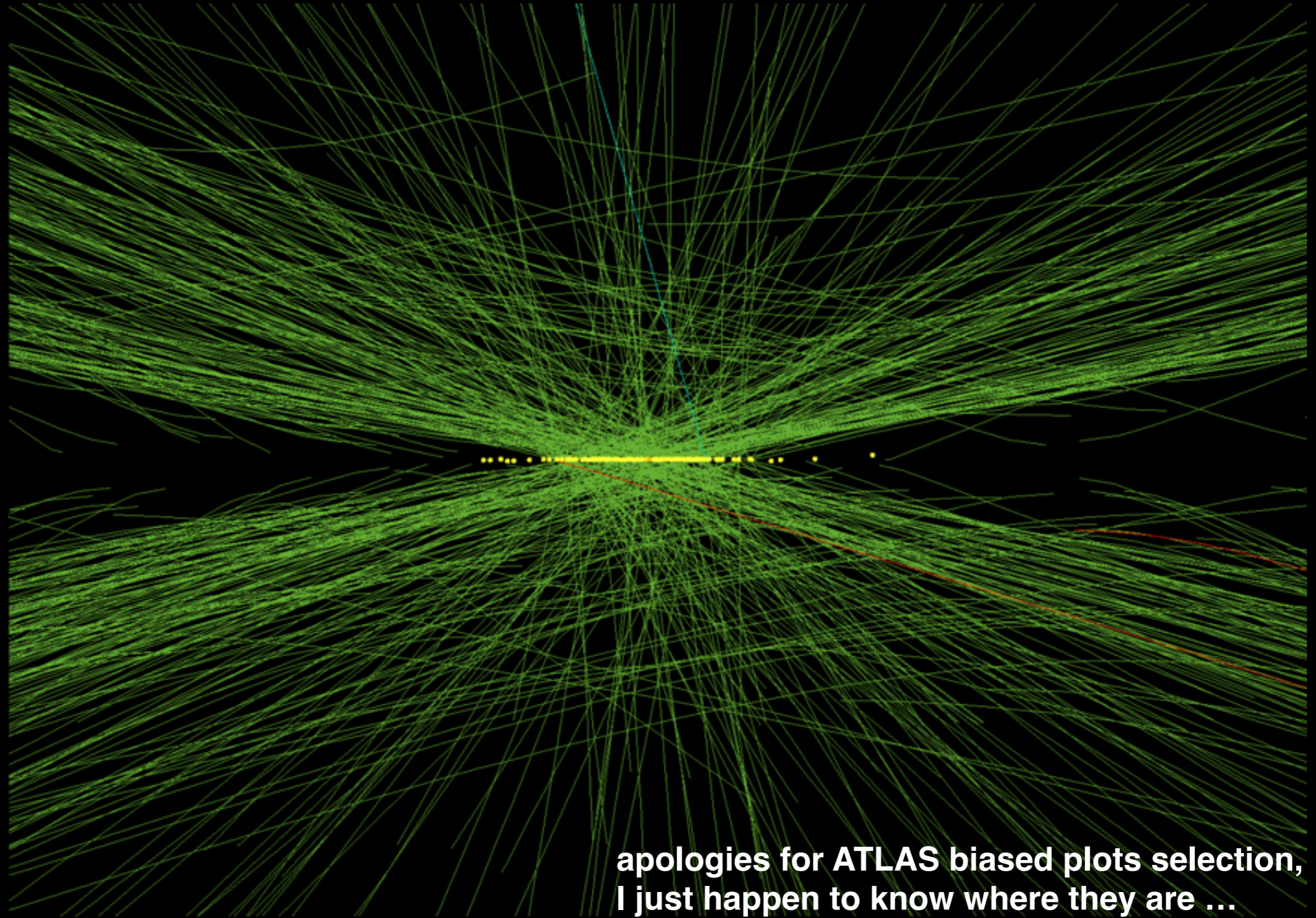
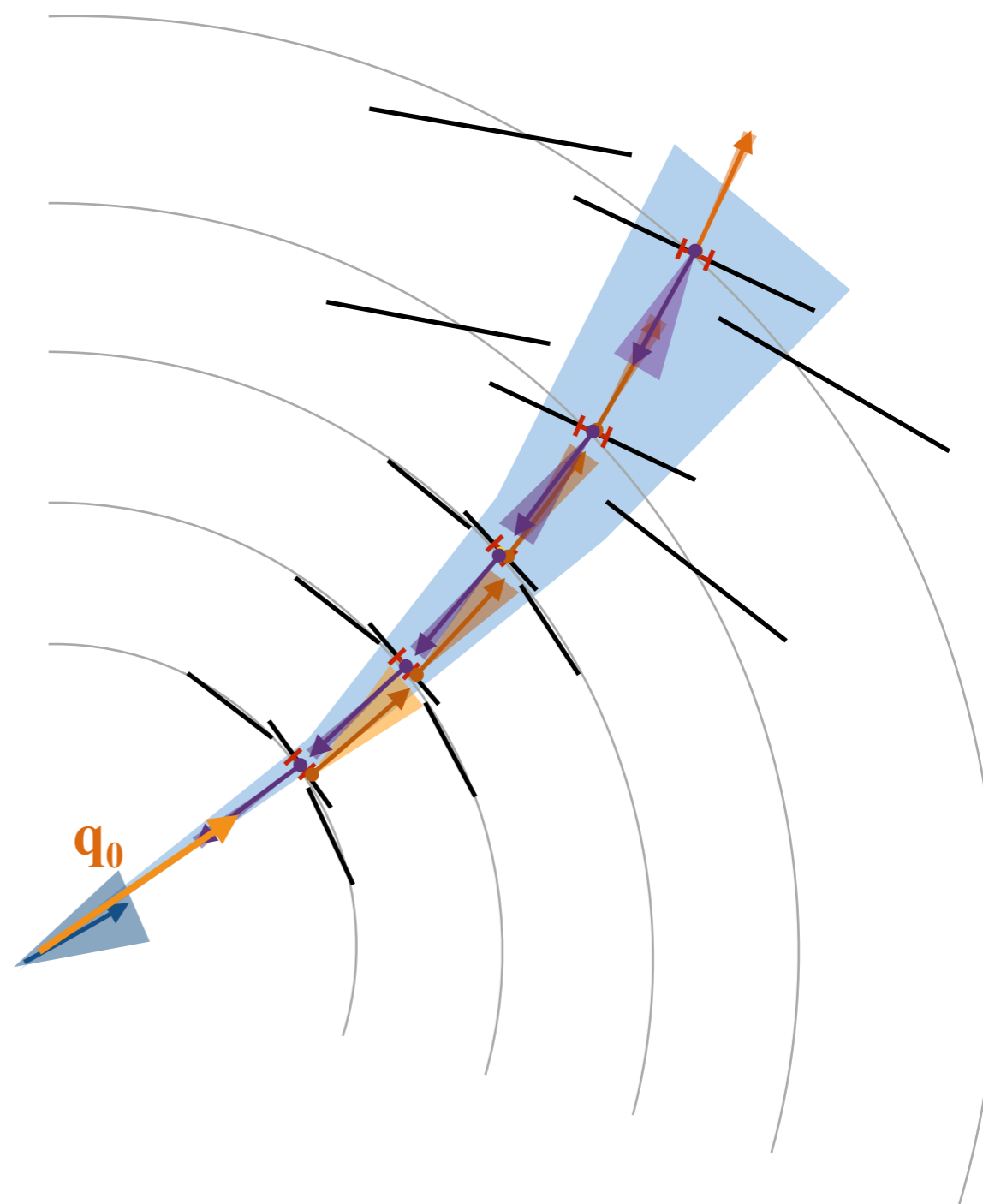


