Imperial College London

Applications of Computational Physics

Patrick Dunne - Imperial College London

Overview

- Show applications of the techniques from lectures
- Focus on details of implementation in complex high energy physics research situations
  - Neutrino physics at T2K
  - Hadron collider physics at CMS

What do we mean by complex?

- Simple examples covered in lectures, e.g. minimising χ^2:

$$\chi^2(a,\alpha) = \sum_{i=1}^{N^{data}} \frac{\left(d_i^{obs} - d_i^{model}(a,\alpha)\right)^2}{\sigma_i^2}$$

- This has uncorrelated errors and a 2 paramter physics model
- Plausible that an analytic solution could be found quickly
- This will not be the case with applications shown today

T2K

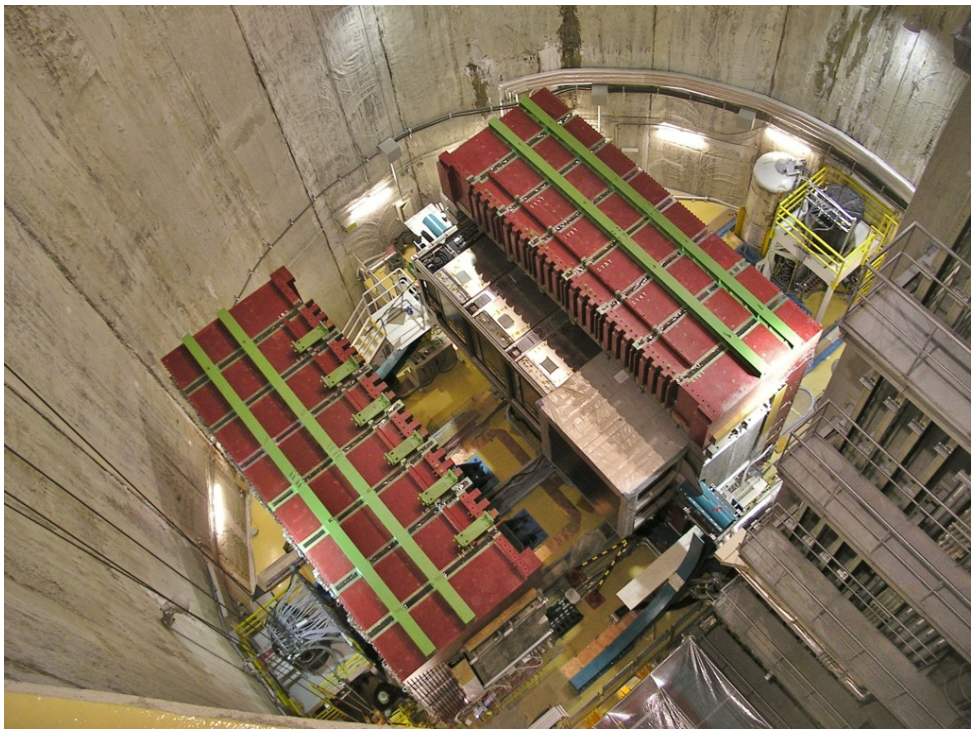
- Searching for neutrino oscillations in a $\nu_\mu/\bar{\nu}_\mu$ beam
- Looking for matter-antimatter asymmetry in lepton sector

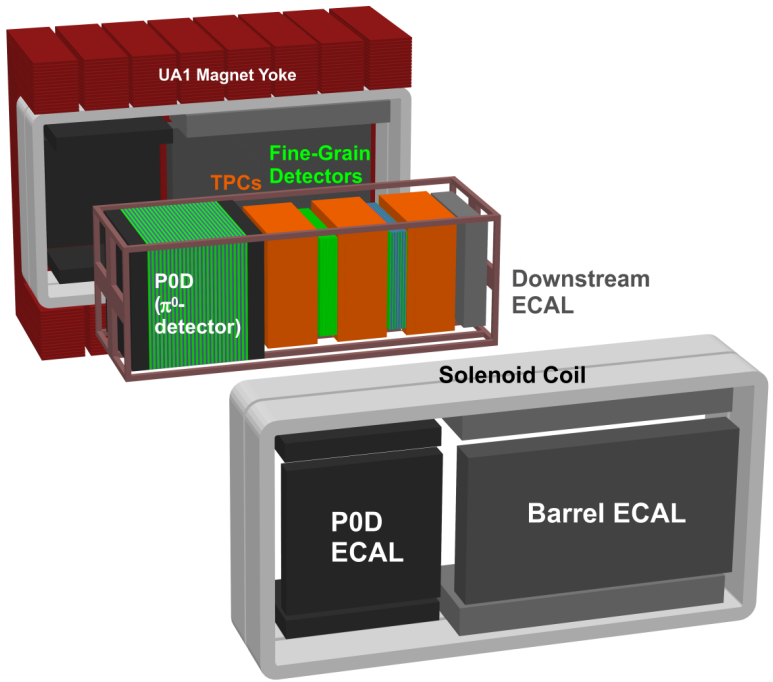
T2K

- 463 members from 61 institutions in 11 countries

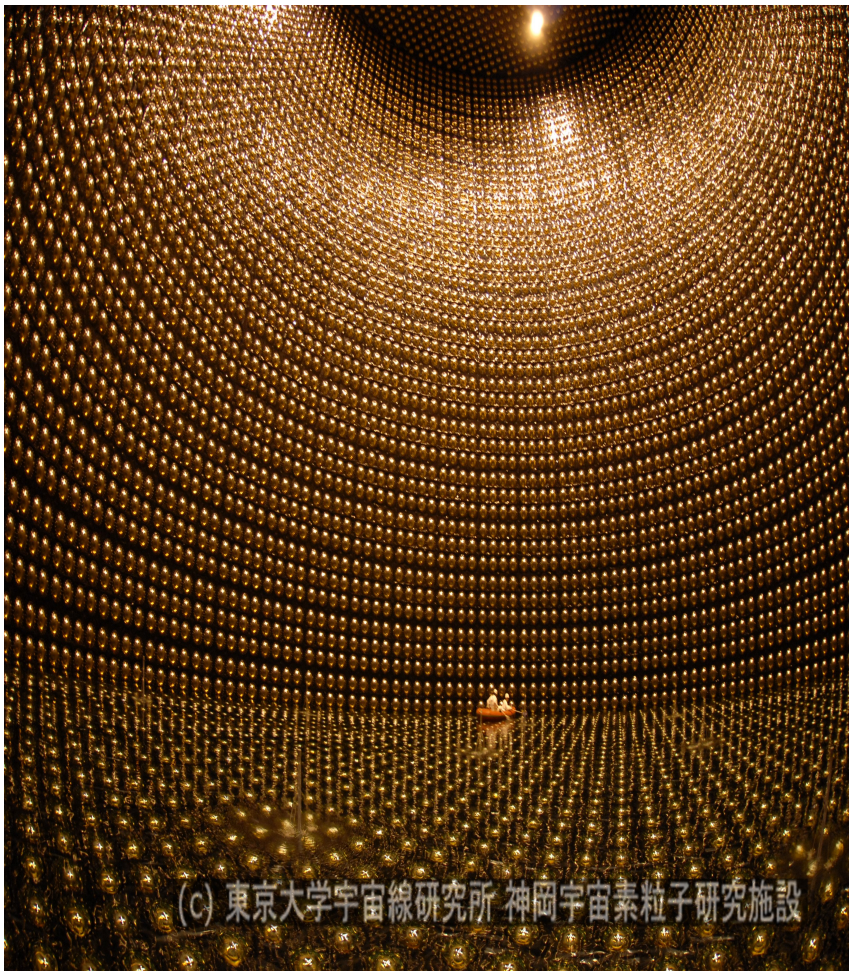
T2K: ND280

- Multi-subsystem neutrino detector 280m from beam origin
- Gives information about the beam before oscillation and interaction cross-section on different materials

