Progress update — Jiangmei Yang

Remaining updates on Tuesday or Thursday

- Email me your preference; I may randomize

Simulation

Progress update — Jiangmei Yang

Remaining updates on Tuesday or Thursday

- Email me your preference; I may randomize

Simulation:

- Just a taste of the ideas
- No homework, but I will post code
- Worth playing with over break to add to tool belt

Numeric simulations are a common tool in solving real problems, because…

Real problems are not simple enough for analytic solutions.
- Perturbative expansions are conceptually useful
- But you don't always have small parameters

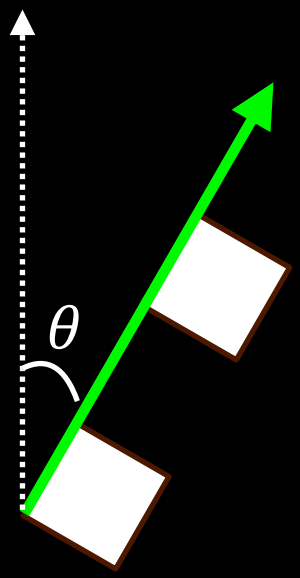
Real problems involve multiple phenomena merged together
- Convolving them further complicates analytic solutions.

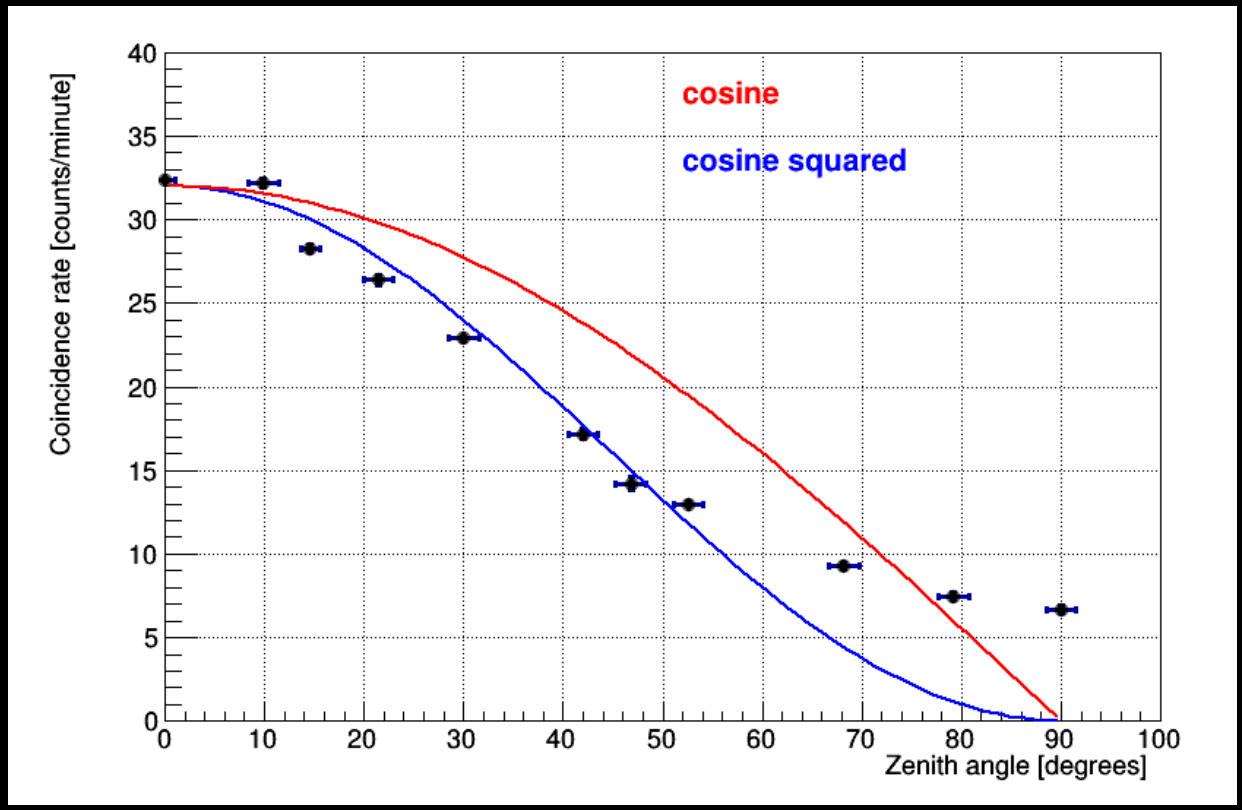
Real problems often involve "random" processes
- Where "random" means unpredictable except for μ and σ.