Part 3 - Vertex reconstruction, analysis & reality

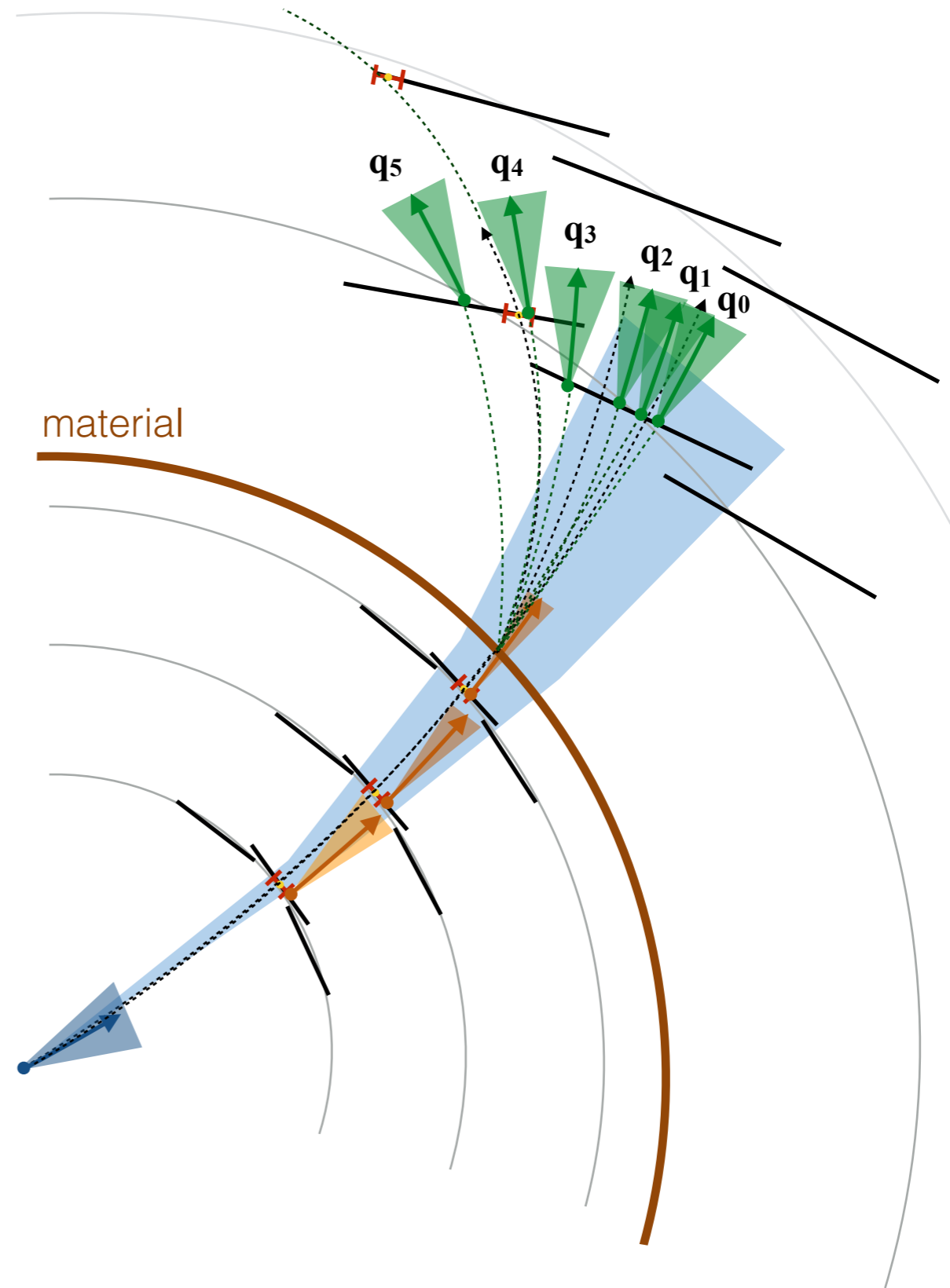


Recap of yesterday

- ▶ We've found tracks
 - global and local pattern recognition algorithms
- ▶ We've fitted those tracks
 - least squares estimator fit, e.g. global χ^2 minimization, Kalman filter
- ▶ Discussed the fit output
- ▶ Touched upon "ghost tracks"
 - we will hear a bit more about that though
- ▶ Dedicated electron fitting

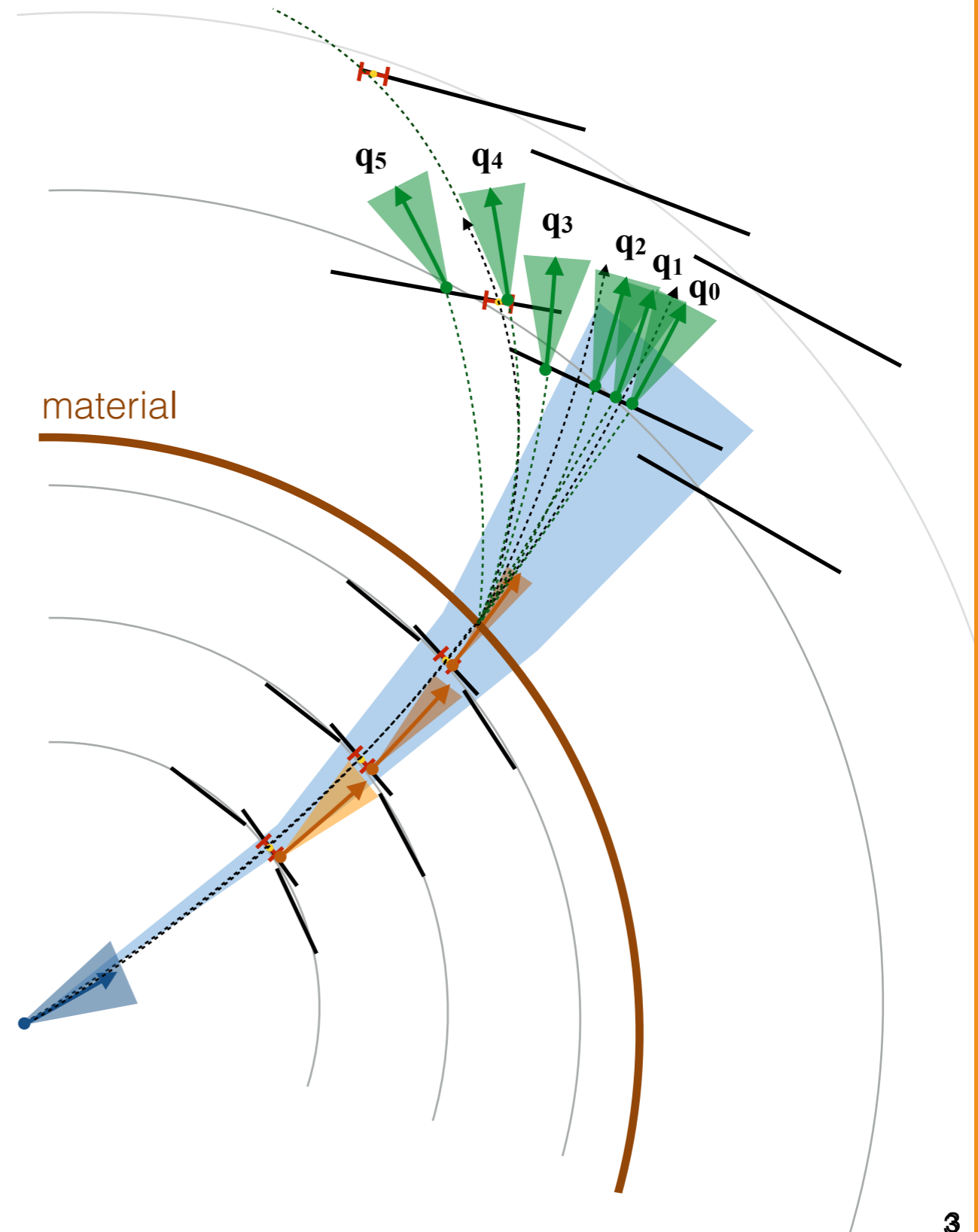
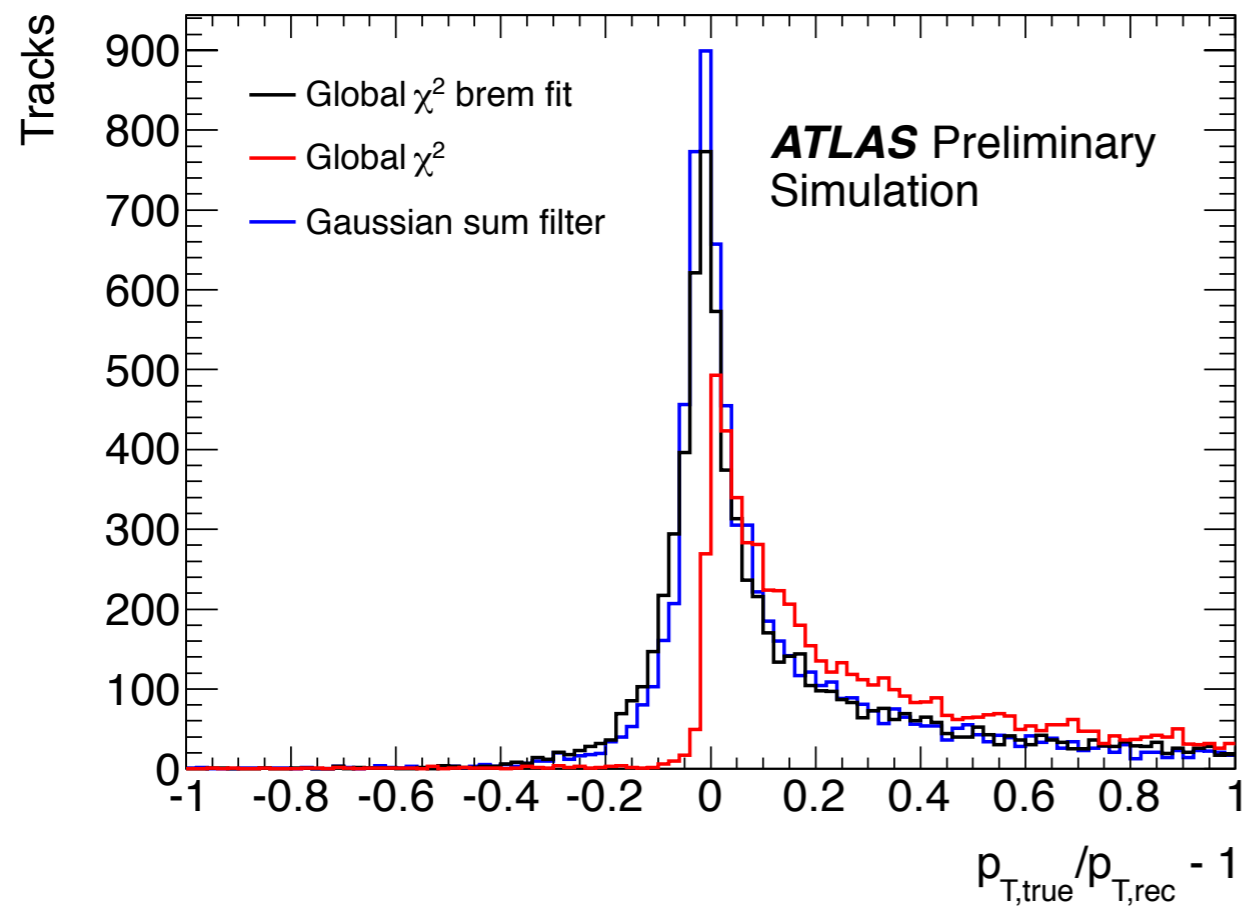


Dedicated electron fitting



Dedicated electron fitting

- ▶ It works !

