in weeks-months time scale

Design data structure with these constraints in mind.

Stages of Reconstruction

- Usually the tracking detectors consist of
  - Several layers of position finding devices
  - Often multiple technologies to find a most optimum situation
- Magnetic fields are often non-uniform and there are enough material in the tracking detectors
  - There are loopers in the tracking devices
  - Trajectories deviate from helical structure
- Even with ideal calibration and with perfect ambiguity resolution, the preprocessing will give rise to hit patterns and they need
  - To be associated to candidate trajectories of charged particles (Pattern recognition)
  - To be tested for goodness of association and for extraction of kinematic parameters (Fitting)
- Do this reconstruction at 2 levels
  - Locally in the context of a given detector system
  - Globally by combining information across the various detectors

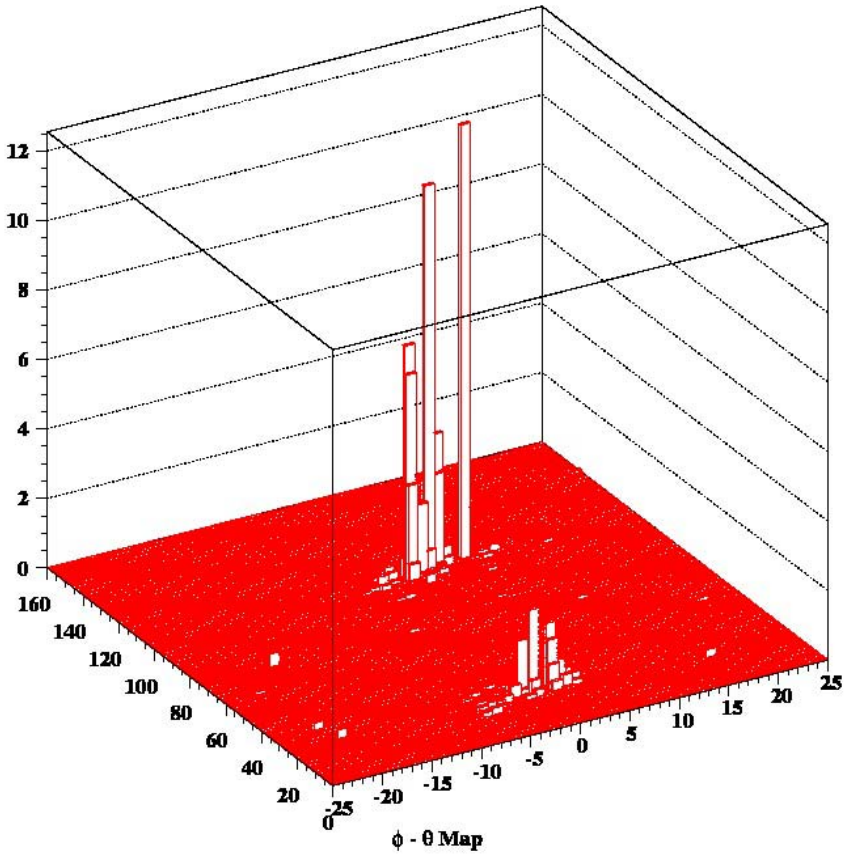
Particle Identification



- Combine all information coming from a detector system:
  - Tracker detects charged particles
  - EM Calorimeter measures energy of e/γ
  - Hadron (+EM) Calorimeter measures energy of jets and missing energy
  - Muon detectors tags muon and complement its energy measurement

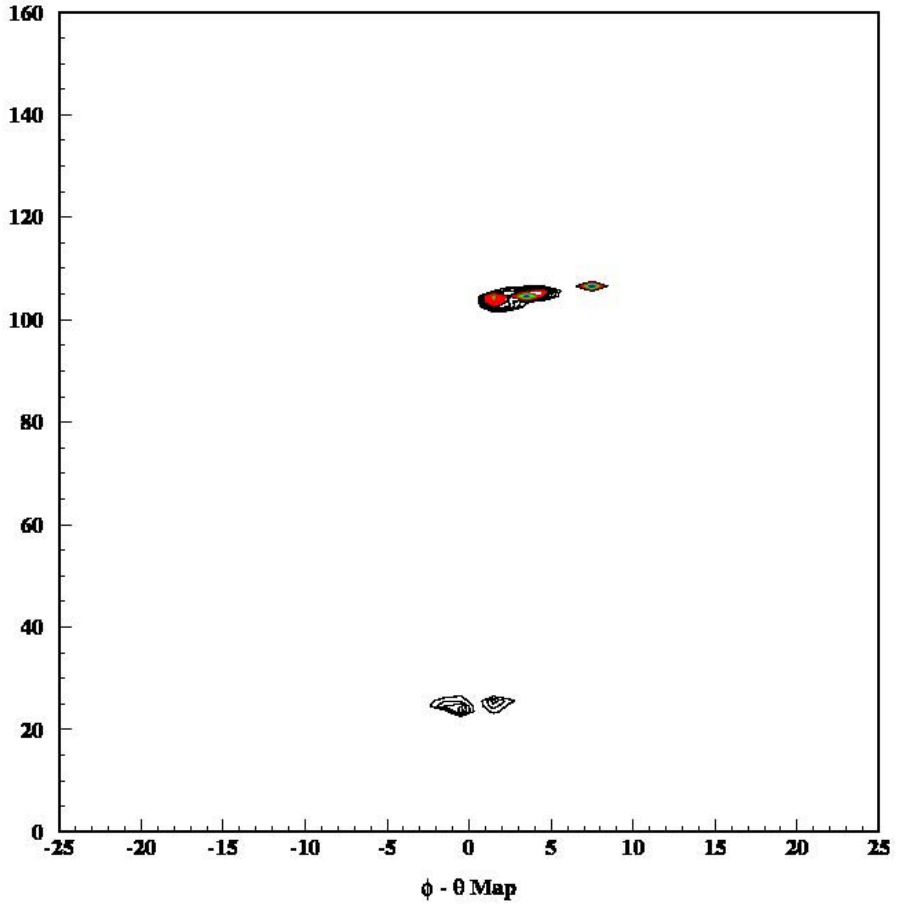
Pattern Recognition in Calorimeters

- Particles lose their energies through a cascade of interactions
- Shower is formed due to each particle with finite lateral and longitudinal size
- If the calorimeter has sufficient granularity each particle will deposit its energy to a number of cells
- Calorimeters are usually designed such that granularity does not exceed the limiting size determined by shower fluctuation
  - Hits belonging to a single particle would be contiguous in space
- This is the basic assumption behind pattern recognition algorithm



Cluster Algorithm

- Algorithm:
  - Sort all hits according to energy deposits
  - Choose the hottest cell as the root hit
  - Look at neighbours (in space) with energy deposits above threshold
    - Find their neighbours
    - Iterate
  - The connected hits (called clusters) to be removed from the starting list of hits and repeat from the next root
  - Look for local maxima inside each cluster



Localization

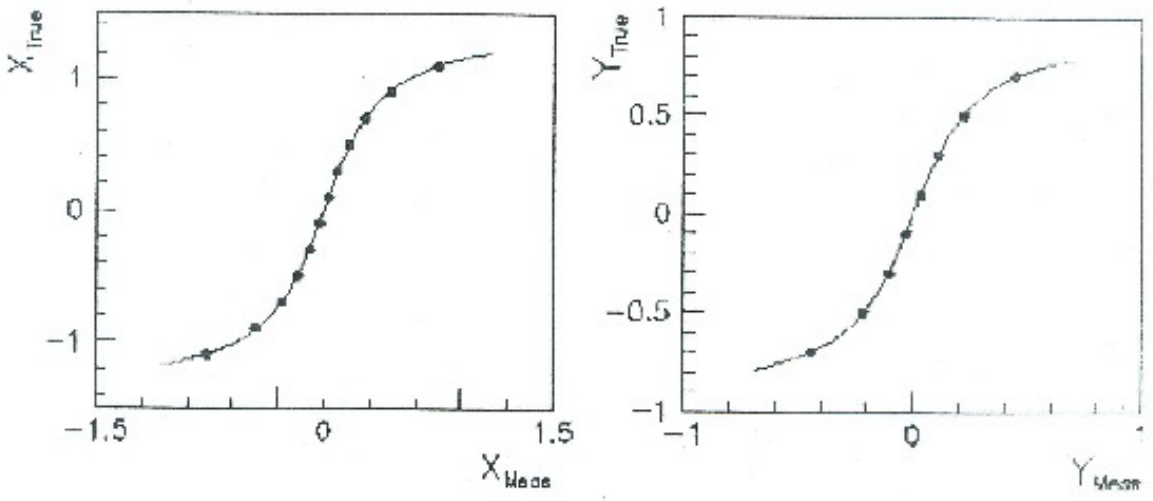
- Localize cluster centres using the centre of gravity method

$$x_{COG} = \frac{\sum w_i \cdot x_i}{\sum w_i}$$

- Correct impact point as obtained from x_{CoG} through some suitable function

$$x_{True} = \alpha_1 \tan^{-1}\left[\alpha_2 \cdot (x_{meas} - x_0)\right] + \alpha_3 \cdot (x_{meas} - x_0)$$

- For crystals with 2x2 cm^2 $PbWO_4$,
  - if one uses $x(y)_{Meas}$ the resolution is several mm
  - If one uses corrected $x(y)_{True}$ the resolution is ~0.3 mm
- Resolution improves slightly with energy
- If one uses logarithmic weighting, the mapping between $x(y)_{Meas}$ and $x(y)_{True}$ is almost linear



Clusters to Jets

- Clusters found from spatial relationship in calorimetric cells are due to showers of ≥1 stable particle(s)
- In hadronic final state, stable hadrons are due to fragmentation/ hadronization of hard partons → Jets (closely related set of hadrons, in this context clusters)
- Possible algorithm
  - start with momenta $\vec{p_j}$ of observed particles (clusters)
  - for each pair of particles i, j (i≠j) define a quantity ρ_{ij} as the likelihood of i and j being in the same jet
  - In general $\rho_{ij} = \rho_{ji}$ and smaller value of ρ_{ij} implies that it is more likely for I and j to come from the same jet
  - join particles according to ρ_{ij} and stop reclustering by some other criterion
- How to choose ρ_{ij}? – no unique answer
  - Two possibilities and many variations in each of these types which lead to a large number of jet algorithms