Real problems often have multiple unknowns
- Solve for x, y, z, and λ simultaneously.

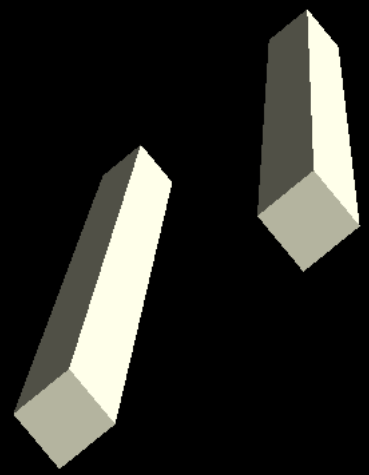
We can solve problems like these by collecting data and fitting it to a model that incorporates all known effects via a numerical simulation.
E.g., climate models, galaxy formation models, cosmology models.

An example related to this class is the zenith angle measurement





But the geometry is really 3D but not spherically symmetric.
A 2D cosine squared is too simplistic.
There is a spread of trajectories that would give coincidence.
Multiple particles could cause coincidence events.
Secondary interactions could cause coincidence from 1 particle
I can test all of these with a simulation…

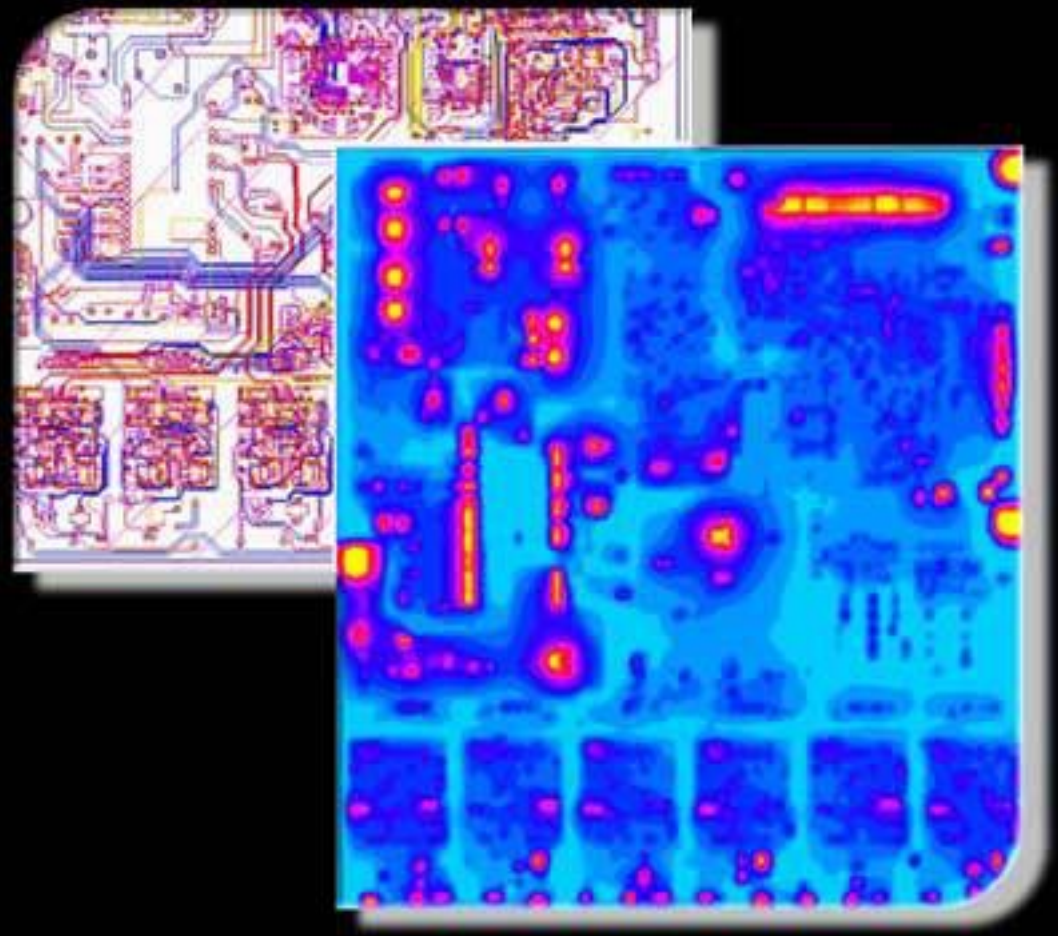


There are two common approaches:

Discrete steps, aka finite elements

Stochastic simulations, called Monte Carlo methods

FEA = finite element analysis is a tool to solve for thermal or mechanical equilibrium when only the nearest neighbor dynamics can be predicted.



MC methods use random sampling of distributions to produce simulated data sets to compare to measured data sets. Useful for validating full chain of analysis.

I will show an example of each.

Discrete steps to solve a differential equation:

Lagrangian mechanics gives you the differential equations of motion, but they are often very difficult to solve analytically without approximations.

For example, a simple pendulum gives $\ddot{\theta} + \frac{g}{L} \sin\theta = 0$

In the small angle approximation it is just SHM. $\ddot{\theta} + \frac{g}{L}\, \theta = 0$