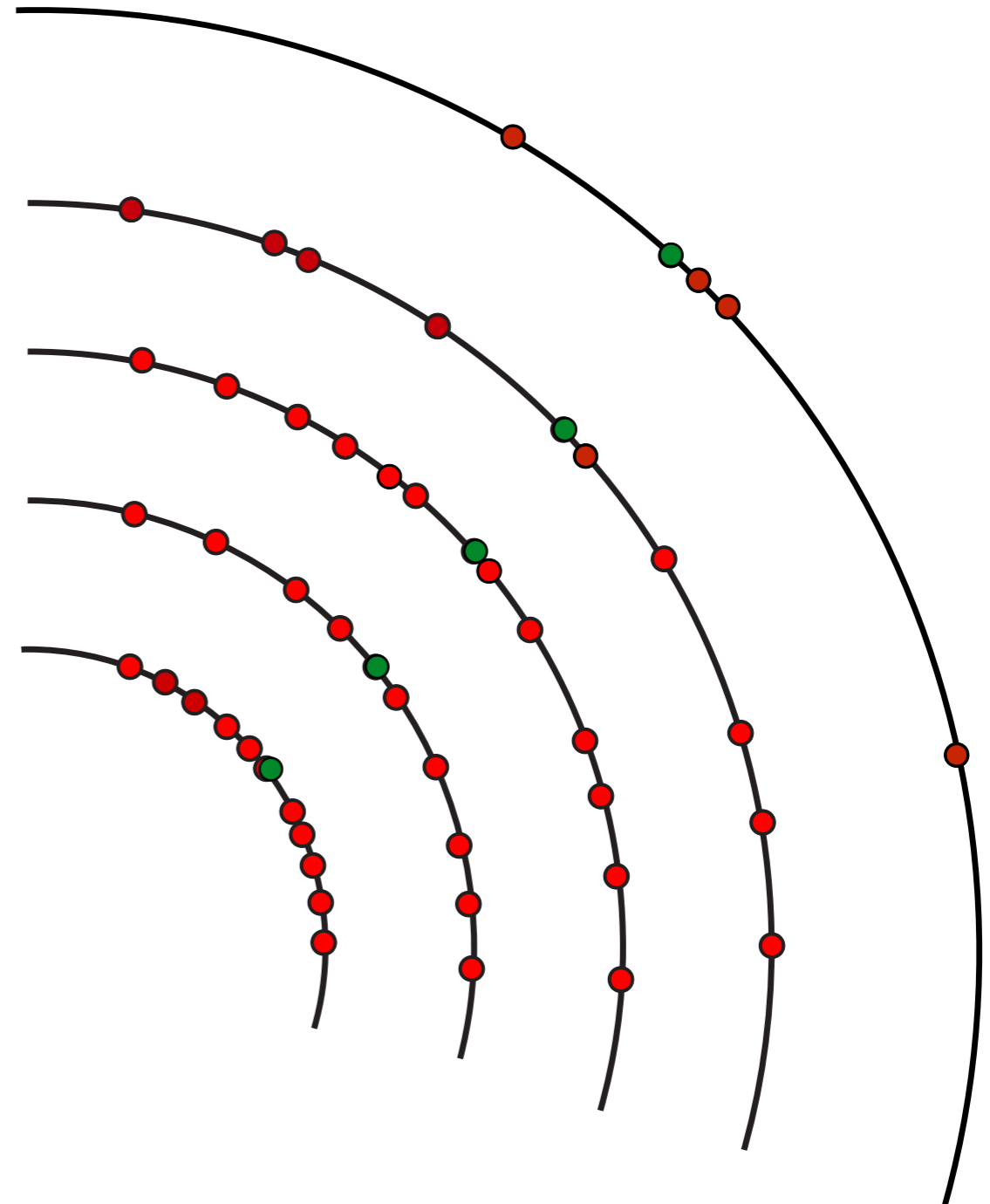


Revisiting enemy No. 2: **ghosts**

- ▶ So far we discussed “fakes” (ghost tracks) at seed level

good track

not so good track



Revisiting enemy No. 2: **ghosts**

- ▶ So far we discussed “fakes” (ghost tracks) at seed level

good track

many compatible hits

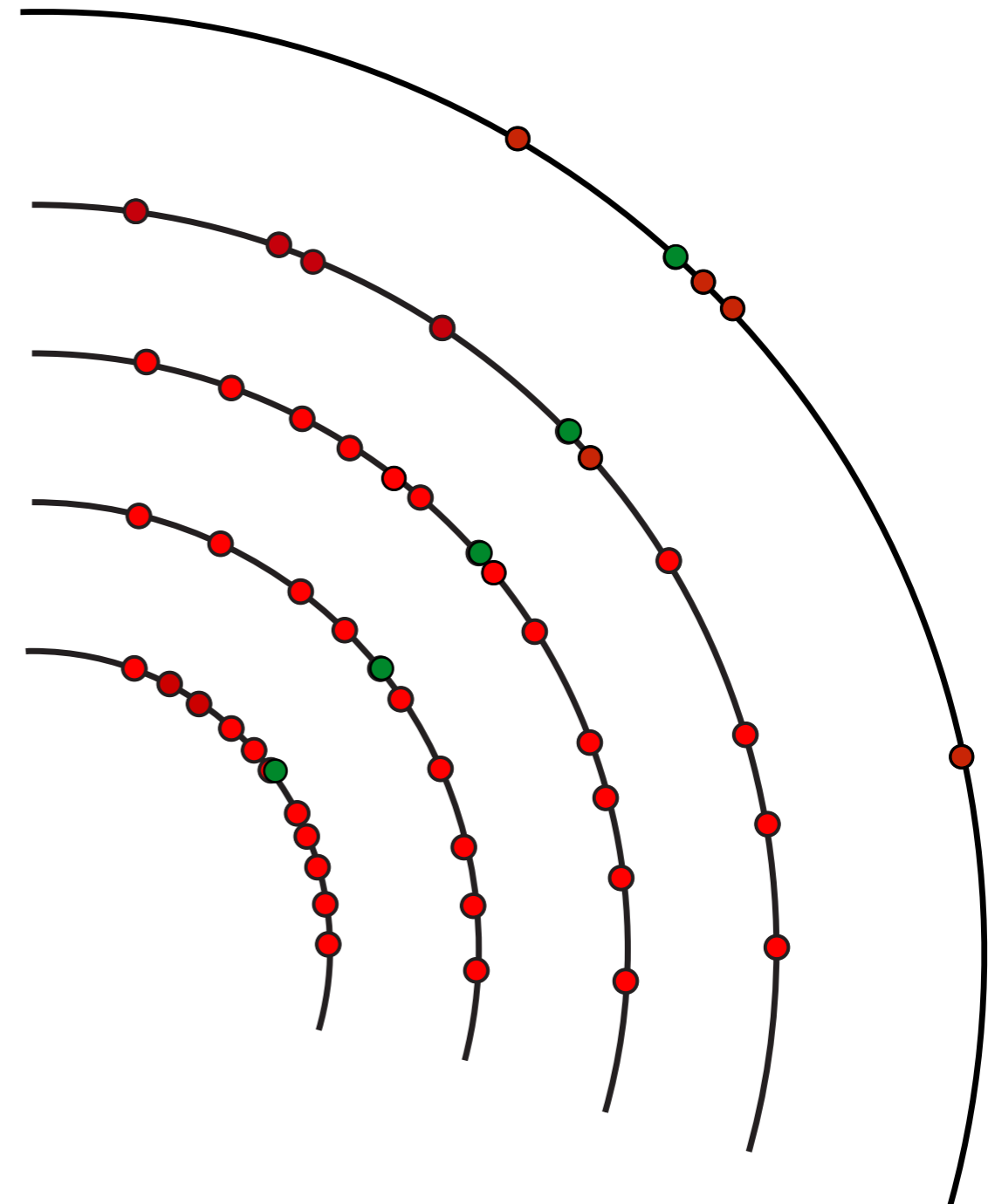
completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