Jet Algorithms

- Choice for ρ_{ij}
  - ρ_{ij} should only give relative ordering of likelihood. If $\rho' = f(\rho)$ is a strictly increasing function of $\rho \rightarrow$ use of ρ' instead of ρ would give the same answer
  - Values of ρ_{ii} are themselves meaningful. In language of pattern recognition $\vec{p_j}$'s are embedded in the so called feature space and ρ_{ij} can be identified with the metric in the space
- In addition to the choice of ρ_{ij}, steps to build the jets (recombination scheme, cutoff, …) could be different from algorithm to algorithm $\rightarrow$ need several parameters which need to be tuned to a given application

Hierarchical Jet Algorithm

- Similarity of two objects is defined by a numerical value ρ_{ij} with $0 \le \rho_{ij} \le 1$

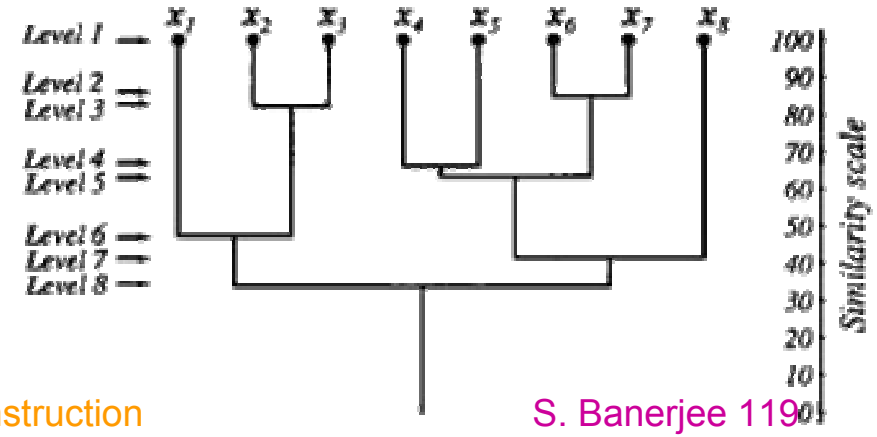
  $\rho_{ij} = 1$ for identical objects

  $= 0$ for extremely different objects

- Choose $\rho_{ij} = (1+\cos\theta_{ij})/2$ where θ_{ij} is the angle between i an j in the CM system
  - start from a system of N particles → N groups with each group consisting of 1 particle each
  - ρ_{ij} is computed for N(N+1)/2 combinations. Combine the group with largest value of ρ_{ij}
  - if i and j are combined to a single group (called m), the corresponding entries are removed from the similarity matrix and new group m is added with

    $\rho_{km} = \min(\rho_{ik},\rho_{jk})$ with $k \neq i$, $k \neq j$
  - Complete the tree

From the hierarchy one has to cut at some value of Δρ to classify the event as an n-jet event



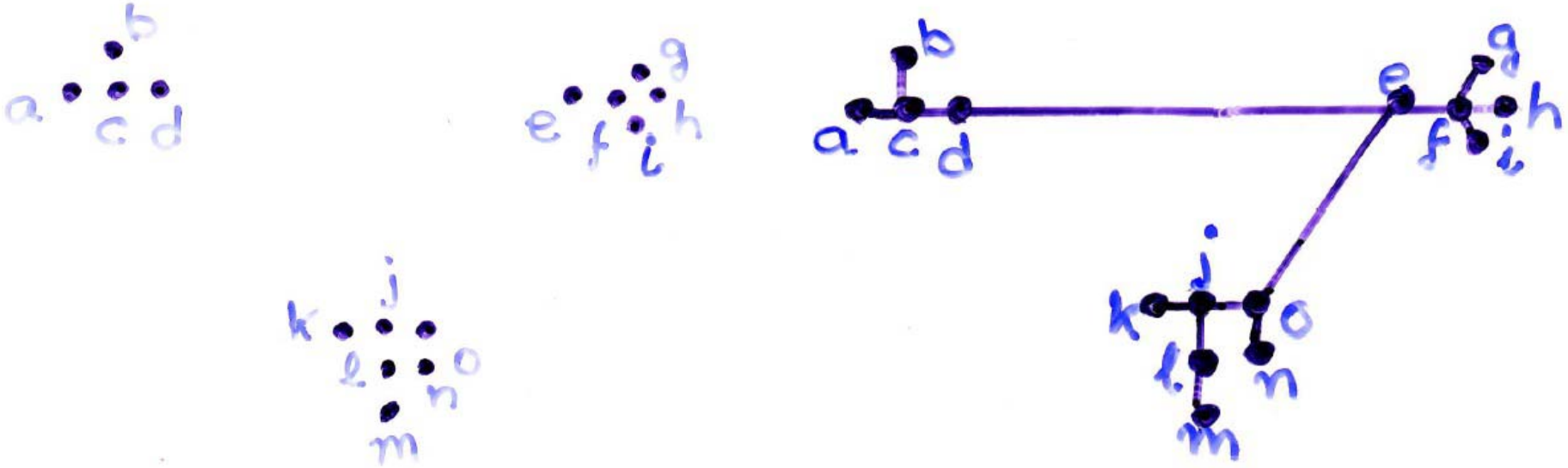
Hierarchical Jet Algorithm (II)

- In one such algorithm, define a quantity M-tricity (T_M)

$$T_M = \sum_{g=1}^{M} \frac{|\sum_{i=1}^{m_g} p_i^g|}{\sum_{i=1}^{N} |p_i|}$$

- For correct jet multiplicity $T_M \rightarrow 1$ and TM is a monotonically increasing function
- So the classification is done by choosing
  - $T_M > T_{cut}$
  - $D_M \equiv T_M - T_{M-1} > D_{cut}$

Minimum Spanning Tree



- Wish to connect a set of points in space → join elements such that
  - There is a connected path between any 2 points
  - Total length of the connecting elements is a minimum
- Terminologies:
  - Node: coordinate points spanned by the tree
  - Edge: lines connecting the nodes
  - Bridging Edge: if the removal of the edge make two sub-trees
  - Non-bridging Edge: if the removal isolates a node

Minimum Spanning Tree (II)

Define the distance metric

$$\rho_{ij} = \theta_{ij}^2 w_{ij}(p_i, p_j)$$

where θ_{ij} = angle between the two particle directions
w_{ij}= weighting matrix ~ $w_i(p_i) \cdot w_j(p_j)$ [$w(p) \sim \frac{1}{p}$ ok]

- Procedure:
  - Join all nodes by edges
  - Compare length of bridging edges to the length of a typical edge
  - If $\rho_{ij} > R \cdot \rho_{median}$ (R ≈ 2) → bridging edge is inconsistent
  - Break the longest inconsistent edge
  - Repeat similar treatment to all sub-trees

Two Step Algorithm

- Find pre-clusters from the primary source
- Find clusters using pre-clusters

→ two resolution parameters and may be 2 ordering variables

PLUTO/CELLO approach:

- Order the particles by energy an use particle directions $\hat{n}_j = \frac{\vec{p_j}}{|\vec{p_j}|}$
- One particle is a member of only one pre-cluster. Two particles will belong to the same pre-cluster if

$$\rho_{ij} \equiv \hat{n_i} \cdot \hat{n_j} > \cos\alpha$$

α predefined and set to ≈ 30°

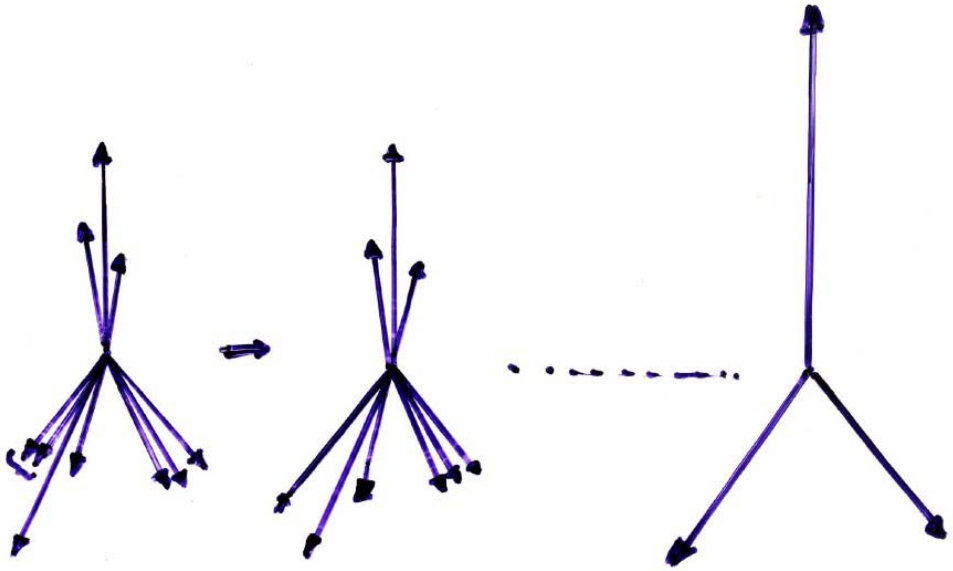
- Direction of the pre-cluster is determined $\hat{n_{D_i}} = \frac{\sum_{k\in D_i} \vec{p_k}}{|\sum_{k\in D_i} \vec{p_k}|}$
- Consider all pre-clusters. One pre-cluster is a member of only one cluster. Any two pre-clusters Di, Dj will belong to the same cluster if

$$\hat{n_{D_i}} \cdot \hat{n_{D_j}} > \cos\beta$$ β can be set to ≈ 45°

- Compute energy, direction of the cluster (≡ Jet)

$$E_{C_i} = \sum_{k\in C_i} E_k \qquad \hat{n_{C_i}} = \frac{\sum_{k\in C_i} \vec{p_k}}{|\sum_{k\in C_i} \vec{p_k}|}$$

Successive Recombination Scheme



- Start with N clusters each containing one particle (or calo-cluster)
- Compute ρ_{ij} for all combinations
- Find i,j for which ρ_{ij} is the smallest
- Regroup $i,j \rightarrow l$; remove i,j from the list of clusters; insert l with its 4-momentum computed from those of i,j; re-compute ρ_{ij}
- Repeat this process till all ρ_{ij}'s exceed a cut-off (jet resolution parameter
- Ordering variable ≡ test variable for freezing jet formation (need not be the same)