Discrete steps to solve a differential equation:

Lagrangian mechanics gives you the differential equations of motion, but they are often very difficult to solve analytically without approximations.

For example, a simple pendulum gives $\ddot{\theta} + \frac{g}{L}\sin\theta = 0$

In the small angle approximation it is just SHM. $\ddot{\theta} + \frac{g}{L}\,\theta = 0$

But a real pendulum also has drag. $\ddot{\theta} + \frac{b}{m}\,\dot{\theta} + \frac{g}{L}\sin\theta = 0$

Solve that!

Discrete steps to solve a differential equation:

Lagrangian mechanics gives you the differential equations of motion, but they are often very difficult to solve analytically without approximations.

For example, a simple pendulum gives $\ddot{\theta} + \frac{g}{L}\sin\theta = 0$

In the small angle approximation it is just SHM. $\ddot{\theta} + \frac{g}{L}\,\theta = 0$

But a real pendulum also has drag. $\ddot{\theta} + \frac{b}{m}\,\dot{\theta} + \frac{g}{L}\sin\theta = 0$

We can solve that with a simple numerical calculation!

Just use $\dot{\theta} \approx \frac{\Delta\theta}{\Delta t}$ and $\ddot{\theta} \approx \frac{\Delta\dot{\theta}}{\Delta t}$

We can solve that with a simple numerical calculation!

$$\ddot{\theta} = -\frac{b}{m}\,\dot{\theta} - \frac{g}{L}\sin\theta$$

Algorithm is:

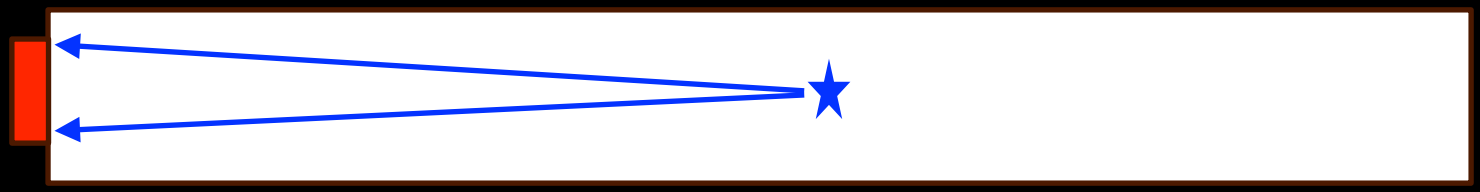
1). Set theta and its velocity, $\dot{\theta}$, to the initial conditions
2). Calculate acceleration, i.e., $\ddot{\theta}$, from differential eom.
3). Update the velocity with $\Delta\dot{\theta} = \ddot{\theta}\ \Delta t$
4). Update theta with $\Delta\theta = \dot{\theta}\ \Delta t$
5). Loop back to (2)

