Statistical Methods in Particle Physics / WS 13

Lecture IV

Monte Carlo Simulations

Niklaus Berger

Physics Institute, University of Heidelberg

Emmy Noether-Programm
Deutsche Forschungsgemeinschaft
DFG

Part IV:

Monte Carlo

Monte Carlo Methods

Replace analytical calculations by random sampling

Usually with a computer

4.2. Mersenne Twister

```
 uint32_t MT[624];
 uint32_t index = 0;

void initialize_generator(int seed) {
     index = 0;
     MT[0] = seed;
     for (int i = 1; i < 624; i++) {
         MT[i] = MT[i-1]*1664525 + 1013904223;
     }
 }
void extract_number() {
     if(index == 0) {
         generate_numbers();
     }
     uint32_t y = MT[index];
     y = y ^ (y>>11);
     y = y ^ (y<<7 & 2636928640);
     y = y ^ (y<<15 & 4022730752);
     y = y ^ (y>>18);
     index = (index + 1) % 624;
     return y;
 }
```

Create a length 624 array to store the state of the generator

initialize the generator with a seed,

then fill array using linear congruential generator

Get a tempered number - recreate the index after 624 numbers have been used

Bit wise exclusive or with shifted y and some "magical" numbers

Increase index

Mersenne Twister

```
void generate_numbers() {
    for(int i = 0; i < 624; i++ {
        uint32_t y = (MT[i] & 0x80000000);
                + (MT[(i+1) % 624] & 0x7fffffff);

        MT[i] = MT[(i+397) % 624]^(y>>1);
        if((y % 2) != 0) {
            MT[i] = MT[i] ^ 2567483615;
        }
    }
}
```

Generate a new array of numbers

Get bit 31 of MT[i] and bits 30 to 0 of MT[i+1]

Mix numbers...

If y is odd, do an exclusive or with yet another number

Mersenne Twister

```
void generate_numbers() {
    for(int i = 0; i < 624; i++ {
        uint32_t y = (MT[i] & 0x80000000);
                + (MT[(i+1) % 624] & 0x7fffffff);

        MT[i] = MT[(i+397) % 624]^(y>>1);
        if((y % 2) != 0) {
            MT[i] = MT[i] ^ 2567483615;
        }
    }
}
```

Generate a new array of numbers

Get bit 31 of MT[i] and bits 30 to 0 of MT[i+1]

Mix numbers...

If y is odd, do an exclusive or with yet another number

- Passes almost all tests for randomness
- Has a very long period of $2^{19937} - 1$ $(\approx 4 \cdot 10^{6001})$
- Is implemented in TRandom3 and what you should use
- Gives you a uniform distribution - basis for everything else

4.7 Monte Carlo for Particle Physics

Usually we want to know an efficiency ε:

- If we produce N_{true} reactions of a certain type, how many of them do we see (N_{obs})
- This is essentially a Monte Carlo integration:

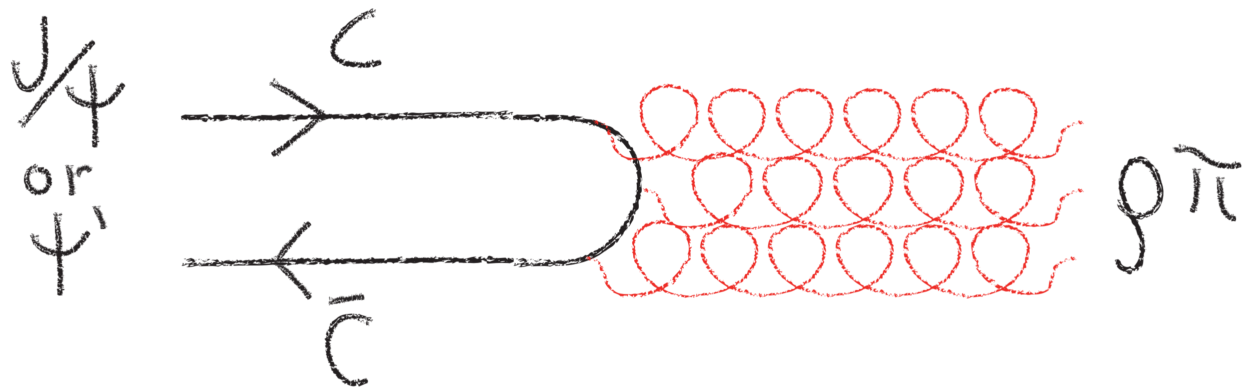
$$\varepsilon = \frac{\int_{\Omega,\Phi} \varepsilon(\Omega,\Phi)\sigma(\omega)d\omega d\phi}{\sigma \int L dt}$$

  where Ω denotes the phase space for the reaction and Φ "everything" that can happen in the detector. Often the cross-section σ is also calculated via MC integration.

- The integral over Ω gets three dimensions per final state particle, Φ is of exteremely high dimension (and will never have a closed form)

4.8 Full Example: Measurement of the branching fractions $J/\psi \rightarrow \pi^+\pi^-\pi^0$ and $\psi' \rightarrow \pi^+\pi^-\pi^0$

- Most abundant hadronic decay of the J/Ψ, but strongly suppressed in Ψ' decay
- Why? Not understood....
- Start studies by measuring branching fractions precisely



Need a lot of charmonium

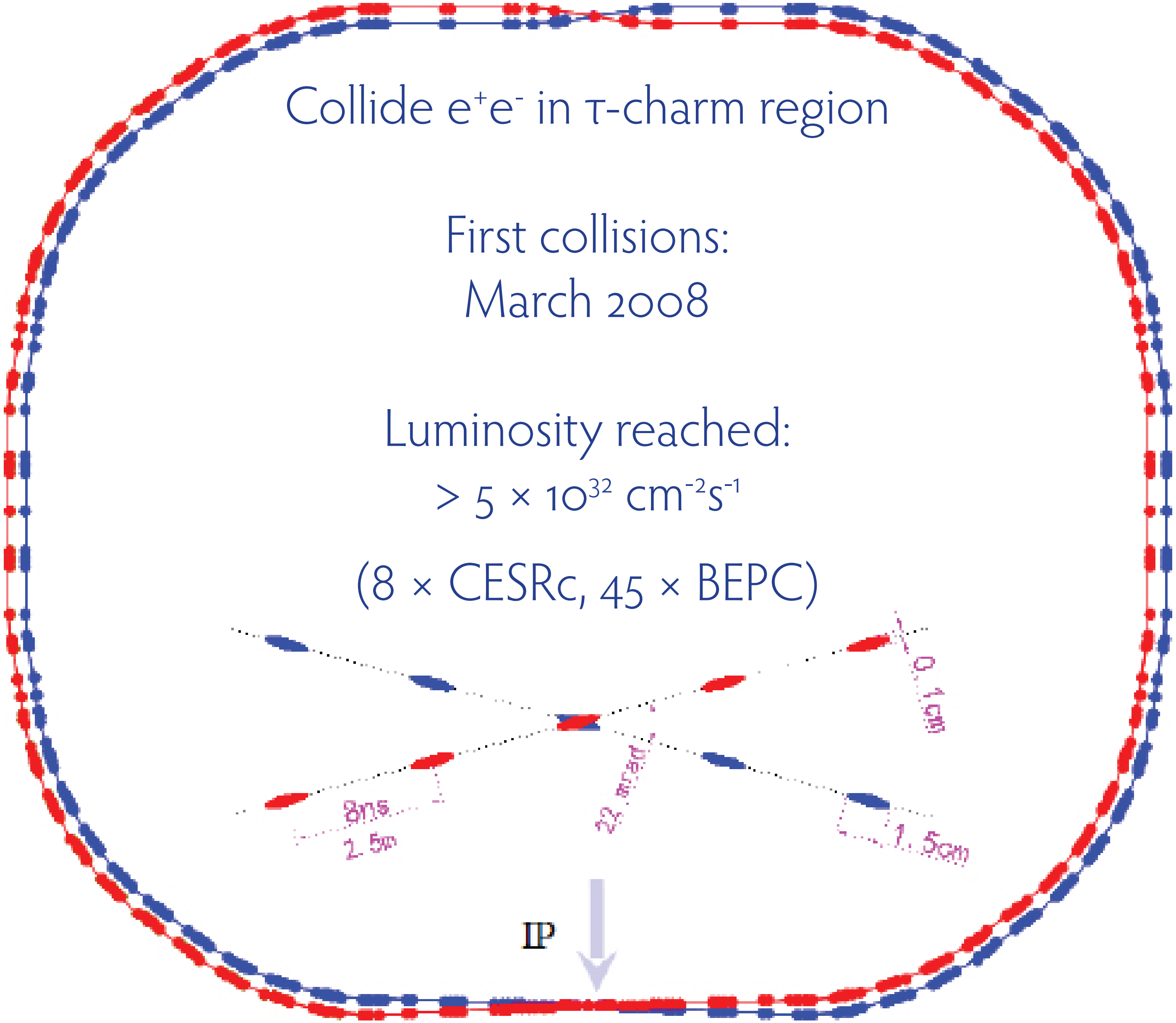
The Beijing Electron-Positron Collider II