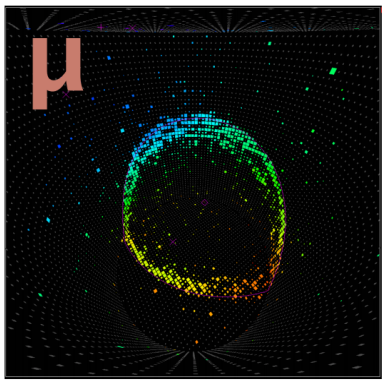


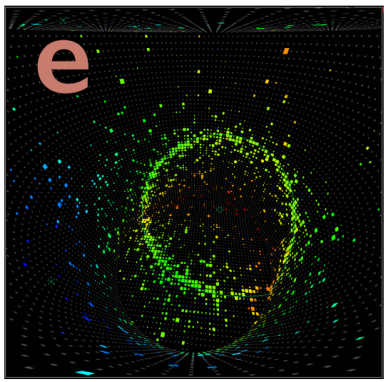


T2K: SK

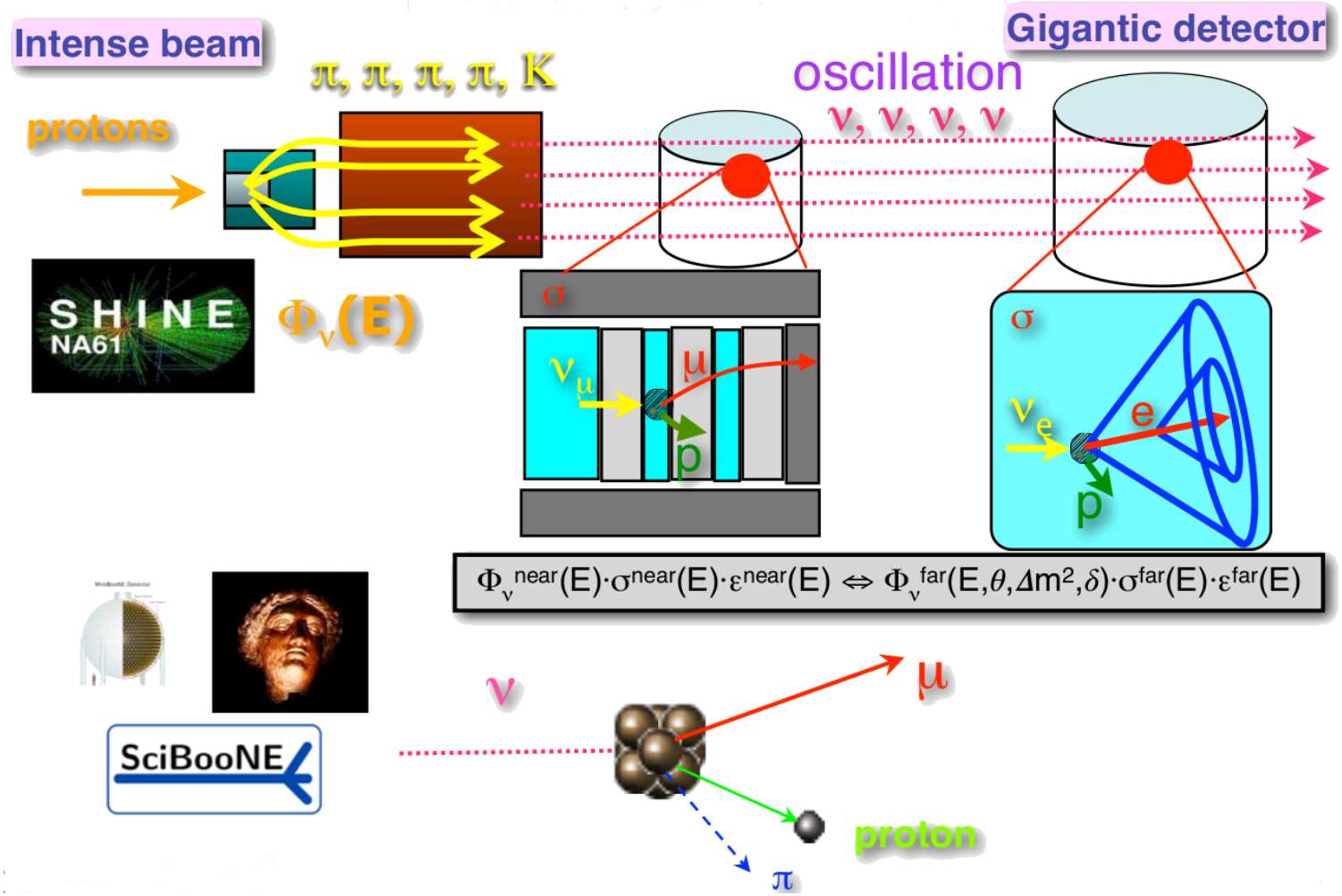


- 50kT ultra pure water
- 13,000 photon detectors
- Sees charged particles through Cherenkov light
- Measures the beam after oscillation



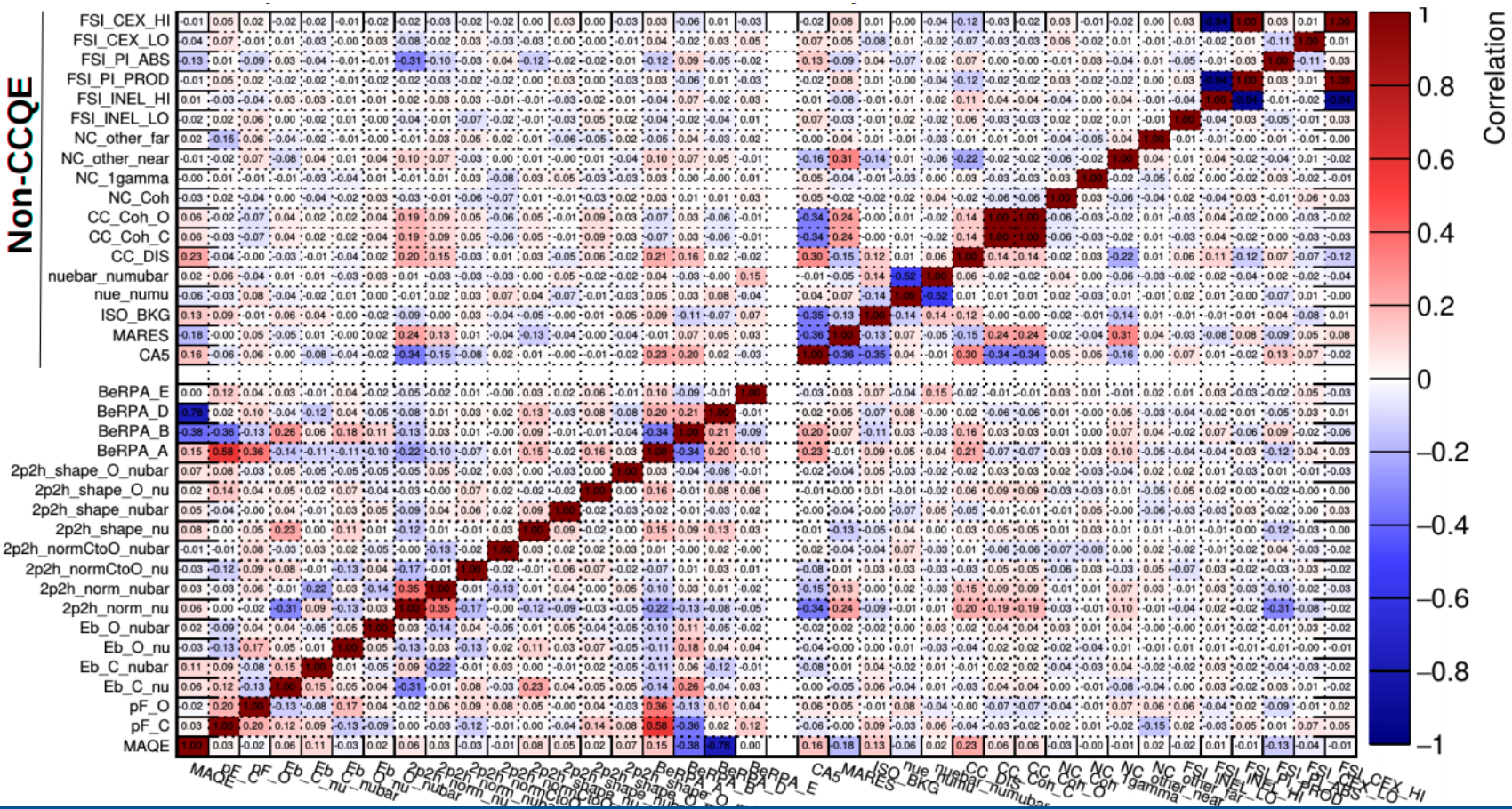


How do you actually do the analysis?



What do we mean by complex?

