MONTE CARLO METHODS

Eric Prebys
Phy 40
Fall 2025

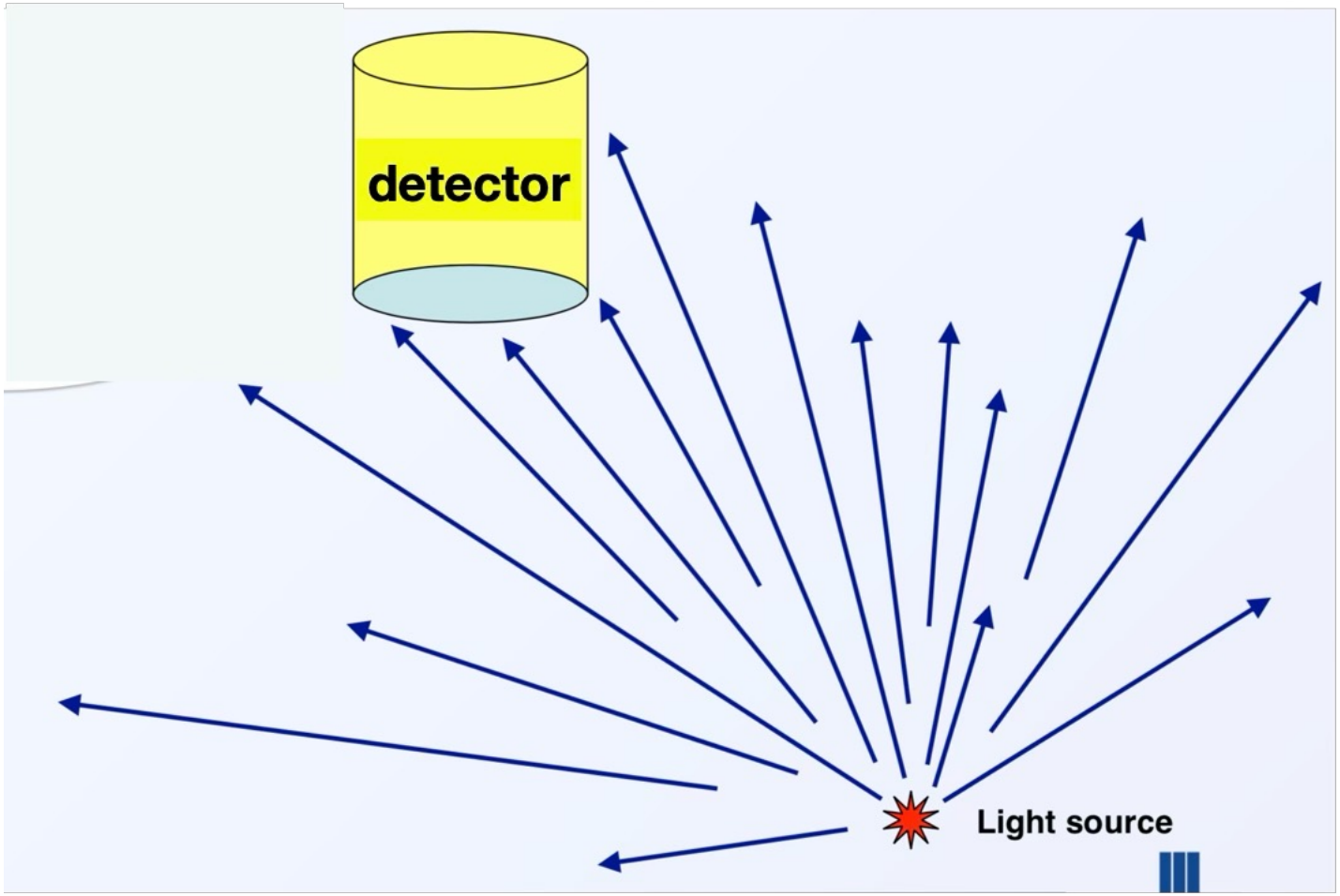
Monte Carlo Methods

- "Monte Carlo" is a generic term for a statistical analysis based on random distributions.
  - Named for Monte Carlo, Monaco, famous for
  - gambling
- Monte Carlo methods are used in a huge range of scientific applications in all sorts applications
  - Detector performance
  - Medical treatments
  - Particle accelerator modeling
  - Material science
  - Climate science
  - Etc, etc

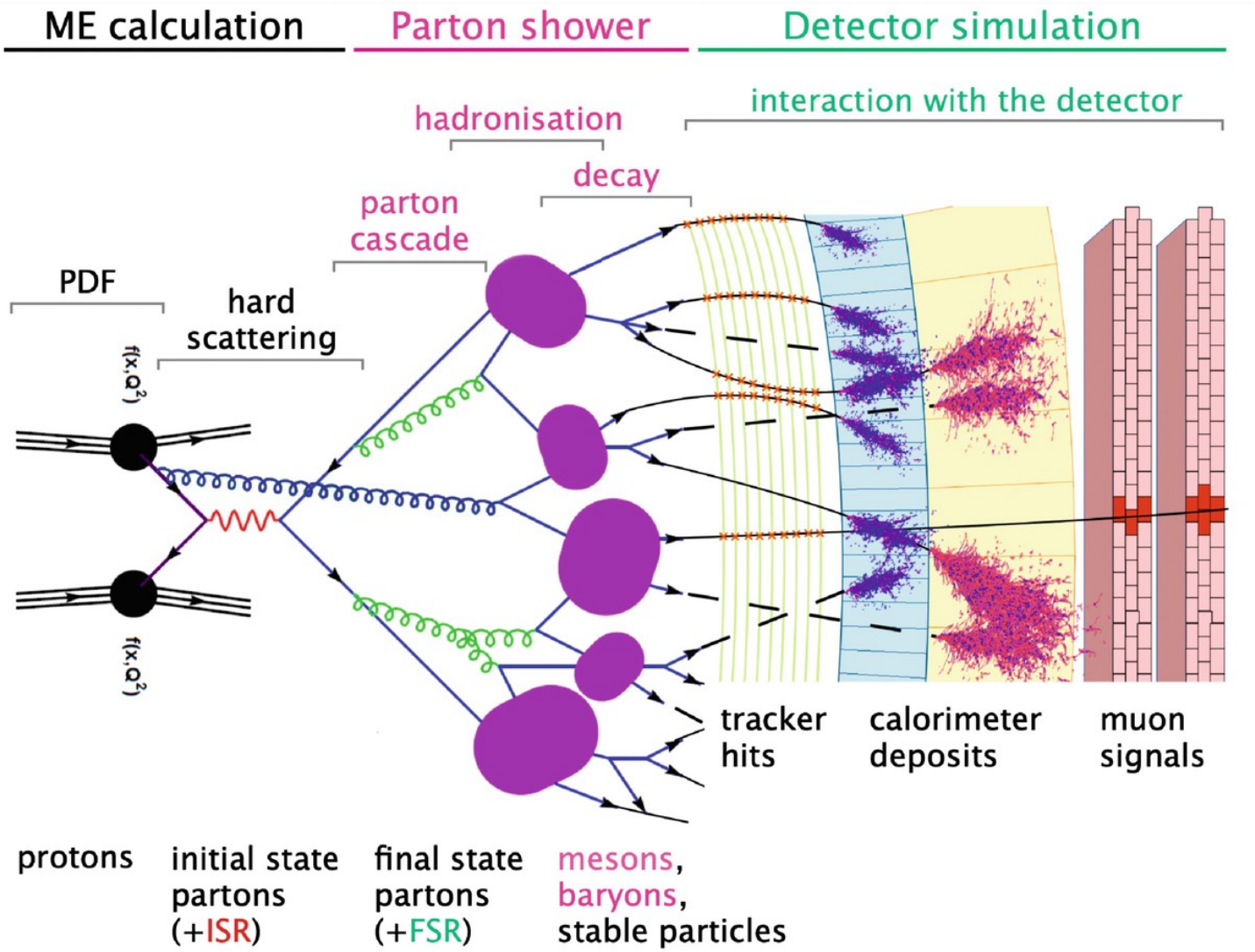
- In modern experiments, the physics analysis can be viewed as a part of the the Monte Carlo analysis, rather than vice versa.

Example: Detector Acceptance

- How many particles from a particular source will hit a detector



Complicated Example: Full Event Simulation at CERN

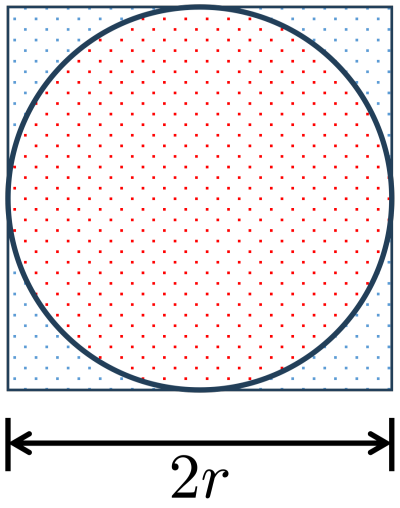


Markov Chain

- Most of our Monte Carlo examples involve what is called a “Markov Chain”
- A Markov Chain is a stochastic process in which each subsequent state depends only on the previous state.
  - Sometimes referred to as “memorylessness”.
- Some non-Markov chains can become Markov chains with more information
  - For examples, if I’m playing Black Jack, the probability of the cards coming up in each deal are non-Markov to the players.
  - However, the process becomes Markov if I include the cards remaining in the deck after each deal.

Simple Example: Calculating π with Monte Carlo

- We're going to throw a set of points to uniformly populate a square of sides *2r*.
- We expect the fraction of the points that fall inside of an inscribed circle of radius *r* to be the ratio of the areas
  - Let's call them "hits"



$$p \equiv \frac{N_{hits}}{N_{points}} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}$$

$$\sigma_p = \sqrt{\frac{(1-p)}{N_{hits}}}$$

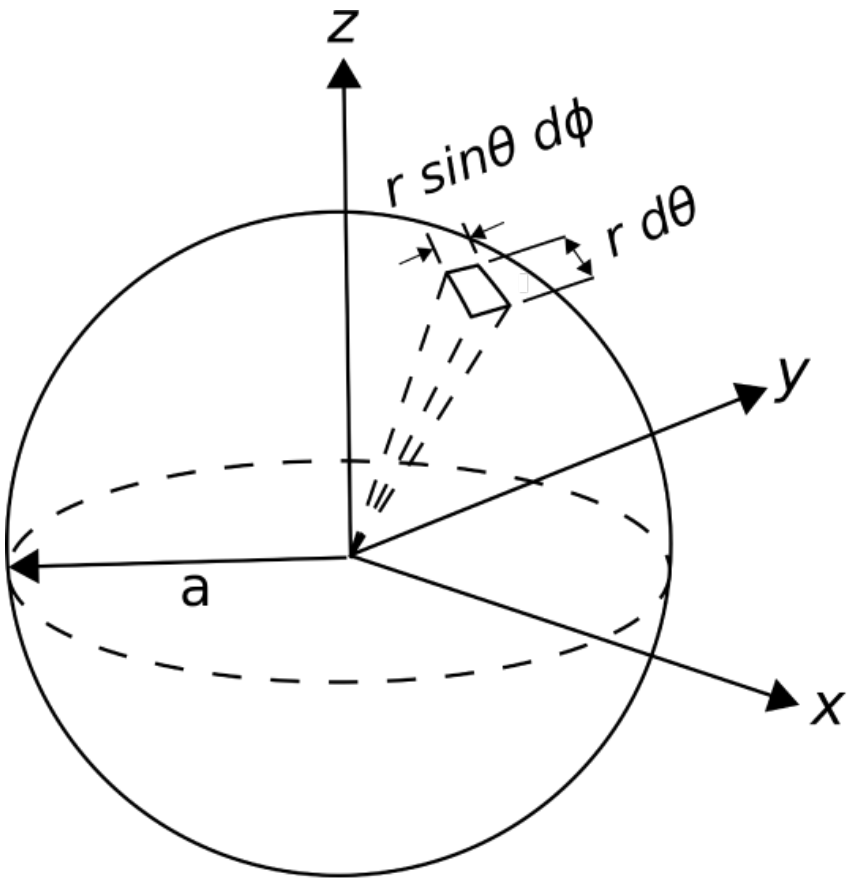
$$\pi = 4p$$

$$\sigma_\pi = 4\sigma_p$$

- You're going to do a variation of this in lab, so I'll stop here.