not so good track



Revisiting enemy No. 2: **ghosts**

- ▶ So far we discussed “fakes” (ghost tracks) at seed level

good track

many compatible hits

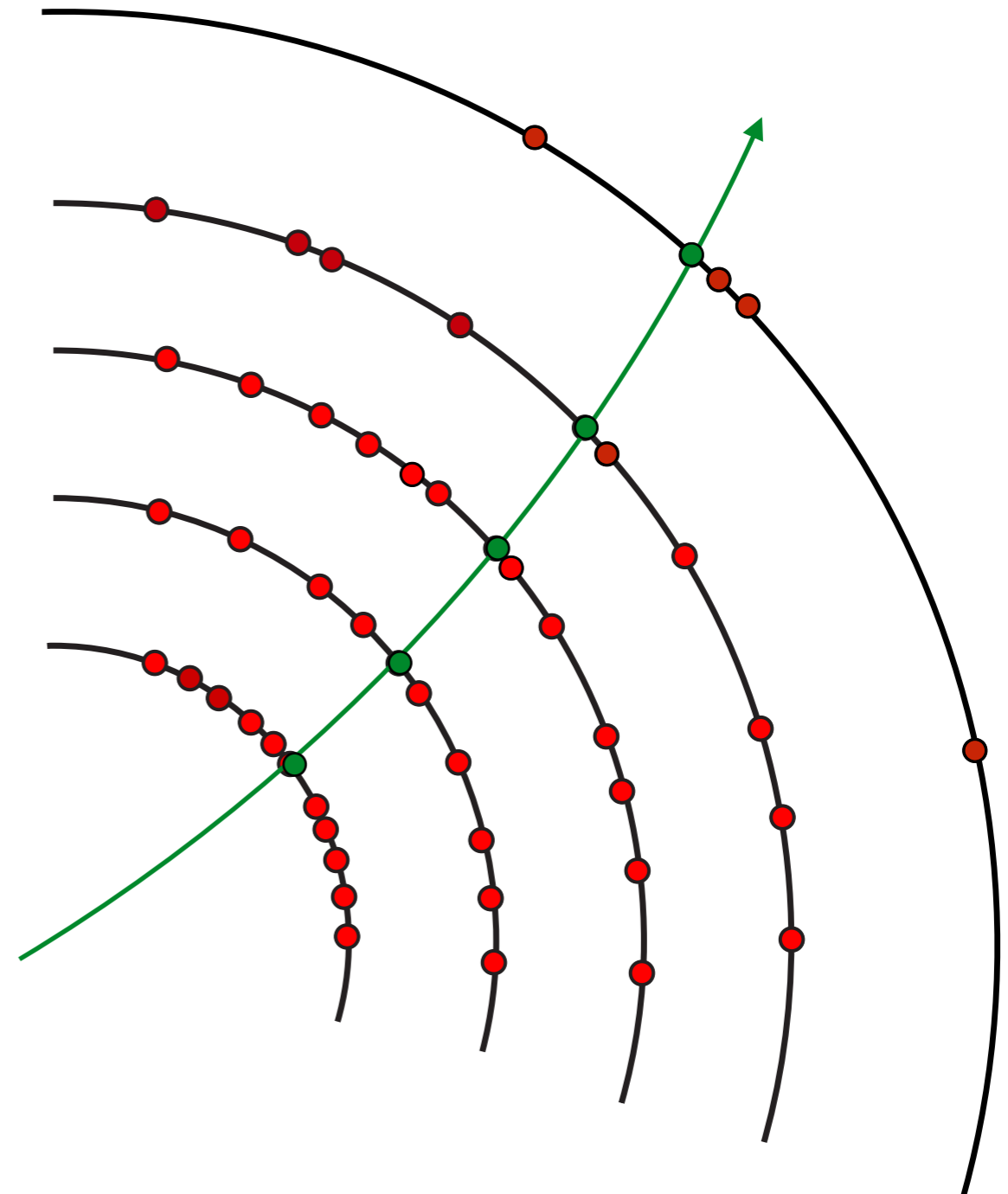
completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