- 6 oscillation model parameters
- More than 700 correlated systematic parameters



Bayesian statistics: Posterior probability

- Given a model can calculate probability of seeing data, D:

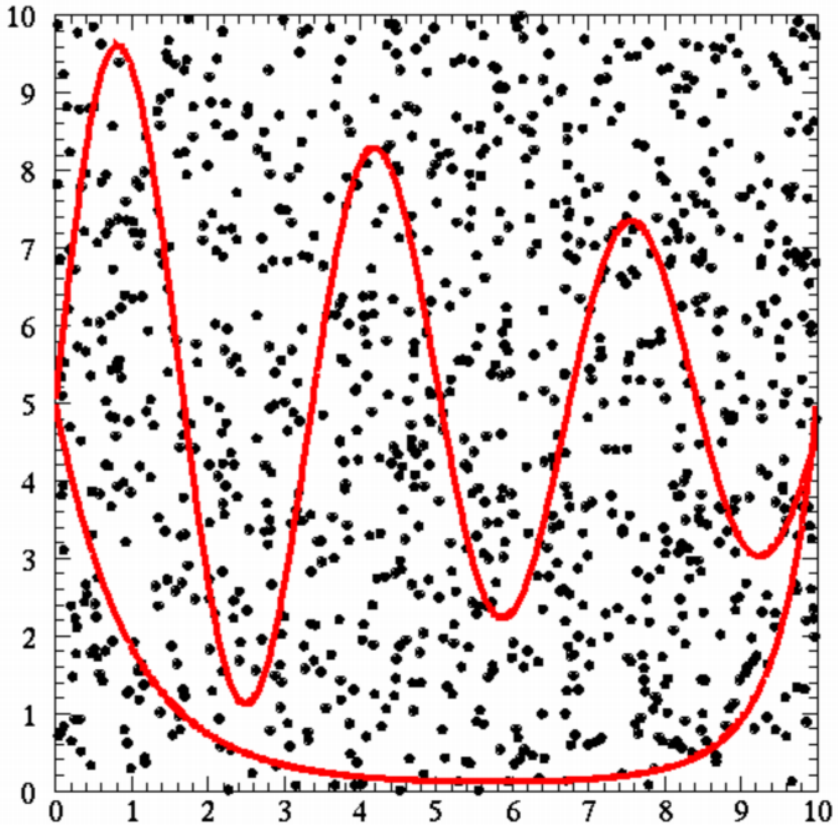
$$P(D, \vec{\theta}) = P(D|\vec{\theta})P(\vec{\theta})$$

- Use Bayes theorem to turn into a statement about parameters of interest:

$$P(\vec{\theta}|D) = \frac{P(D|\vec{\theta})P(\vec{\theta})}{\int P(D|\vec{\theta})P(\vec{\theta}\mathrm{d}\vec{\theta}}$$

- This is a function of over 700 variables!
- Need to eliminate all the 'nuisance' parameters
- Bayesians do this by integrating over them

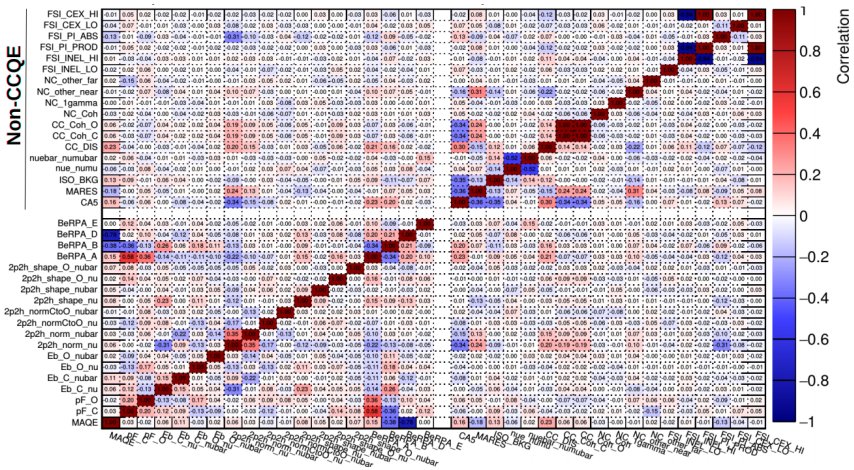
Using Monte Carlo methods



Standard MC

- Randomly choose points in your space
- Calculate the average value of your function at the points
- Multiply by area sampled

Using Monte Carlo methods



Problem

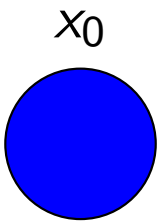
- Probability is low at most points
- Space is very very large!
  - Over 700 dimensions
- Need a huge number of points

Solution: Markov Chain Monte Carlo (MCMC)

- Need to target where the steps are thrown
- Use a Markov chain to 'choose' points
  - Next point location depends on current point
- Need to avoid bias through choice of points
  - Make sure density of points $\propto$ probability density
- How do you design a Markov chain to pick points proportional to a distribution?

Metropolis-Hastings Algorithm

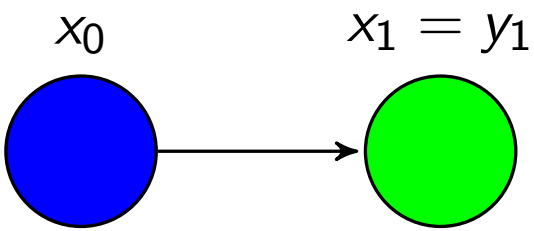
- Pick a starting point x_0
- Calculate Probability $P(x_0)$
- Don't need to know the distribution, just calculate it at a point



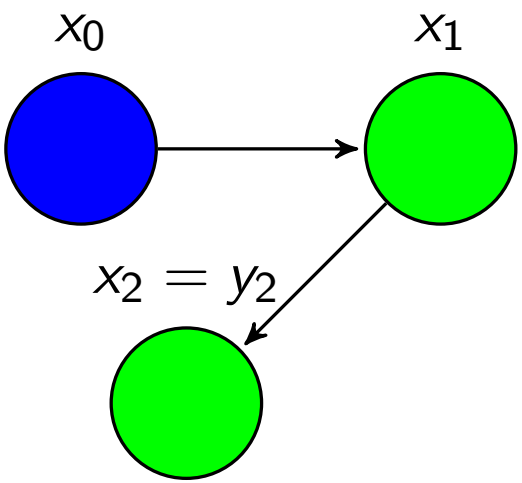
Metropolis-Hastings Algorithm

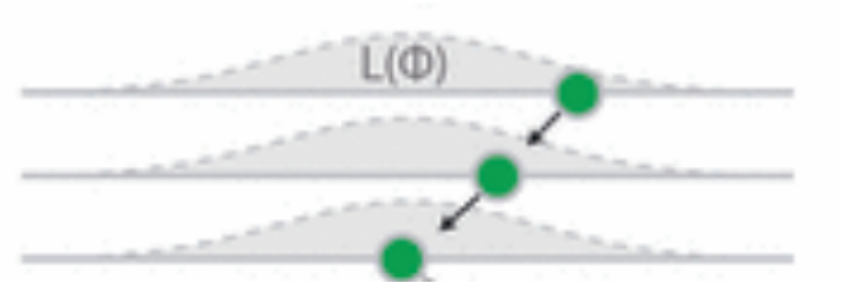
- Propose a new point y_1 using a 'proposal function': $q(y_1|x_0)$
- Set $x_1 = y_1$ with probability:

$$\alpha = min\left(1, \frac{P(y_1)q(y_1|x_0)}{P(x_0)q(x_0|y_1)}\right)$$

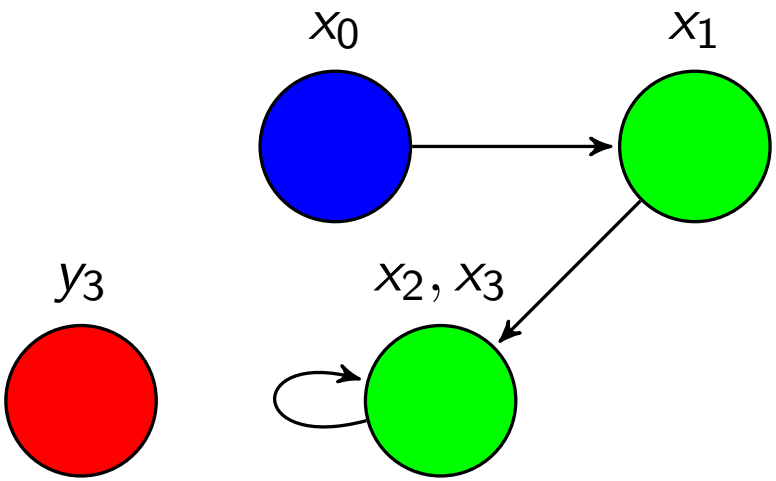


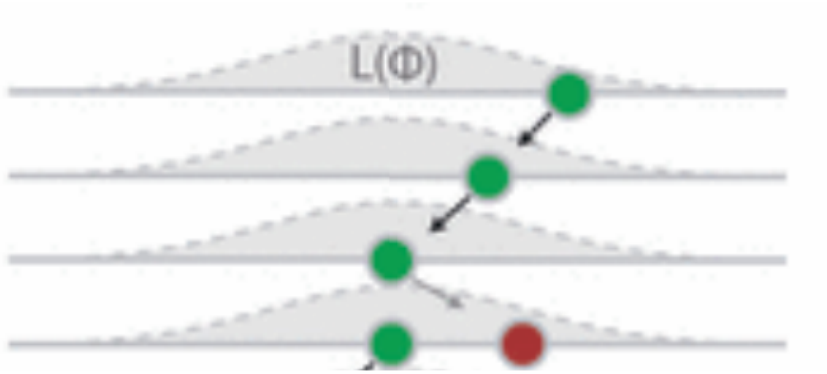
Metropolis-Hastings Algorithm





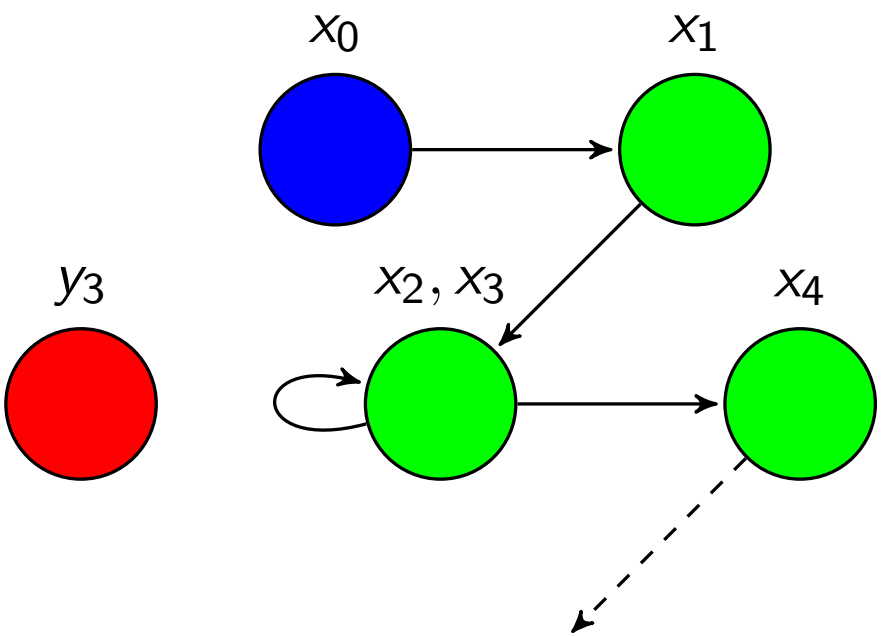
Metropolis-Hastings Algorithm

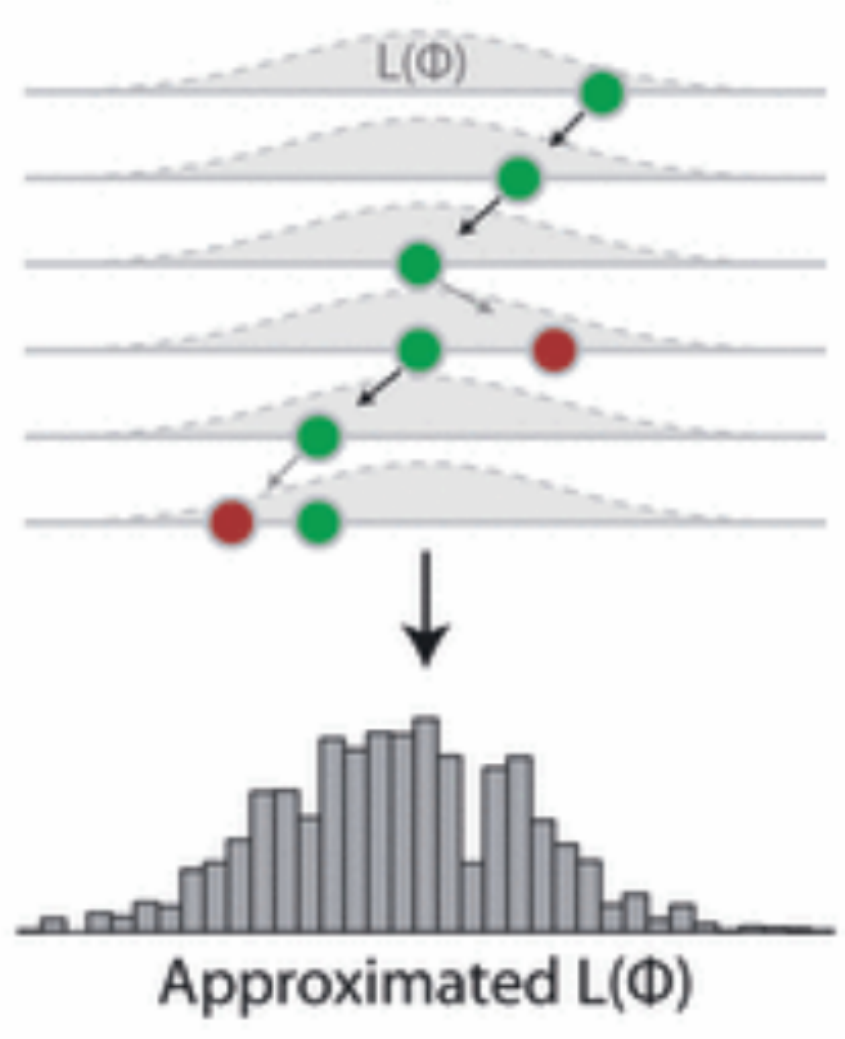




- $P(y_3) < P(x_2)$ so $\alpha < 1$ and it might be rejected

Metropolis-Hastings Algorithm





- Repeat until you build up a picture of the probability

How do we know the MCMC is representative?

- There are three conditions for a Markov Chain to converge to being representative of a distribution:
  1. Irreducibility: Must be possible to get from one point to any other point in a finite number of steps
  2. Recurrence: Each step after convergence must be a sample from the distribution
  3. Aperiodicity: The chain must not move periodically through the same sequence of states
- Need to choose the right proposal function and acceptance probability

Which Proposal Function do we use?

Continuous parameter

- Use a multivariate Gaussian
- Take into account correlation between parameters
- Satisfies irreducibility ✓
- Satisfies aperiodicity ✓

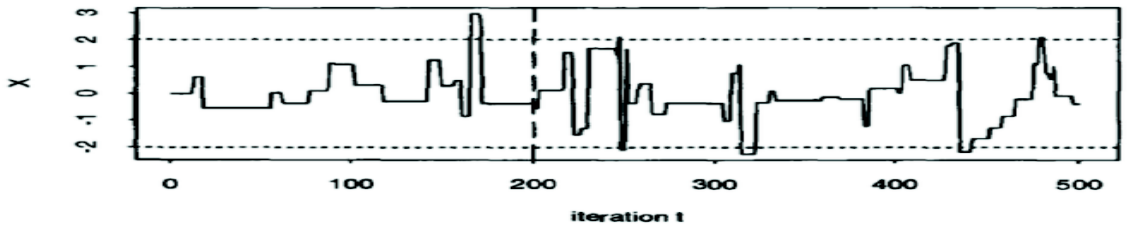
Discrete parameter

- All discrete parameters we look at have 2 values
- 50% chance of flipping for any step
- Satisfies irreducibility ✓
- Satisfies aperiodicity ✓

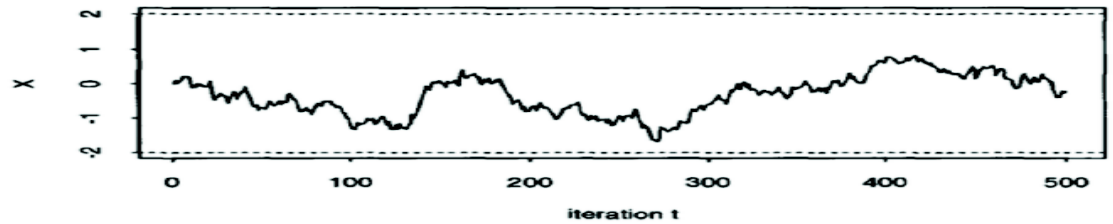
- Both of these are symmetrical so:
  - $\rightarrow \alpha = min\left(1, \frac{P(y_1)}{P(x_0)}\right)$
- Each step is therefore a sample from P
  - Satisfies recurrence ✓

Which Proposal Function?