Spherical Distributions

- If we look at the differential surfaces area of a sphere



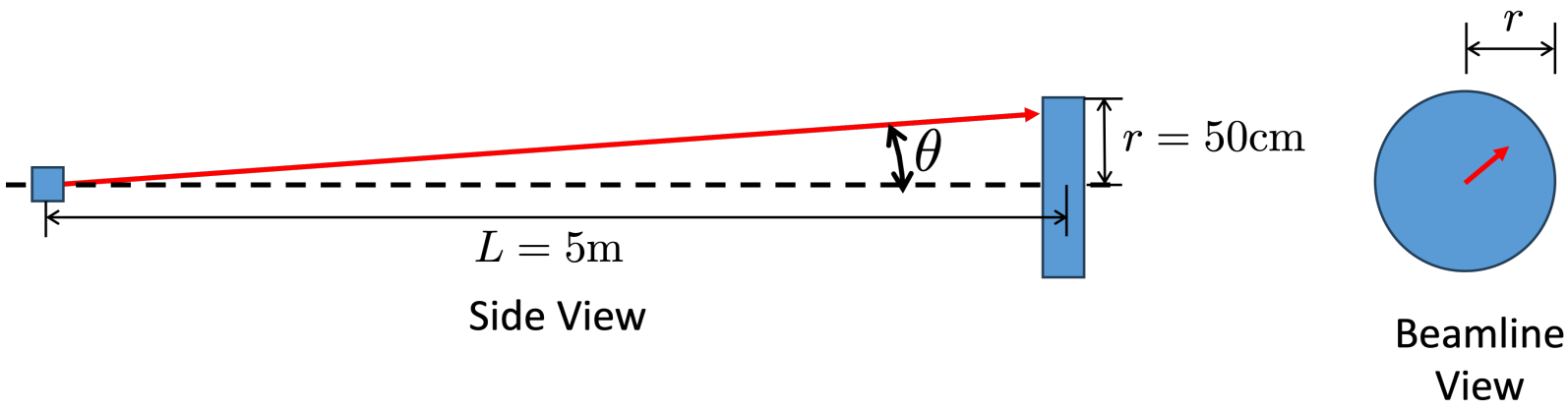
$$
\begin{aligned}
dA &= (r d\theta)(r \sin\theta d\phi) \\
&= r^2 (\sin\theta d\theta) d\phi \\
&= r^2 d(\cos\theta) d\phi \\
&\equiv r^2 d\Omega
\end{aligned}
$$

A distribution that's uniform in ϕ and $(\cos\theta)$ will be spherically symmetric

Solid angle

Detector Acceptance Example

- If we generate a spherically symmetric distribution of particles, how many will it a target of 50 cm radius 5 m away?



- We expect the fraction to be the ratio of the areas

$$N_{hits} = N_{points} \frac{\pi r^2}{4\pi L^2} = N_{points} \frac{1}{4} \left(\frac{r}{L} \right)^2$$

Code Example

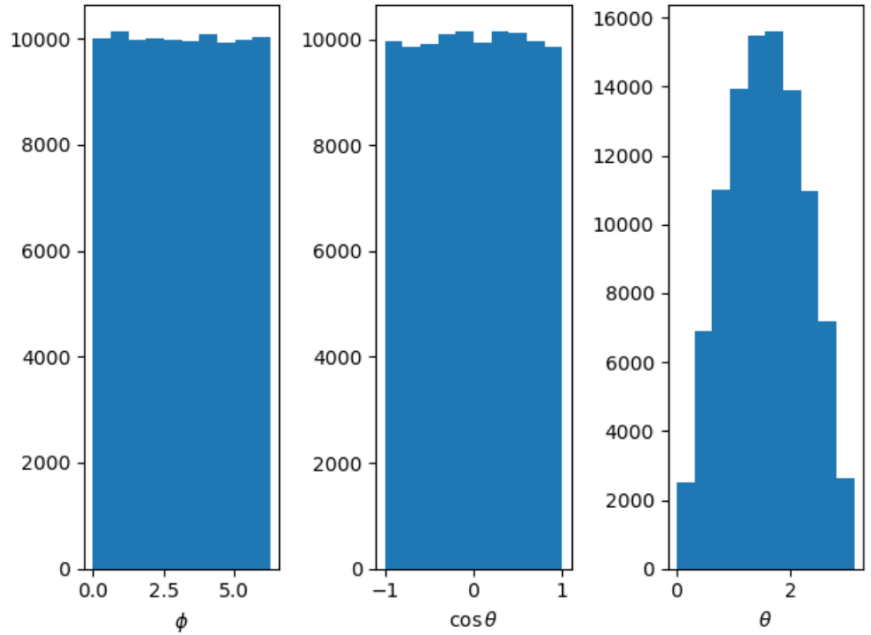
```
npoints = 100000
# randomly distribute cosine theta and phi
costheta = np.random.uniform(-1.,1.,npoints)
phi = np.random.uniform(0,2*math.pi,npoints)
# Create a Pandas dataframe
points = pd.DataFrame({'costheta':costheta,'phi':phi})
# Calculate theta
points['theta'] = np.arccos(points['costheta'])
# Histogram all three
fig,(ax1,ax2,ax3) = plt.subplots(1,3)
ax1.hist(points['phi'])
ax1.set_xlabel(r'$\phi$')

ax2.hist(points['costheta'])
ax2.set_xlabel(r'$\cos\theta$')

ax3.hist(points['theta'])
ax3.set_xlabel(r'$\theta$')

plt.tight_layout()  # Fixes overlaps
plt.show()
```

Calculate θ from $\cos\theta$



Does this make sense?

Project to Target

```
# Now let's see how many hit a 50 cm target 5m away
L = 5.
points['tantheta']=np.tan(points['theta'])
points['rho'] = L*points['tantheta']
# Drop the backwards going tracks!
hits = points.query("costheta>0.")
hits = hits.query("rho<.5")
nhits = len(hits)
dnhits = math.sqrt(nhits*(1-nhits/npoints))
expected = npoints*(.5)**2/L**2/4
print("Nhits: %d +- %.1f, expected: %.1f (%.2f sigma)"%
      (nhits,dnhits,expected,(nhits-expected)/dnhits))
```

```
Nhits: 223 +- 14.9, expected: 250.0 (-1.81 sigma)
```

Calculate $\tan\theta$

Calculate radius at target

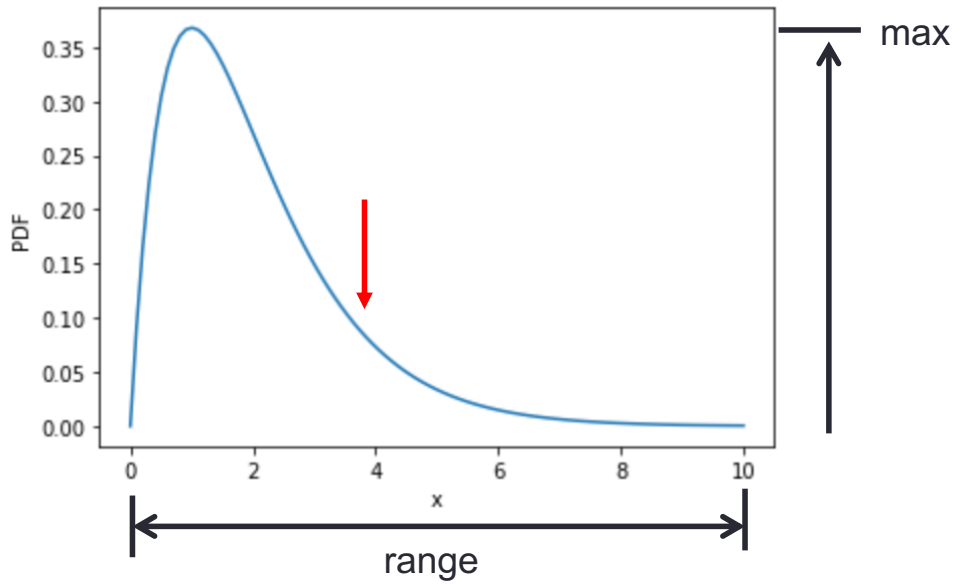
Drop backward going tracks!

How many *are* ρ<*50* cm?

Compare to expectations.