Let's do this in a simple python program

Ray tracing

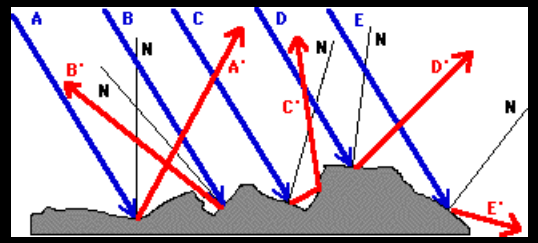
Another example of a discrete step method is ray tracing.

Light produced in the scintillator bars is produced isotropically.
What fraction hits the SiPM?
Some of it goes straight, we could calculate that from the SiPM's solid angle.



But this misses several effects:

- Light is produced along a path of excitation by the cosmic ray
- Some light is absorbed by the scintillator as it propagates
- Some light reflects from the back end
- Some light undergoes total internal reflection
  - But surface is not perfectly smooth so not a simple calculation
- Some light escapes and is reflected back in from wrapping.

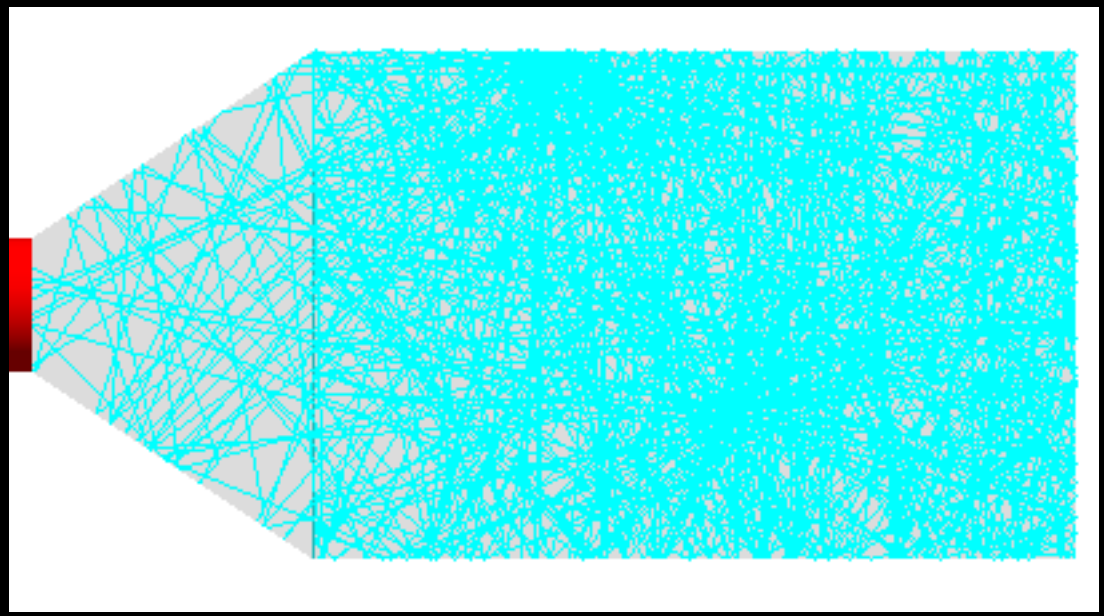


Ray tracing

We could track the path of each photon until it hits a surface and:

- Account for absorption as a random exponential process based on pathlength
- Determine total internal reflection, with a random effect for roughness
- Absorb or reflect escaped photons on the wrapping
- Repeat until the photon is absorbed.