not so good track



Revisiting enemy No. 2: **ghosts**

- ▶ So far we discussed “fakes” (ghost tracks) at seed level

good track

many compatible hits

completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

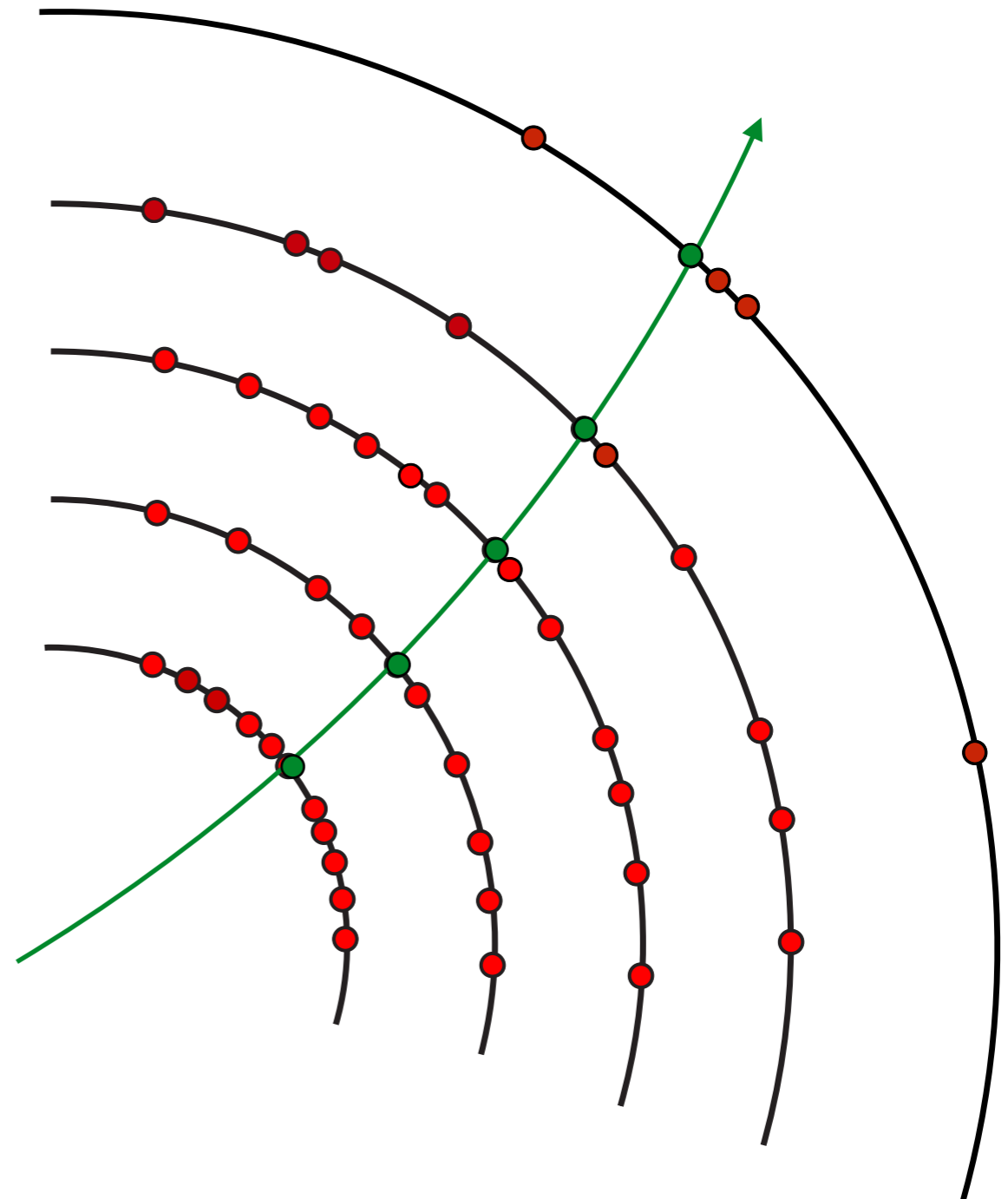
not so good track

short tracks

holes

shared hits

bad fit quality,
outliers



Revisiting enemy No. 2: **ghosts**

- ▶ So far we discussed “fakes” (ghost tracks) at seed level

good track

many compatible hits

completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

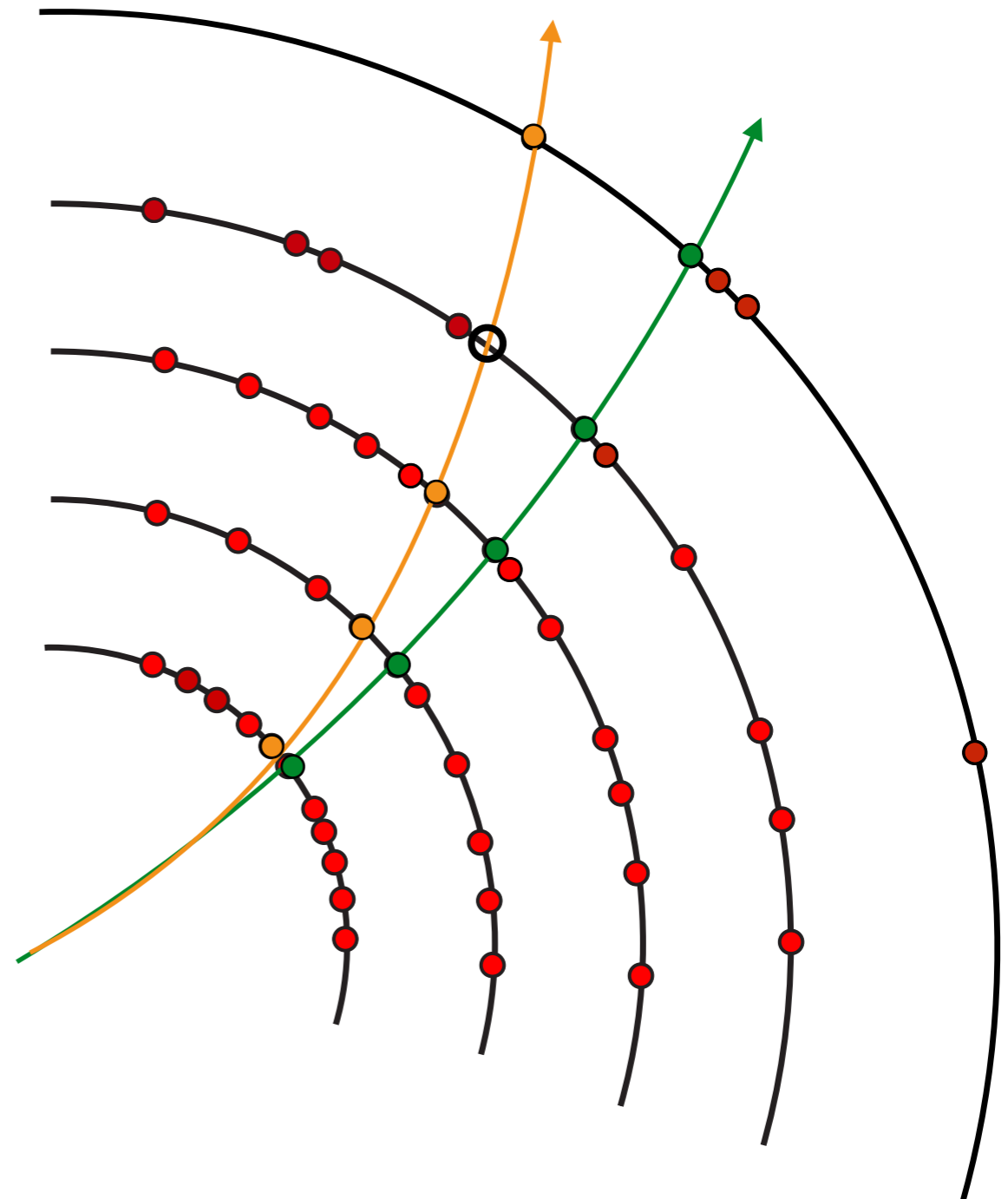
not so good track

short tracks

holes

shared hits

bad fit quality,
outliers



Revisiting enemy No. 2: **ghosts**

- ▶ So far we discussed “fakes” (ghost tracks) at seed level

good track

many compatible hits

completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

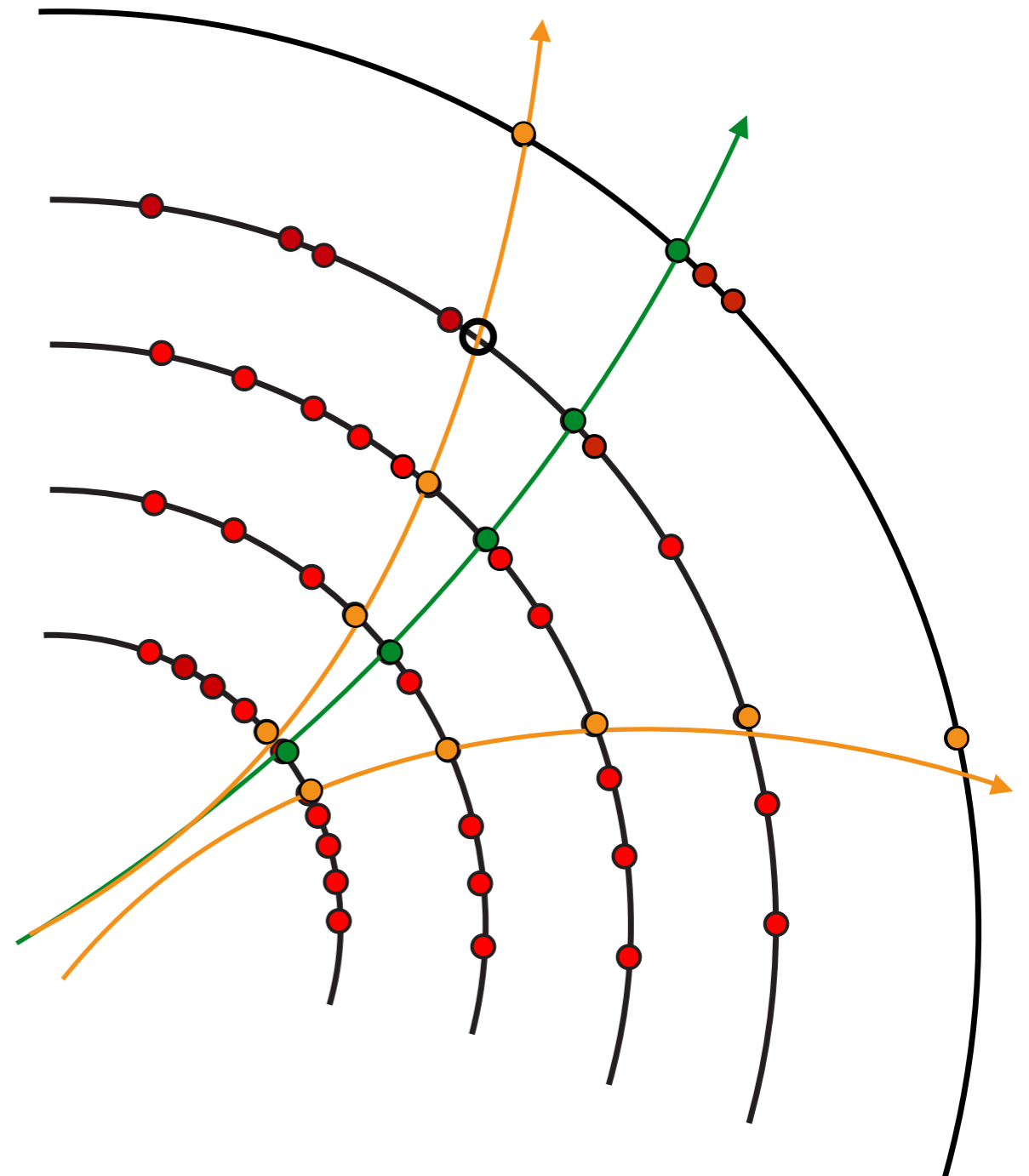
not so good track

short tracks

holes

shared hits

bad fit quality,
outliers



Revisiting enemy No. 2: **ghosts**

- ▶ So far we discussed “fakes” (ghost tracks) at seed level