Reminder: Arbitrary Distributions

- Example:



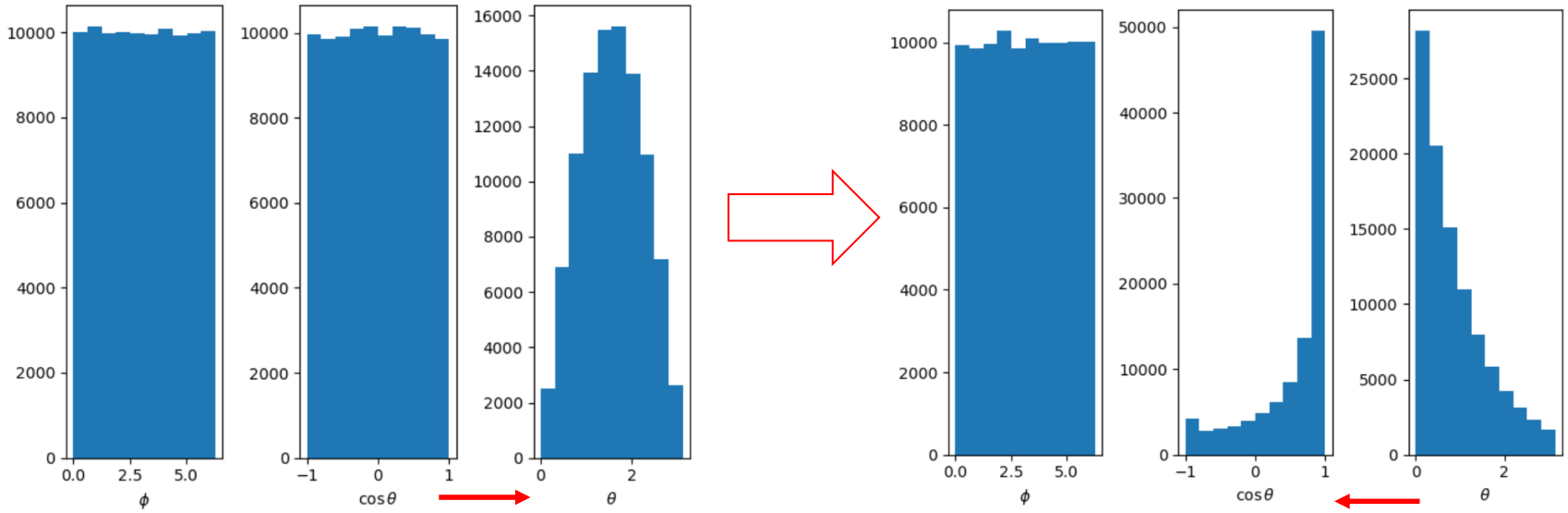
- Step 0: Choose a range for your random numbers
- Step 1: Generate a random value of *x* which is *uniformly* distributed over the range.
- Step 2: Evaluate the PDF at this point
- Step 3: Is the PDF value at this point greater than a random number between 0 and that maximum value of the function?
  - Yes? Return the value of *x*
  - No? Go back to step 1

Arbitrary Angular Distribution

- Instead of uniform cos(θ), try $\mathrm{PDF}(\theta) = e^{-\theta}$

```
costheta = np.random.uniform(-1.,1.,npoints)
phi = np.random.uniform(0,2*math.pi,npoints)
points = pd.DataFrame({'costheta':costheta,'phi':phi})
points['theta'] = np.arccos(points['costheta'])
```

```
theta = mydist(thetadist,npoints,0.,math.pi,1)
phi = np.random.uniform(0,2*math.pi,npoints)
points = pd.DataFrame({'theta':theta,'phi':phi})
points['costheta'] = np.cos(points['theta'])
```



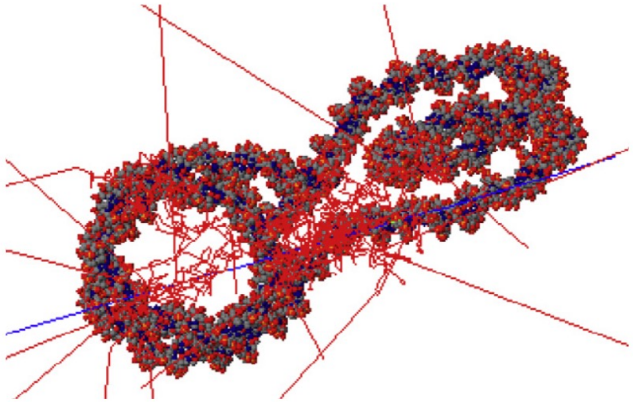
Nhits: 9873 +- 15.5, compared to 240 +- 15.5 before

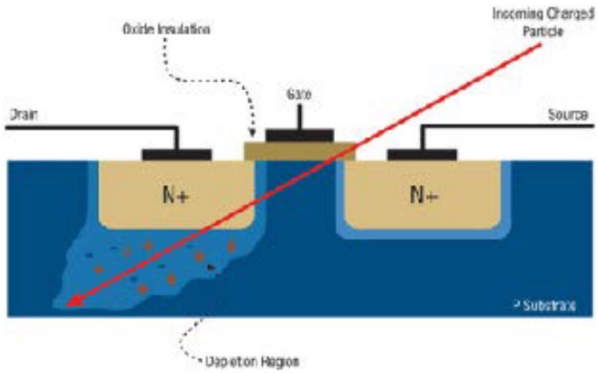
GEometry ANd Tracking (GEANT)

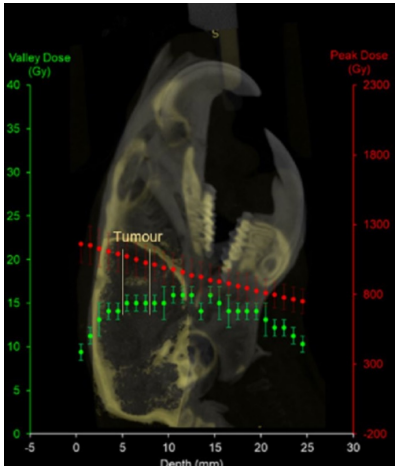
- There are many Monte Carlo tools for different applications, but by far the most widely used is GEANT.
- GEANT is a system of simulation and visualization tools, developed at CERN for particle physics.
- The first version of GEANT was written in FORTRAN in 1974.
- The capabilities of GEANT were greatly expanded in GEANT3 (still in FORTRAN) from 1982 through the early 1990s to support the experiments at the LEP collider at CERN.
- Beginning in the 1990s, GEANT4 was entirely re-written in object-oriented C++.
- It has since branched out from HEP to become a ubiquitous simulation tool in many fields
- A host of specialized physics libraries has been developed to support the different user communities.

Other Users of Geant4 Include

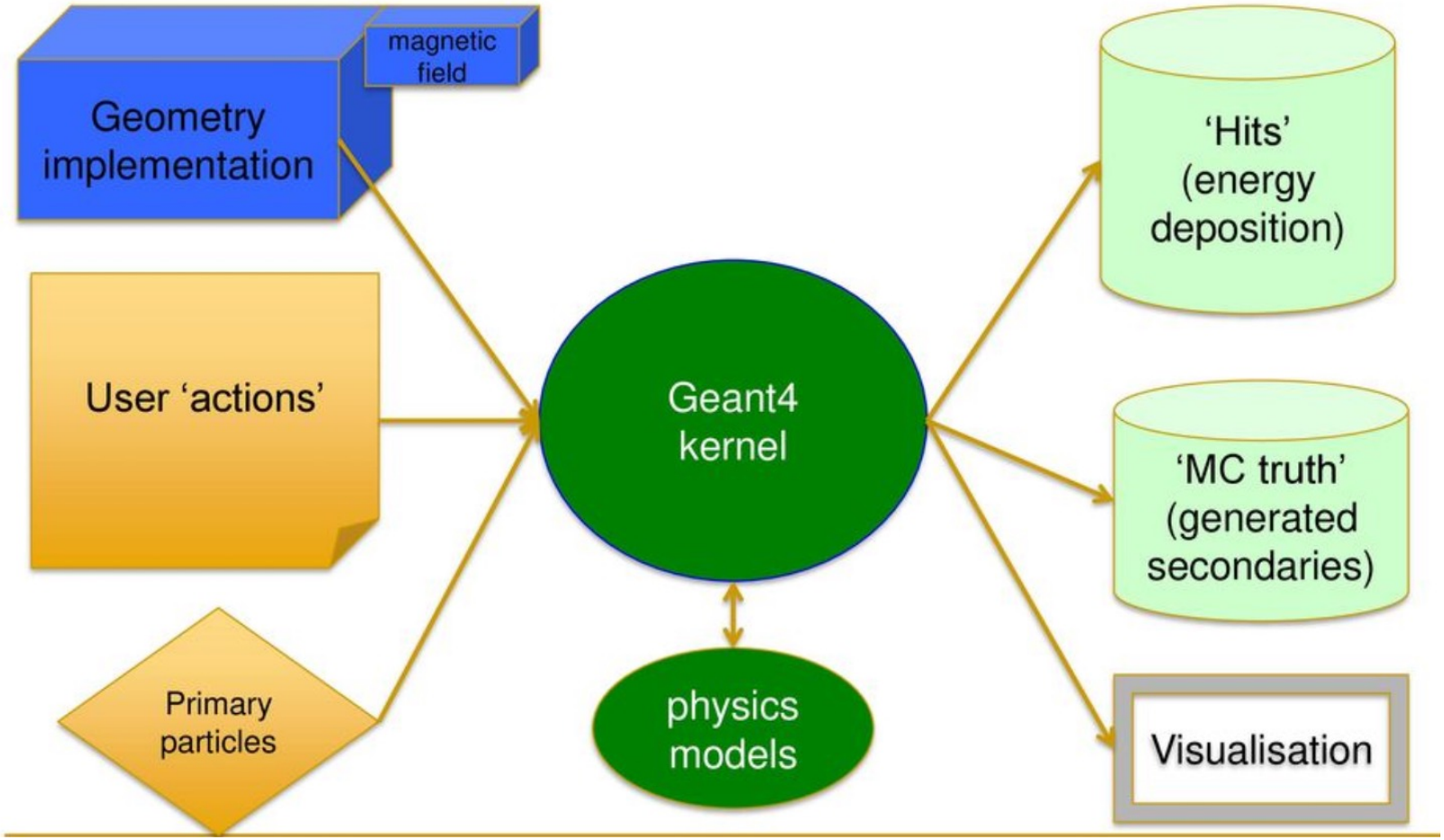
- Nuclear physics
- Hadron therapy
- Radiobiology
- Positron Emission Tomography (PET)
- Spacecraft shielding
- Radiation damage
- Single event effects (SEE) in electronics
- Neutron spallation targets
- ALL types of detector development
- Many others







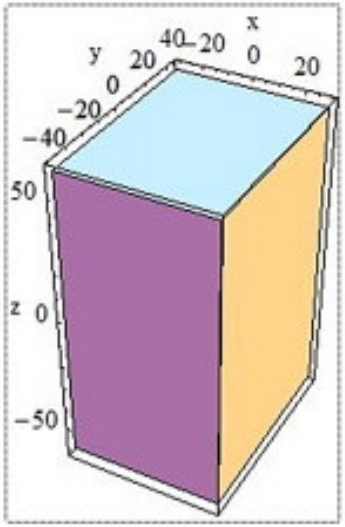
General Structure of Geant4

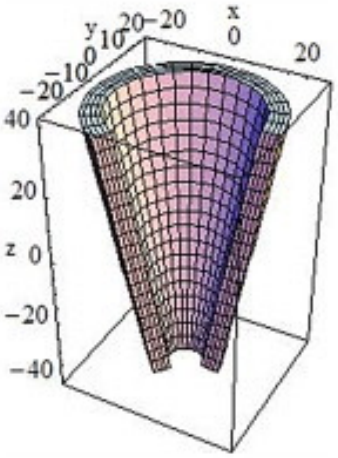


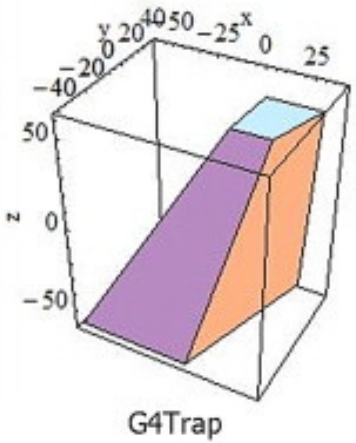
- Handles for user customization at every level!