Collide e^+e^- in τ-charm region

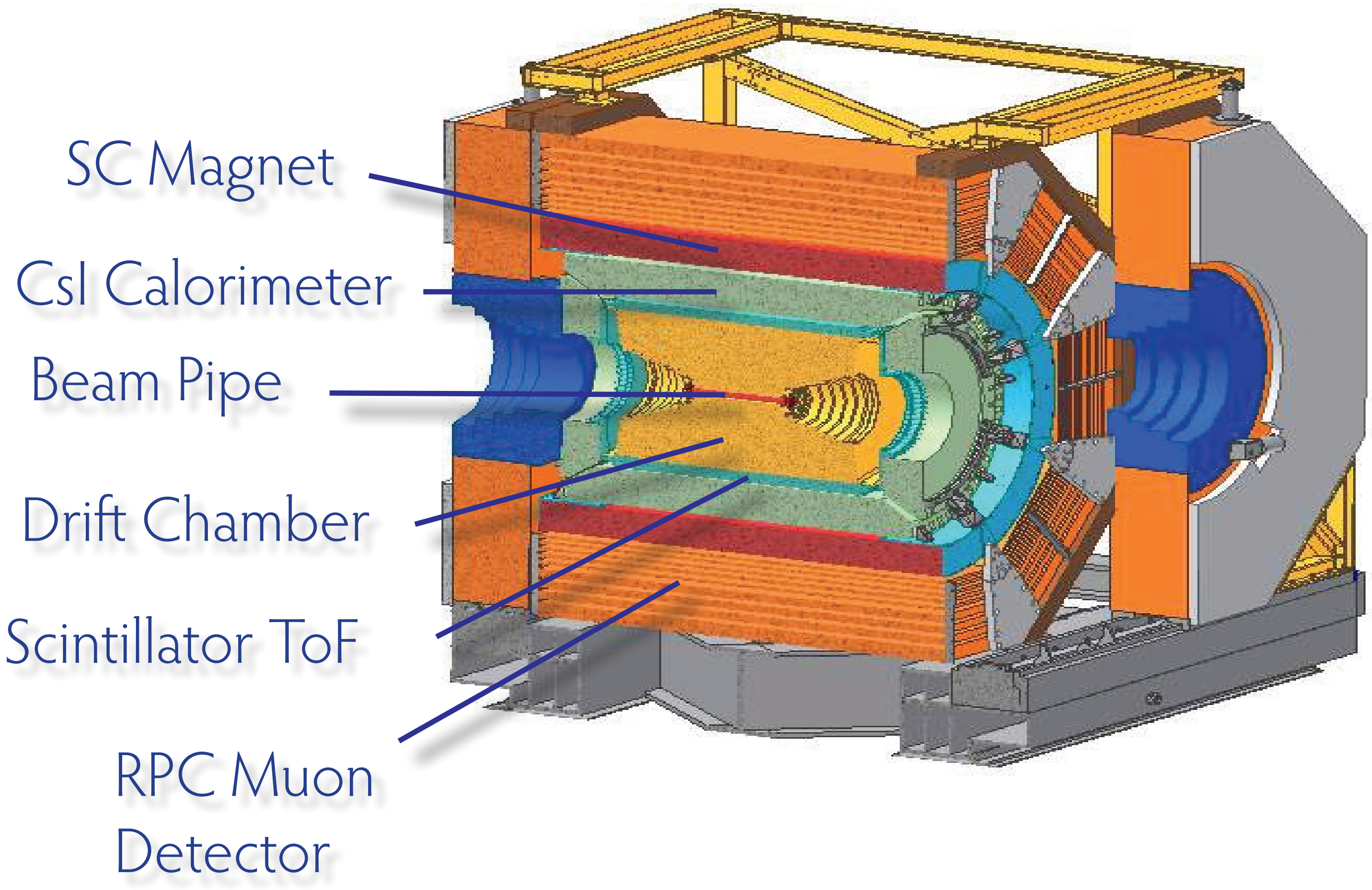
First collisions:
March 2008

Luminosity reached:
$> 5 \times 10^{32}\ \mathrm{cm^{-2}s^{-1}}$

(8 × CESRc, 45 × BEPC)



The Beijing Spectrometer III



Excellent tracking and calorimetry:

Tracks:

$\sigma_p/p = 0.58\%$ @ 1 GeV/c

Photons:

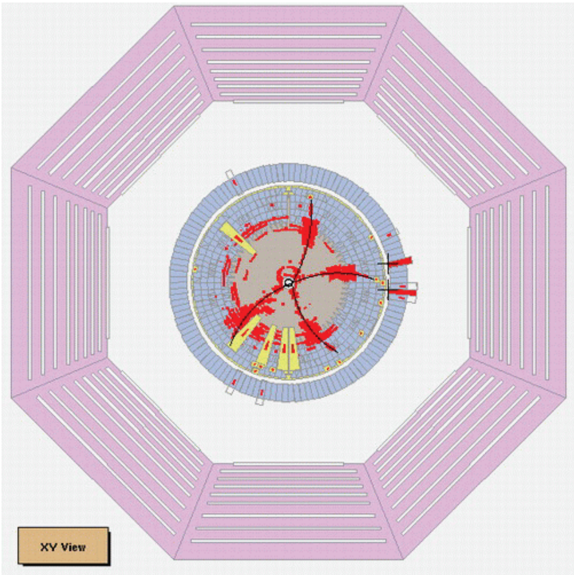
$\sigma_E/E = 2.5\%$ @ 1 GeV

Read-out at up to 6 KHz

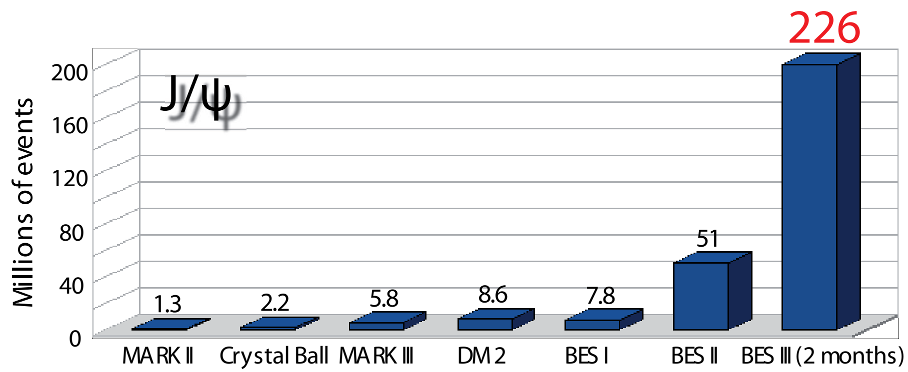
Data!

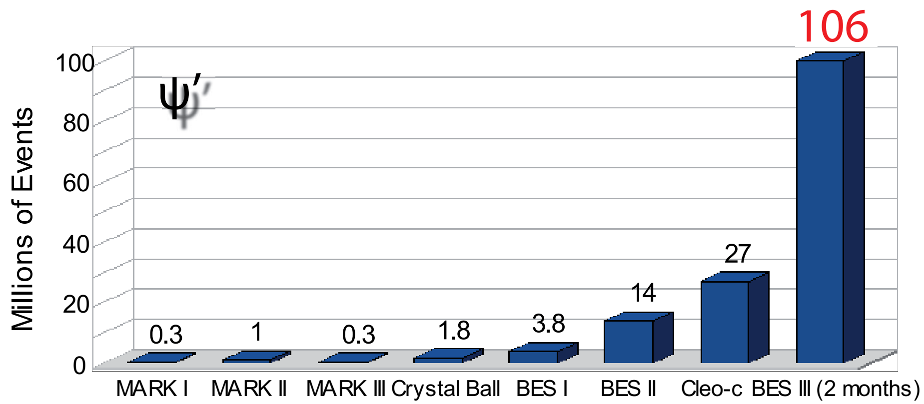
At the time of this measurement (2010), BES III had collected:

- 226 Millions of J/ψ
- 106 Millions of ψ'
- 2.9 fb^{-1} at the ψ(3770)
- 0.5 fb^{-1} at 4010 MeV