good track

many compatible hits

completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

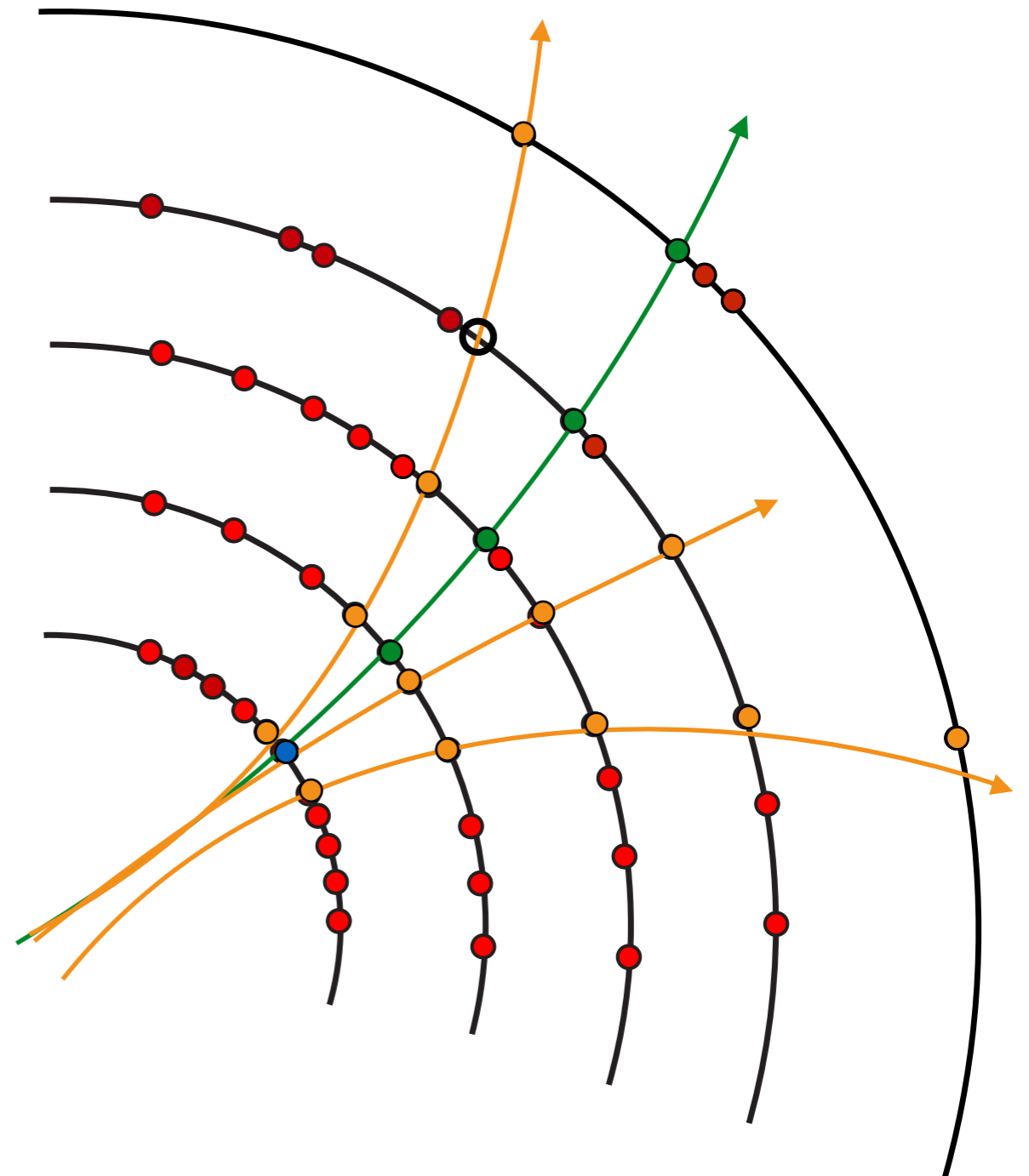
not so good track

short tracks

holes

shared hits

bad fit quality,
outliers



Revisiting enemy No. 2: **ghosts**

- ▶ So far we discussed “fakes” (ghost tracks) at seed level

good track

many compatible hits

completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

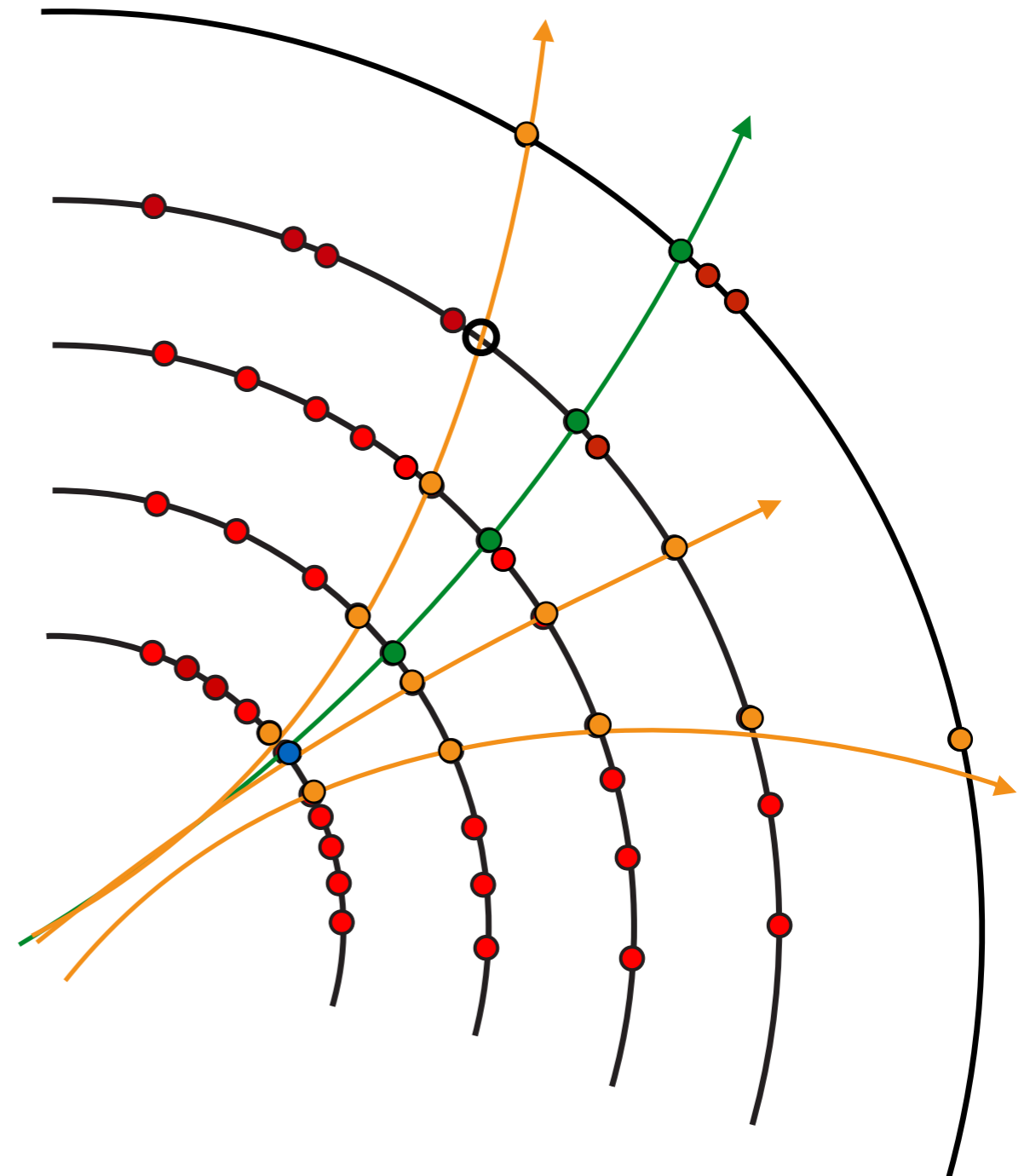
not so good track

short tracks

holes

shared hits

bad fit quality,
outliers



- ▶ Can we be sure ?

Revisiting enemy No. 2: **ghosts**

- ▶ So far we discussed “fakes” (ghost tracks) at seed level

good track

many compatible hits

completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

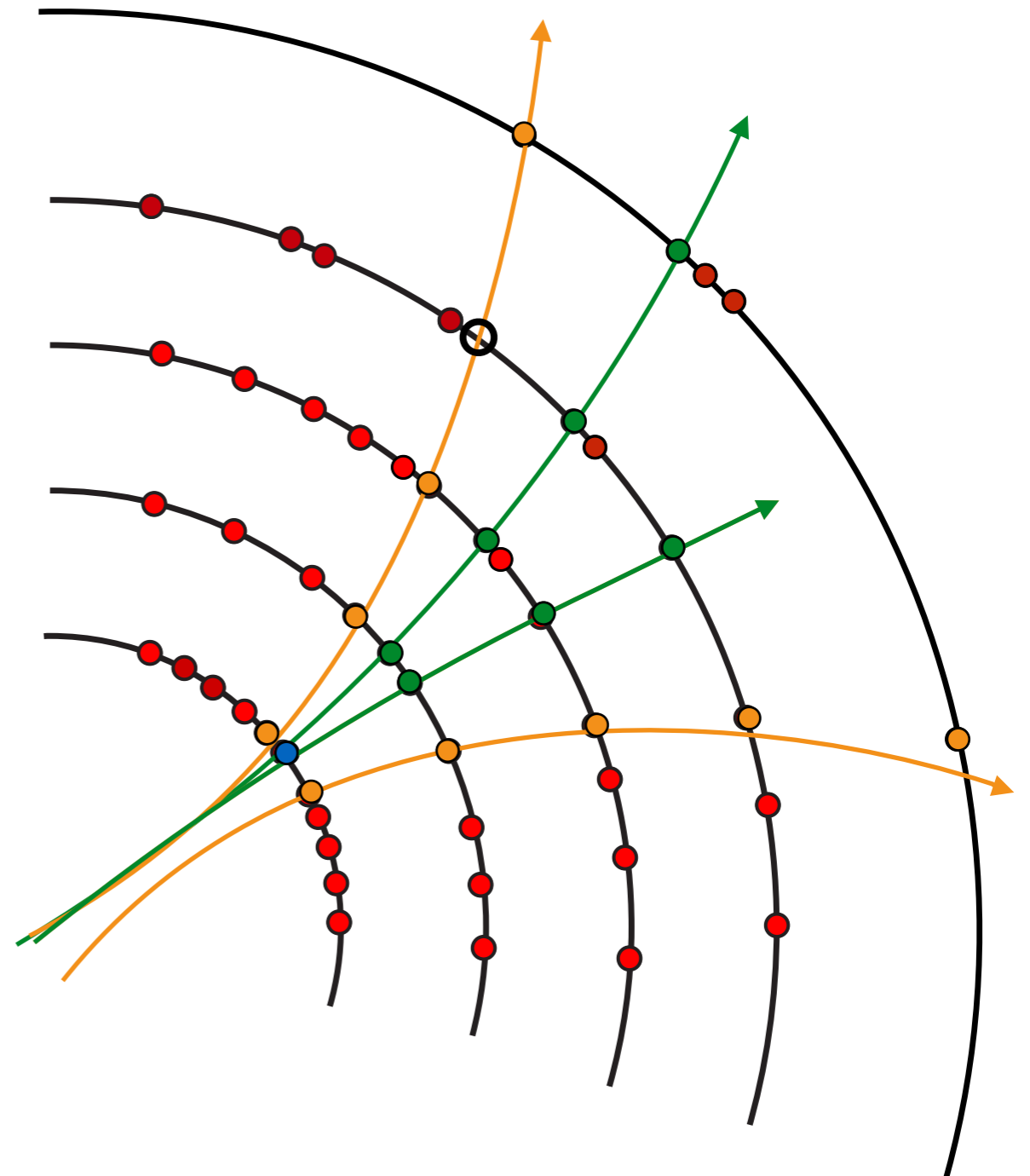
not so good track

short tracks

holes

shared hits

bad fit quality,
outliers



- ▶ Can we be sure ?

Revisiting enemy No. 2: **ghosts**

- ▶ Some of the characteristics can only be checked after all track candidates are found