Here is an example for a scintillator coupled to a light guide and a PMT on the end. A small fraction of the photons are detected.



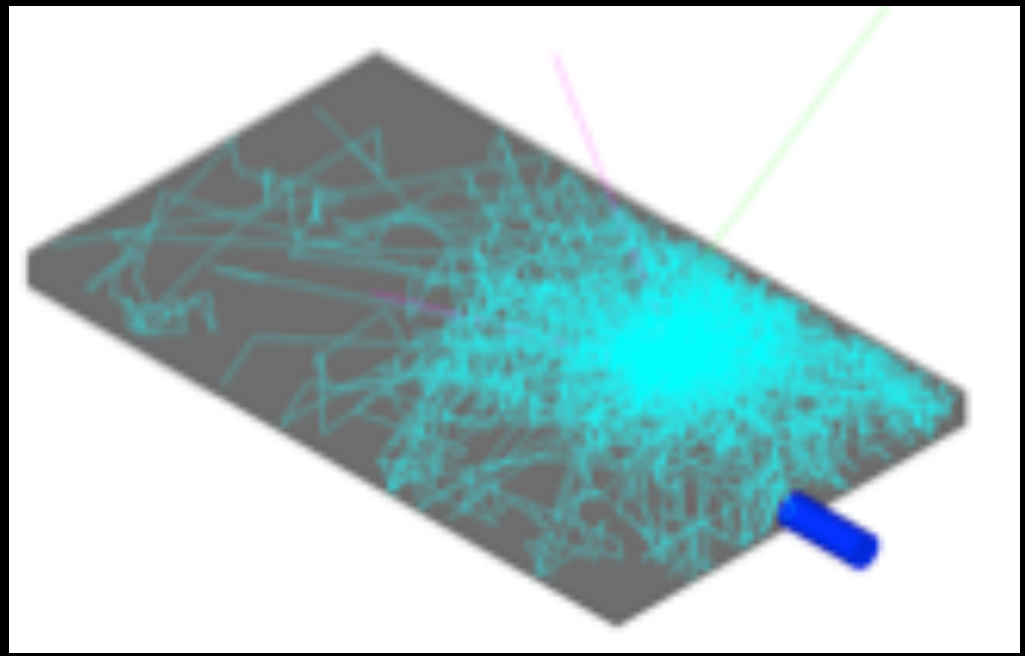
After simulating this for a large number of cosmic ray events, we could just use mean and RMS values for photon yield parameterized across the surface. This mixes discrete steps and Monte Carlo methods.

Ray tracing

We could track the path of each photon until it hits a surface and:

- Account for absorption as a random exponential process based on pathlength
- Determine total internal reflection, with a random effect for roughness
- Absorb or reflect escaped photons on the wrapping
- Repeat until the photon is absorbed.

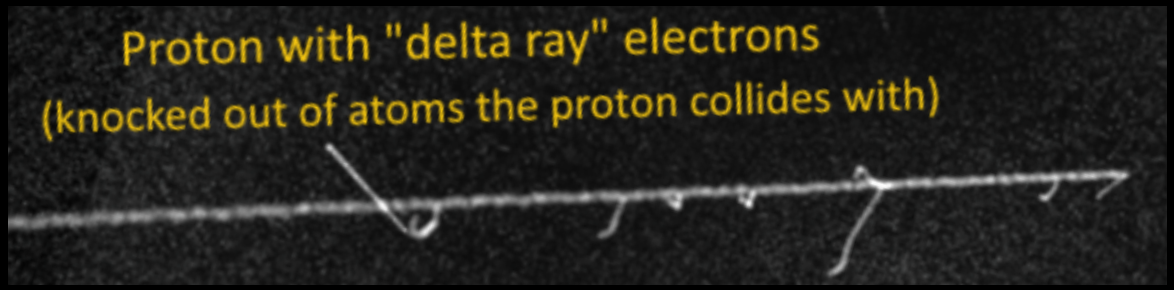
Here is an example for a scintillator coupled to a light guide and a PMT on the end. A small fraction of the photons are detected.