Jets in e^+e^- Interactions

- Ambiguity lies in
  - Definition of ρ
  - Recombination of p_i, p_j to obtain p_{ln}

Name	ρ_{ij}	Recombination Scheme
Jade	$\frac{2E_iE_j}{E_{vis}^2}(1-\cos\theta_{ij})$	$p_l = p_i + p_j$
k_T	$\frac{2}{E_{vis}^2}\min E_i^2, E_j^2(1-\cos\theta_{ij})$	$p_l = p_i + p_j$
E	$\frac{(p_i+p_j)^2}{s}$	$p_l = p_i + p_j$
p	$\frac{(p_i+p_j)^2}{s}$	$\vec{p_l} = \vec{p_i} + \vec{p_j};\ E_l = \lvert\vec{p_l}\rvert$
Geneva	$\frac{8E_iE_j}{9(E_i+E_j)^2}(1-\cos\theta_{ij})$	$p_l = p_i + p_j$
E_0	$\frac{(p_i+p_j)^2}{s}$	$E_l = E_i + E_j;\ \vec{p_l} = \frac{E_l}{\lvert\vec{p_i}+\vec{p_j}\rvert}(\vec{p_i}+\vec{p_j})$

Jets in hadron colliders

- Traditionally hadron machines used to have different flavours of cone algorithms. But more recently (LHC era), similar successive recombination schemes are used.
- Here in addition to distance between two particles i,j (ρ_{ij}), distance from the beam (ρ_{iB}) is computed and the minimum is determined among ρ_{ij} 's and ρ_{iB} 's. If distance between particles is found to be minimum, they are combined. If ρ_{iB} is found to be minimum, i is declared as a jet and taken out from the list.

Name	ρ_{ij}	ρ_{iB}	Recombination Scheme
k_T	$\min(k_{Ti}^2, k_{Tj}^2)\frac{\Delta R_{ij}^2}{R^2}$	k_{Ti}^2	$p_l = p_i + p_j$
Aachen-Cambridge	$\frac{\Delta R_{ij}^2}{R^2}$	1	$p_l = p_i + p_j$
Anti-k_T	$\min(k_{Ti}^{-2}, k_{Tj}^{-2})\frac{\Delta R_{ij}^2}{R^2}$	k_{Ti}^{-2}	$p_l = p_i + p_j$

where $\Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$

and R is a jet radius parameter and ≈ 1 in many applications

Maximum Likelihood Method

- Problem: try to obtain the best estimate of a parameter which is a continuous variable
- For discrete variables likelihood ratio method is used → probability of any 2 different values of a parameter is the ratio of probabilities of getting experimental results assuming the two different values of the parameter
- Use the same principle for continuous variable:

$f(x, m)$ = truly normalized distribution function

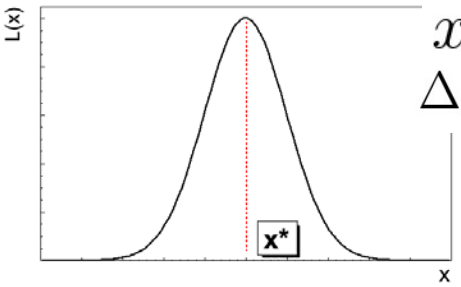
$$\int f(x, m)\, dm = 1$$

$m \equiv$ measurement; $x \equiv$ parameter

Likelihood function: $\mathcal{L}(x) = \prod_{i=1}^{N} f(x, m_i)$

Joint probability density function of getting a particular experimental results m_1, …. m_N

Relative probability of x can be obtained from $\mathcal{L}(x)$ vs x



$x^\star$ = most probable value of x (Maximum Likelihood solution)

Δx = RMS spread of x about $x^\star$

$$= \left[\frac{\int (x-x^\star)^2 \mathcal{L}(x)dx}{\int \mathcal{L}(x)dx}\right]^{\frac{1}{2}}$$

Likelihood Method

For $N \to \infty$, $x^\star$ approaches true value (x_0) of x

For multiple parameters $x_1, \cdots x_\ell$ determine $\mathcal{L}(x_1, \cdots x_\ell)$ and solve ℓ simultaneous equations

$$\frac{\partial W}{\partial x_i}\Big|_{x_i = x_i^\star} = 0$$

where $W = \ln \mathcal{L}(x_1, \cdots x_\ell)$

For measurements with Gaussian errors:

$$\mathcal{L}(x) = \prod \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{\{m_i - \bar{m}(x)\}^2}{2\sigma_i^2}\right]$$

$$W = -\frac{1}{2}\sum_{i=1}^{N} \frac{\{m_i - \bar{m}(x)\}^2}{2\sigma_i^2} - \sum_{i=1}^{N} \ln(\sqrt{2\pi}\sigma_i)$$

The solution: minimize χ^2 with respect to x

Error on Parameter

For large number of measurements N, $\mathcal{L}(x)$ approaches Gaussian distribution

$$\mathcal{L}(x) \sim \exp\left[-\frac{h}{2}(x - x^\star)^2\right]$$

with $\frac{1}{\sqrt{h}}$ as RMS spread of x about $x^\star$

Likelihood Method

$$W = -\frac{h}{2}(x - x^\star)^2 + constant$$

$$\frac{\partial W}{\partial x} = -h(x - x^\star) \qquad \frac{\partial^2 W}{\partial x^2} = -h$$

$$\Delta x = \frac{1}{\sqrt{h}} = \left(-\frac{\partial^2 W}{\partial x^2}\right)^{-\frac{1}{2}}$$

If $f(x, m)$ follows Gaussian distribution

$$\frac{\partial W}{\partial x} = \sum \frac{m_i - x}{\sigma_i^2}$$

$$\Rightarrow \Delta x = \left[\sum \frac{1}{\sigma_i^2}\right]^{-\frac{1}{2}}$$

If $\mathcal{L}(x)$ is truly Gaussian, $\frac{\partial^2 W}{\partial x^2}$ is the same for all values of x

Otherwise it is better to use

$$\frac{\partial^2 \bar{W}}{\partial x^2} = \frac{\int \frac{\partial^2 W}{\partial x^2}\mathcal{L}(x)dx}{\int \mathcal{L}(x)dx}$$

Estimate number of events required to measure a parameter with a given accuracy → determine $\frac{\partial^2 \bar{W}}{\partial x^2}$ averaged over many experiments with N events

For 1 event $\frac{\partial^2 \bar{W}}{\partial x^2} = \int \frac{\partial^2 \ln f}{\partial x^2} f\, dm$ For N events $\frac{\partial^2 \bar{W}}{\partial x^2} = N \int \frac{\partial^2 \ln f}{\partial x^2} f\, dm$

Likelihood Method

$$\frac{\partial^2 \ln f}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{1}{f}\frac{\partial f}{\partial x}\right) = -\frac{1}{f^2}\left(\frac{\partial f}{\partial x}\right)^2 + \frac{1}{f}\frac{\partial^2 f}{\partial x^2}$$

$$\Rightarrow \int \frac{\partial^2 \ln f}{\partial x^2} f\,dm = -\int \frac{1}{f}\left(\frac{\partial f}{\partial x}\right)^2 dx + \int \frac{\partial^2 f}{\partial x^2} dm = -\int \frac{1}{f}\left(\frac{\partial f}{\partial x}\right)^2 dx + \frac{\partial^2}{\partial x^2}\int f\,dm$$

$$\int f\,dm = 1 \qquad \rightarrow \text{the second term drops out}$$

$$\frac{\partial^2 \bar{W}}{\partial x^2} = -N\int \frac{1}{f}\left(\frac{\partial f}{\partial x}\right)^2 dm$$

$$\Rightarrow \Delta x = \frac{1}{\sqrt{N}}\left[\int \frac{1}{f}\left(\frac{\partial f}{\partial x}\right)^2 dm\right]^{-\frac{1}{2}}$$

Want to measure coefficient of electron energy distribution in muon decay: $f(x,m) = (1+x.m)/2$ with 1% accuracy for x = -(1/3). Find N

Multiple Parameters

Measure ℓ parameters $x_1, x_2, \cdots x_l$ in an experiment with N events

If x_i's are uncorrelated, .i.e., $\overline{(x_i - x_i^\star)(x_j - x_j^\star)} = 0$ for $i \neq j$, then

$$\Delta x_i = \left(-\frac{\partial^2 W}{\partial x_i^2}\right)^{-\frac{1}{2}}$$

Likelihood Method

In general

$$W(x) = W(x^\star) + \sum_{i=1}^{l} \frac{\partial W}{\partial x_i}\Big|_{x_i^\star} \beta_i - \frac{1}{2} \sum_i \sum_j H_{ij}\beta_i\beta_j + \cdots$$

with $\beta_i = x_i - x_i^\star$

$$H_{ij} = -\frac{\partial^2 W}{\partial x_i \partial x_j}\Big|_{x_i^\star x_j^\star}$$

$$\frac{\partial W}{\partial x_i}\Big|_{x_i^\star} = 0$$

$$\ln \mathcal{L}(x) = W(x^\star) - \frac{1}{2} \sum_{i,j} H_{ij}\beta_i\beta_j + \cdots$$

Neglecting higher order terms

$$\mathcal{L}(x) = C \cdot \exp(-\tfrac{1}{2} \textstyle\sum_{i,j} H_{ij}\beta_i\beta_j)$$

$\rightarrow \ell$ dimensional Gaussian surface
(approximate $\mathcal{L}(x)$ Gaussian-like in the region $x_i \approx x_i^\star$)

H is a symmetric matrix $\rightarrow$ canbe diagonalized through unitary transformation

$$UHU^{-1} = h \quad \text{with } h \text{ diagonal}$$

and

$$\tilde{U} = U^{-1}$$

Likelihood Method

Let $\beta = (\beta_1, \cdots \beta_l)$ and $\gamma = \beta U^{-1}$

$\overline{\gamma_i \gamma_j} = \delta_{ij} h_i^{-1}$ [l dimensional Gaussian surface becomes product of l Gaussians]

$$\overline{\beta_i \beta_j} = \sum_{a,b} \overline{\gamma_a \gamma_b} U_{ai} U_{bj}$$

$$= (U^{-1} h U)^{-1}_{ii} = H^{-1}_{ii}$$

Thus $\overline{(x_i - x_i^\star)(x_j - x_j^\star)} = H^{-1}_{ij}$ with $H_{ij} = -\frac{\partial^2 W}{\partial x_i \partial x_j}$

Averaging over repeated experiments

$$\overline{H_{ij}} = N \int \frac{1}{f} \left(\frac{\partial f}{\partial x_i}\right) \left(\frac{\partial f}{\partial x_j}\right) dm$$

Measure range and straggling coefficient of mono energetic particles – estimate uncertainties $\mathcal{L}(x_1, x_2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi} x_2} \exp[-\frac{(m_i - x_1)^2}{2x_2}$.