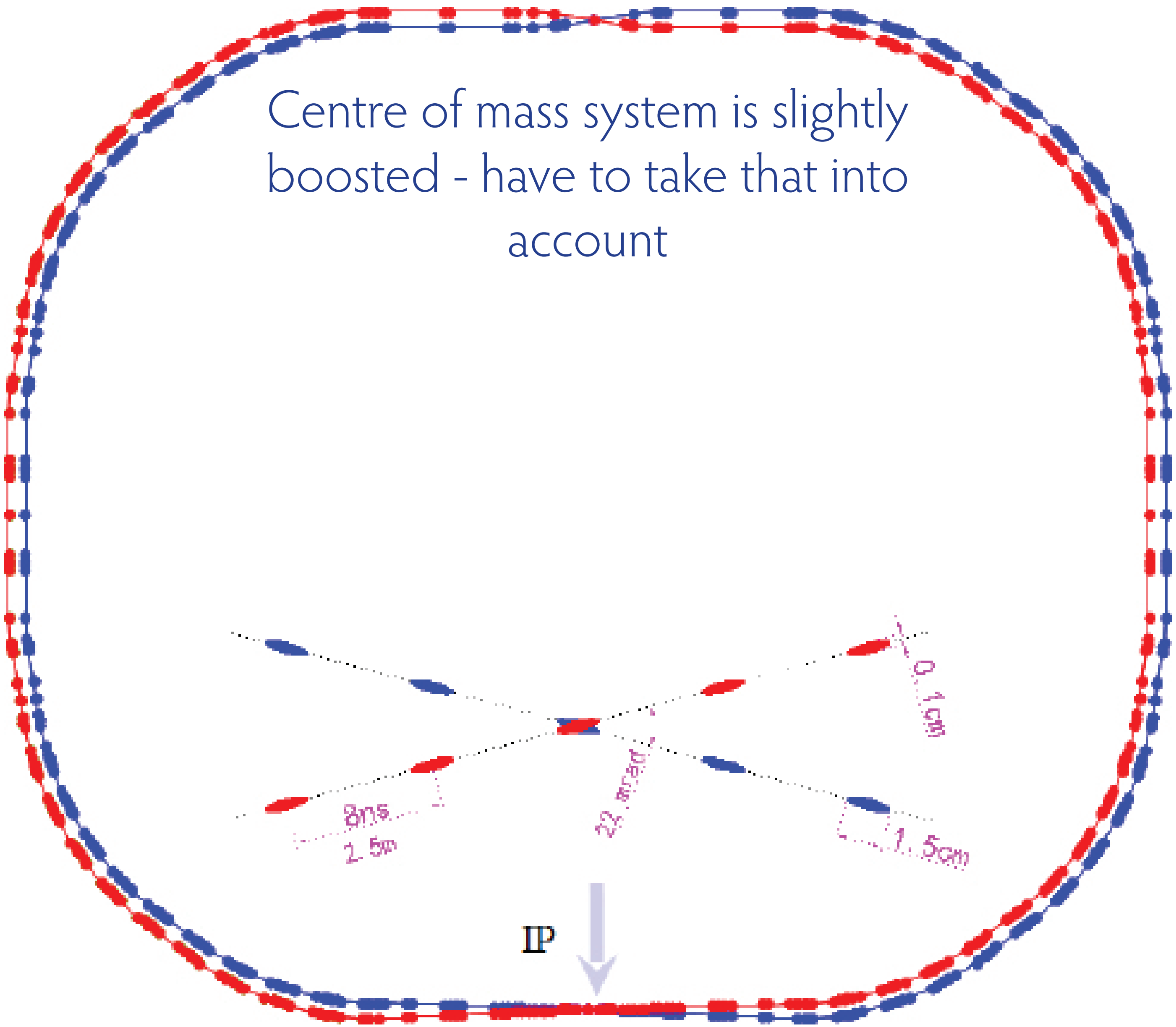
First hadronic event, July 2008





Monte Carlo Simulation: Initial State

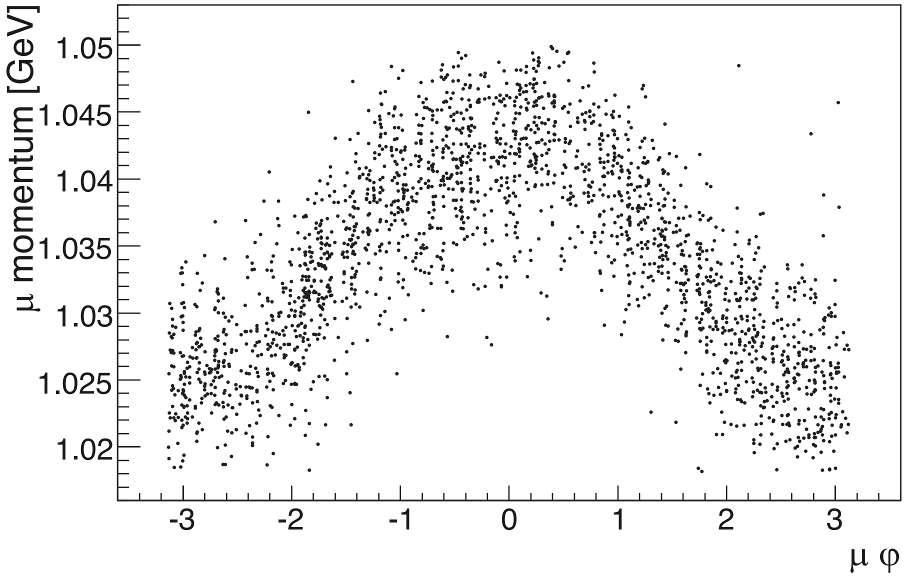
Centre of mass system is slightly boosted - have to take that into account



Always check...

Here use J/Ψ → $\mu^+\mu^-$ as a cross-check channel

- Muons have fixed momentum in the J/Ψ rest frame
- In the lab frame - not so much
- Apply boost

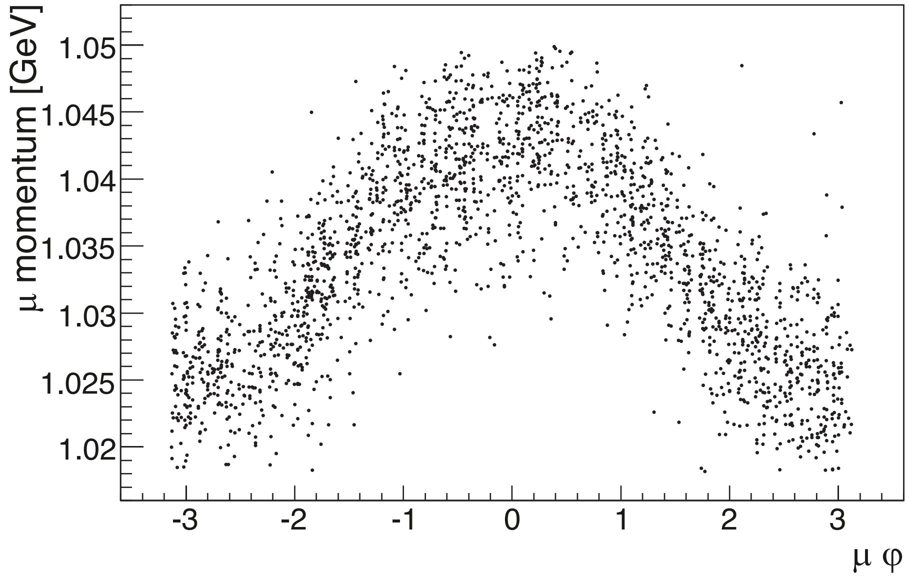


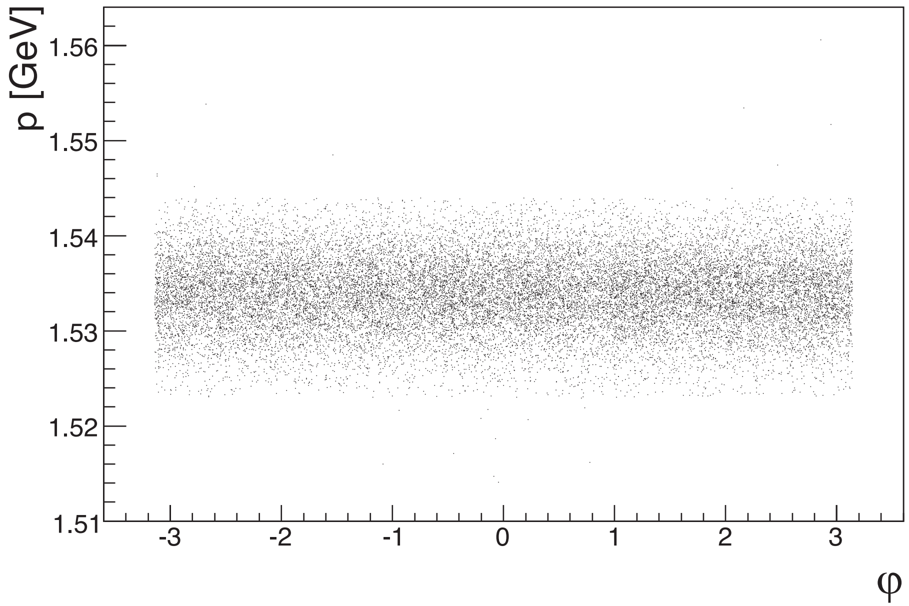
Always check...

Here use J/Ψ → $\mu^+\mu^-$ as a cross-check channel

- Muons have fixed momentum in the J/Ψ rest frame
- In the lab frame - not so much
- Apply boost
- And all is fine (if you applied the correct boost)

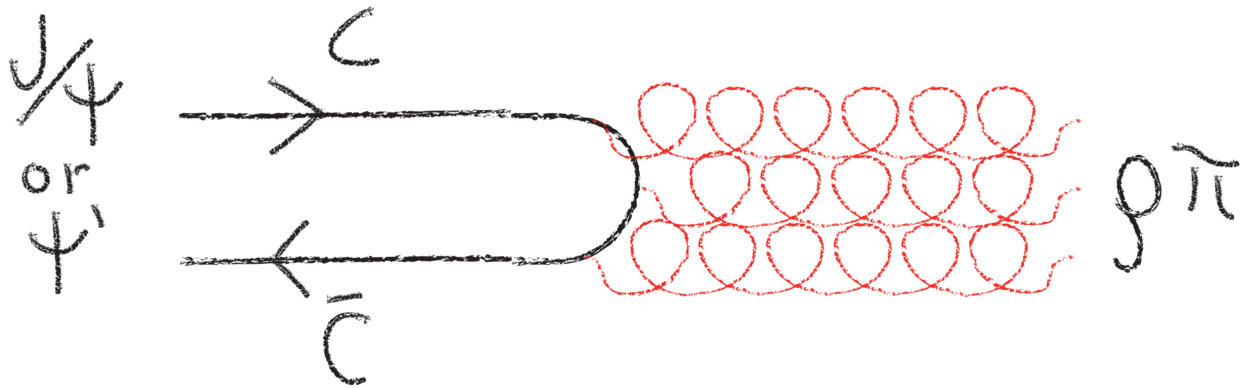
- Actually not - momenta slightly too low - energy loss in the detector





Generate physics process

- No good model for the hardonic part available (that is why we are doing the measurement in the first place...)
- Generate various processes:
  - Flat three particle phase space
  - Phase space for two pions coming from a vector resonance (a ρ)
  - And a simulation involving a $\rho(770)$ peak structure
- Using the EvtGen software package gives you 4-vectors for the final state particles (π^+, π^- and two photons from the π^0)