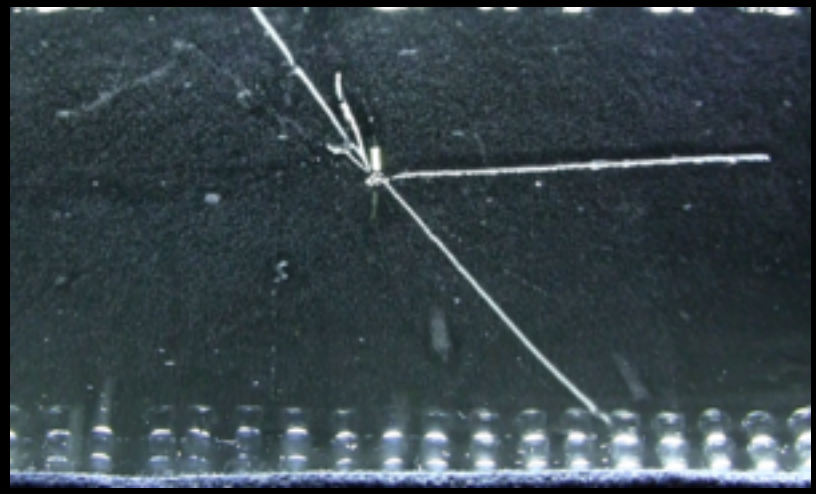
After simulating this for a large number of cosmic ray events, we could just use mean and RMS values for photon yield parameterized across the surface. This mixes discrete steps and Monte Carlo methods.