Pattern Recognition in Tracker

- Non-destructive detectors observe a set of hits which are due to passage of charge particles
  - Hit ≡ one or more signal(s) which can be related to a spatial position
  - Given the Hit positions, try to identify the tracks which are responsible for the Hits
- There are two types of pattern recognition code:
  - Global: all hits enter into an algorithm in the same way and a list of tracks is produced (algorithm is linear with number of tracks)
    - Histogramming, Template matching, …..
  - Local: select a track candidate at a time starting with a few points and then predict if additional points belong to the candidate (computing time increases faster than linear with number of tracks)
    - Track following, road, ….
- A good track finding algorithm gives the same set of tracks irrespective of the order in which the points appear in the method

Curvature Sampling Method

Principle: Define a set of n different functions of coordinates and enter the function values in a n-dimensional histogram, Tracks will appear as peaks in the n-dimensional histogram

Tracker measurements (r, ϕ, z) are quantized (particularly r)
→ precompute finite number of quantities of interest

$$\phi - \sin^{-1}\left(\frac{C \cdot r}{2}\right)$$

C ≡ sampled signed curvature

If C is close to the curvature of the track of interest, the function → φ_0 of the track at r = 0

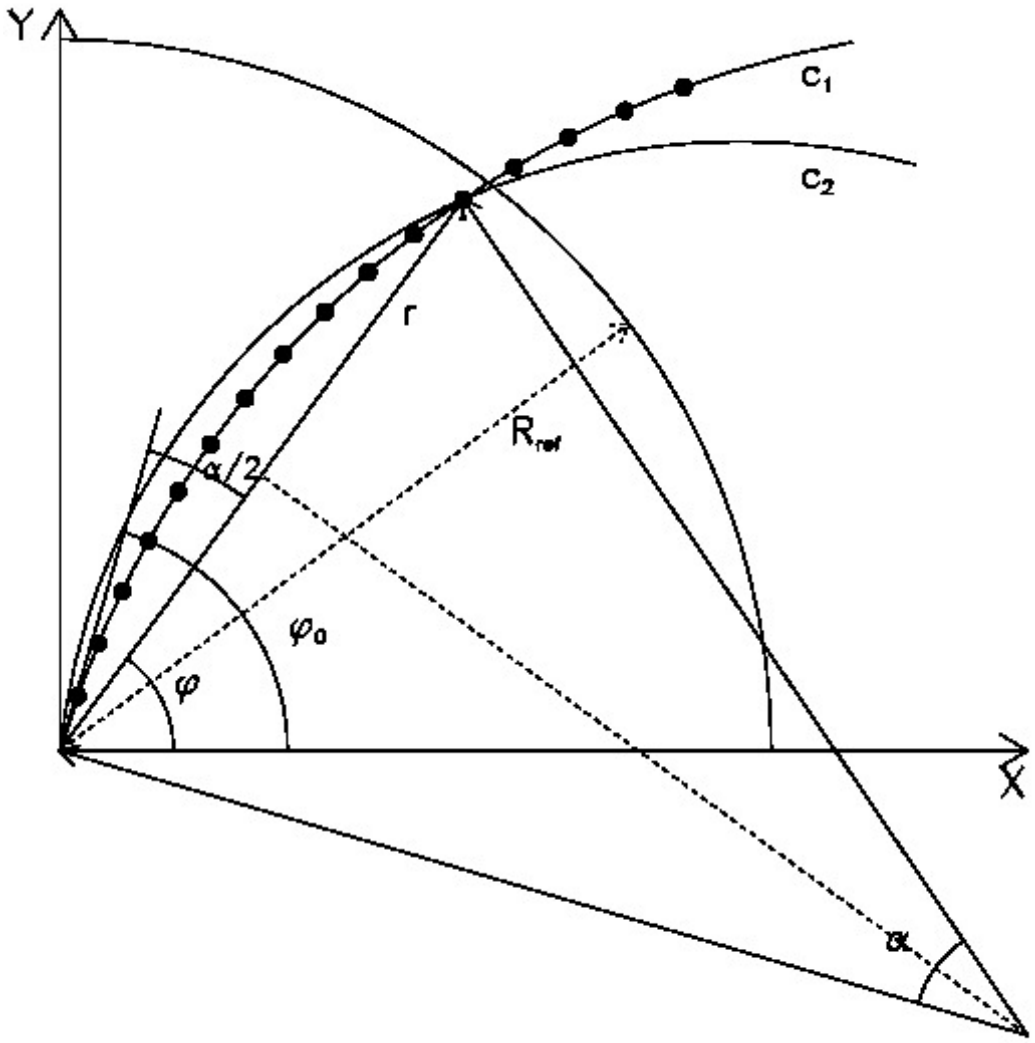
It will be the same for all measurements of the track → peak in the histogram at a fixed φ_0 for given C

Given a solenoidal magnetic field, tracks originating from the primary vertex will follow a circle in the transverse plane. So it can be paramtrized as:

$$f_i(R_{ref}) = \phi_i - \sin^{-1}\left(\frac{Cr_i}{2}\right) + 2\sin^{-1}\left(\frac{CR_{ref}}{2}\right)$$

$f_i(R_{ref})$ ≡ azimutal angle of the trajectory at $r = R_{ref}$

Curvature Sampling Method (II)



- Sample points for all curvatures between - $C_{\max}$ and $C_{\max}$ in steps of ΔC. All points belonging to a single trajectory will have the same f if computed with the closest sampled curvature → peak in the scatter plot f vs C

$$\phi_0 = \phi + \frac{\alpha}{2}$$

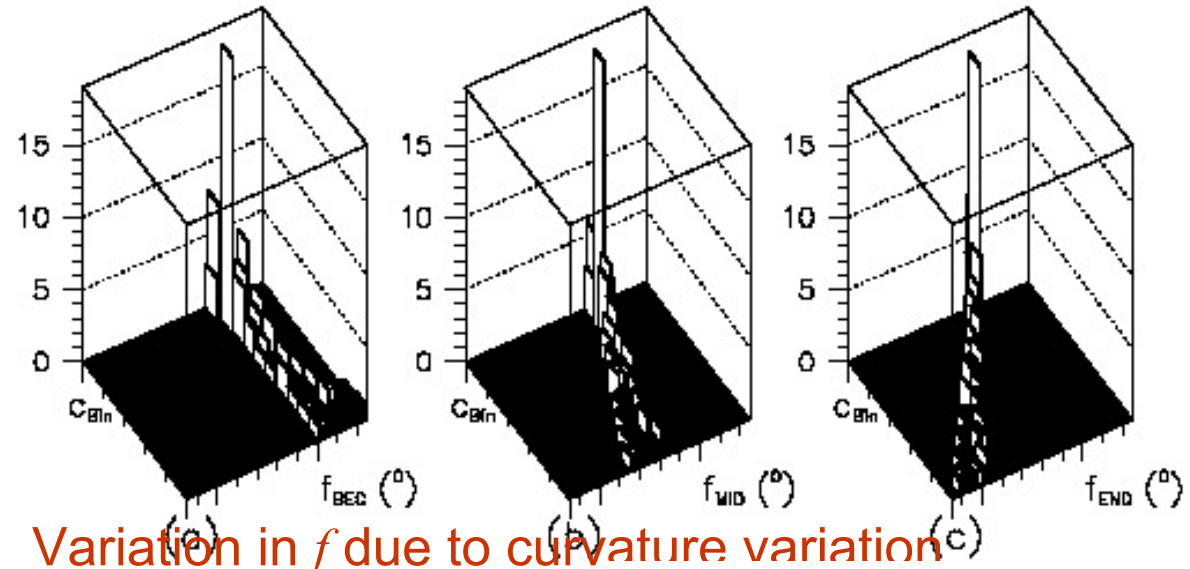
$$\sin\left(\frac{\alpha}{2}\right) = \frac{r}{2} \cdot \frac{1}{\rho}$$

$$\frac{1}{\rho} = -C$$

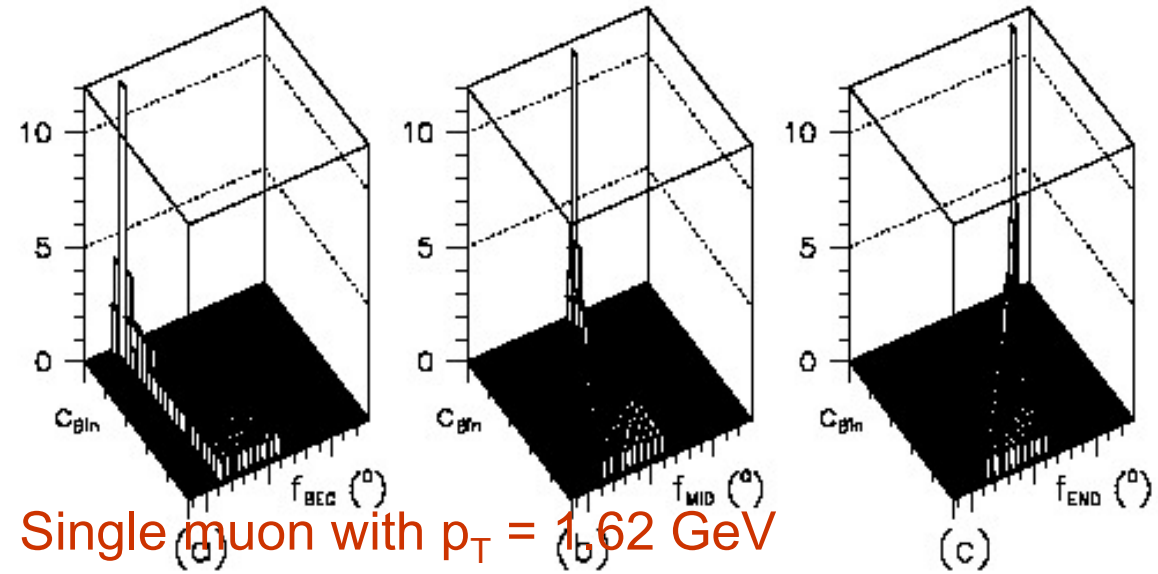
$$\Rightarrow \phi_0 = \phi - \sin^{-1}\left(C \cdot \frac{r}{2}\right)$$