good track

many compatible hits

completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

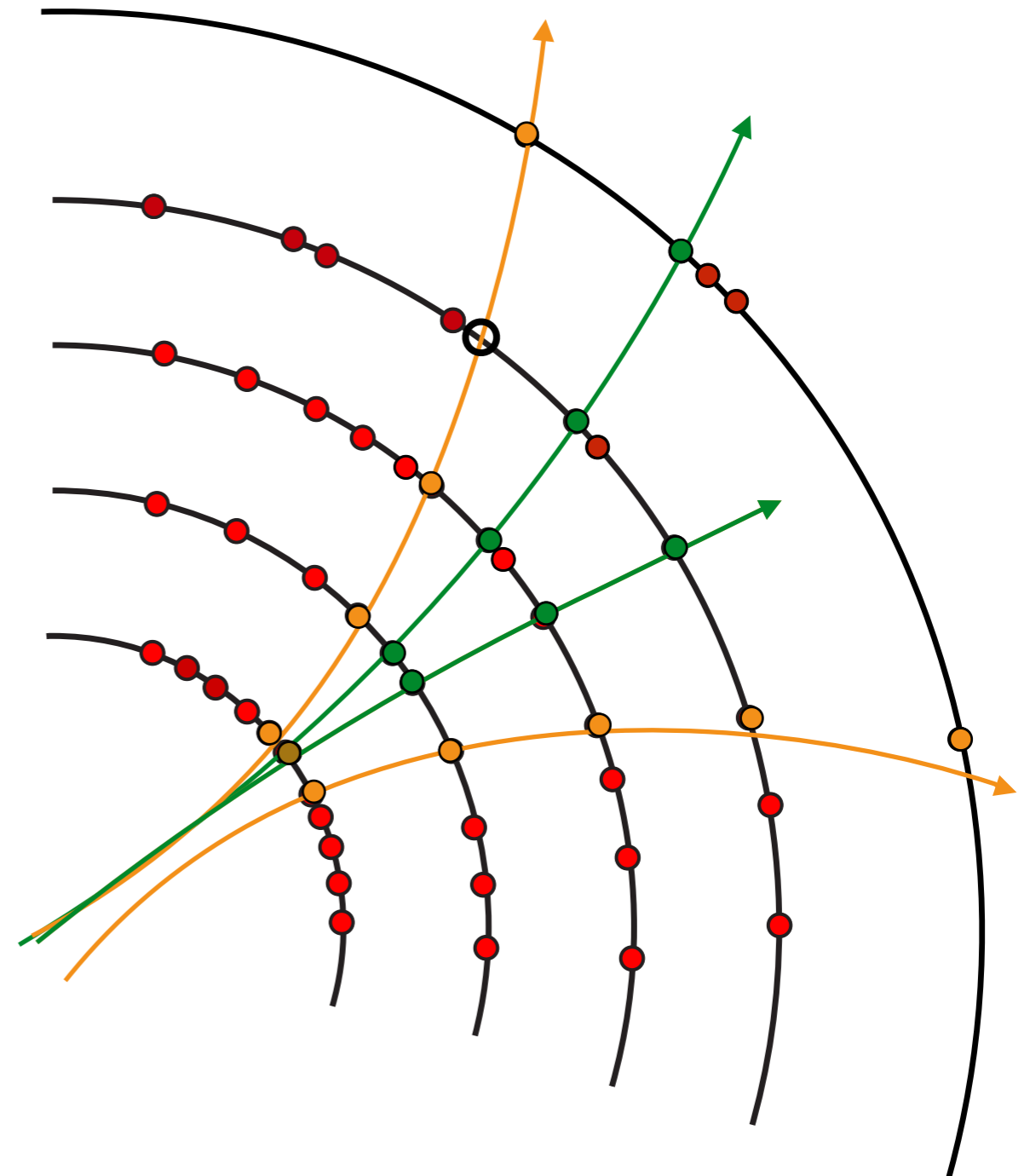
not so good track

short tracks

holes

shared hits

bad fit quality,
outliers



Revisiting enemy No. 2: **ghosts**

- ▶ Some of the characteristics can only be checked after all track candidates are found

good track

many compatible hits

completeness

uniqueness

low χ^2/ndf

small impact parameter
(for primaries)

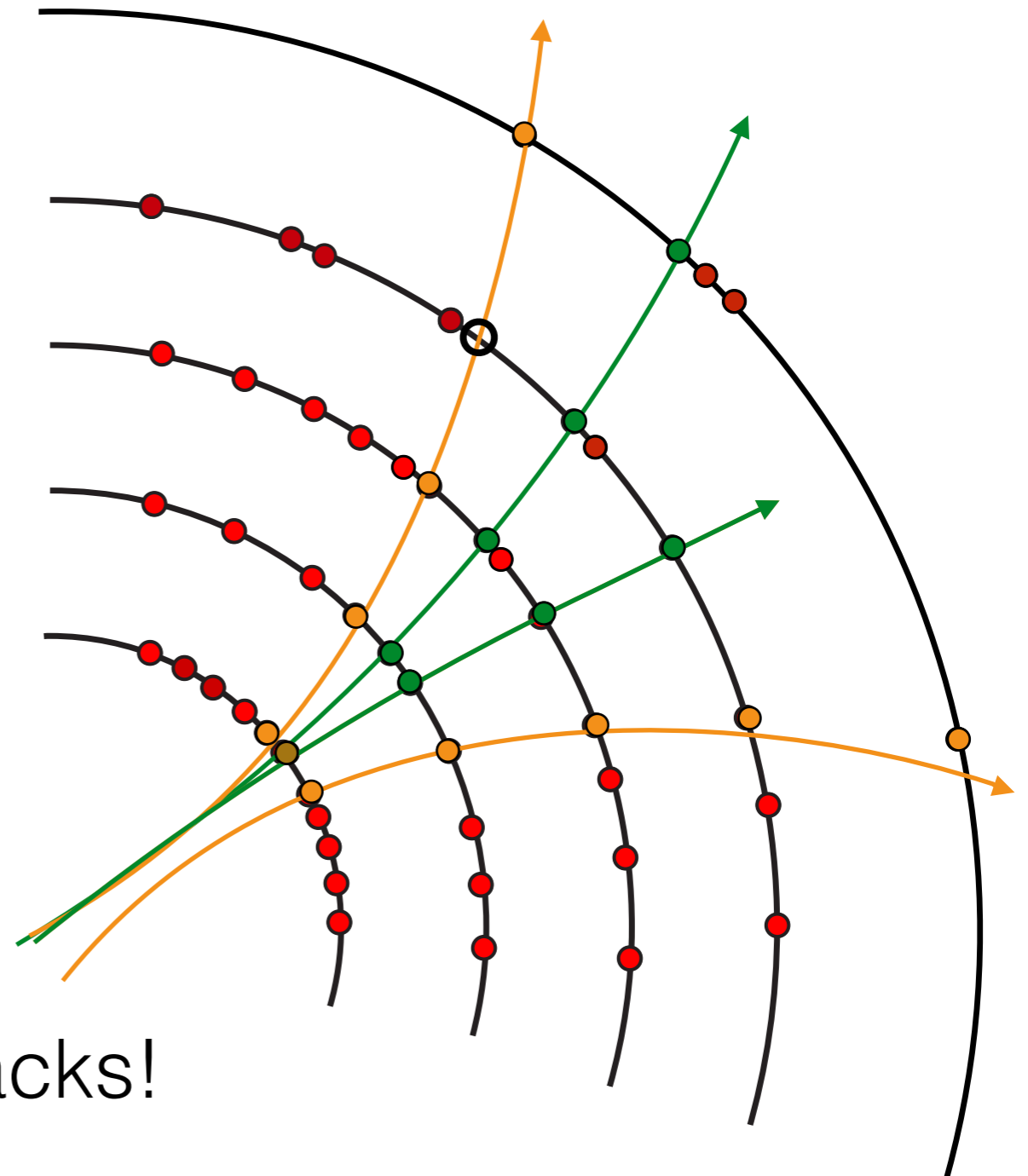
not so good track

short tracks

holes

shared hits

bad fit quality,
outliers



give scores and rank the tracks!

Vertex reconstruction

- ▶ We've found tracks, let's use them: vertex reconstruction
- ▶ Similar problem to track reconstruction
 - consists of vertex finding and vertex fitting, (primary) input are tracks

Vertex reconstruction

- ▶ We've found tracks, let's use them: vertex reconstruction
- ▶ Similar problem to track reconstruction
 - consists of vertex finding and vertex fitting, (primary) input are tracks
- ▶ Vertex fitting

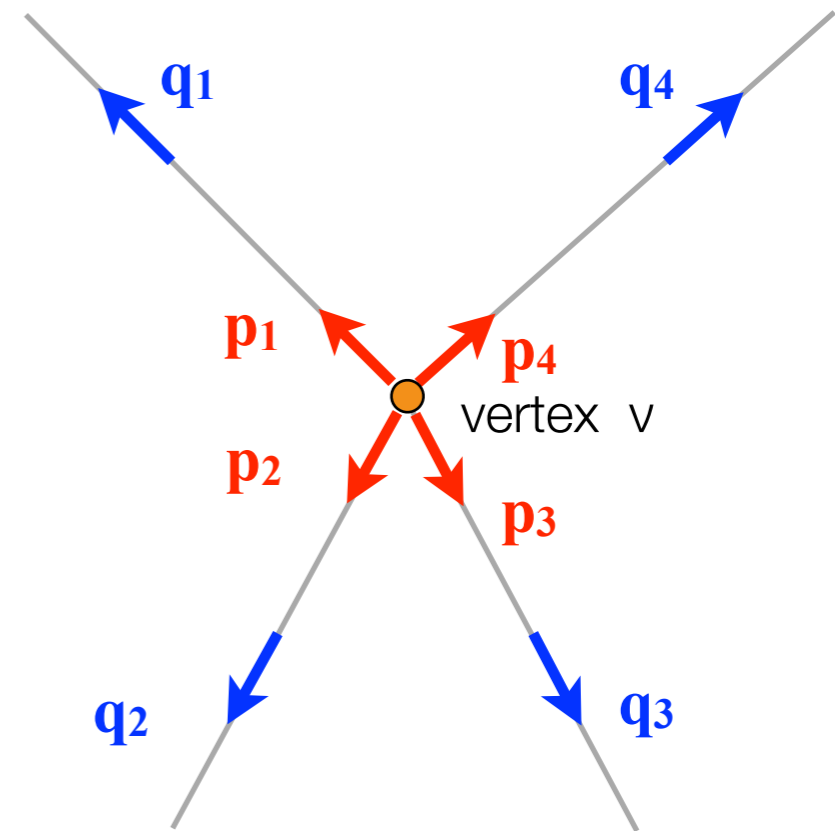
$$q_i = h_i(v, p_i) + \varepsilon_i$$

with: h_i = dependency of track parameters on vertex and parameters at vertex

ε_i = error of q_i

Jacobians: $A_i = \frac{\partial h_i(v, p_i)}{\partial v}$ $B_i = \frac{\partial h_i(v, p_i)}{\partial p_i}$

$$h_i = f \circ \tilde{q}(v, p_i) \quad \text{with:} \quad v = (v_x, v_y, v_z)$$
$$p_i = (\theta_i, \phi_i, Q_i/P_i)$$



Vertex reconstruction

- ▶ We've found tracks, let's use them: vertex reconstruction
- ▶ Similar problem to track reconstruction
 - consists of vertex finding and vertex fitting, (primary) input are tracks
- ▶ Vertex fitting