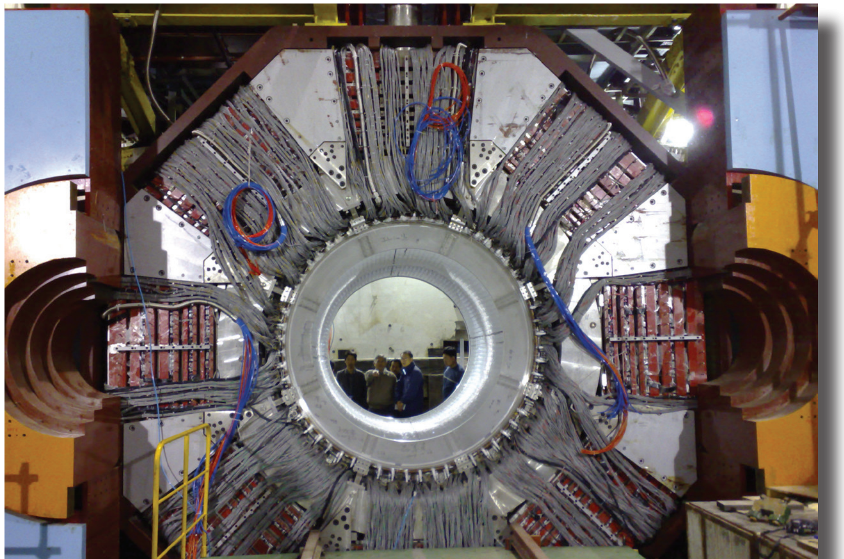


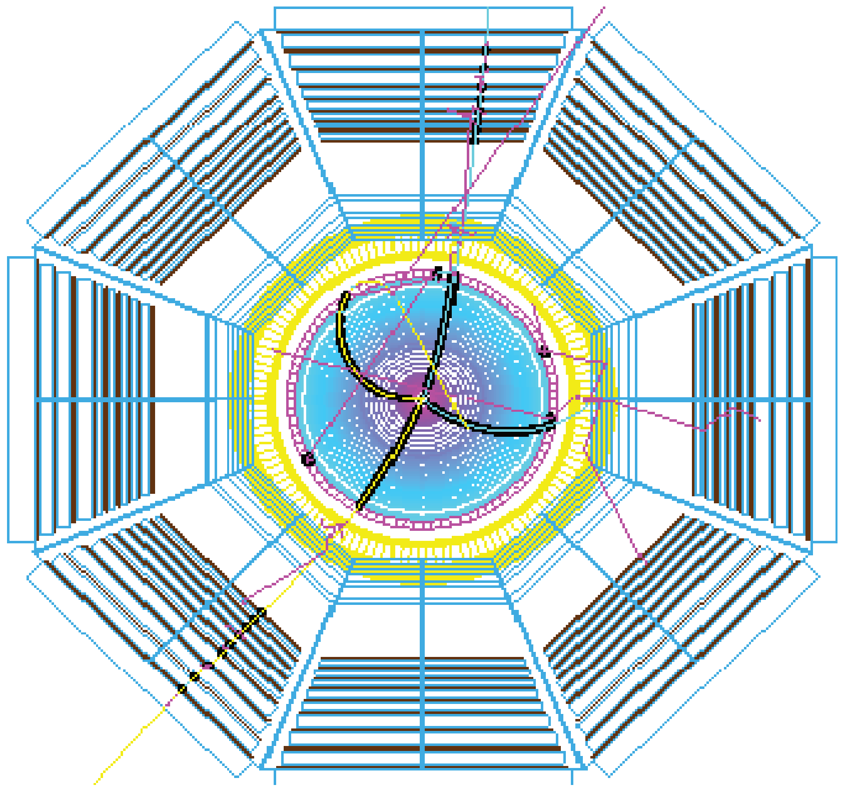
Have particles interact with the detector: Geant

Use the Geant4 software package - contains the collected community knowledge of particle interactions with matter - implemented as Monte Carlo

- Need a model of the detector geometry:
  - Where is there material? What is it exactly made of?
  - How do the magnetic fields look?
  - Where are there active elements?

- This will never be perfect: cables, cable ties, details of the mechanics, exact isotopic composition of things ...





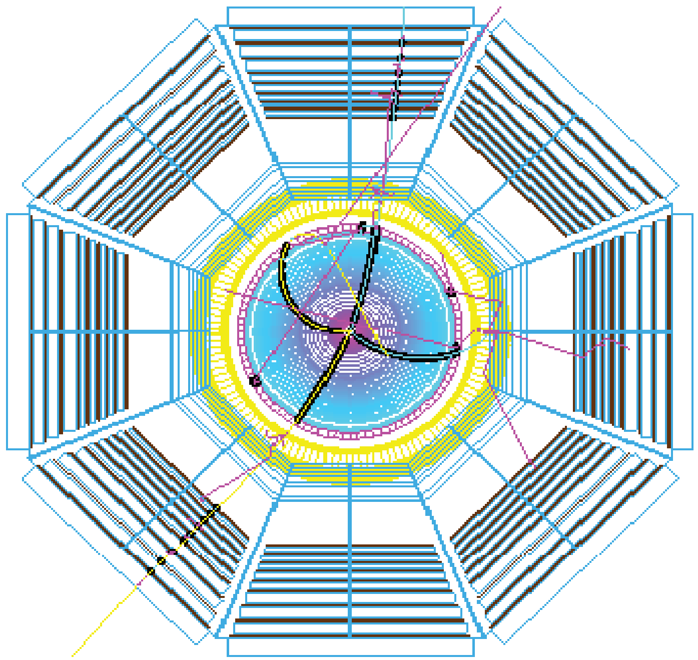
Have particles interact with the detector: Geant

Need to make a lot of choices for physics models

- Down to which range should secondary particles be created? tracked?
- How to treat Coulomb scattering in material? Scattering by scattering or with a multiple scattering parametrization?
- How to treat showers? Create and track all secondaries or parametrize? How about hadronic interactions?
- Should optical photons be generated and tracked in a scintillator?
- What to do with (thermal) neutrons?

All involve a trade-off between computing time and precision
 - after all we want to generate millions of events.

Often minutes of computing time per event even with simplified models

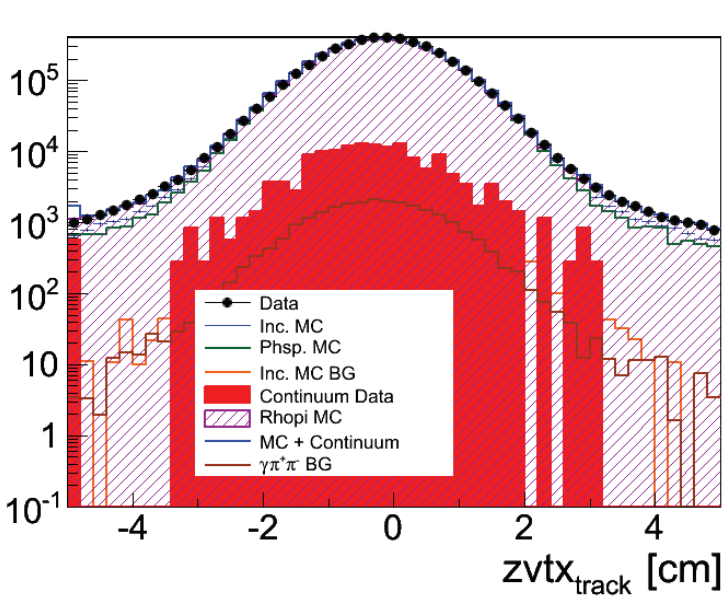


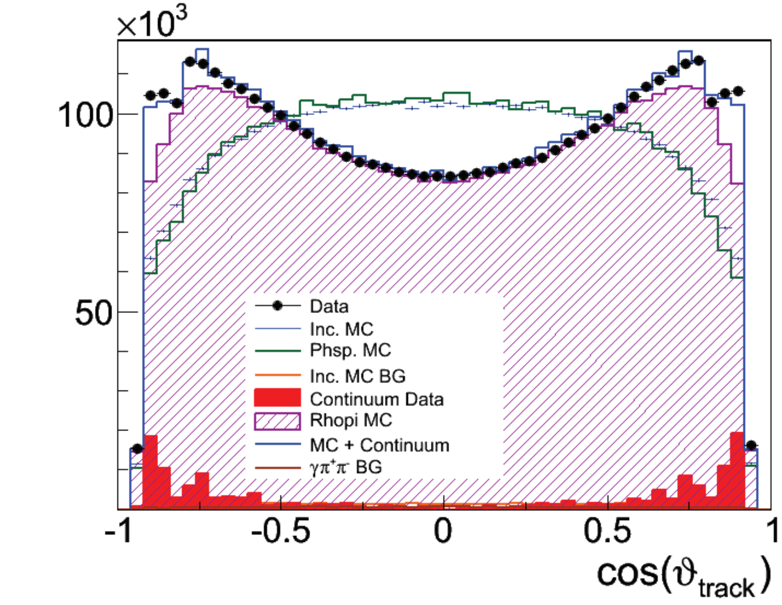
Niklaus

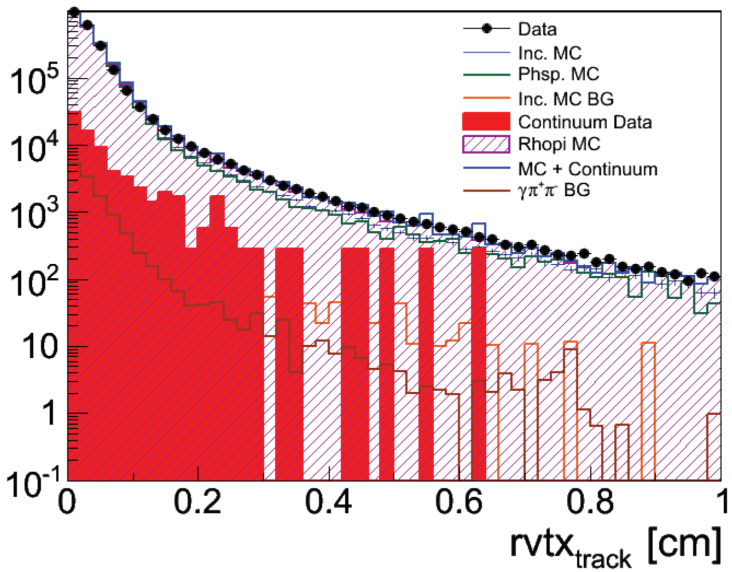
Simulate detector response

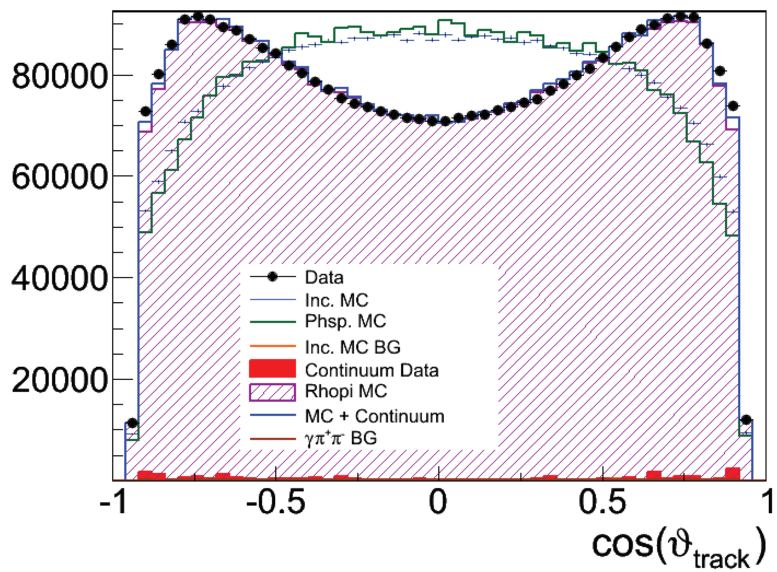
- Geant4 will give e.g. energy deposit in a drift chamber gas or number of photons in a scintillator
- We need to provide a model of charge (light) collection, amplifier response, electronics response, noise etc.
- Also very important: Model of detector defects - broken channels, misalignment etc.
- Output should be similar to real data, with the difference that we know what we started with
- Can then run the same reconstruction algorithms