Curvature Sampling Method (III)

Single muon with p_T = 41.4 GeV



Variation in f due to curvature variation smallest at middle point: $\delta f = (R_{ref} - \frac{r}{2}) \cdot \delta C$

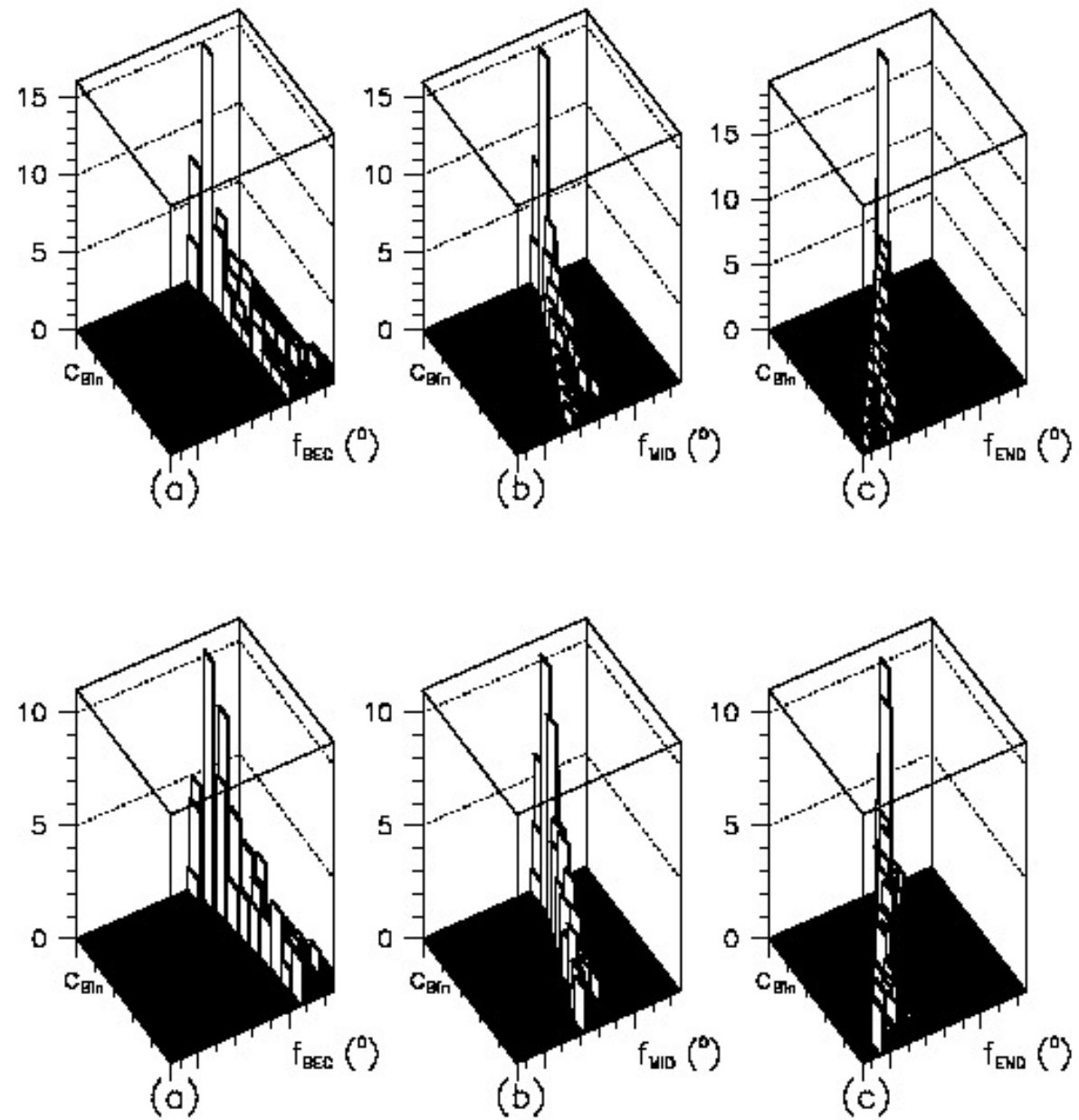


Single muon with p_T = 1.62 GeV

- Choose 3 values for reference R
  - near origin
  - at maximal R
  - half way through
- Take a realistic detector → CMS tracker R_{ref}^{End} = 120 cm
- Consider momentum range to be covered p_T > 1 GeV
- Bin size optimized to match p_T coverage and computing time: for 50 bins in C (either sign), $\Delta C \geq 2 \times 10^{-4}$ cm^{-1}

See clean peak even for low p_T tracks

Curvature Sampling Method (IV)



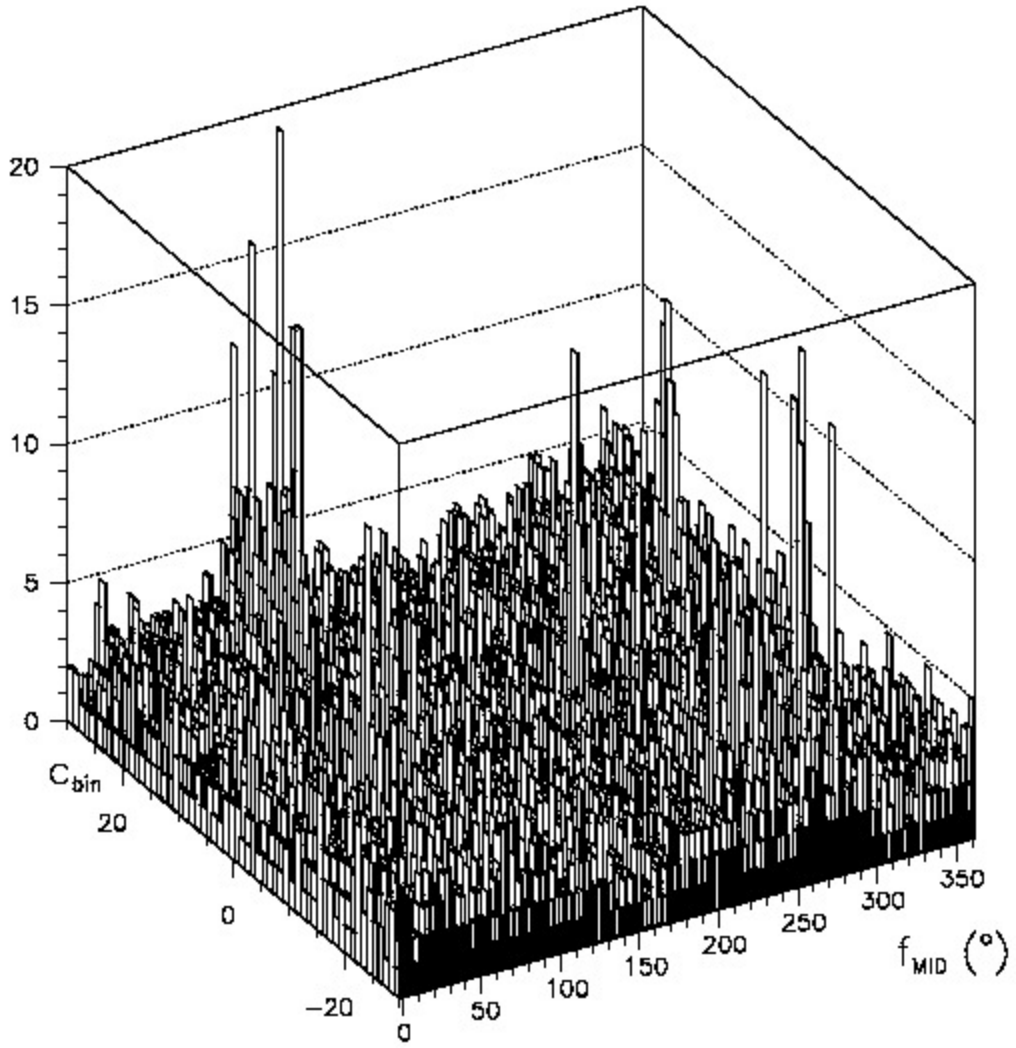
- Randomize the hit position by 200-500 μm (detector resolution is better than 50 μm) – algorithm is robust against this
- Introduce transverse shift of the origin by a few mm (tracks from secondary vertex) – peak are still recognized, but values of f, C at the peak are biased:

$$\phi = \phi_0 + \alpha \cdot r + \beta \cdot r^3 + \frac{d}{r}$$

$$\alpha = \frac{C}{2}, \quad \beta = -\frac{\alpha^3}{6}$$

d is the impact parameter

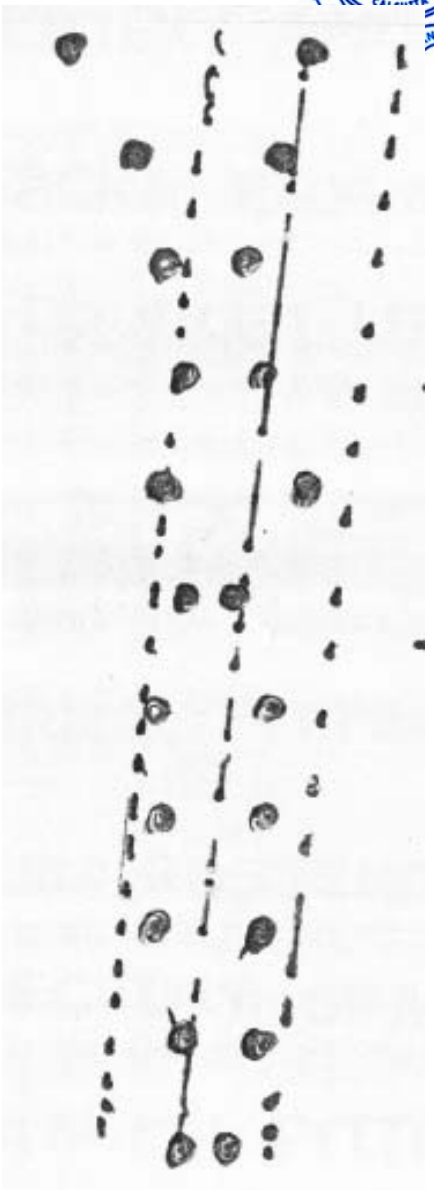
Curvature Sampling Method (V)