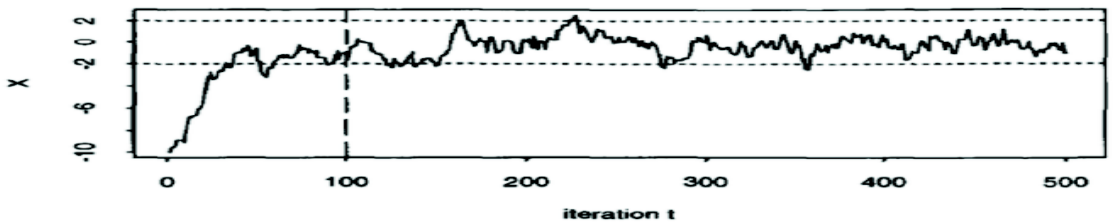
- Need to pick a width for the Gaussian
- Too wide and a lot of steps will be rejected:



- Too narrow takes too long to 'forget' initial conditions:

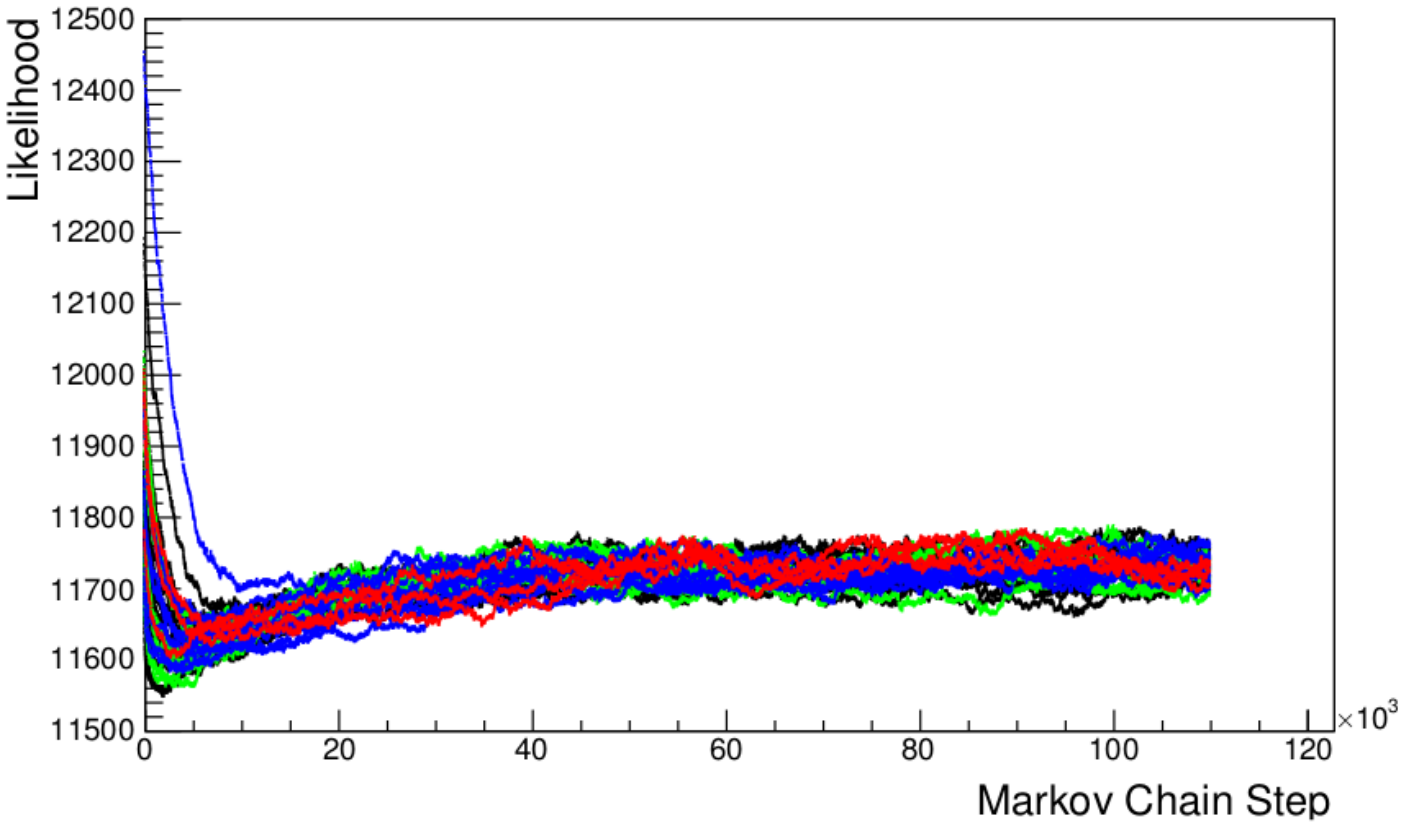


- Just right and it explores space and converges quickly:



MCMC: Burn In

- Initial period not representative of distribution
- We cut off the first 20k steps of the chain

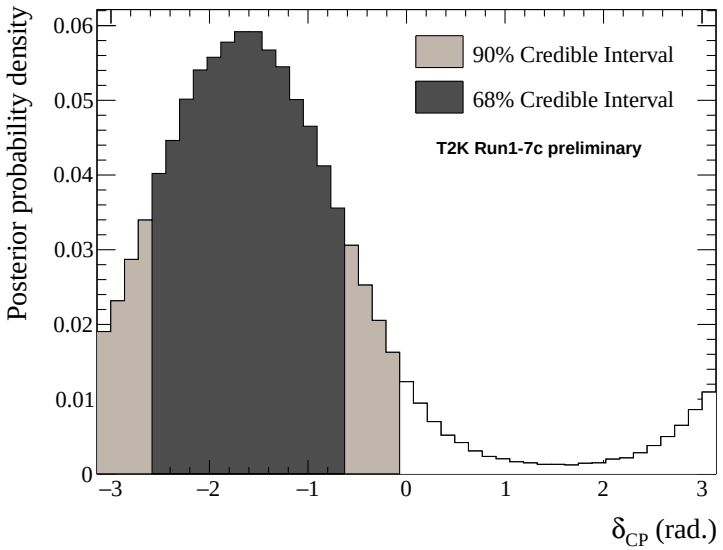


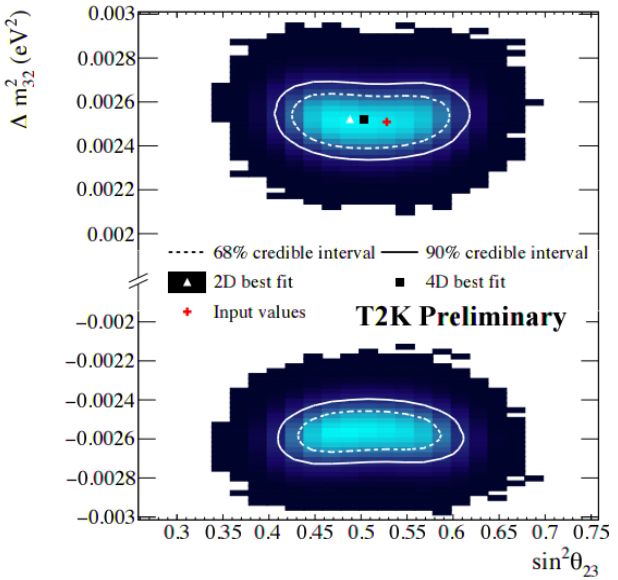
Implementation

- Representative chain needs several million steps
- Takes ∼24 hours using multiple machines and GPU acceleration
- Code written in C++ and CUDA (GPU langauge)
  - We have 8 active developers
  - s/step ∼0.5 with GPU, ∼2.5 without GPU

MCMC: Results

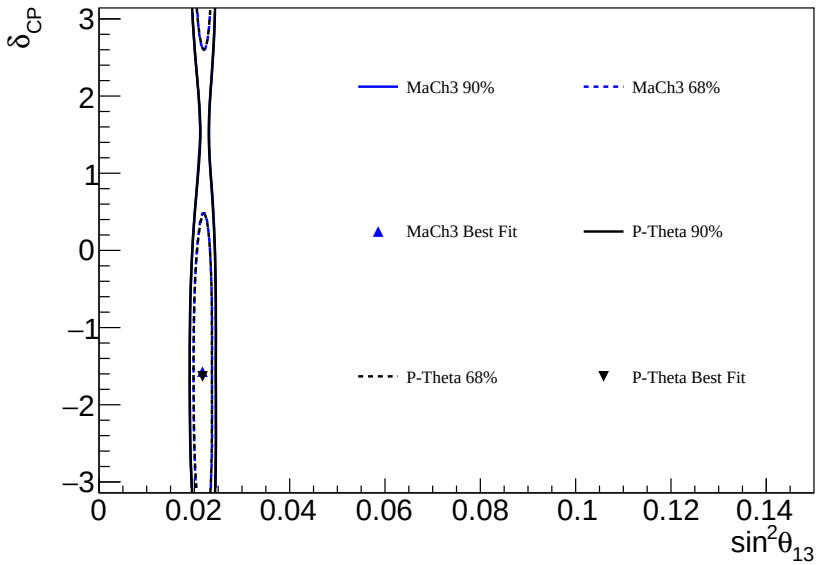
- Draw contours around regions containing 68/90% of all steps
- Region inside/outside contour is 'allowed'/'excluded'