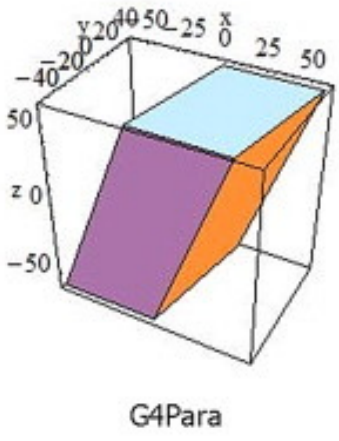
Geometry in Geant4

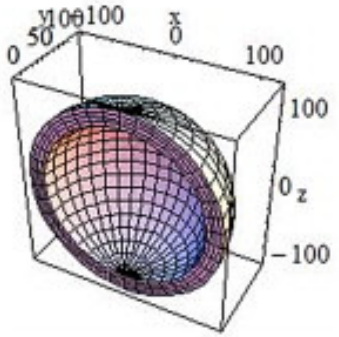
- Geometry is built using a variety of “primitives”

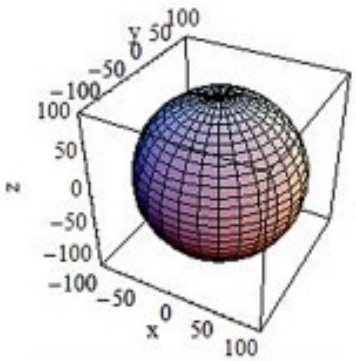


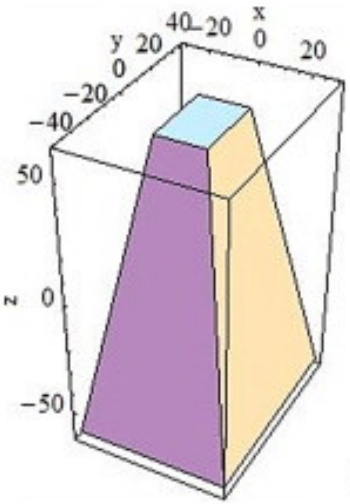


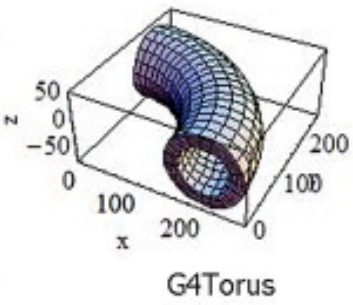






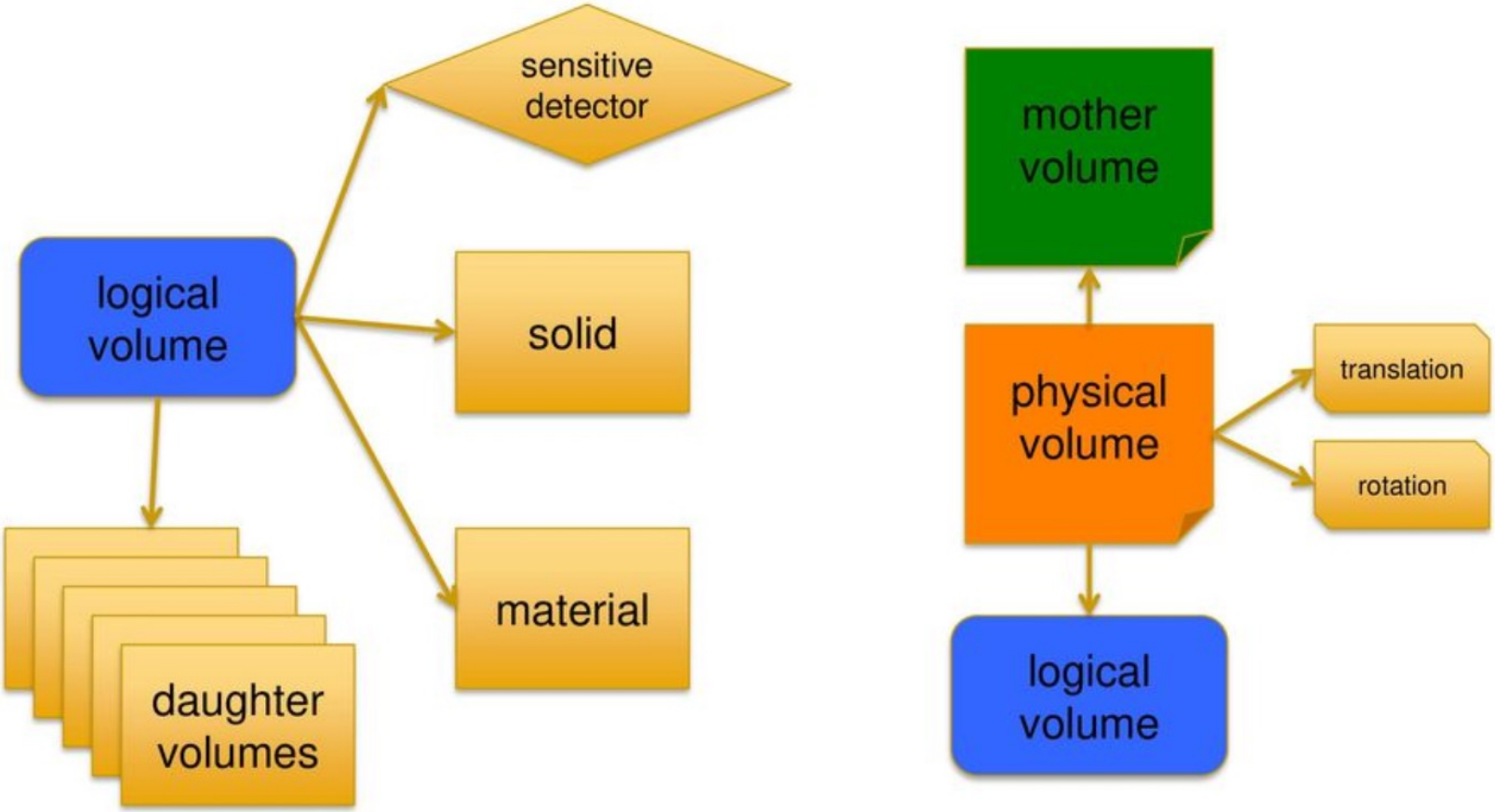






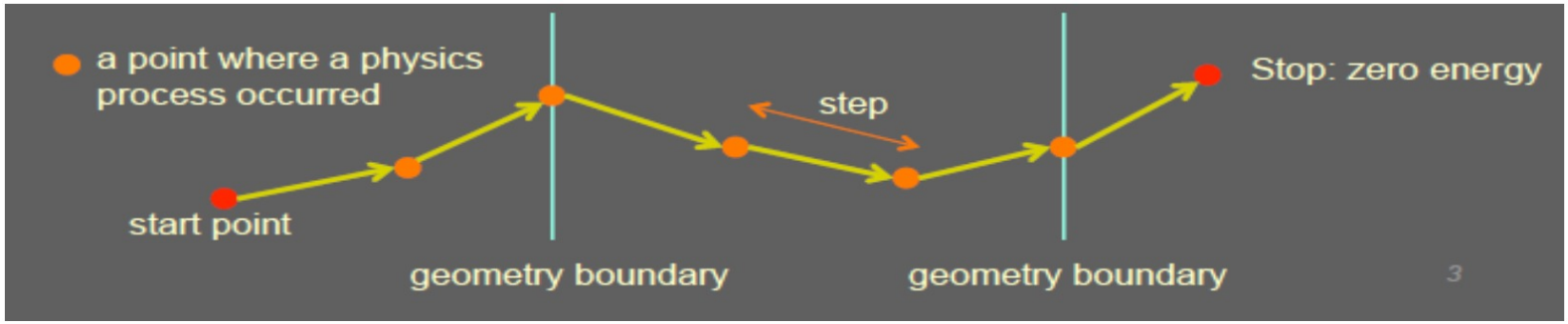
Geometry (cont'd)