- Try the method in a multi-track environment: a Pythia6 multi-hadron event
  - Try an automatic peak finder by demanding at least half of the number of layers should be there in the peak
  - All candidates are found and they match well with the original tracks
  - Can be improved by demanding compatibility in z-measurements
  - Bin size should be optimized taking care the detector resolution

Road Method

- Pick initial points near the beginning and end regions of the detector
- For tracks in B-field, pick points in the middle region
- Use simple model of trajectory and define a road depending on the precision of hit points
- Pick hits compatible with a trajectory in the road defined by seed hits
- Modify the road with the hit collection and try for additional hits
- The road should be optimum in width
  - too narrow will result good hits being missed (inefficiency)
  - Too wide will result too many candidates (slow in speed)



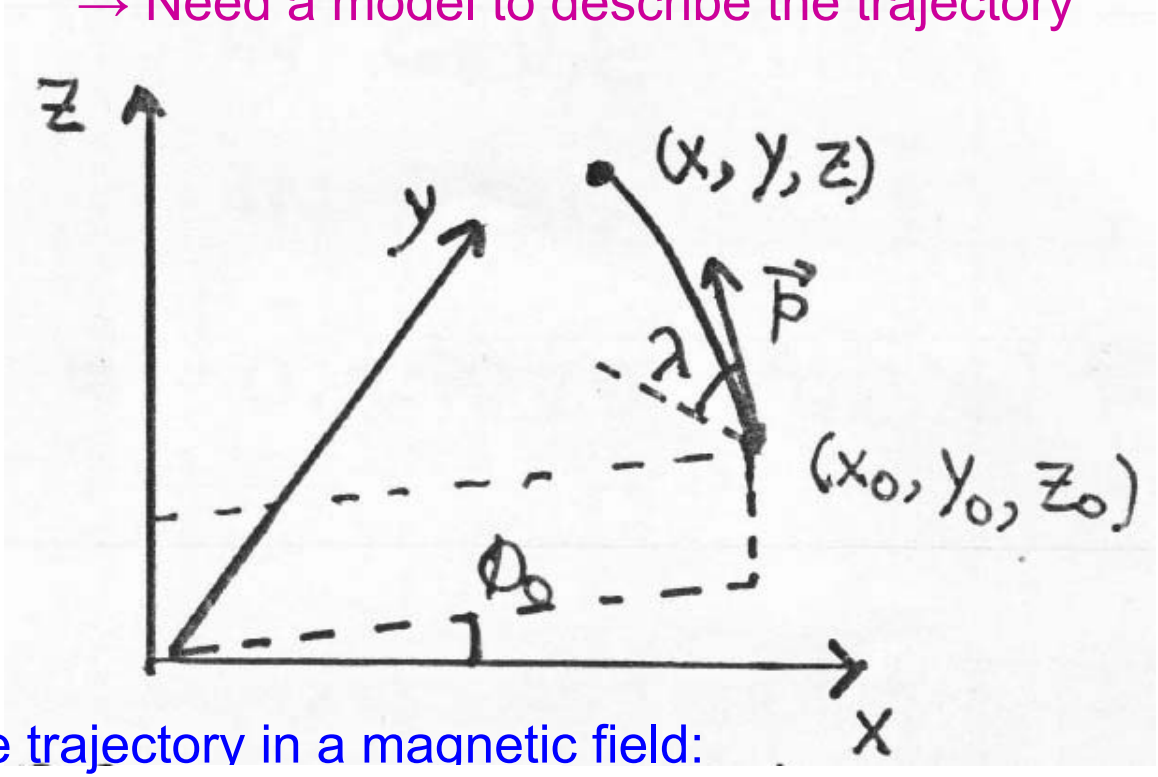
Tree Method

- Combine a pair of hits by some adjacency criteria to form a doublet
- Take one doublet (root) and combine it with a new doublet using 'doubl adjacency' criteria (could be sharing of hits)
- If this branch is valid by some validity criteria (sagitta, ….) attach the ro to this new doublet
- Replace the root by the newly connected doublet and repeat adjacency and branch validity test
- Continue this process to build a tree
- Introduce a depth parameter which measures the distance from the firs hit of root in terms of doublet
- The open end of the tree is called a leaf
- Trace back from leaf, look for unused branch, take the next best branch continue to build other trees
- Keep a compatible set of longest trees
- Basic object is a doublet. Each doublet of an ideal track originating from the vertex carries the same information as the track
- Track originating from the vertex can be used at
  - Doublet level: doublets point to small area centred around origin
  - Track level: move out rapidly picking upall acceptable doublets

Geometrical Fitting

- Pattern recognition associates a set of hits to a trajectory of a charged particle
- Geometrical fitting extracts accurate measurements of the track parameters

→ Need a model to describe the trajectory



Particle trajectory in a magnetic field:

$$\frac{d^2\vec{x}}{dS^2} = \frac{q}{p}\frac{d\vec{x}}{dS} \wedge \vec{B}$$

where S = distance along trajectory

Geometrical Fitting (II)

- Neglecting energy loss, multiple scattering this will be described by a helix in a uniform B-field
- For B-field along z-axis, particle motion will be circular when projected on the x-y plane and dip angle λ (out of x-y plane) is constant
- The projected curvature: $\frac{1}{\rho} = \frac{qB}{p\cos\lambda}$
- Any point on the helix will be given by

$$
\begin{aligned}
x(S) &= \rho[\cos(\phi_0 + \frac{S}{\rho}\cos\lambda) - \cos\phi_0] + x_0 \\
y(S) &= \rho[\sin(\phi_0 + \frac{S}{\rho}\cos\lambda) - \sin\phi_0] + y_0 \\
z(S) &= S\sin\lambda + z_0
\end{aligned}
$$

- To introduce energy loss substitute $\rho \rightarrow \rho(S)$
- For non-uniform B-field introduce $\rho \rightarrow \rho(x, y, z)$

Geometrical Fitting (III)

- Break trajectory into segments (configuration where planes ofdetector is parallel to x-y plane
- If (x_n, y_n, z_n) is a point on the trajectory, (x, y) at $z = z_{n+1}$ can be estimated using

$$
\begin{aligned}
x_{n+1} &= x_n + x'_n h_n + \frac{1}{2} x'' h_n \\
x'_{n+1} &= x'_n + x''_n h_n \\
y_{n+1} &= y_n + y'_n h_n + \frac{1}{2} y'' h_n \\
y'_{n+1} &= y'_n + y''_n h_n
\end{aligned}
\qquad (h_n = z_{n+1} - z_n)
$$

where

$$
\begin{aligned}
x'' &= \frac{q}{p}\frac{dS}{dz}\left[x'y'B_x - (1+x'^2)B_y + y'B_z\right] \\
y'' &= \frac{q}{p}\frac{dS}{dz}\left[(1+y'^2)B_x - x'y'B_y - x'B_z\right]
\end{aligned}
$$

with $x' = \frac{dx}{dS}$; $y' = \frac{dy}{dS}$ and $\frac{dS}{dz} = (1 + x'^2 + y'^2)$

- Fit the measured space points by minimizing χ^2 defined as

$$\chi^2 = \sum_{i=1}^{n} \left[\frac{x_i - f(w, z_i)}{\sigma_i}\right]^2$$

Geometrical Fitting (IV)

where x_i = Measured coordinates at planes $z = z_i$

σ_i = Error on measuring the point

w = Trajectory parameters $(x_0, y_0, p, \lambda, \phi)$

f = Model for the trajectory

Obtain the track parameters using

$$\frac{\partial \chi^2}{\partial w_i} = 0$$

Also get the covariance matrix

- In the least square method, a linear expansion is used:

$$f(\underline{w}) = f(\underline{w_0}) + A \cdot (\underline{w} - \underline{w_0}) + \mathcal{O}\left((\underline{w} - \underline{w_0})^2\right)$$

where $A = \frac{\partial f(w)}{\partial w}$ at $w = w_0$

- In general one uses measurement variables $\underline{m}$. Solution to minimization gives

$$\underline{w} = \underline{w_0} + (A^\dagger W A)^{-1} A^\dagger W \left[\underline{m} - f(\underline{w_0})\right]$$

Where the weight matrix W is evaluated by inverting the covariance matrix (V) obtained from 2 independent contributions

$$\epsilon = \epsilon_{Detector} + \epsilon_{MS}$$

Geometrical Fitting (V)