And then compare with data

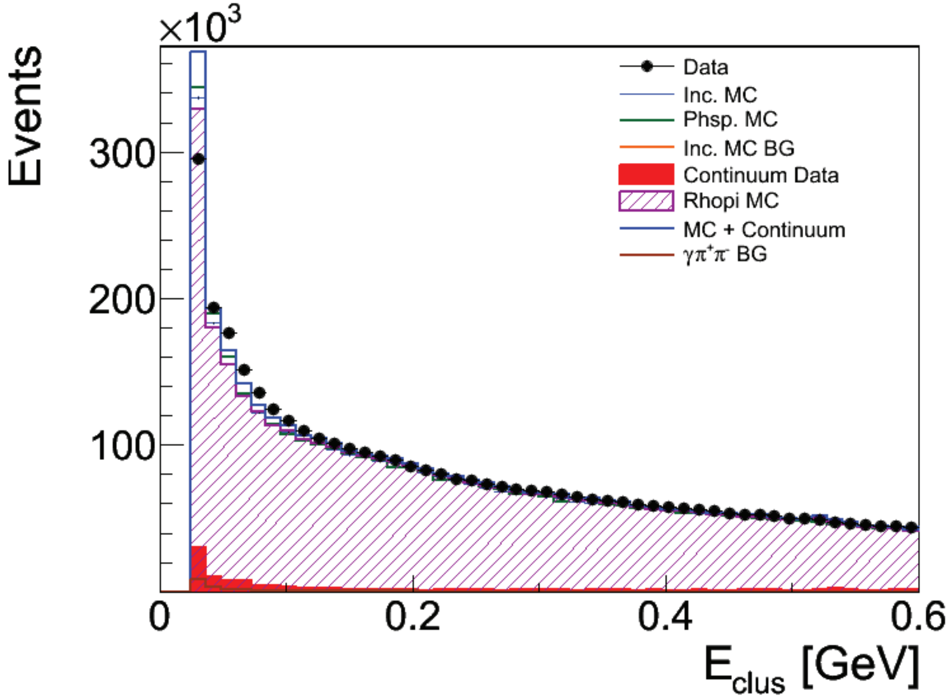


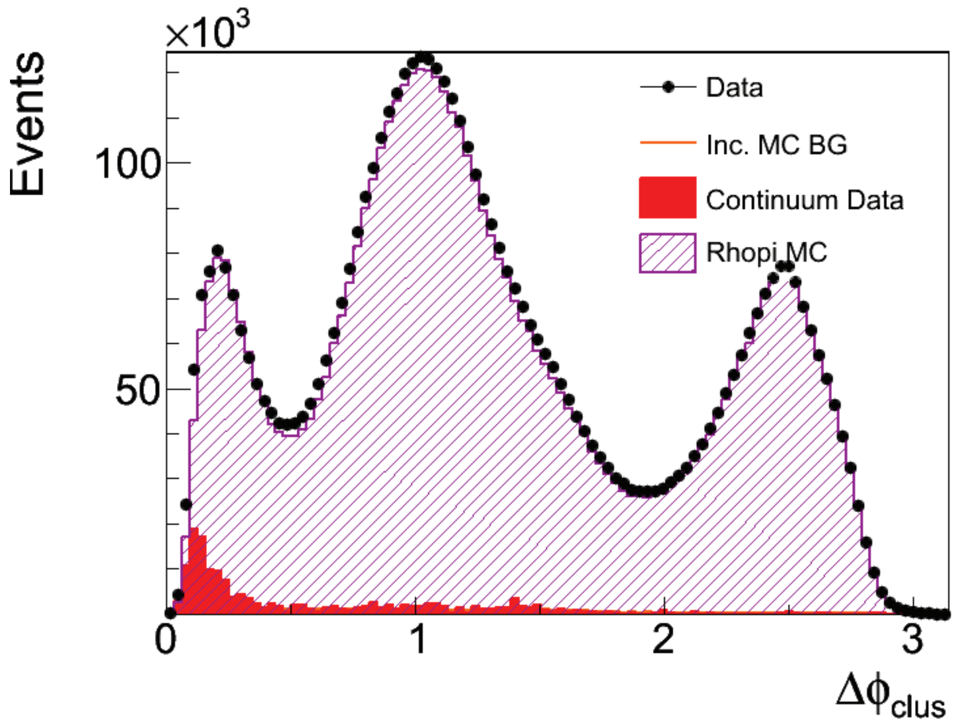






And keep comparing...





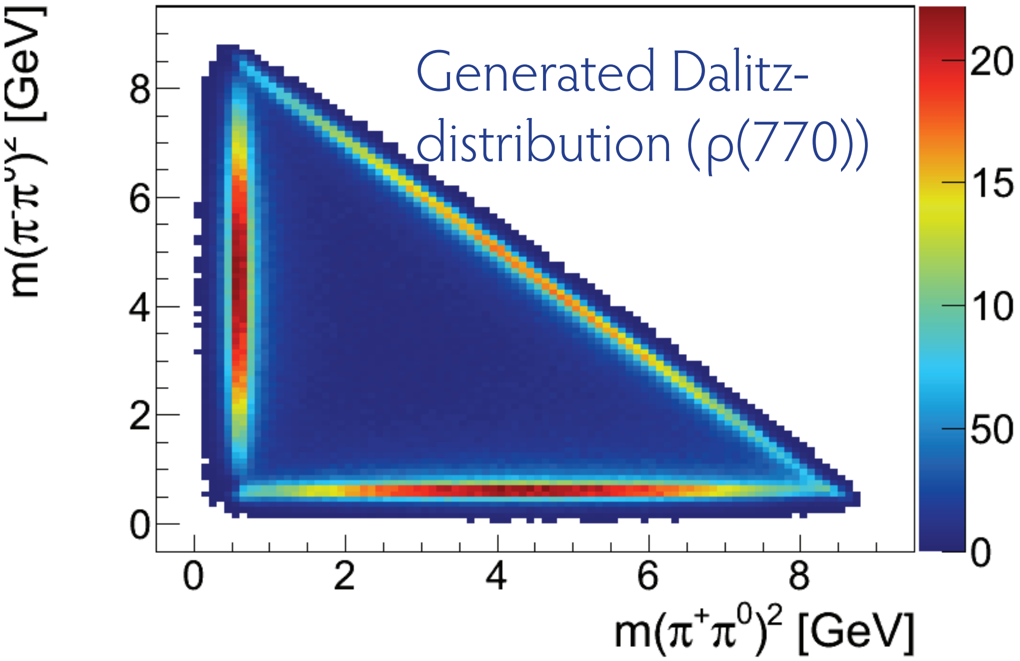
- Keep fixing the simulation, cleaning up the data and removing bugs from the code until you have "reasonable agreement"
- Remaining differences enter the "systematic error" - more discussion later

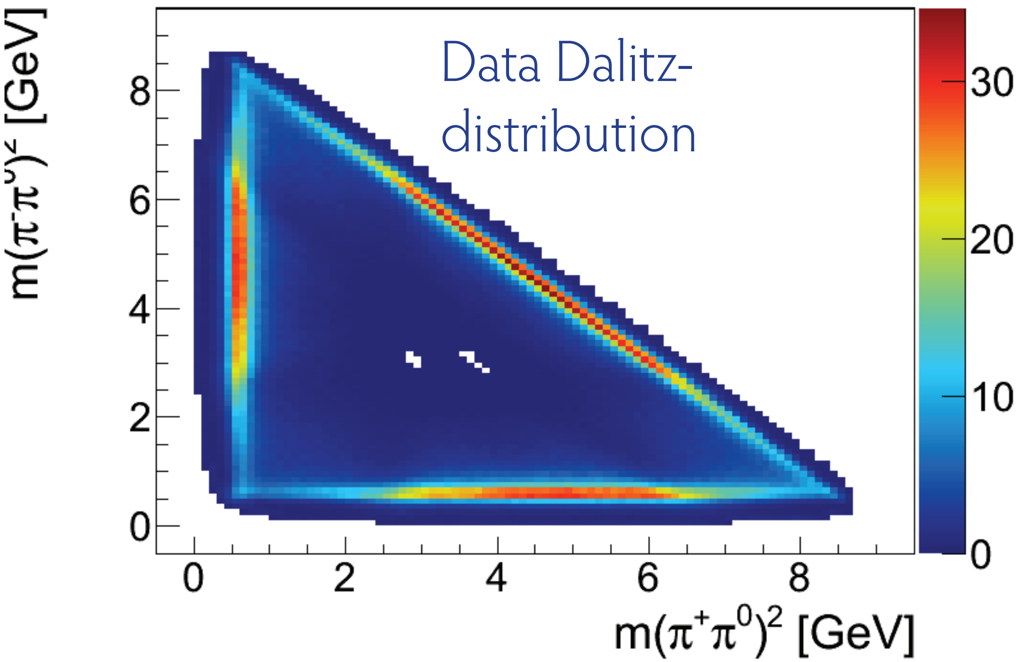
As often as possible, use data input

Here data input was used:

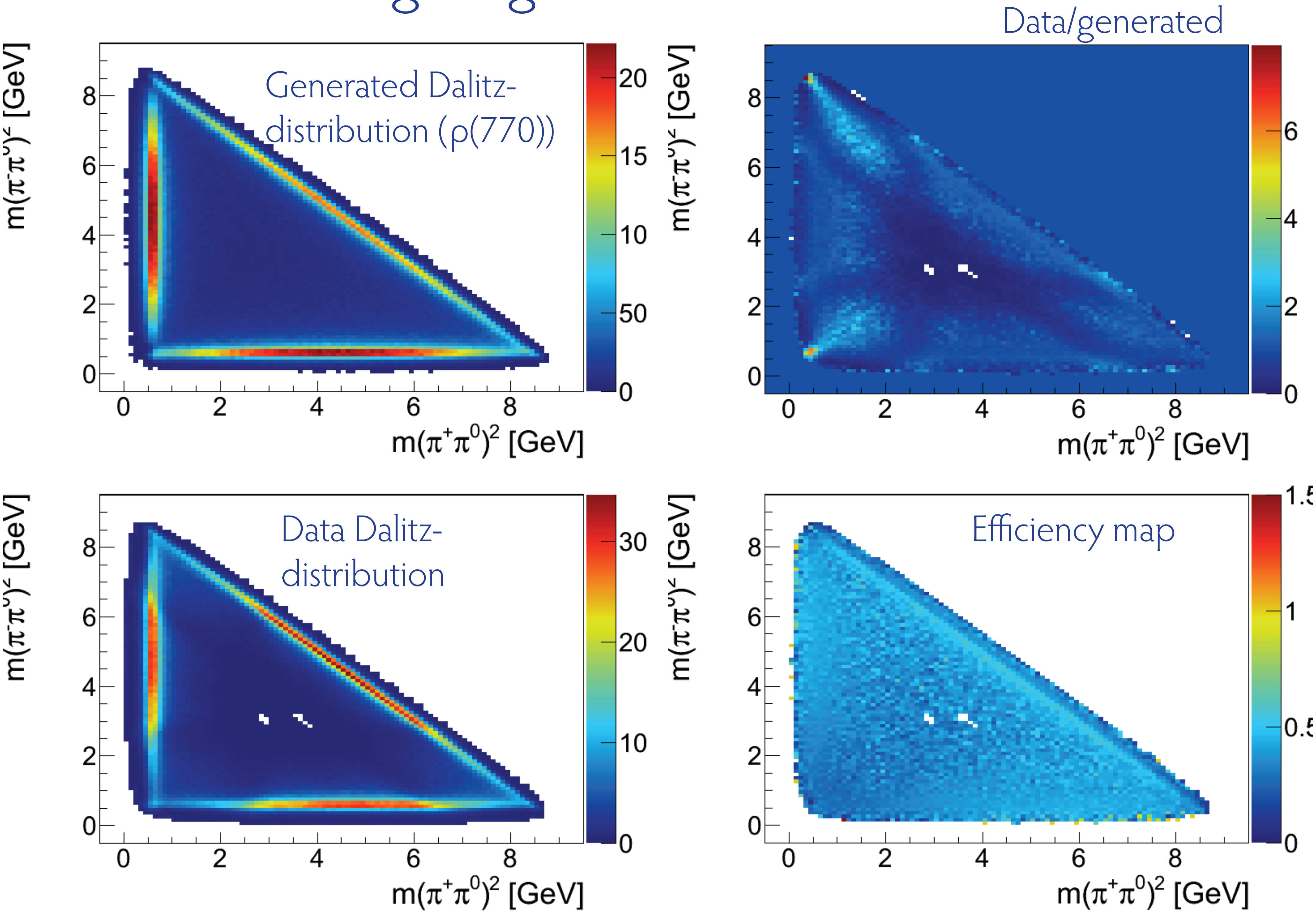
- For the continuum background ($e^+e^- \rightarrow \pi^+\pi^-\pi^0$ without a J/Ψ) by detuning the beam
- For the dynamics of the decay...

Kinematic re-weighting

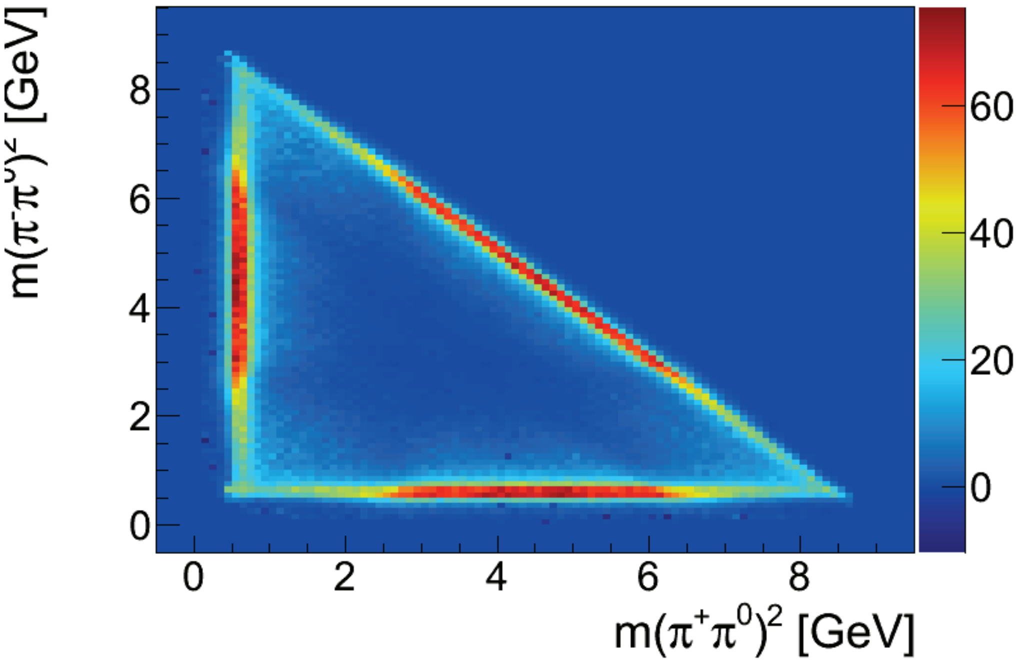




Kinematic re-weighting



After weight



- Get back "true" distribution
- Only works if there are data events
- Corrections for totally inefficient regions need to come from model

$$BF = \frac{N_{sel} - N^{BG}_{continuum} - N^{BG}_{resonance}}{N_{\psi} \cdot \epsilon_{MC} \cdot \epsilon_{trig} \cdot BF(\pi^0 \to \gamma\gamma)}$$

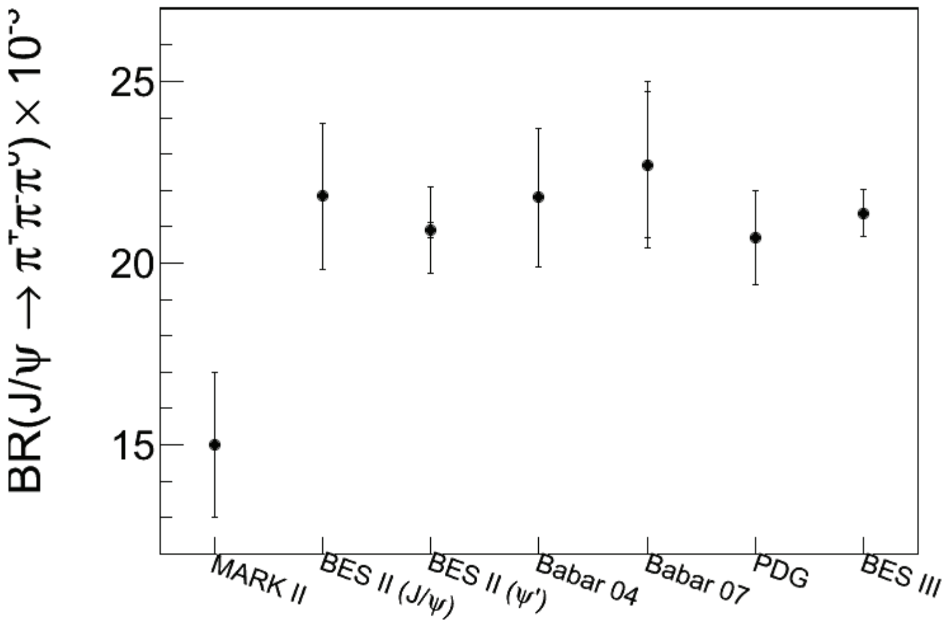
Results

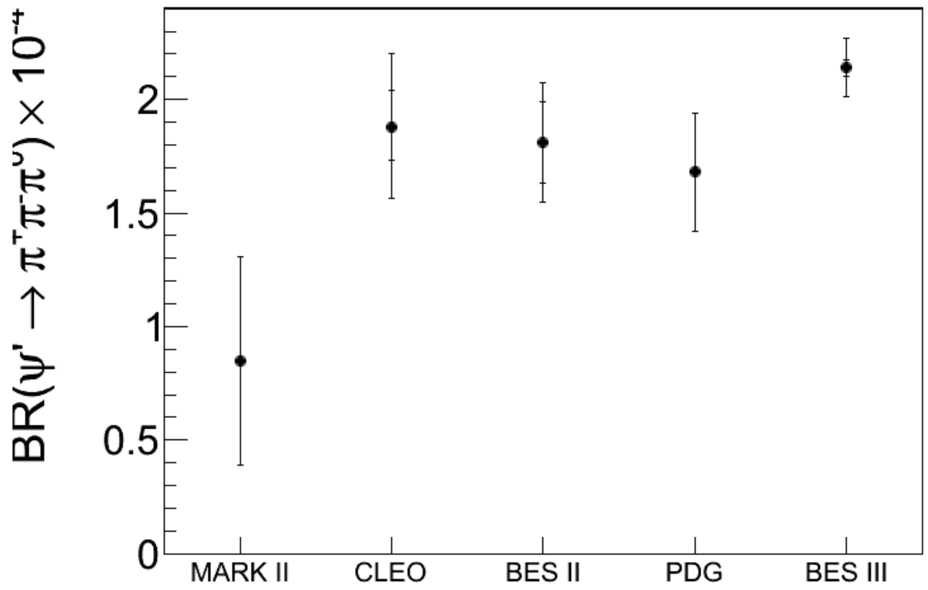
The branching fraction for $J/\psi \longrightarrow \pi^+\pi^-\pi^0$ is measured to be