- Volumes can be translated, rotated, embedded, etc

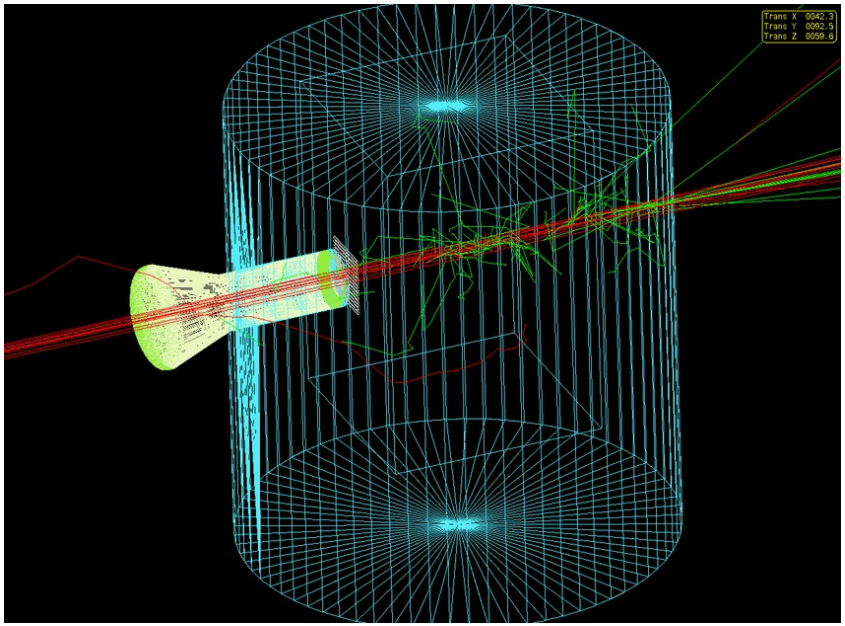


Tracking in Geant4

- Particles are “stepped” through the material, depositing energy along the way

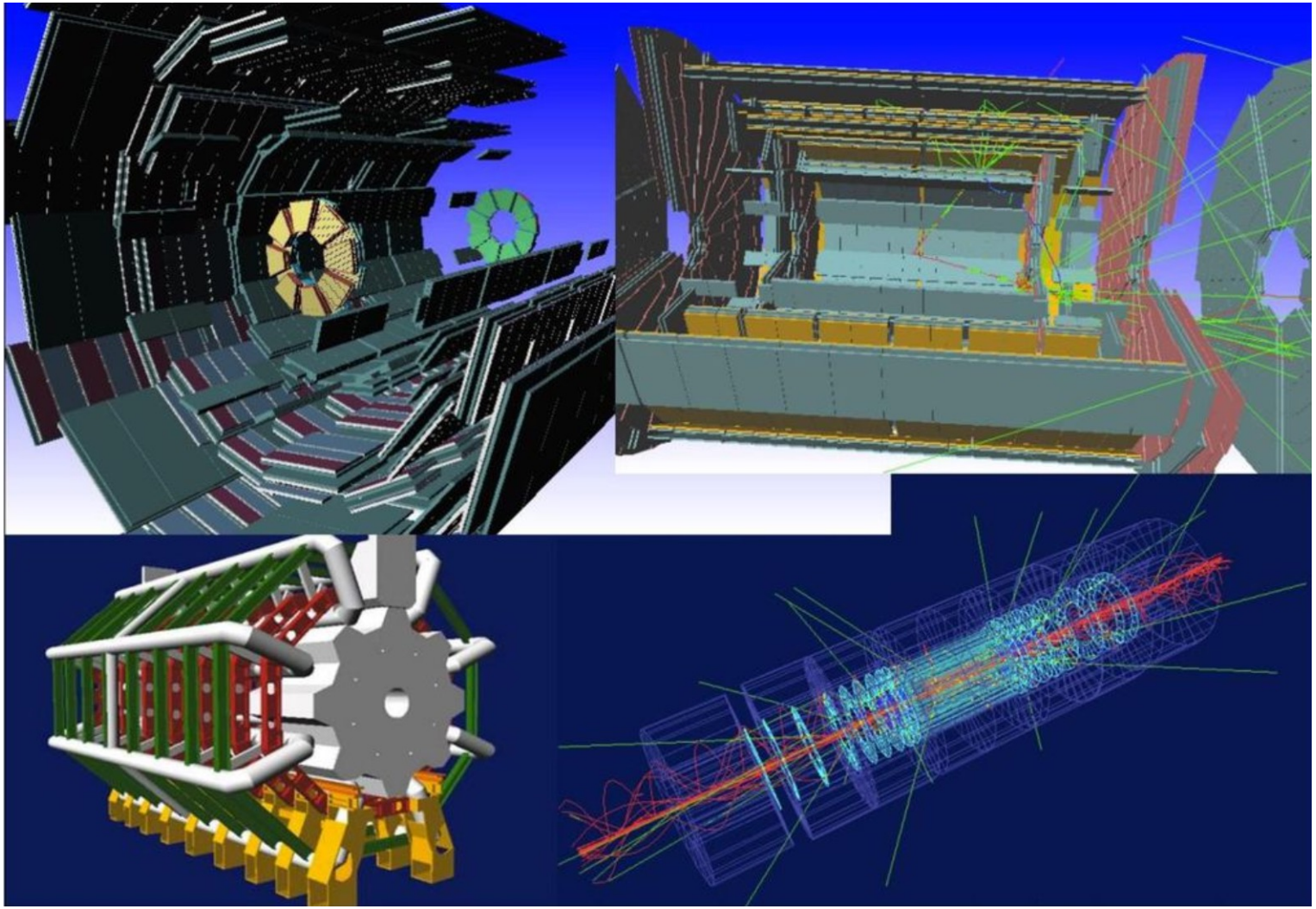


  - If it’s a sensitive volume the deposited energy is “scored”
- Secondaries:
  - Secondaries can be created through decay or interaction
  - They are also tracked, down to a specified cutoff energy



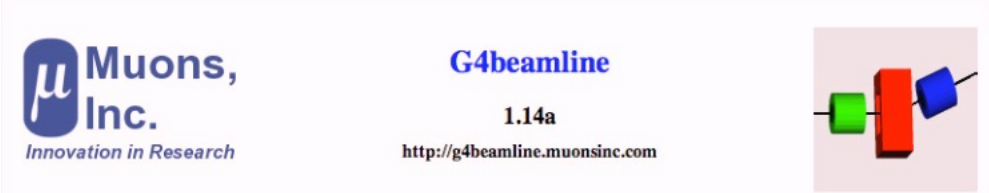
Example: ATLAS at LHC

- Done with full Geant4



Geant4 is Horrible!

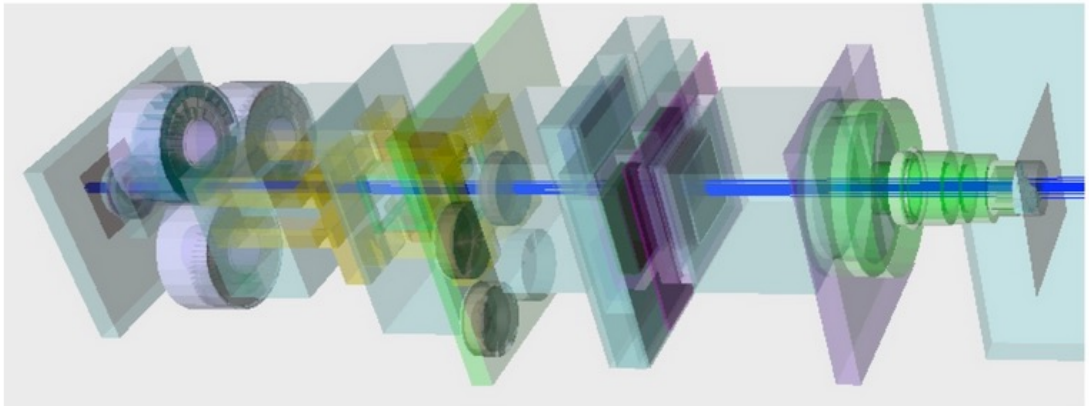
- Geant4 is kind of a nightmare to use.
- Even the simplest programs are quite complex to set up.
- For this reason, numerous scripted/GUI “wrapper” programs had been written to ease the pain of using it. Examples:
  - g4beamline: beamline (sort of) and detector modeling

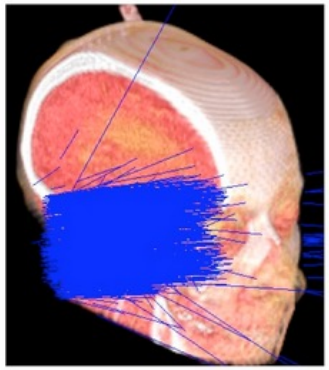


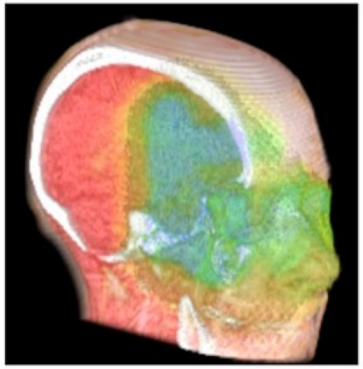
  - BDsim: accelerator modeling
  - TOPAS: medical physics