Full MC particle physics simulation packages include many more effects

Delta rays



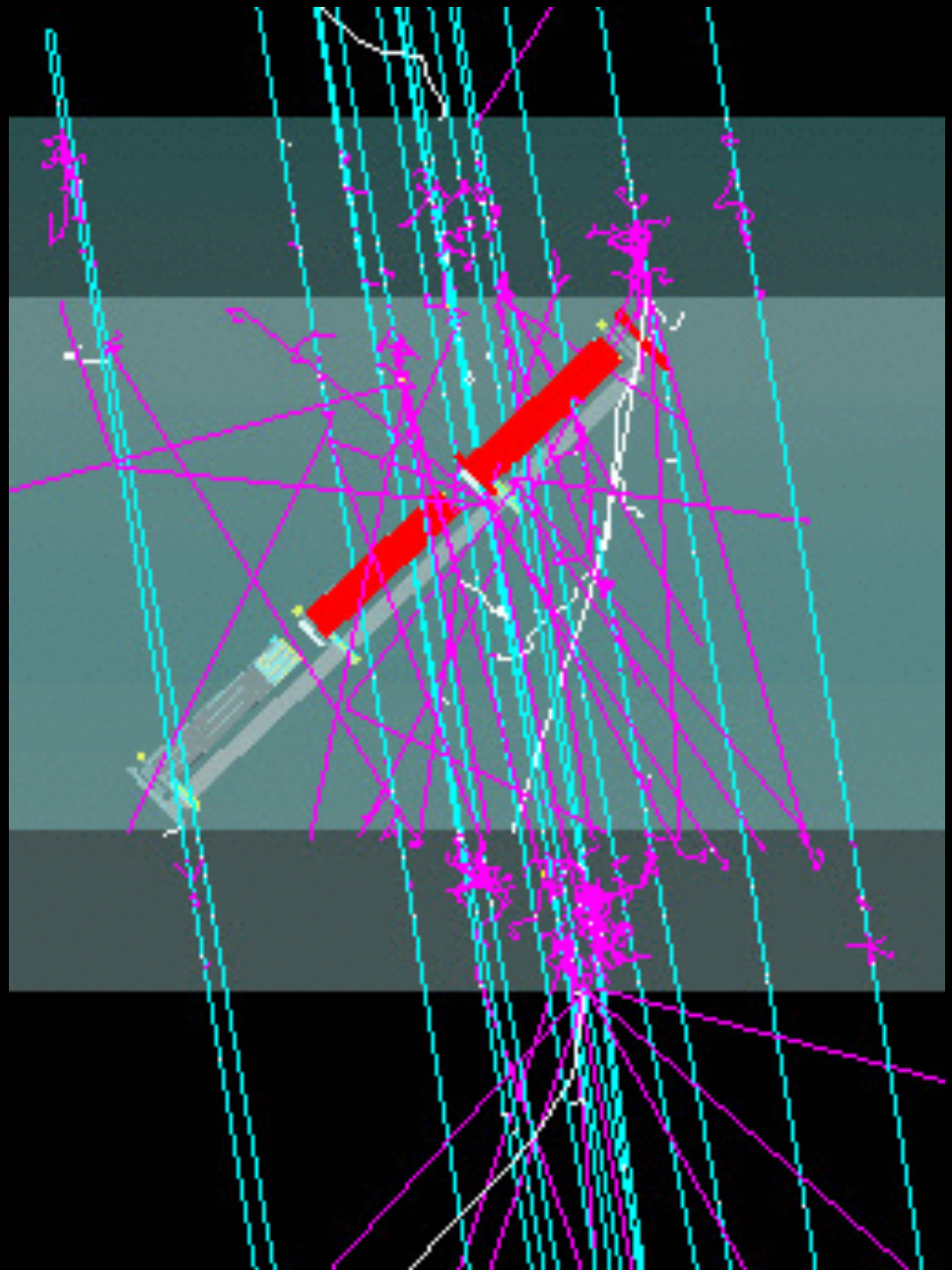
Nuclear interactions

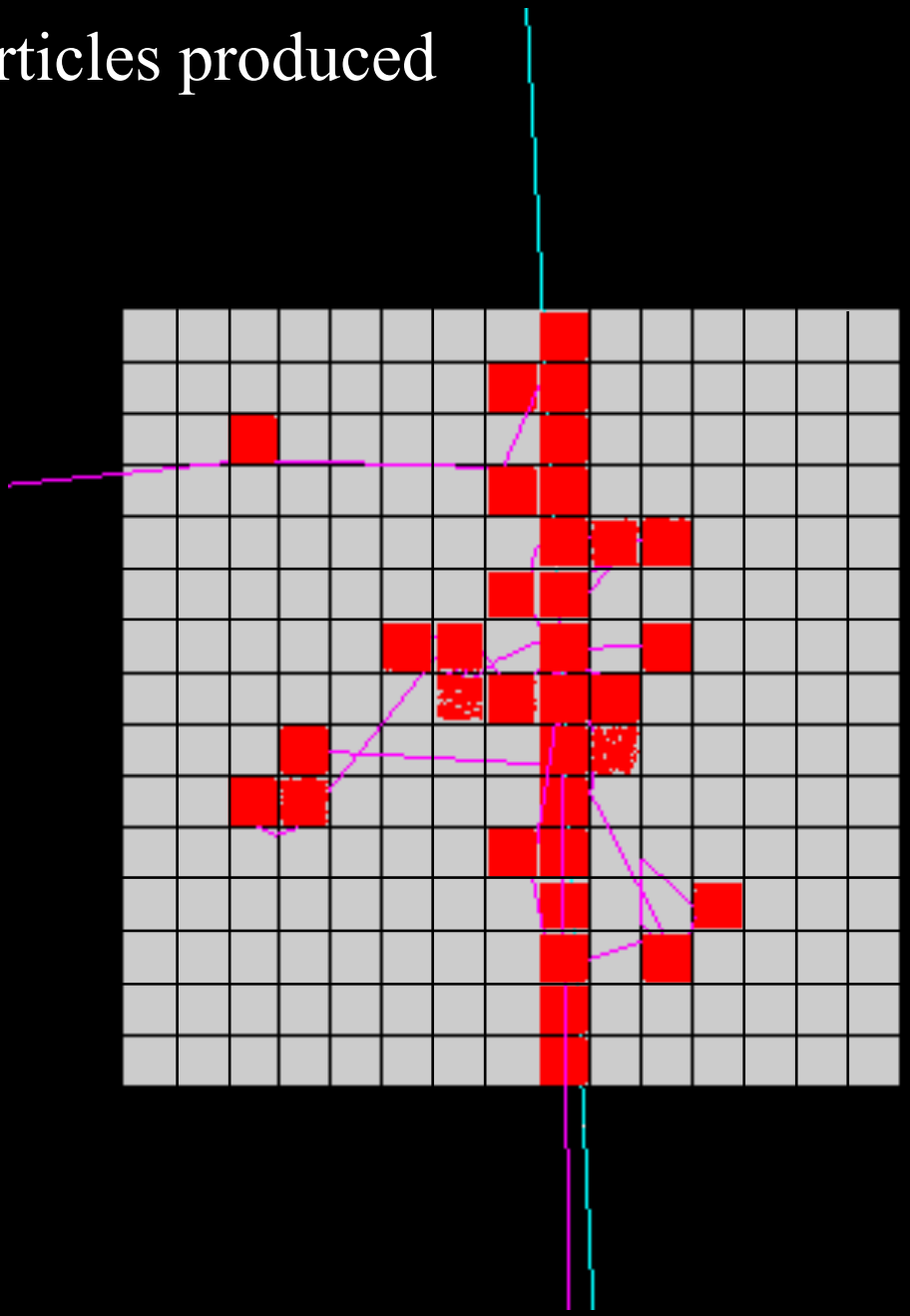


K-shell x-rays

Full simulation packages include many more effects

There are potentially many secondary particles produced



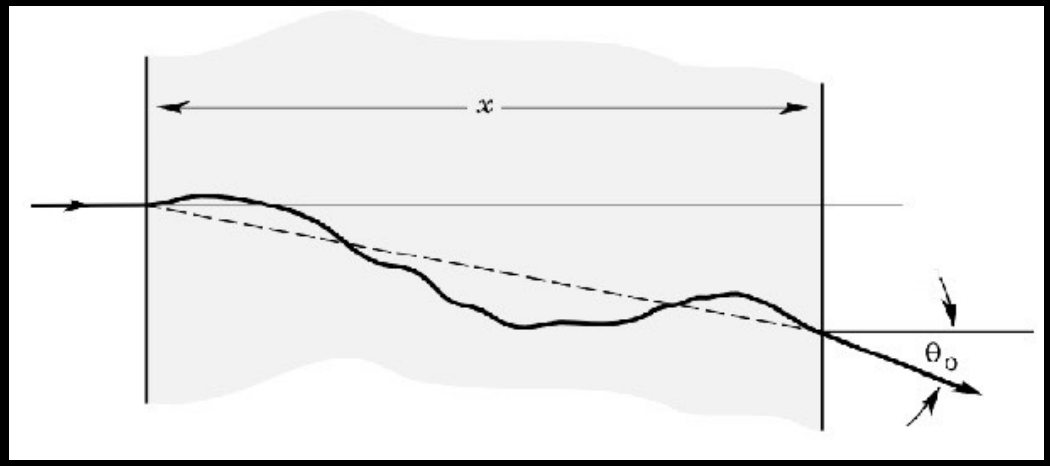


Cosmic ray tracing

So let's try to emulate the effect of bars in detecting a cosmic ray flux with some input angular distribution.

I'll use a simple, brute force approach.

Treat them as straight lines, ie ignore multiple scattering.



Avoid complicated geometry by taking small steps & sum path length
This would allow including multiple scattering later.

Cosmic ray tracing algorithm

Randomly set cosmic ray position and direction in a plane above detector.
Iteratively move it along its path in small steps.
Check if it is "in" a detector during this step, if so accumulate the pathlength.
Once it moves below a lower plane, record pathlengths for the 2 detectors.
If they are both above some minimum threshold, call it a coincidence event.

This allows us to rerun the simulation with different settings such as:
Vary the distribution of initial cosmic ray directions
Vary the thresholds for coincidence.
Add in multiple scattering
Add possibility of secondaries
Add possibility of two cosmic rays coming together.

And we can vary parameters for each of these until the simulation fits the data.