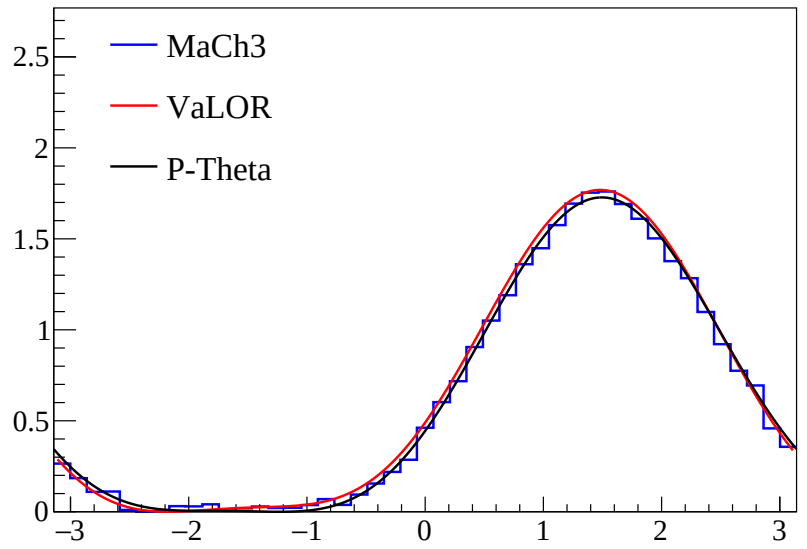




Validation

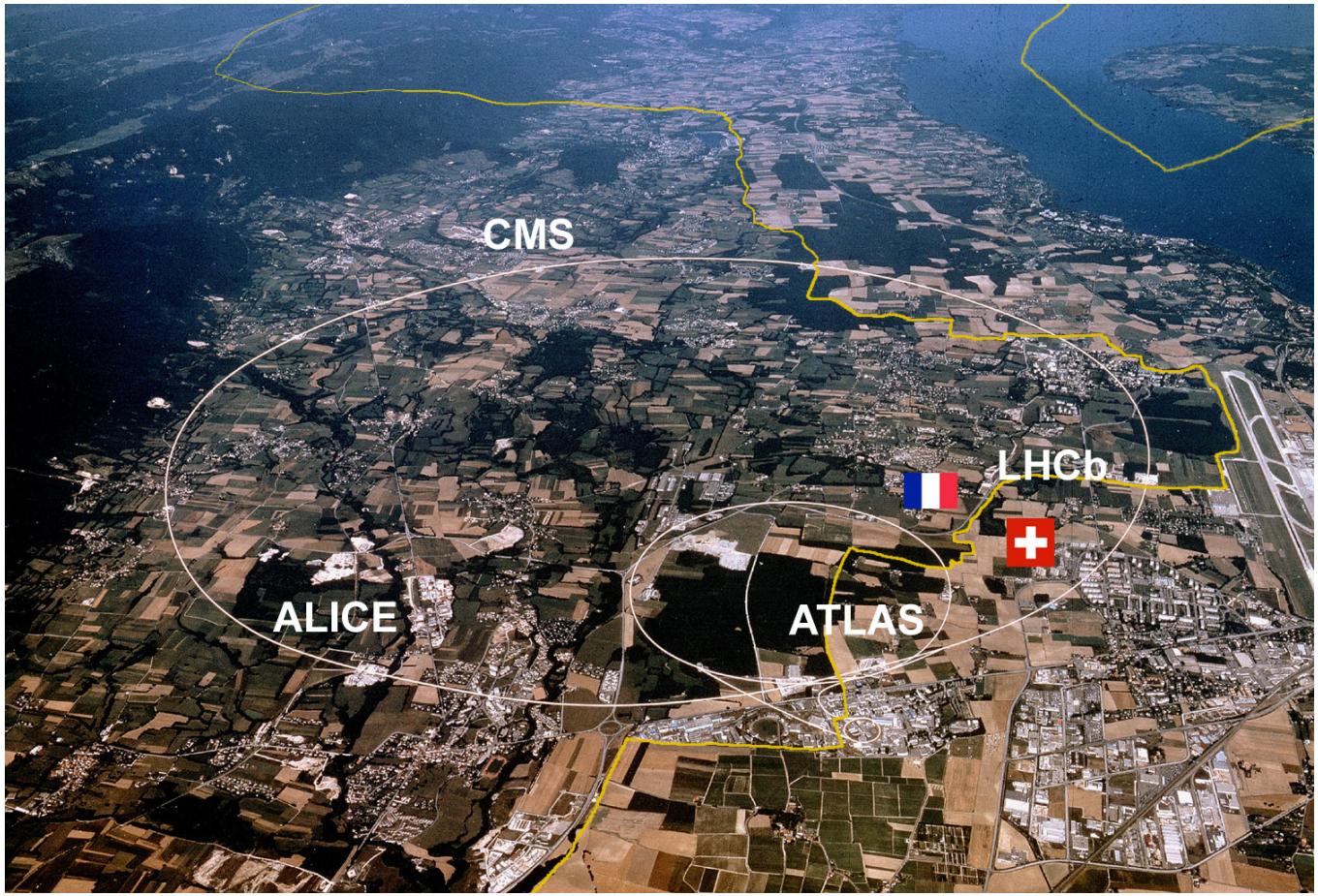
- Highly complex code and inputs so validation necessary
- T2K has two other oscillation fitters
- Validate results from all three fitters against each other





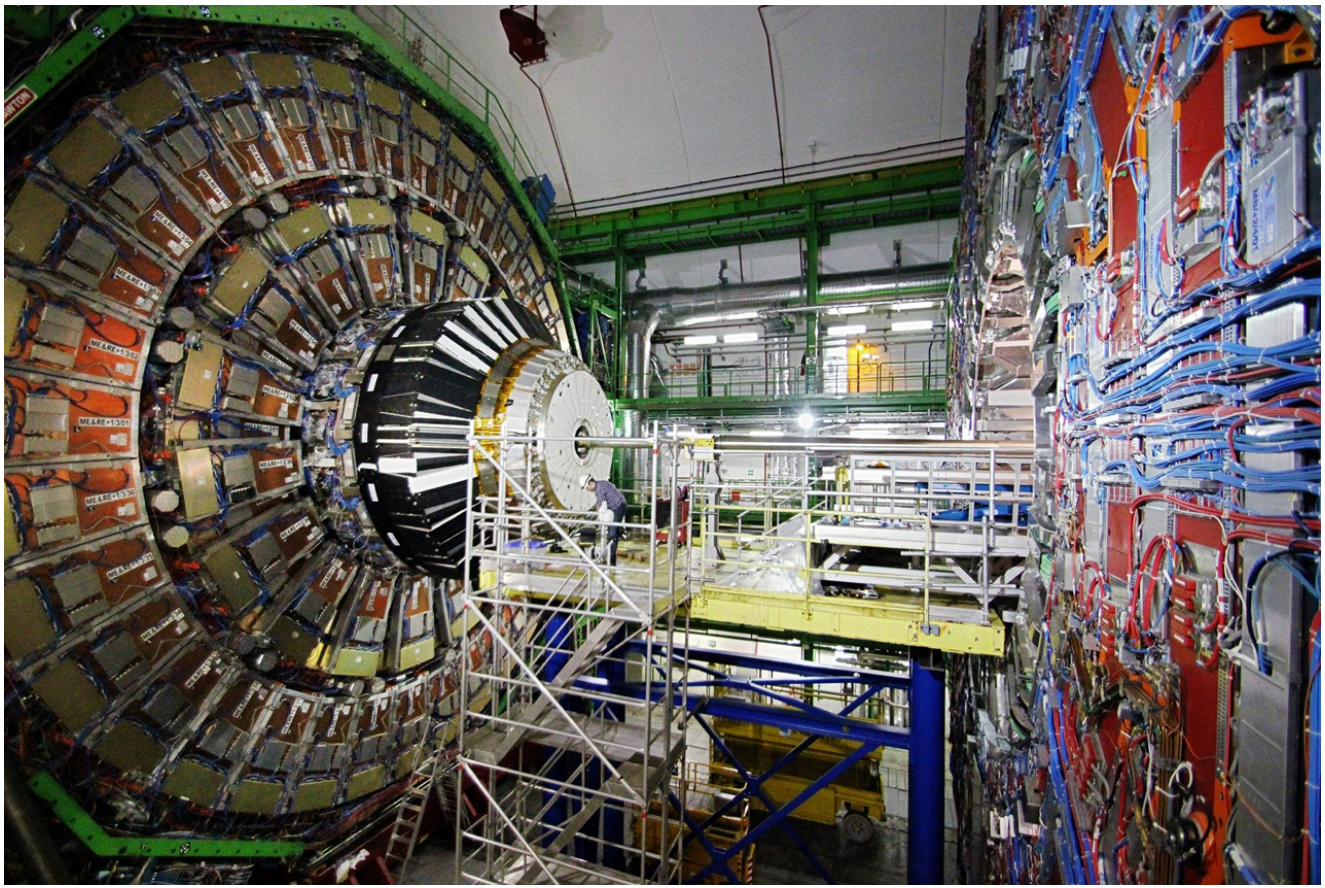
The Large Hadron Collider (LHC)