Example: Proton Therapy

- Implemented in TOPAS

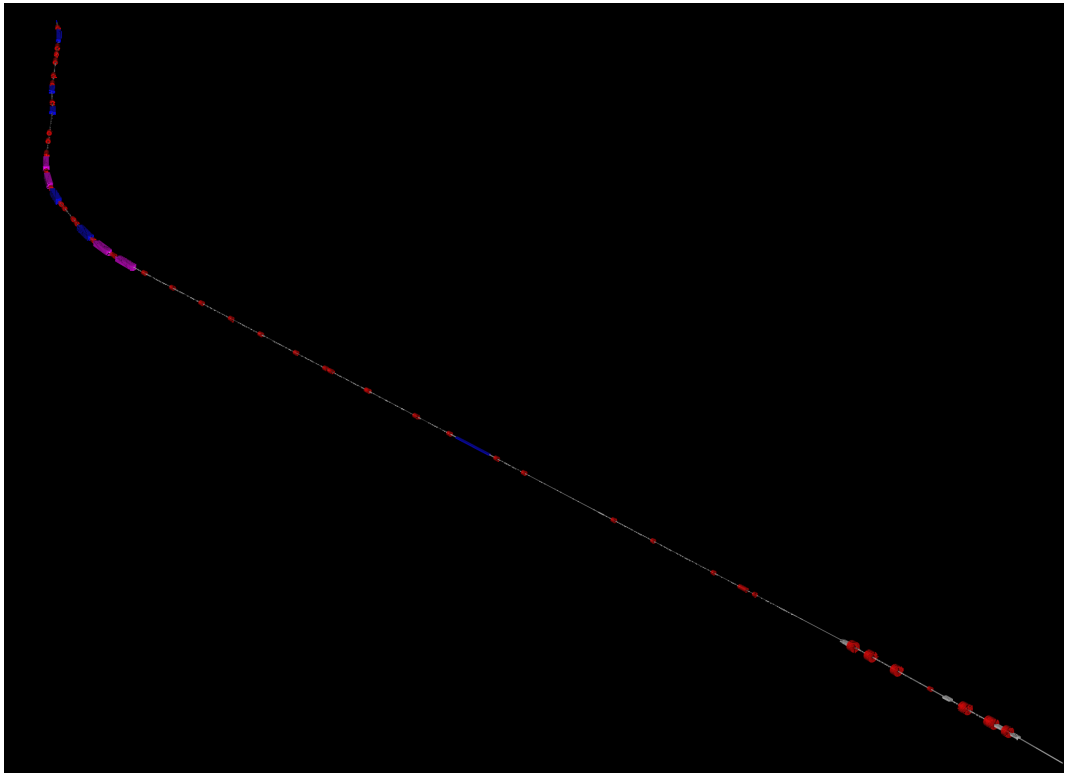


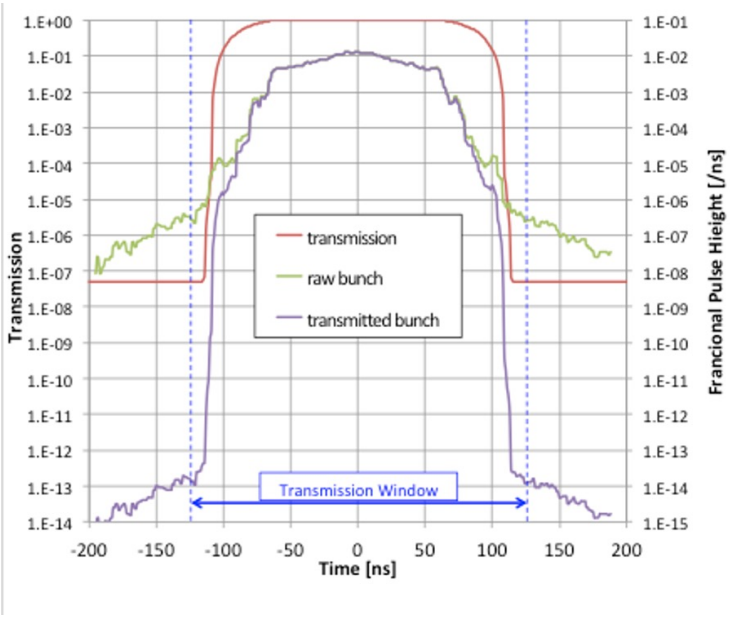




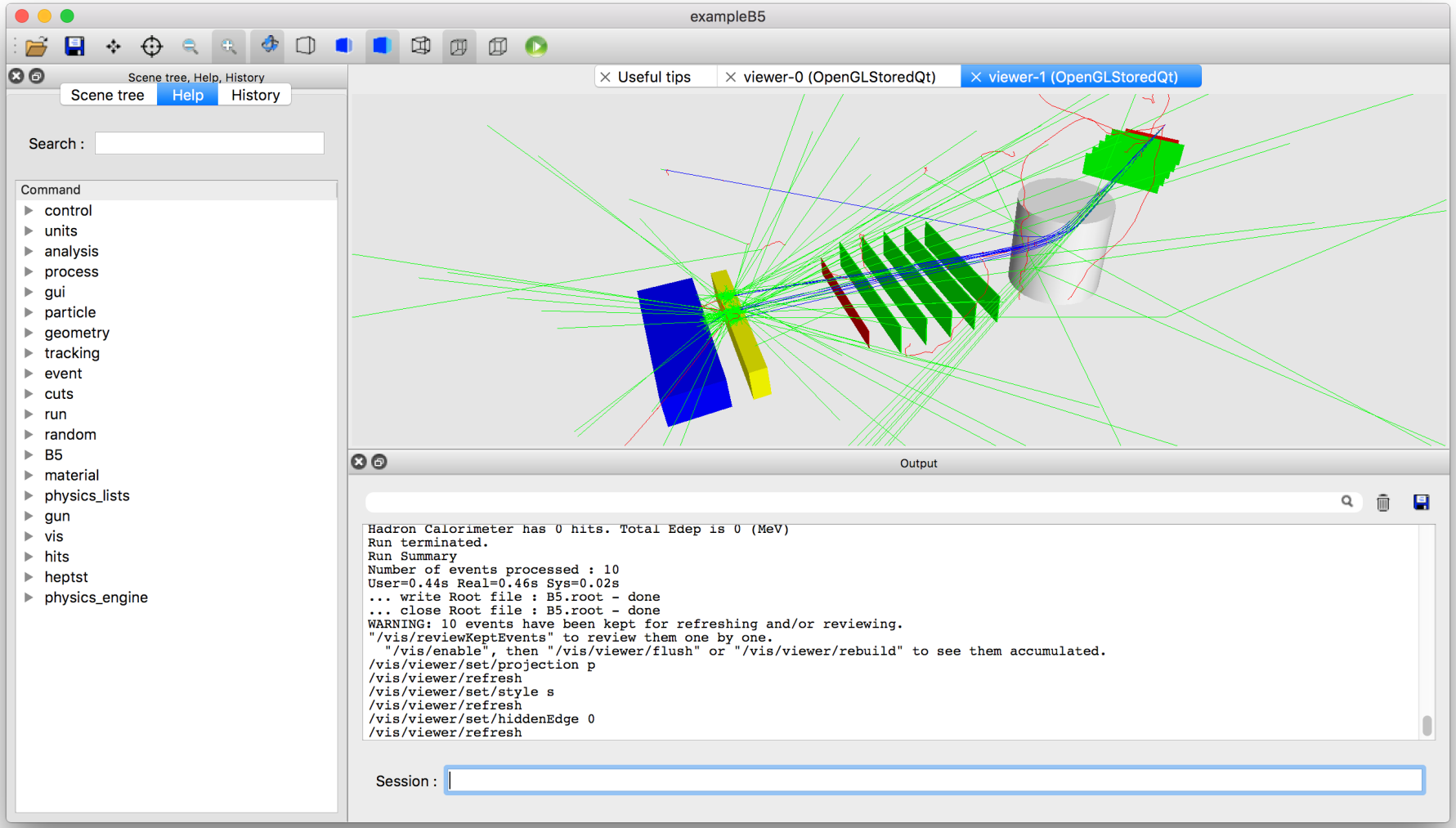
Example: Mu2e Beamline

- Implemented with g4beamline
  - Used to model “extinction” of out-of-time beam





Example: Particle Scattering with BDsim



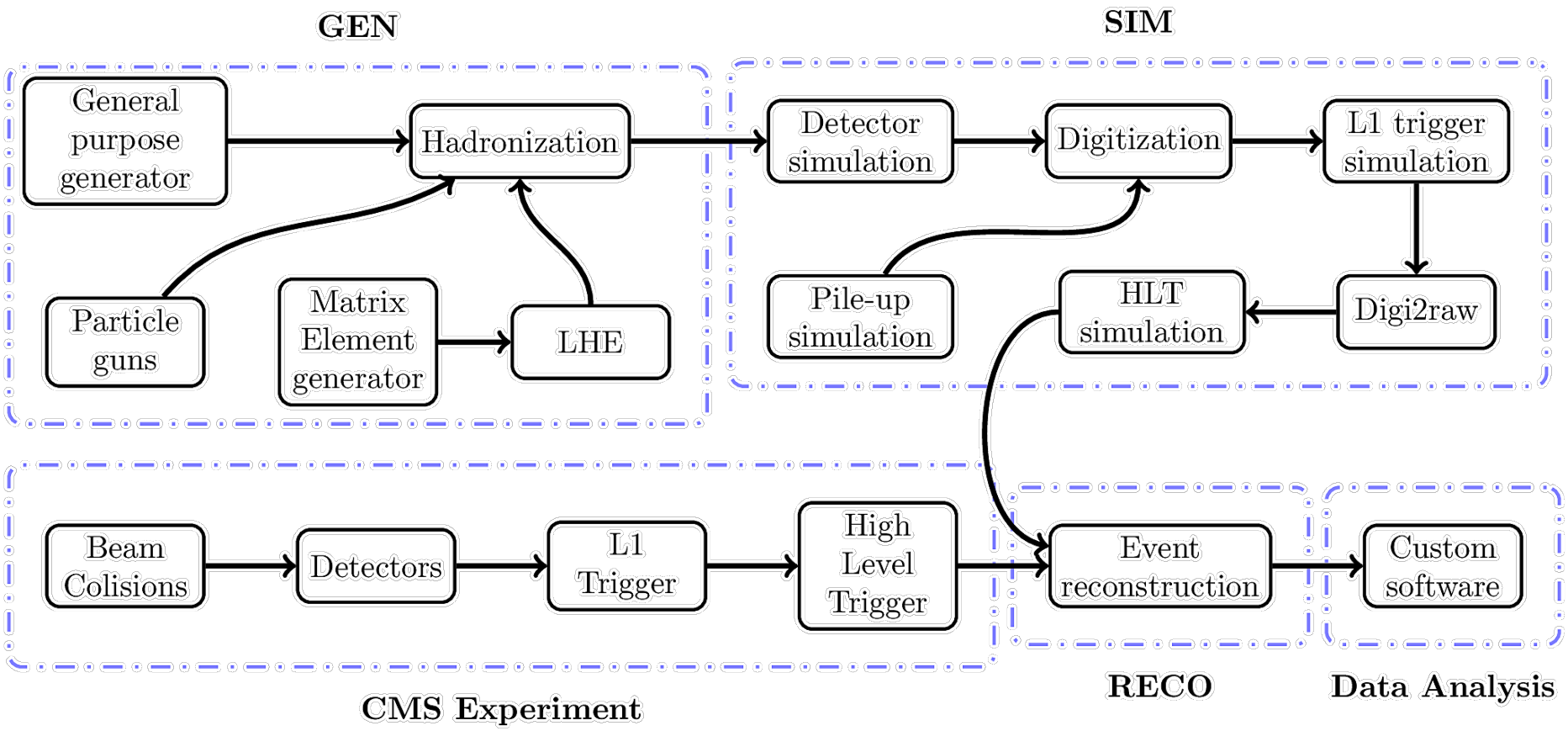
Types of Monte Carlo Studies

- There are a range of Monte Carlo studies, going from…
  - "Fast" or "Toy"
    - Underlying physical processes are simulated
    - Detected and/or reconstructed data are dropped, misclassified, or "smeared" according to measured or anticipated acceptance, confusion matrices, and resolution.
  - "Full" or "hits based"
    - Physical processes are simulated.
    - These are used to generate the lowest level *signals* in the detector
      - Tracker hits
      - Calorimeter signals
      - Cherenkov light
      - Etc
    - Noise hits are added if appropriate
    - These are then reconstructed an analyzed in *exactly the same way as the real data*.
      - The only difference is that you know the "true" answer
- Fast Monte Carlo methods are appropriate for conceptual analyses, but full Monte Carlo studies are required for the release of any physics results.

Computing Requirements

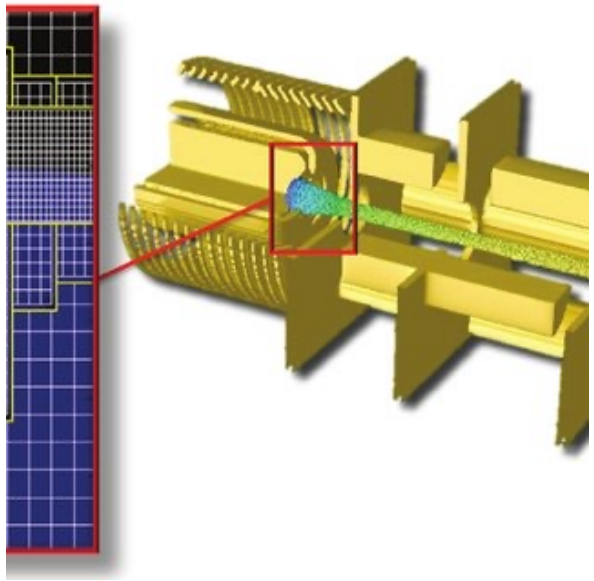
- A single Monte Carlo event requires *more* computing time than a real event
  - The hits take the same amount of time to reconstruct and analyze, but you have to add the time it takes to create them.
- There will always be systematic uncertainties on the physics of simulations, but we would prefer that Monte Carlo statistics to not add significantly to our overall uncertainty
  - We need at least as many Monte Carlo events as real events
  - Preferably more
- Monte Carlo production is a major part of any physics analysis

Example: CMS Monte Carlo Overview



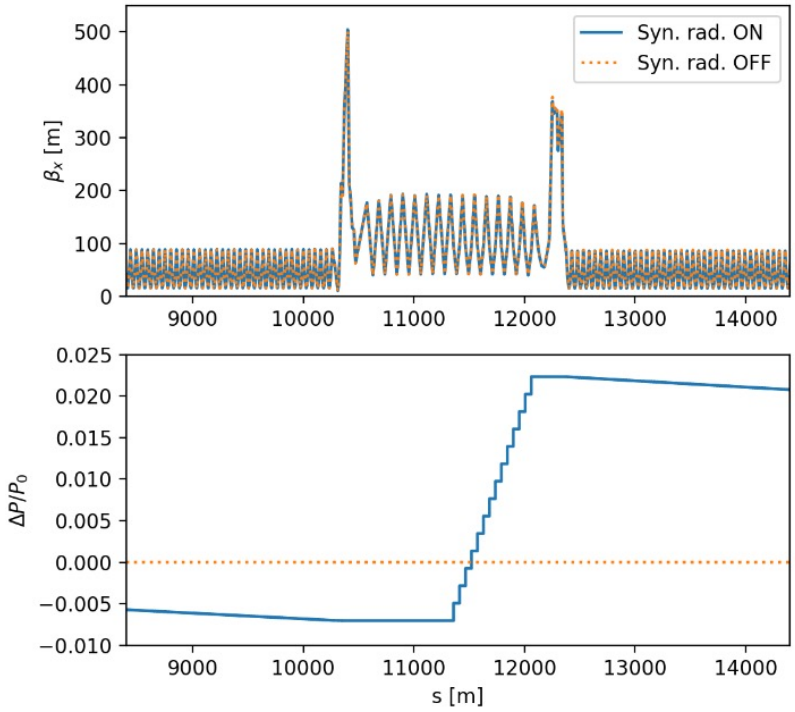
Other Monte Carlo Tools: Accelerator Physics