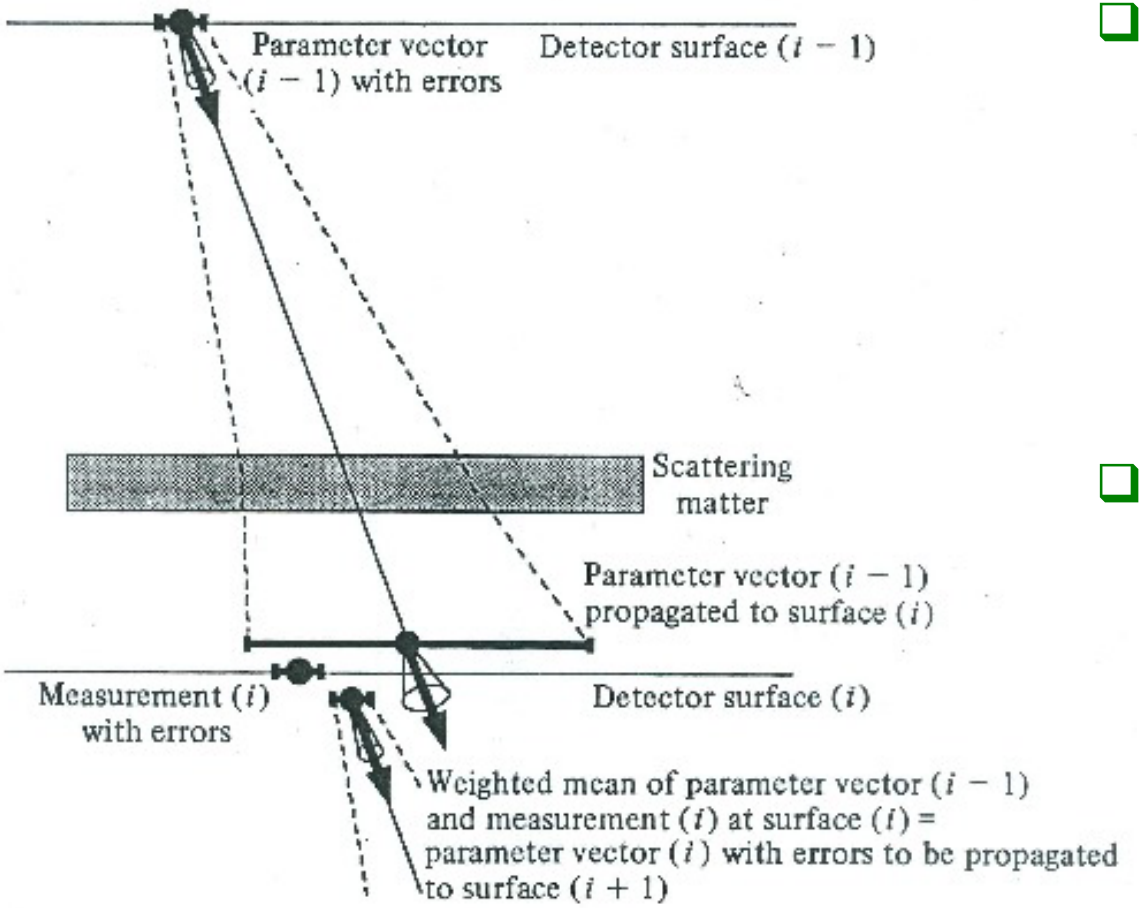
- So V is not diagonal and is to be inverted numerically. For n measured points, covariance matrix is of size nxn and inversion time would be ~ n^3.
- If the matrix inversion is tried in every step of pattern recognition to throw away wrong ambiguous solutions, it is worse
- Why not try a recursive track fitting procedure: Kalman Filter
- Let the track parameters $\underline{w}_i$ known at a surface with it covariance matrix C
- Propagate the parameter vector $\underline{w}_i$ and the covariance matrix to the next surface $i+1$

$$\begin{aligned} \underline{w}_{i+1}^{(i)} &= f_{i+1}(\underline{w}_i) \\ C_{i+1}^{(i)} &= D_{i+1}^{(i)} C_i D_{i+1}^{(i)\dagger} \\ D_{i+1}^{(i)} &= \frac{\partial f_{i+1}}{\partial \underline{w}_i} \end{aligned}$$

with f_{i+1} the precise track model between i and $i+1$.

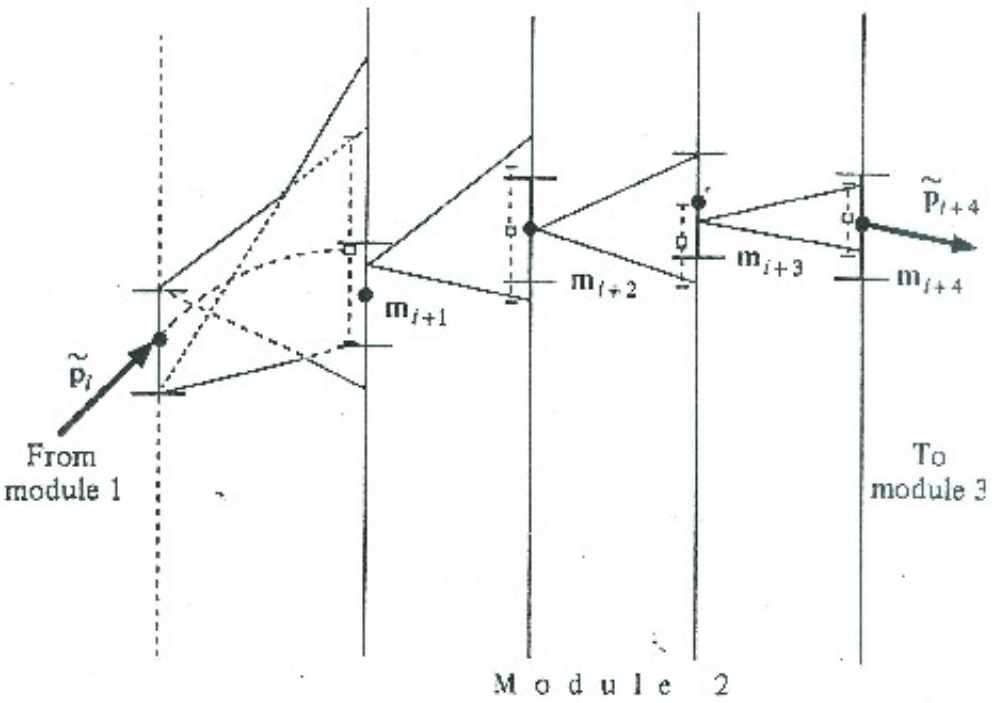
- Now use the measurements at the surface $i+1$

Geometrical Fitting (VI)



- Estimate at the position $i+1$ a properly weighted mean of actual measurement at the surface $i+1$ and the prediction based on information of preceding surfaces
- This allows evaluation of increment in χ^2 and hence of overall χ^2

Geometrical Fitting (VII)



- The progressive fit is suitable for combined track finding and track fitting
- There is no large matrix to be inverted and number of computations increases only linearly with number of measurements
- The estimated track parameters closely follow the real path of the particle
- Linear approximation of the track model need to be valid only between two surfaces of measurement

Geometrical Fitting (VIII)

- Track parameters at a point include information of preceding detectors and known with precision defined by those measurements
- For less confusion in less dense region tracks are normally propagated back from less precise detectors (in less dense region) to more precise detectors
- Some smoothing algorithm is required along with track propagation and recursive estimation of track parameters at all intermediate points
- Usually track finding is started from either end of a tracking device and an optimum merging is done
  - Find most suitable starting element
  - Make correct estimation of the amount of matter traversed
- In current days detectors which use multiple detector modules and non-negligible material, Kalman filtering is the best way out for track finding

Kinematic Fitting

- Try to derive kinematic quantities from geometric measurements. Geometric fits give rise to some level of precision. If there are other physics constraints, one can improve this precision utilizing these.
- For example let us look into W-mass measurement in a process like

$$e^+e^- \rightarrow W^+W^-$$

$$W^+ \rightarrow q\bar{q}'; \quad W^- \rightarrow q''\bar{q}'''$$

- Here jet directions are very well determined from calorimetric measurements but not the energies. However, we know
  - Energy momentum is conserved in the production of W-boson
  - The two W's have the same rest mass
  - Net initial momentum is 0 and energy is $2E_{Beam}$
  - Beam energy is measured with very high precision
- These information can be utilized to improve the resolution of the jet energies and hence measured W-mass

Kinematic Fitting (II)

- Let us start with the general formulation of least square prblem. Let
  - $\underline{m}$ variables measured in experiments
  - $\underline{m}^0$ measurements of these variables
  - G_M the covariance matrix
  - $\underline{x}$ unmeasured parameters (if any)
  - $f_i(\underline{x}, \underline{m}) = 0$ set of constraint equations $i = 1, \cdots, k$
- The best estimate of measured and unknown quantities are obtained by minimizing

$$\chi^2(m, x, \lambda) = (m - m^0)^\dagger G_m (m - m^0) + 2\lambda^\dagger f(x, m)$$

- Where λ = vector (of k components) of Lagrange multipliers

$$\frac{d\chi^2}{dm} = 0 = 2\left[(m - m^0)^\dagger G_m + \lambda^\dagger f_m(x, m)\right] \quad \text{(A)} \qquad f_m = \frac{\partial f}{\partial m}$$

$$\frac{d\chi^2}{dx} = 0 = 2\lambda^\dagger f_x \quad \text{(B)} \qquad f_x = \frac{\partial f}{\partial x}$$

$$\frac{d\chi^2}{d\lambda} = 0 = 2f(x, m) \quad \text{(C)} \quad \text{Constraint relation}$$

Kinematic Fitting (III)

- In general the constraint equations are not linear. So an iterative procedure is followed:
- Start with initial guess; want to calculate values of $(\nu+1)$th iterations
  - Linearized constraint equation (C) gives

$$f^{\nu+1} = 0 = f^{\nu} + f_x^{\nu}(x^{\nu+1} - x^{\nu}) + f_m^{\nu}(m^{\nu+1} - m^{\nu}) \qquad \text{(D)}$$

  - From equation (A) one gets

$$\begin{aligned} \left(m^{\nu+1} - m^0\right)^{\dagger} G_m &= -(\lambda^{\nu+1})^{\dagger} f_m^{\nu+1} \\ \Rightarrow \left(m^{\nu+1} - m^0\right)^{\dagger} &= -(\lambda^{\nu+1})^{\dagger} f_m^{\nu+1} G_m^{-1} \\ \Rightarrow m^{\nu+1} - m^0 &= -G_m^{-1}(f_m^{\nu+1})^{\dagger}\lambda^{\nu+1} \end{aligned} \qquad \text{(A')}$$

  - Approximate $f^{\nu+1}$ with f^{ν} in the last term and use (D) to get

$$f^{\nu} + f_x^{\nu}\left(x^{\nu+1} - x^{\nu}\right) + f_m^{\nu}\left(m^0 - m^{\nu} + G_m^{-1} f_m^{\nu\dagger}\lambda^{\nu+1}\right) = 0 \qquad \text{(E)}$$

  - Let

$$\underline{R} = f^{\nu} + f_m^{\nu}\left(m^0 - m^{\nu}\right) \qquad \text{(F)}$$

$$\underline{S} = f_m^{\nu} G_m^{-1} (f_m^{\nu})^{\dagger} \qquad \text{(G)}$$

Kinematic Fitting (IV)