- LHC in Geneva is a 27km long, 14 TeV proton synchrotron



Compact Muon Solenoid (CMS) at the LHC

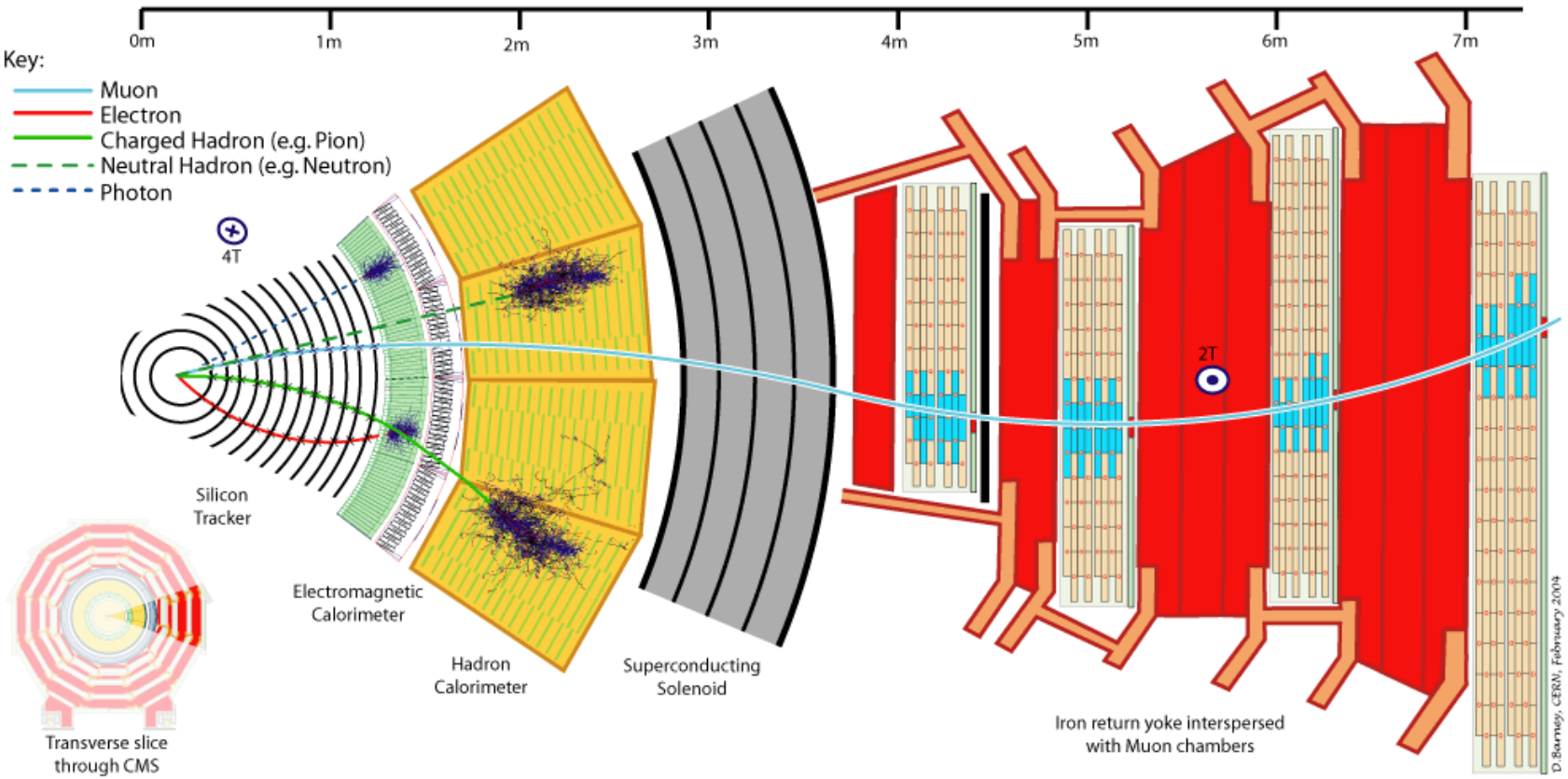
- 75M read-out channels just in inner tracker
- Has to record a collision every 25ns



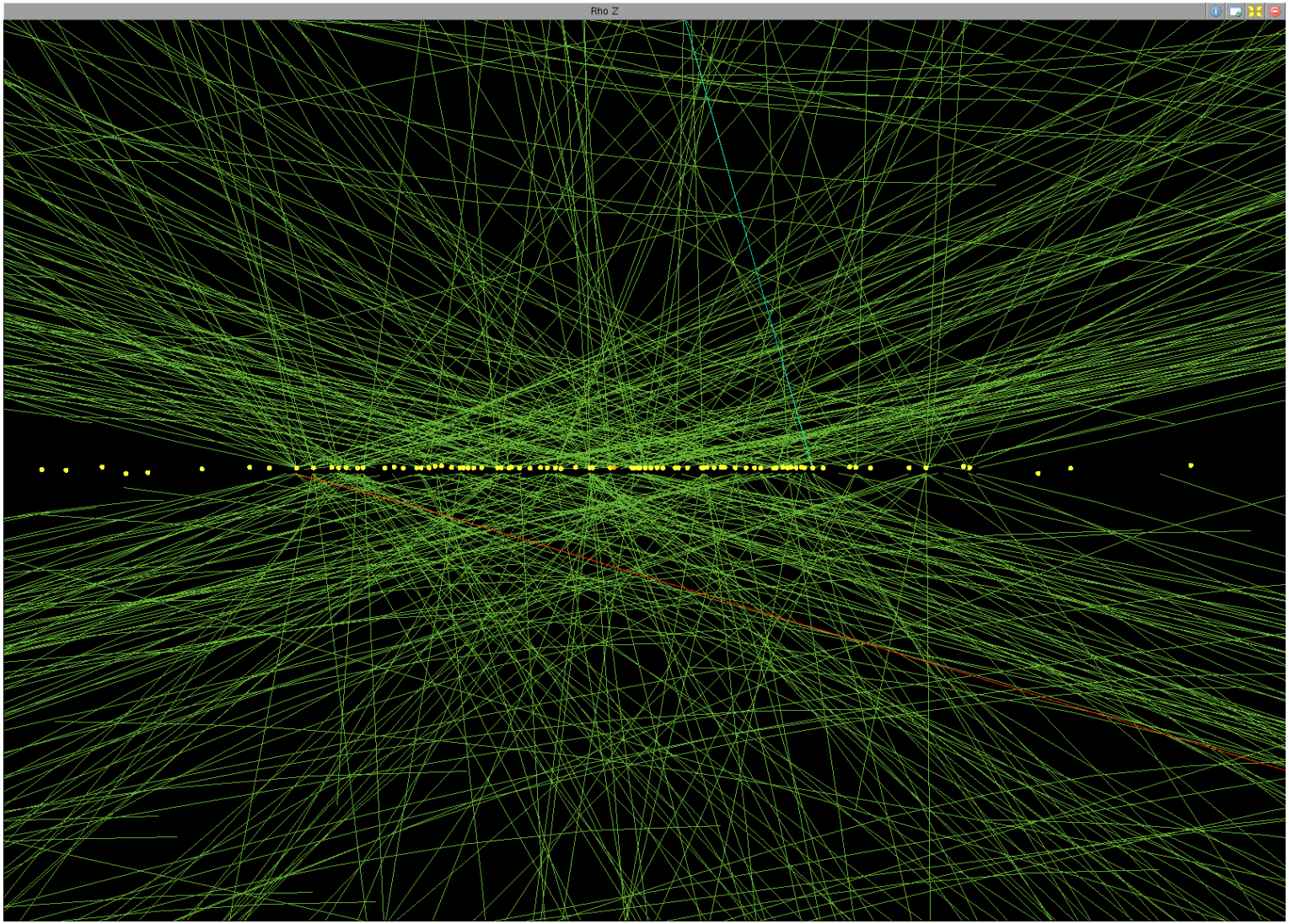
CMS

- Over 5000 active members from 199 institutes in 46 countries

CMS at the LHC



Vertex Finding: Simulated Annealing



Vertex Finding: Annealing

- Start with a set of tracks with positions z_i, uncertainty σ_i
- Want to find the set of vertex positions z_k that they come from
- Lots of local minima that you could get stuck in
- Good candidate for annealing (see lecture 8)