- Historically, accelerator modeling tools are calculational rather than Monte Carlo
  - MADX
  - TRANSPORT
  - Treated particles within bunches are *independent* of one another
  - Did not consider interaction with environment (beam pipe, magnets, etc)
- Modern improvements fall into two categories
  - Interactions with the environment
    - MARS or Geant4-based
  - Interactions with each other (i.e. “space charge”. VERY compute-intensive)
    - WARP
    - OPAL
    - Synergia
    - BLoND
    - edt



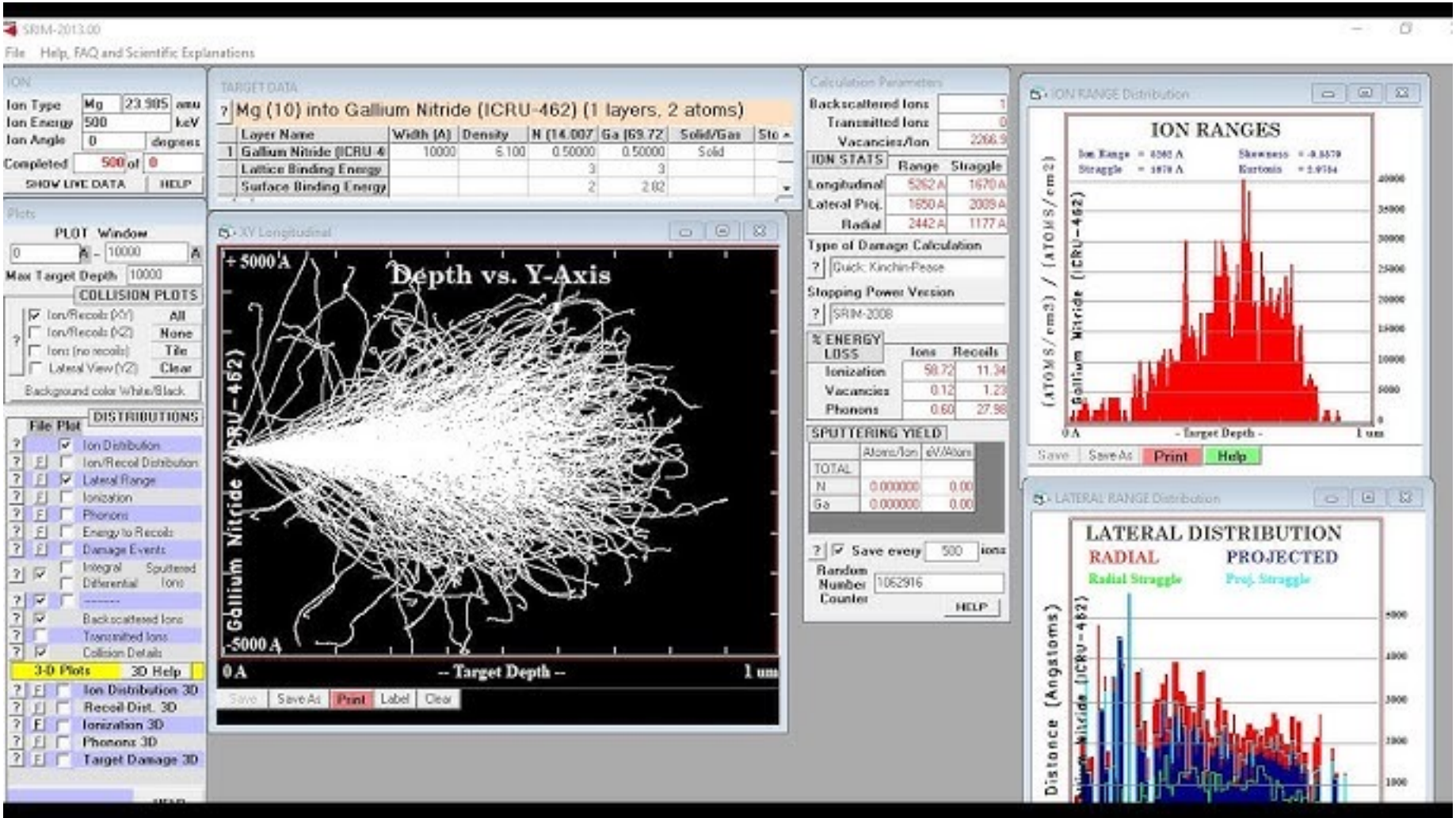
XSuite

- XSuite is a new set of packages for accelerator modeling
- Has now become the standard tool for LHC modeling at CERN.
- User interface is Python based.
  - Xsuite is now part of the standard Python distribution.
  - Install with “pip install xsuite”

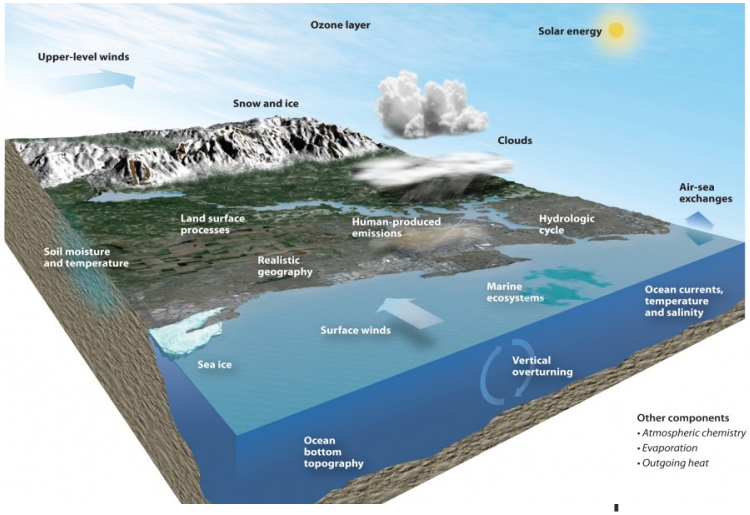


Radiation Effects

- Stopping Range of Ions in Matter (SRIM) is the industry standard



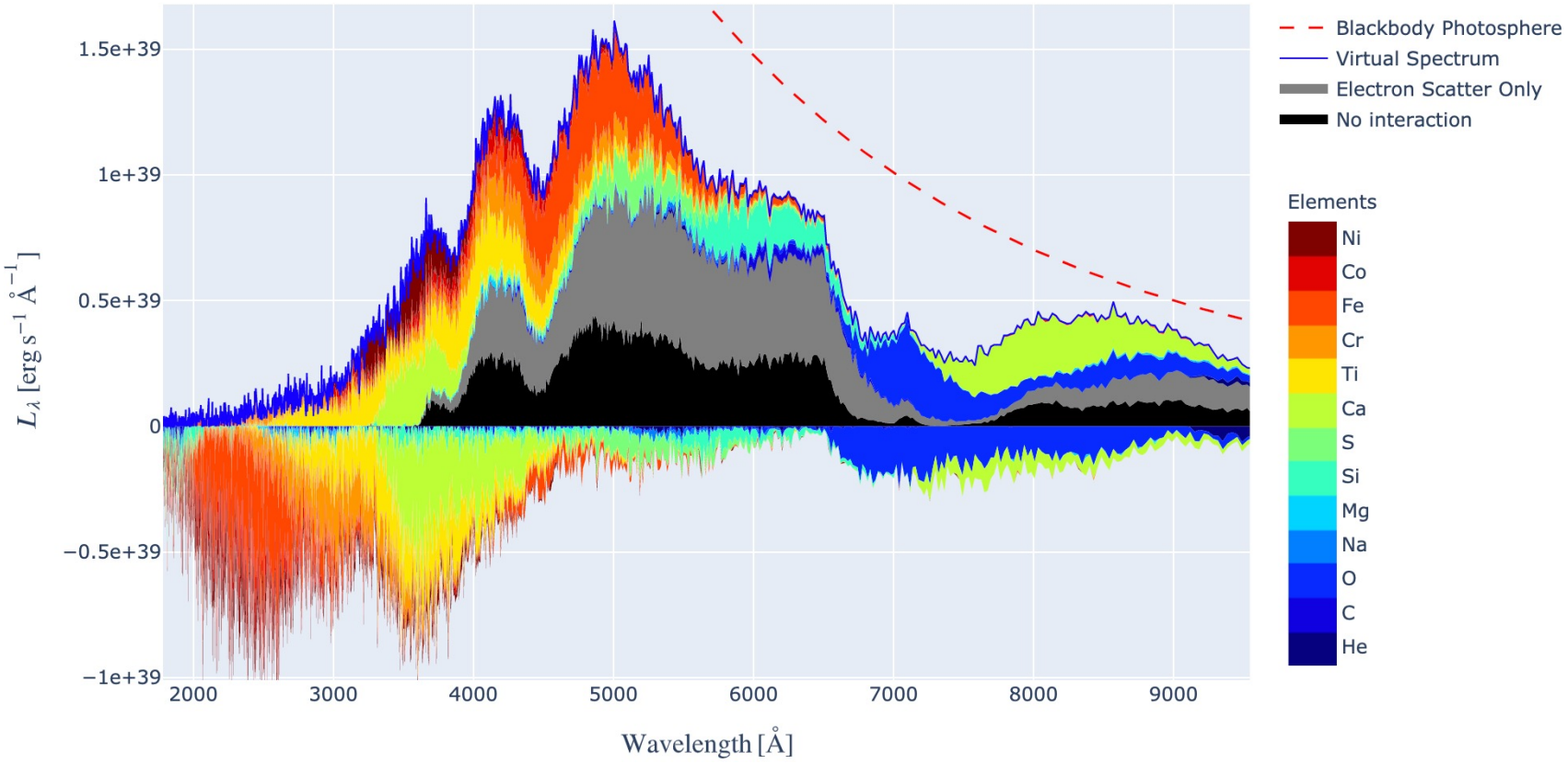
Climate Science



- **CESM (Community Earth System Model):** CESM is a fully-coupled climate model | developed by the National Center for Atmospheric Research (NCAR). It includes components for the atmosphere, ocean, land surface, and sea ice. CESM supports Monte Carlo simulations for uncertainty analysis in climate projections.
- **CLM (Community Land Model):** CLM is a widely by the National Center for Atmospheric Research (NCAR). It's often coupled with atmospheric models and supports Monte Carlo simulations for uncertainty analysis.
- **WRF (Weather Research and Forecasting Model):** WRF is a widely used mesoscale numerical weather prediction system. It has capabilities for conducting Monte Carlo simulations to assess uncertainty in weather forecasts and climate projections.