– $\underline{R}, \underline{S}$ depend on quantities known at iteration ν

$$f_x^\nu \left(x^{\nu+1} - x^\nu\right) + R + S\lambda^{\nu+1} = 0$$

$$\Rightarrow \quad \lambda^{\nu+1} = S^{-1}\left[R + f_x^\nu \left(x^{\nu+1} - x^\nu\right)\right] \qquad \text{(H)}$$

$$= S^{-1}R + S^{-1} f_x^\nu \left(x^{\nu+1} - x^\nu\right)$$

– Put this back in (B)

$$0 = \lambda^\dagger f_x = \left[S^{-1}R + S^{-1} f_x^\nu \left(x^{\nu+1} - x^\nu\right)\right]^\dagger f_x^\nu$$

$$= R^\dagger S^{-1\dagger} f_x^\nu + \left(x^{\nu+1} - x^\nu\right)^\dagger f_x^{\nu\dagger} S^{-1\dagger} f_x^\nu$$

$$\Rightarrow \quad \left(x^{\nu+1} - x^\nu\right)^\dagger f_x^{\nu\dagger} S^{-1\dagger} f_x^\nu = -R^\dagger S^{-1\dagger} f_x^\nu$$

$$\Rightarrow \quad f_x^{\nu\dagger} S^{-1} f_x^\nu \left(x^{\nu+1} - x^\nu\right) = -f_x^{\nu\dagger} S^{-1} R$$

$$\Rightarrow \quad \left(x^{\nu+1} - x^\nu\right) = -\left[f_x^{\nu\dagger} S^{-1} f_x^\nu\right]^{-1}\left[f_x^{\nu\dagger} S^{-1} R\right] \qquad \text{(I)}$$

Kinematic Fitting (V)

- RHS of (I) completely known at iteration ν $\rightarrow x^{\nu+1}$
- Substitute this in (H) $\rightarrow \lambda^{\nu+1}$
- Substitute $\lambda^{\nu+1}$ in (A') $\rightarrow m^{\nu+1}$
- So go from step to step choosing m, x, λ atisfying the constraint equations and minimizing χ^2 simultaneously

- Iterations stop when
  - Constraint equations are balanced to better than the precision required
  - Derivatives $\frac{\partial\chi^2}{\partial m}, \frac{\partial\chi^2}{\partial x}$ are sufficiently close to 0
  - χ^2 change per iteration step is small
- Now $m^{\nu+1}, x^{\nu+1}$ can be expressed explicitly in terms of m^0, G^{-1} Approximate to linear equation

$$m^{\nu+1} = g(m^0); \qquad x^{\nu+1} = h(m^0)$$

- Carry out error propagation

$$G^{-1}_{m^{\nu+1}} = \left(\frac{dg}{dm^0}\right) G^{-1}_m \left(\frac{dg}{dm^0}\right)^\dagger$$

$$G^{-1}_{x^{\nu+1}} = \left(\frac{dh}{dm^0}\right) G^{-1}_m \left(\frac{dh}{dm^0}\right)^\dagger$$

uction

Kinematic Fitting (VI)

- And correlation between measured and unmeasured quantities:

$$\left(\frac{dg}{dm^0}\right) G_m^{-1} \left(\frac{dh}{dm^0}\right)$$

- (A') gives $m^{\nu+1} = m^0 - G_m^{-1} \left(f_m^\nu\right)^\dagger \lambda^{\nu+1}$
- So

$$\begin{aligned}
\frac{dg}{dm^0} &= 1 - G_m^{-1} \left(f_m^\nu\right)^\dagger \frac{d\lambda^{\nu+1}}{dm^0} \\
&= 1 - G_m^{-1} \left(f_m^\nu\right)^\dagger S^{-1} \left[\frac{dR}{dm^0} + f_x^\nu \frac{d(x^{\nu+1} - x^\nu)}{dm^0}\right] && \text{From (H)} \\
&= 1 - G_m^{-1} \left(f_m^\nu\right)^\dagger S^{-1} \left[\frac{dR}{dm^0} - f_x^\nu \left(f_x^{\nu\dagger} S^{-1} f_x^\nu\right)^{-1} f_x^{\nu\dagger} S^{-1} \frac{dR}{dm^0}\right] && \text{Using (I)} \\
&= 1 - G_m^{-1} \left(f_m^\nu\right)^\dagger S^{-1} \left[f_m^\nu - f_x^\nu \left(f_x^{\nu\dagger} S^{-1} f_x^\nu\right)^{-1} f_x^{\nu\dagger} S^{-1} f_m^\nu\right] && \text{From (F)}
\end{aligned}$$

- Similarly

$$\frac{dh}{dm^0} = \left[f_x^{\nu\dagger} S^{-1} f_x^\nu\right]^{-1} f_x^{\nu\dagger} S^{-1} f_m^\nu$$

- Thus variance becomes

$$\begin{aligned}
G_{m^{\nu+1}}^{-1} &= G_m^{-1} - G_m^{-1} f_m^{\nu\dagger} S^{-1} f_m^\nu G_m^{-1} + \\
&\quad G_m^{-1} f_m^{\nu\dagger} S^{-1} f_x^\nu \left(f_x^{\nu\dagger} S^{-1} f_x^\nu\right)^{-1} f_x^{\nu\dagger} S^{-1} f_m^\nu G_m^{-1} \\
G_{x^{\nu+1}}^{-1} &= \left(f_x^{\nu\dagger} S^{-1} f_x^\nu\right)^{-1}
\end{aligned}$$

Kinematic Fitting (VII)

- And the correlation is

$$C_{(mx)^{\nu+1}} = -G_m^{-1} f_m^{\nu\dagger} S^{-1} f_x^{\nu} \left(f_x^{\nu\dagger} S^{-1} f_x^{\nu} \right)^{-1}$$

- So the overall covariance matrix becomes

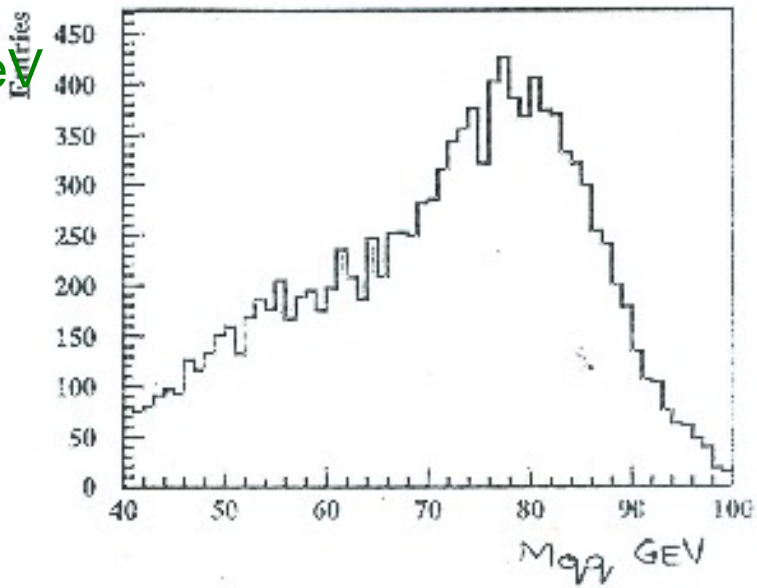
$$\begin{bmatrix} G^{-1}_{m^{\nu+1}} & C_{(mx)^{\nu+1}} \\ C^{\dagger}_{(mx)^{\nu+1}} & G^{-1}_{x^{\nu+1}} \end{bmatrix}$$

- The fit procedure reduces variance on the measured quantities; but introduces correlations where none existed initially

In the current example, all quantities have some measurements; i.e., the vector $\underline{x}$ has no entries. But the measurements were uncorrelated to start with and they become correlated at the end of the fit. So for any derived quantities, one has to use the full covariance matrix to estimate uncertainties

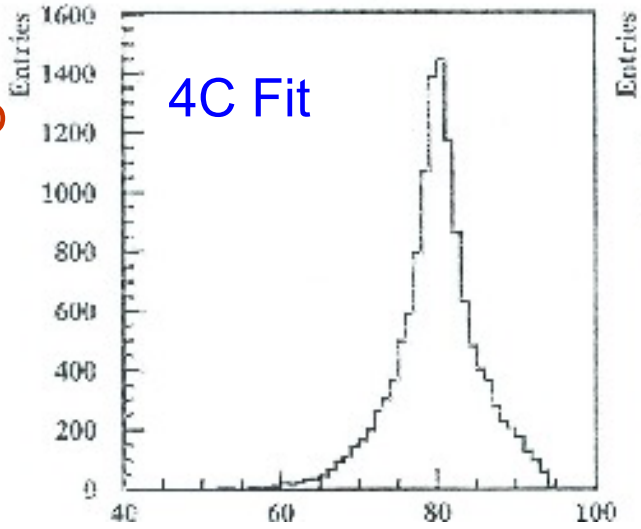
Kinematic Fitting (VIII)

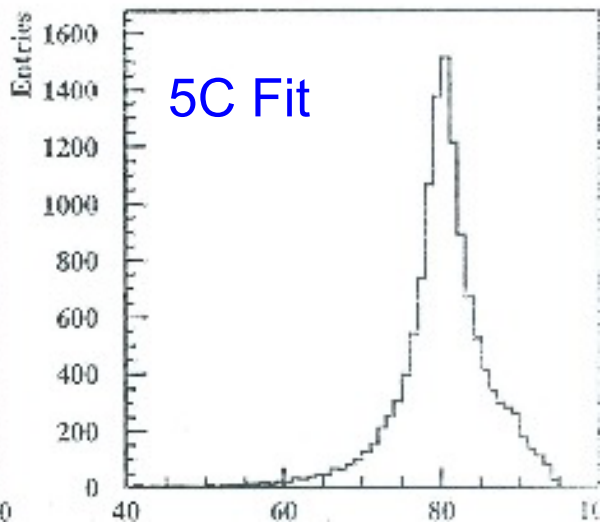
From e^+e^- data at $\sqrt{s}$ = 189 GeV
in the L3 experiment



Measured spectrum

4C Fit: Conserve E/p
5C Fit: + $m_{12} = m_{34}$





Simulation and Reconstruction