$$(2.137 \pm 0.004(stat.)^{+0.064}_{-0.062}(syst.)) \times 10^{-3},$$

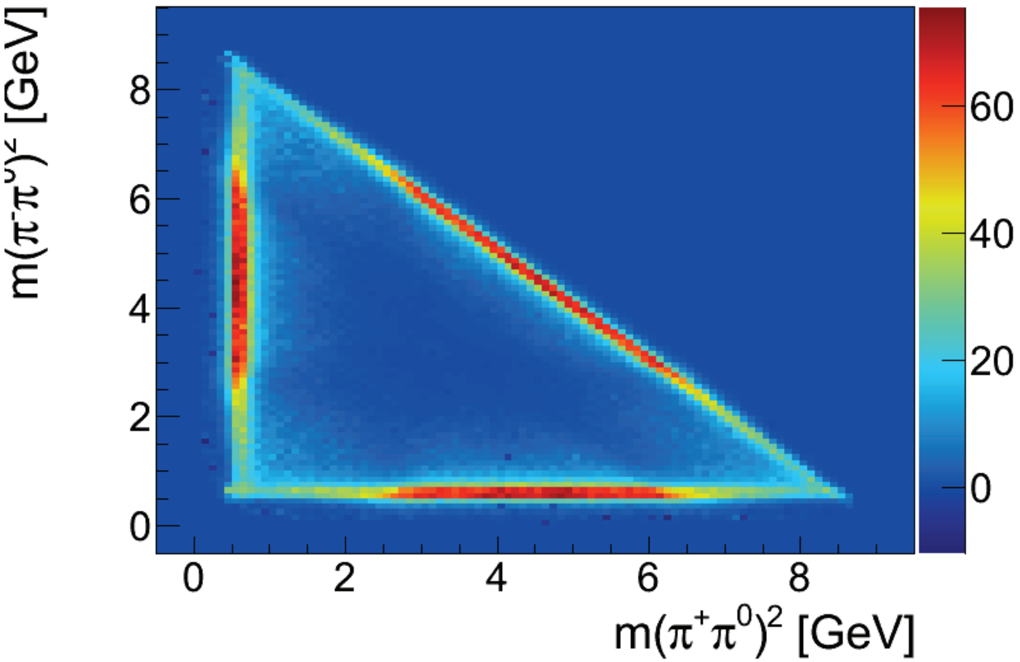
The branching fraction for $\psi(2S) \longrightarrow \pi^+\pi^-\pi^0$ is measured to be

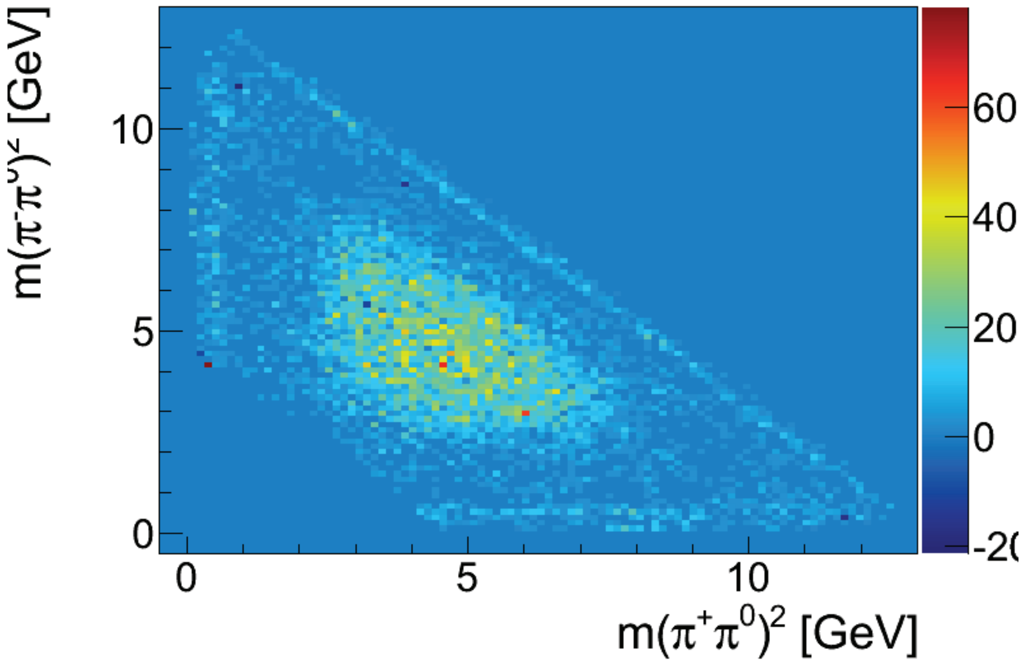
$$(2.14 \pm 0.03(stat.)^{+0.12}_{-0.11}(syst.)) \times 10^{-4},$$