Vertex Finding: Annealing

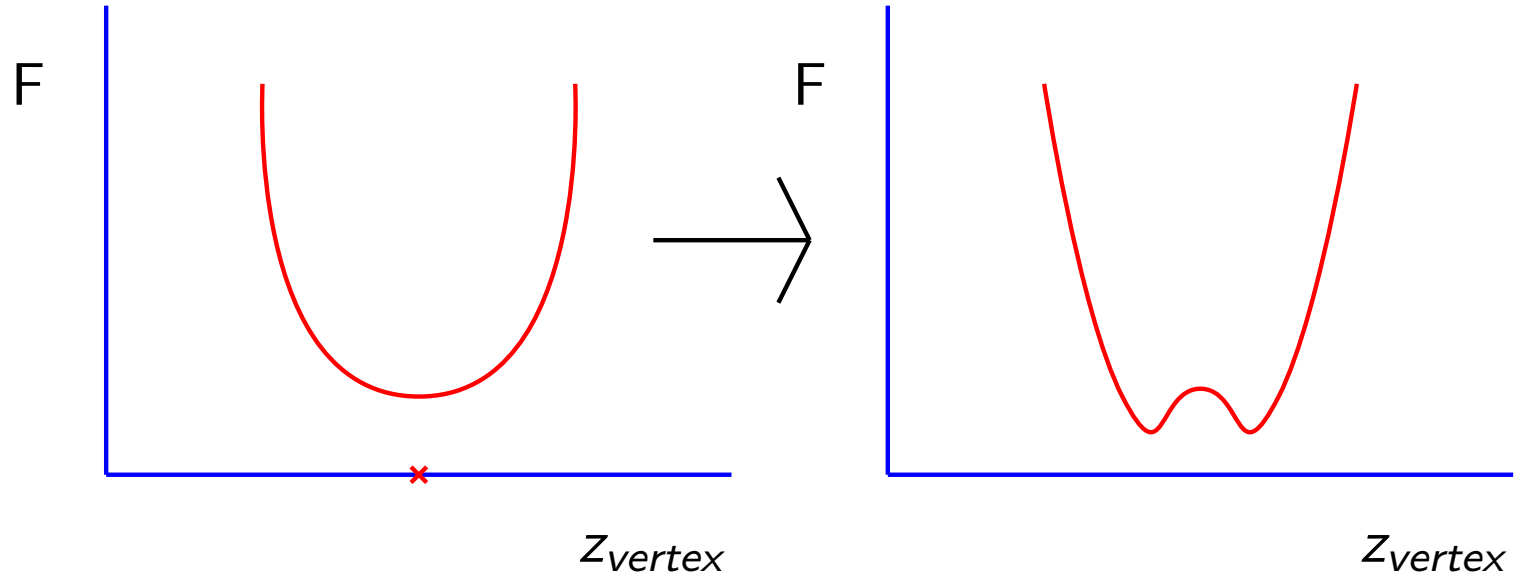
- Define the 'free energy' of the system:

$$F = -T \sum_{i}^{\#tracks} log \sum_{k}^{\#vertices} exp \left[-\frac{1}{T} \frac{\left(z_i^T - z_k^V\right)^2}{\sigma_i^2} \right]^2$$

- Temperature T defines the penalty for a track being far from a vertex

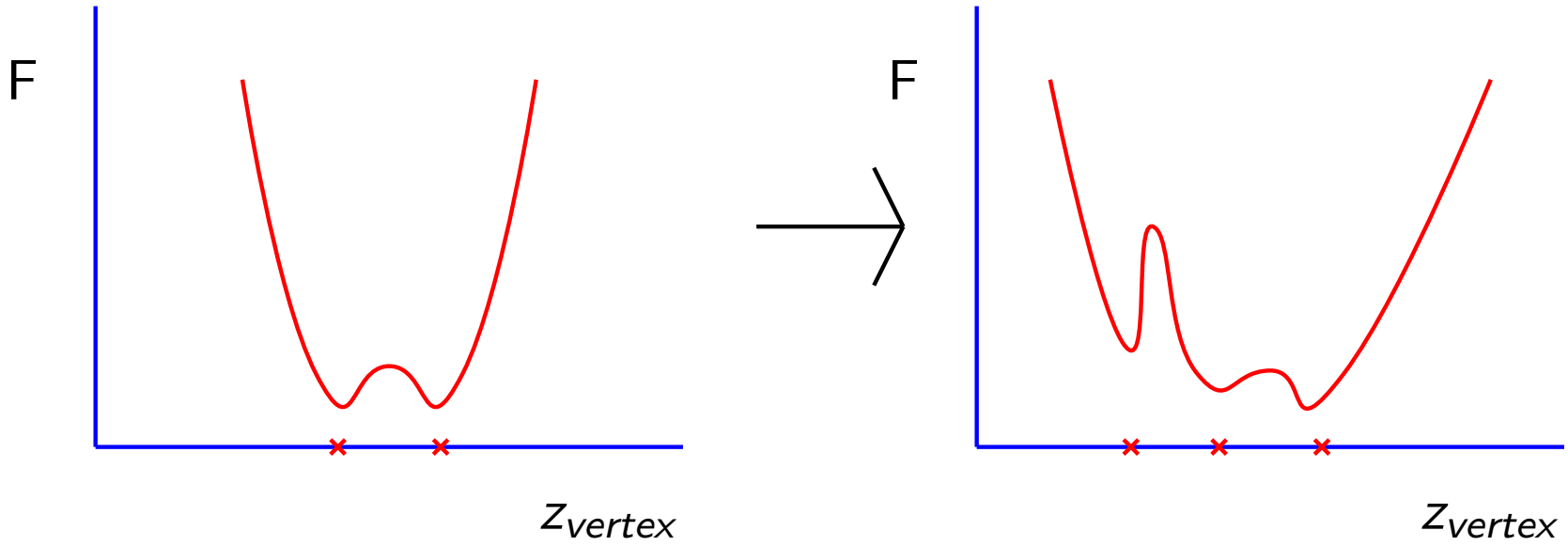
Vertex Finding: Annealing

- Start at high T with a single vertex
- Find the value of z_{vertex} that minimises F
- Reduce T and repeat until the minimum of F turns into a saddle point



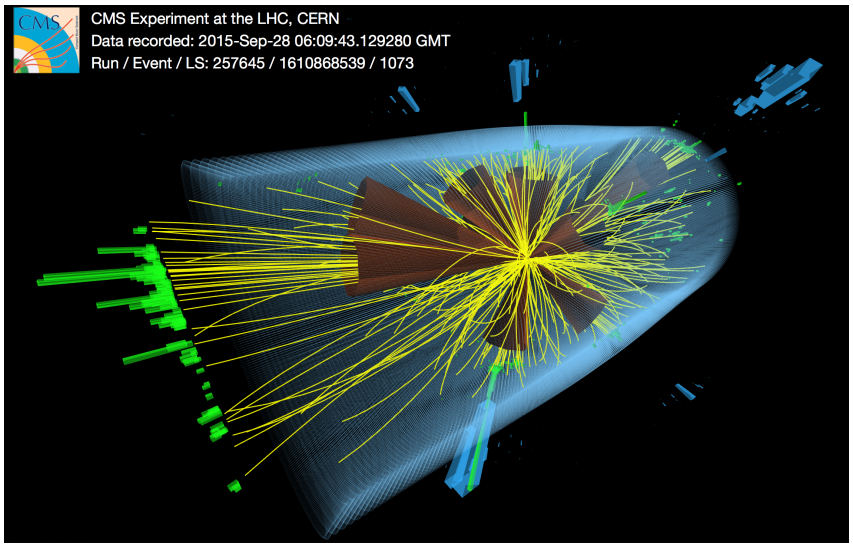
Vertex Finding: Annealing

- If there's a saddle point split the closest vertex into two
- Repeat minimisiation with the new number of vertices
- Continue reducing T and repeating until you reach T_{min}
- Algorithm finds both position and number of vertices



Jet Energy Resolution: Pseudo-Random numbers

- A jet is a collimated spray of hadrons
- In simulation we assume a certain energy measurement resolution
- This is then measured in data and we need to update the simulation



Jet Energy Resolution: Pseudo-Random numbers

- Usually need to slightly worsen the resolution
- Add a random amount of energy drawn from a gaussian with width:

$$\sqrt{c^2 - 1}\sigma_{simulation}$$

- Still want reproducibility for validation against other analysers
- Use a pseudorandom number
- The framework where this was implemented has 8 active developers and is mostly written in C++

Summary

- Showed several applications of the techniques from previous lectures
  - Markov chain Monte Carlo
  - Simulated annealing
  - Pseudo-random numbers
- Described the scale of the challenges faced in particle physics and the size of the collaborations needed to solve them