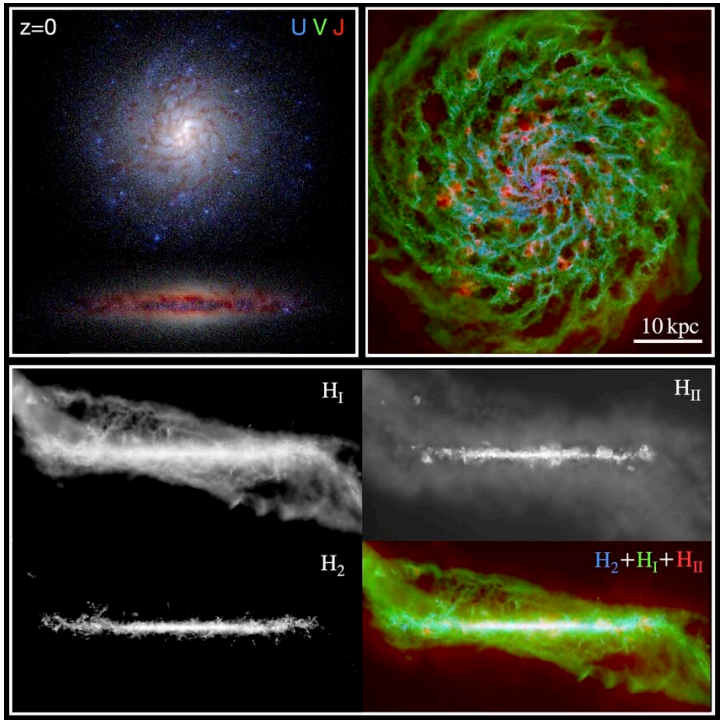
Supernova Simulations

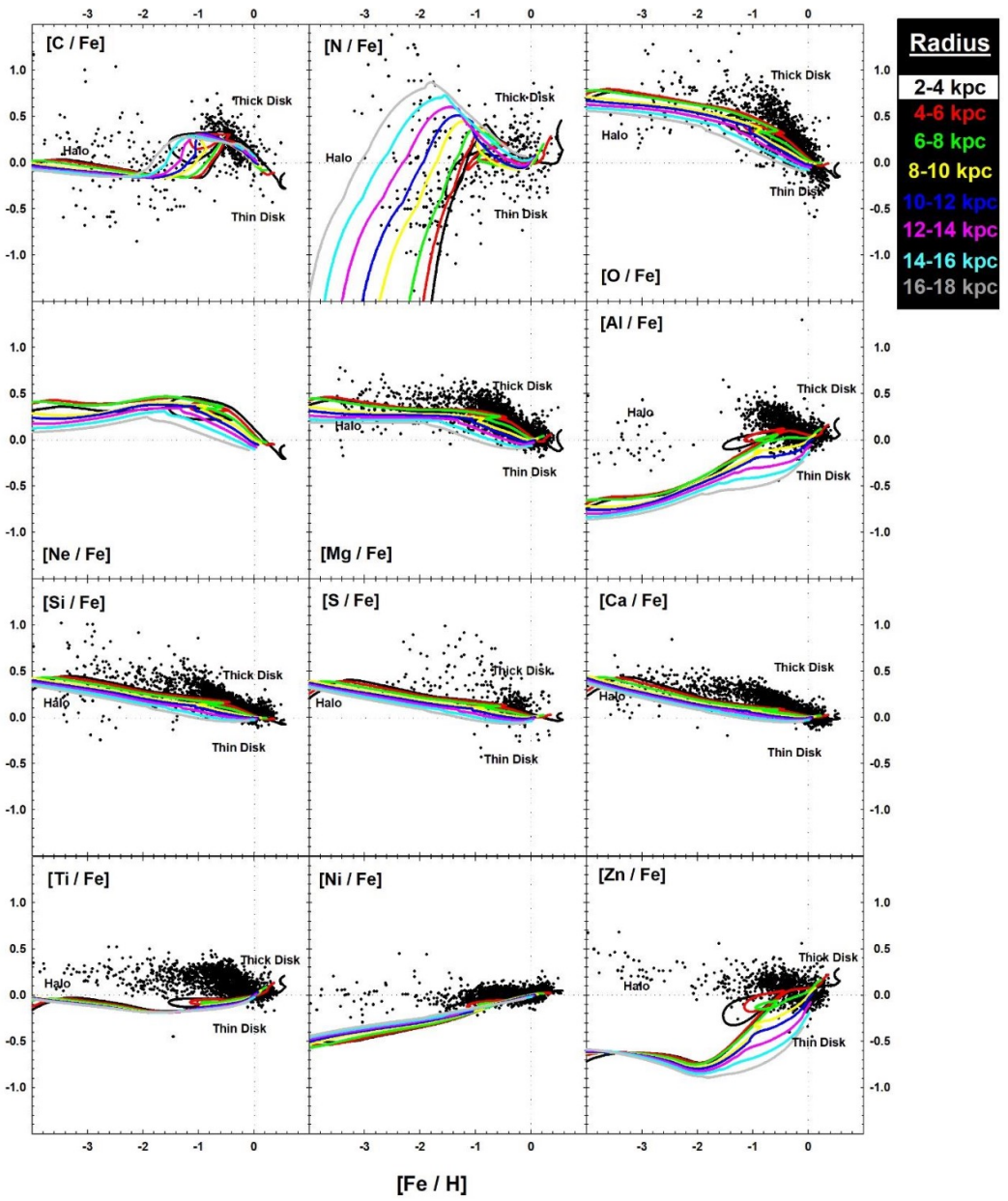
- Several, including TARDIS



Galaxy Formation

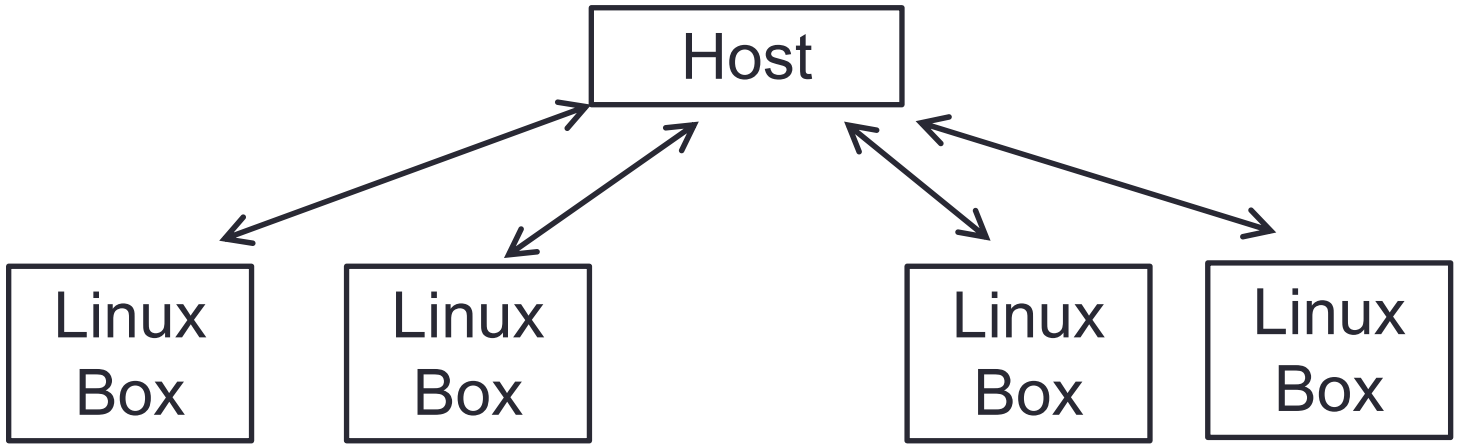
- MANY, incuding
  - GALFORM
  - GADGET
  - RAMSES
  - MUGS
  - Meraxes





Parallel Processing in Monte Carlo Studies (high level)

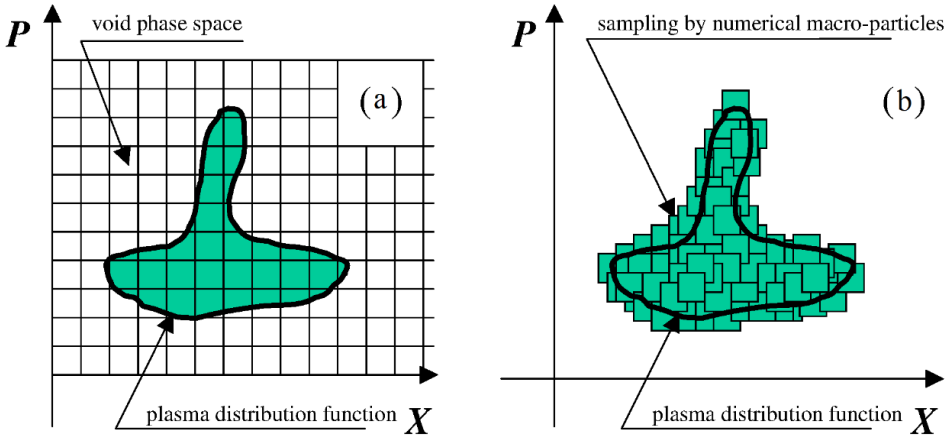
- Because each event is independent, particle physics benefits from "high level parallelism", both for analysis and simulation.
  - Each processing node handles one complete event at a time, and the results are transmitted to a host, which collects them.
  - This was well suited to the advent if inexpensive Unix processors, which were developed for the computer graphics and animation industry around 1990 (Silicon Graphics, HP, Sun, etc)
  - A few year later, Linux made "commodity computing" dirt cheap, and that's how everything is done now.



Low Level Parallel Processing

- A number of challenging simulations (space charge, weather, hydrodynamics, galaxy formation, etc) involve the interaction of individual elements.
  - e.g. there are over 10^{11} proton in each bunch in the LHC at CERN
- Simulating the individual interactions would be beyond any conceivable computing power in the foreseeable future.
- Monte Carlo techniques generally involve “Particle in Cell” (PIC) approaches, in which the aggregate is divided into “cells”, each containing many individual elements (particles, atoms, planets, etc)

Example: plasma simulation



- The problem is now essentially solving a nonlinear differential equation
  - Which is also very hard!

Parallel Processing Applied to PIC

- Generally each processor is assigned a cell or cells
- The problem is that unlike the particle physics example, cells are far from independent. Information is exchanged among cells *throughout the simulation process*
  - The linux farm model would be completely useless.
- Parallelism is achieved through a combination of
  - Multiple cores
  - shared memory
  - accelerating co-processors (GPUs, TPUs, etc)
- This is a growth industry!