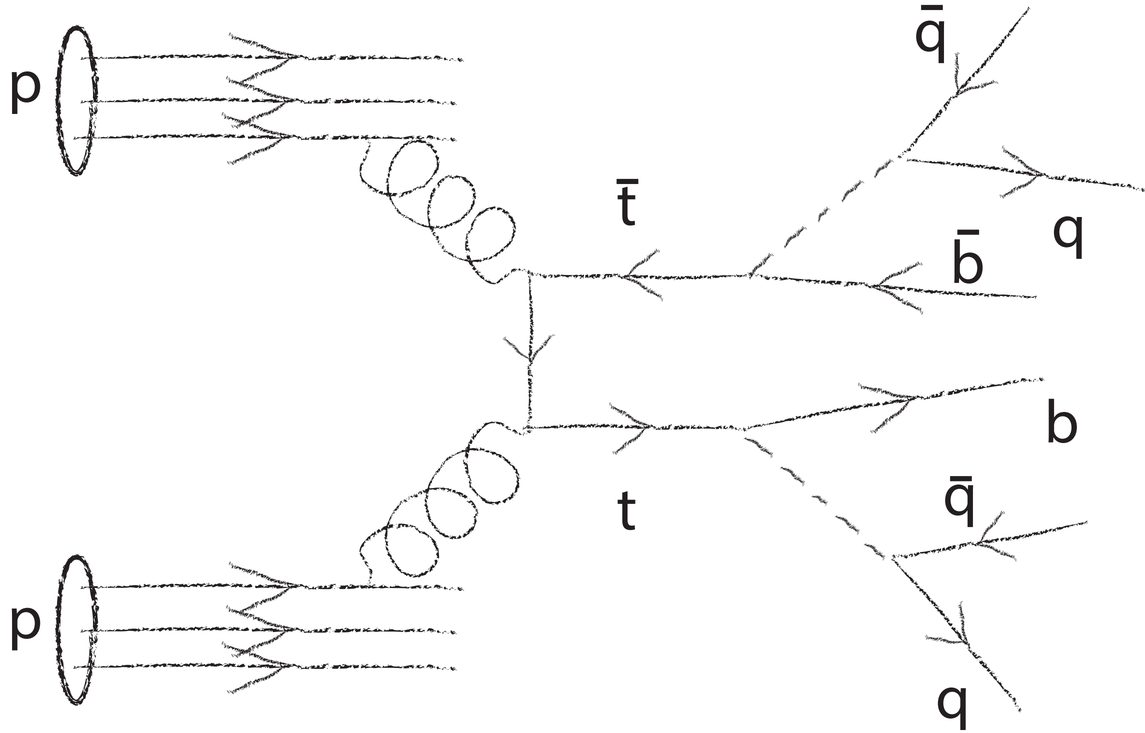


And the Dalitz Plots... - the puzzle is still puzzling



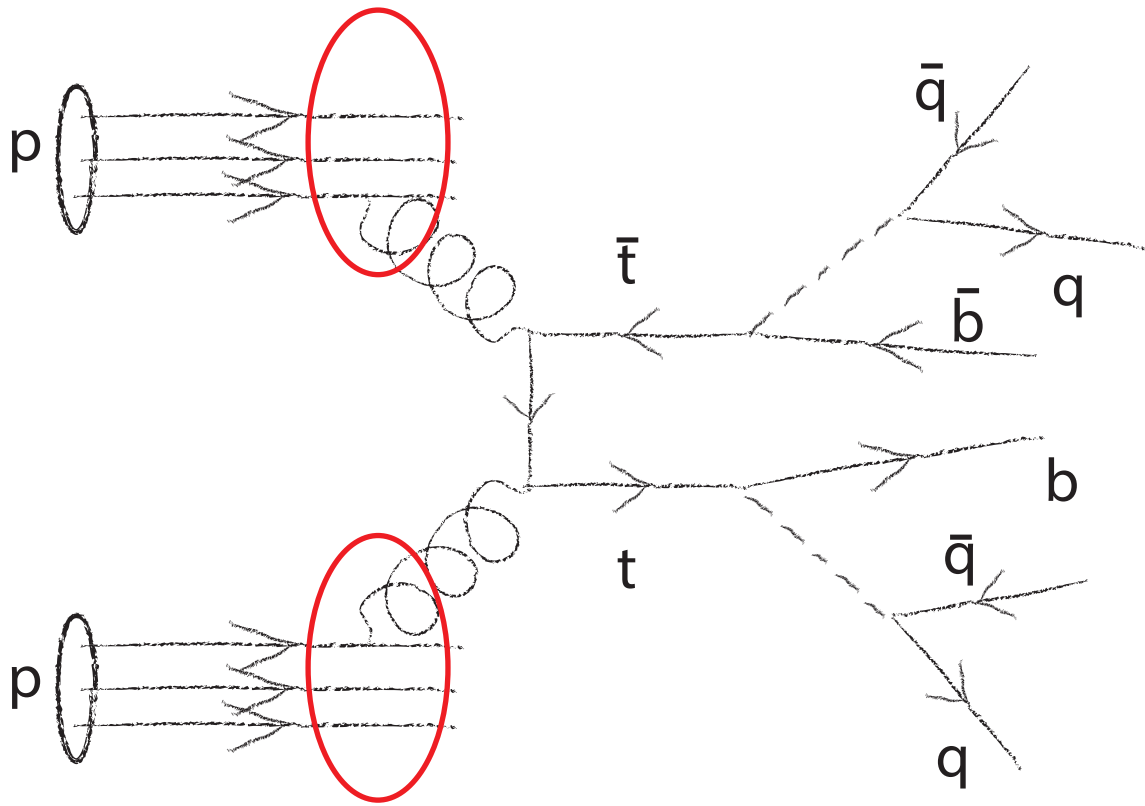


4.9. Monte Carlo for LHC



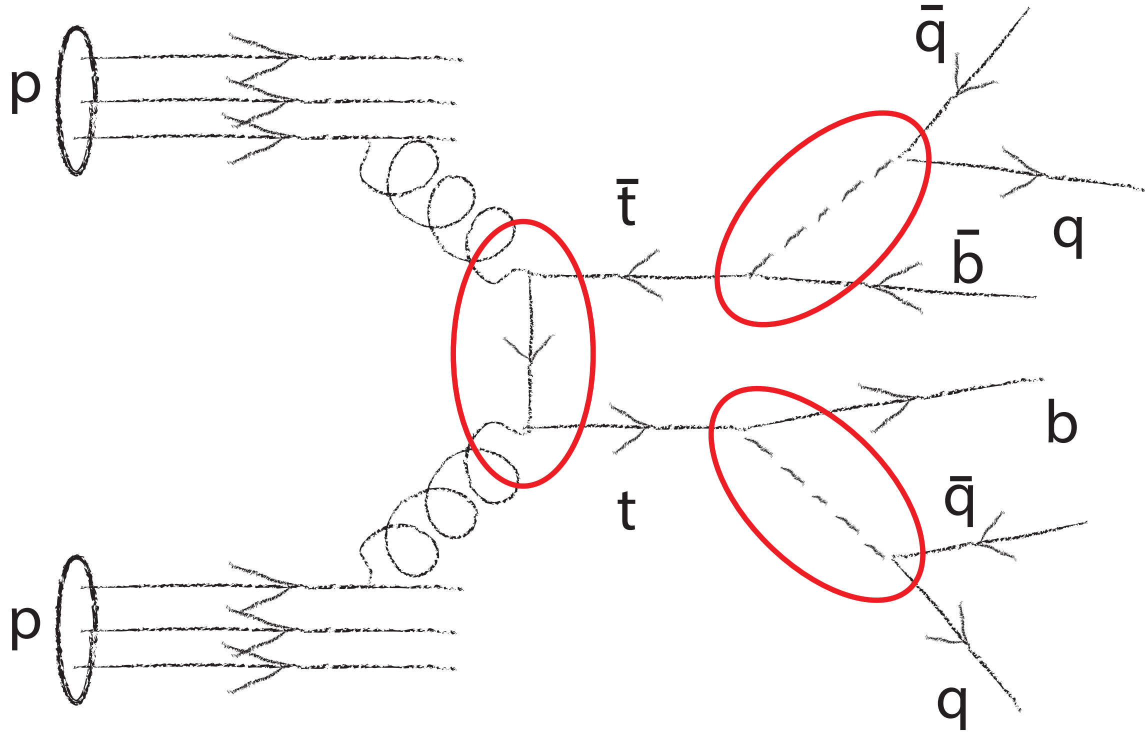
Monte Carlo for LHC

- Randomly draw partons from the protons according to parton density functions



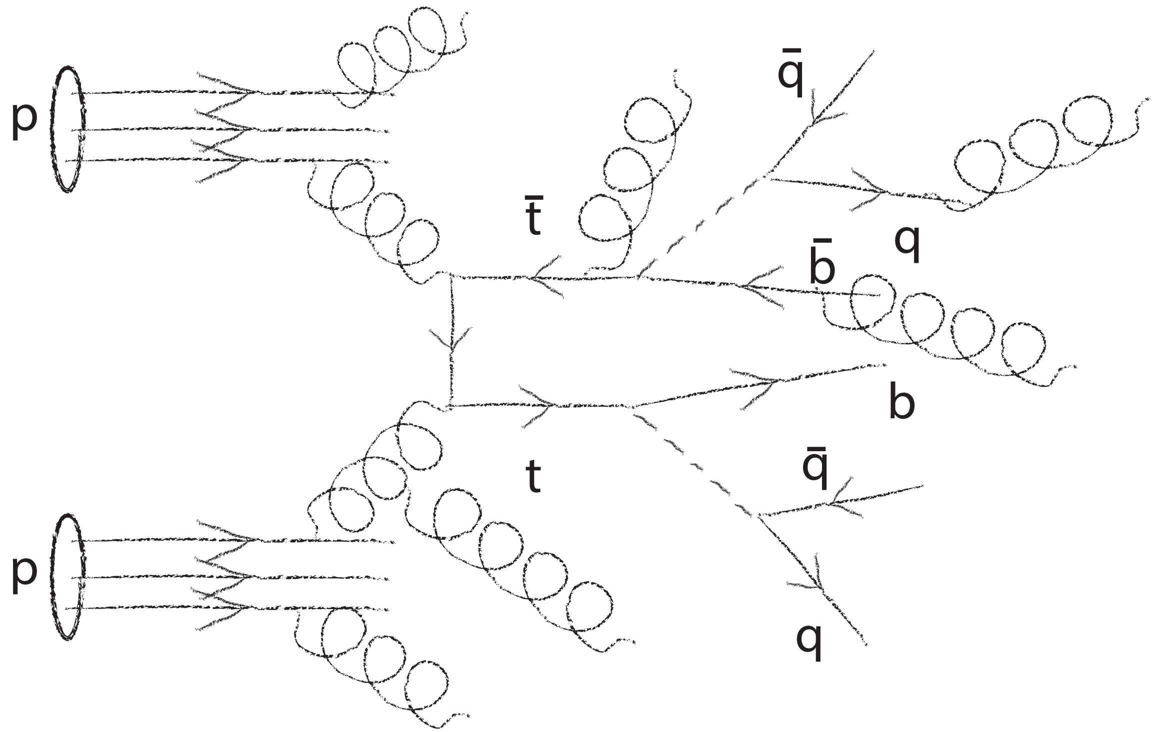
Monte Carlo for LHC

- Randomly draw partons from the protons according to parton density functions
- Usually have a matrix element for the hard process(es) - generate phase space and use accept/reject



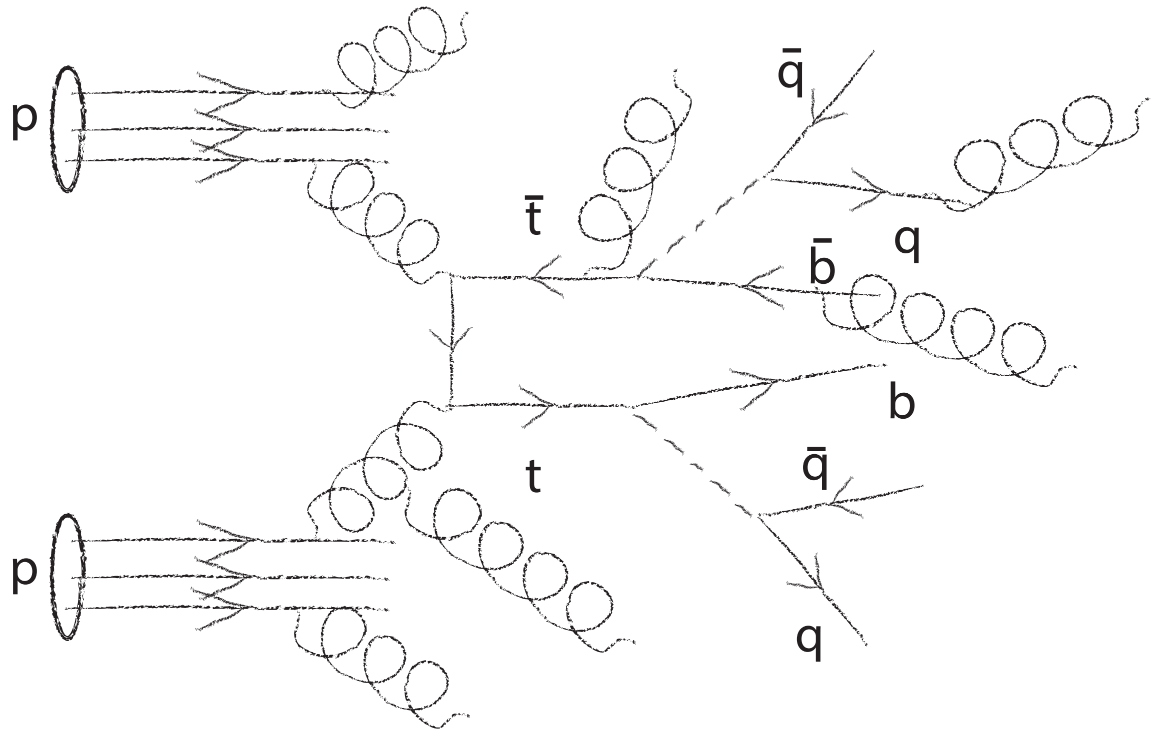
Monte Carlo for LHC

- Randomly draw partons from the protons according to parton density functions
- Usually have a matrix element for the hard process(es) - generate phase space and use accept/reject
- Almost everything can radiate gluons - generate them via MC



Monte Carlo for LHC

- Randomly draw partons from the protons according to parton density functions
- Usually have a matrix element for the hard process(es) - generate phase space and use accept/reject
- Almost everything can radiate gluons - generate them via MC
- And in the end everything has to be colour-neutral hadrons; usually use Pythia MC code to simulate that
- Also have to deal with the remains of the proton (underlying event)
- Course by T. Plehn on all this



Pileup

Many collisions at the same time

- Also have to simulate these pile-up events
- Simulation uses minutes/event