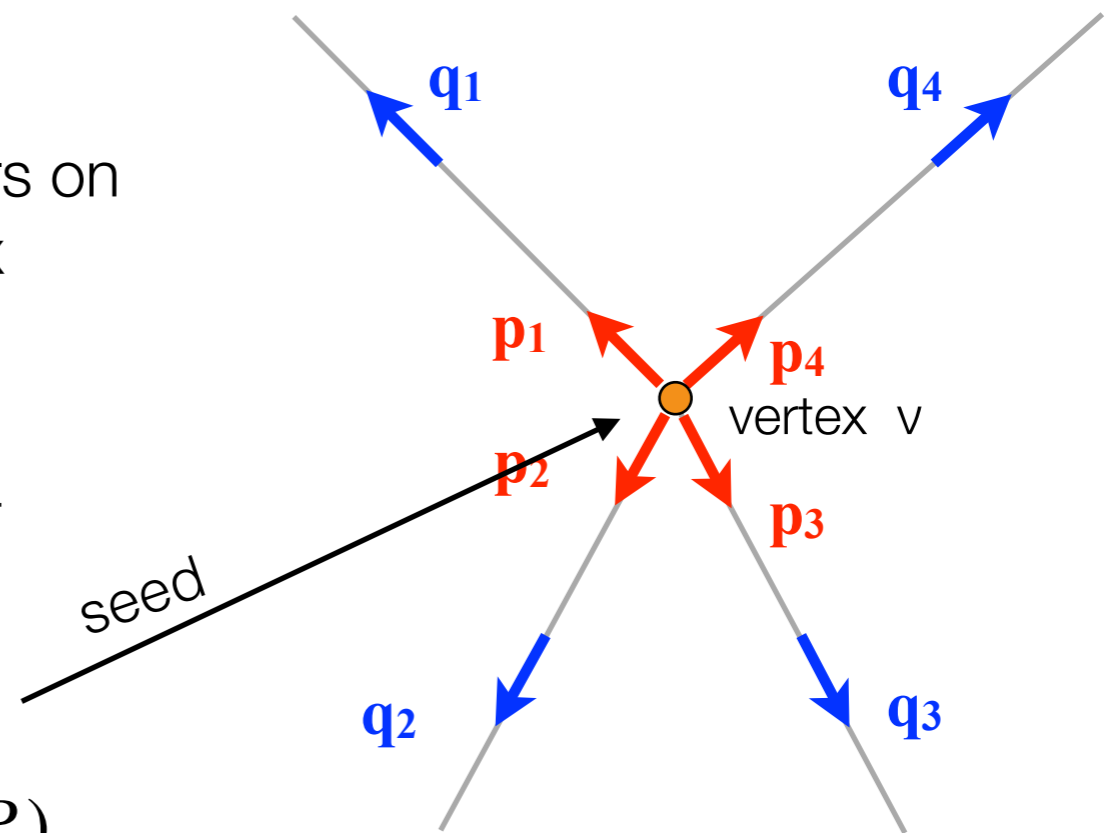
$$q_i = h_i(v, p_i) + \varepsilon_i$$

with: h_i = dependency of track parameters on vertex and parameters at vertex

ε_i = error of q_i

Jacobians: $A_i = \frac{\partial h_i(v, p_i)}{\partial v}$ $B_i = \frac{\partial h_i(v, p_i)}{\partial p_i}$

$h_i = f \circ \tilde{q}(v, p_i)$ with: $v = (v_x, v_y, v_z)$
 $p_i = (\theta_i, \phi_i, Q_i/P_i)$



Vertex fitting

- ▶ And again, you can do a straight forward global χ^2 minimisation

$$\chi^2 = \sum_i \Delta q_i^T G_i \Delta q_i \quad \text{with: } \Delta q_i = q_i - h_i(v, p_i)$$

$V_i = G_i^{-1}$ covariance of the measured q_i

linearize the problem: $v \rightarrow v_0 + \delta v$ and $p_i \rightarrow p_{i,0} + \delta p_i$

$$h_i(v, p_i) \cong h_i(v_0, p_{i,0}) + A_i \delta v + B_i \delta p_i + \text{higher terms}$$

this yields:

$$\chi^2 = \sum_i \left(h_i(v_0, p_{i,0}) + A_i \delta v + B_i \delta p_i \right)^T G_i \left(h_i(v_0, p_{i,0}) + A_i \delta v + B_i \delta p_i \right)$$

minimizing the linearized χ^2 gives the following set of equations:

$$\frac{\partial \chi^2}{\partial v} = 0 \quad \Rightarrow \quad \left(\sum_i A_i^T G_i A_i \right) \cdot \delta v + \sum_i A_i^T G_i B_i \cdot \delta p_i = \sum_i A_i^T G_i \cdot \Delta q_{i,0}$$
$$\frac{\partial \chi^2}{\partial p_i} = 0 \quad \Rightarrow \quad B_i^T G_i A_i \cdot \delta v + B_i^T G_i B_i \cdot \delta p_i = B_i^T G_i \cdot \Delta q_{i,0}$$

with: $\Delta q_{i,0} = q_i - h_i(v_0, p_{i,0})$

Vertex fitting

- Or, you apply a Kalman filter technique

$$\delta v_i = C_i^{-1} \cdot [C_{i-1} \delta v_{i-1} + A_i^T G_i^B \cdot \Delta q_{i,i-1}]$$

$$C_i = (C_{i-1}^{-1} + A_i^T G_i^B A_i)^{-1}$$

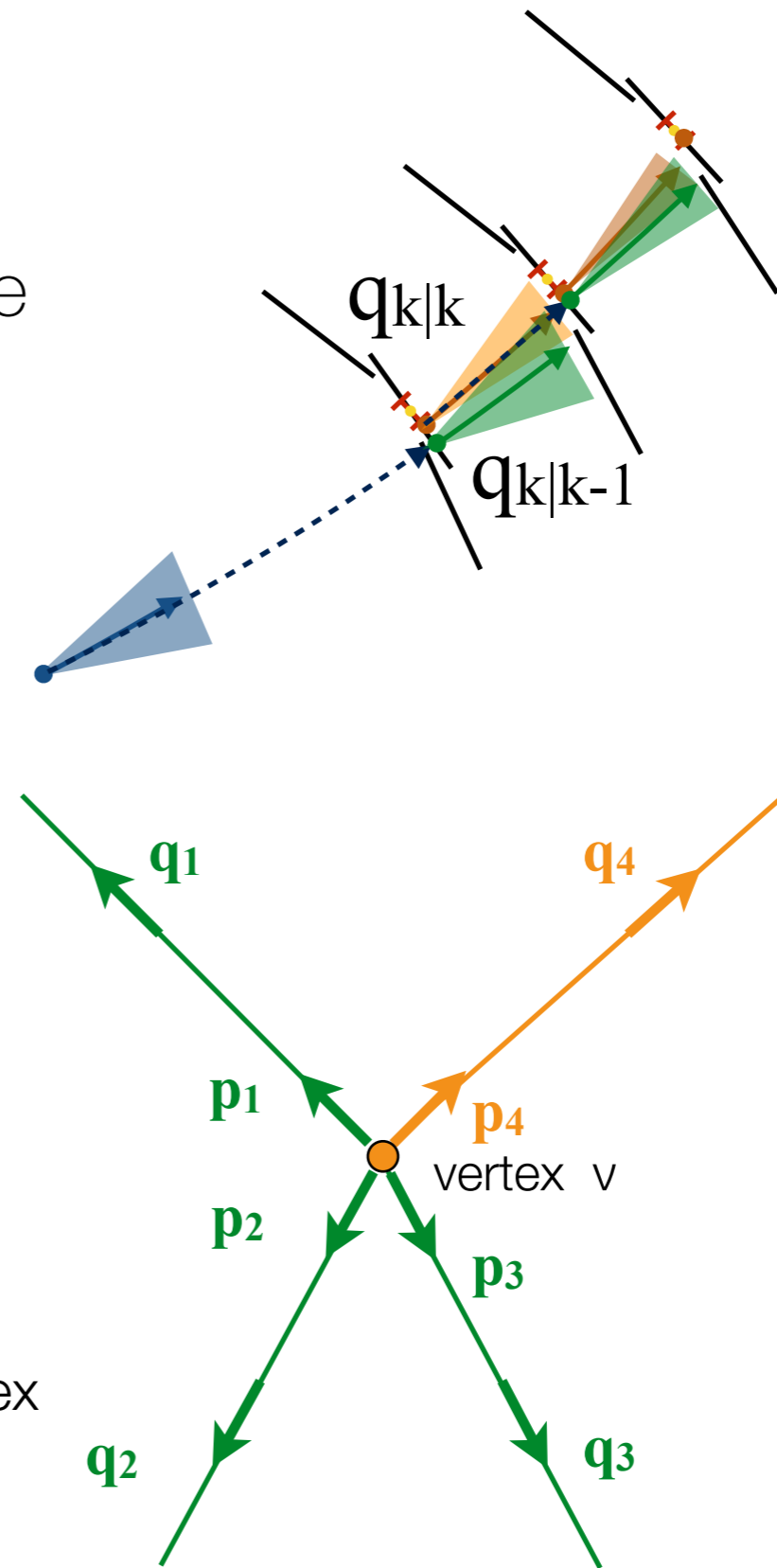
$$\delta p_{i,i} = W_i B_i^T G_i \cdot (\Delta q_{i,i-1} - A_i \delta v_i)$$

$$D_i = W_i + W_i B_i^T G_i A_i C_i A_i^T G_i B_i W_i$$

and the smoother becoming the re-evaluation of the parameters $q_{i,n}$ depending on the fitted vertex after inclusion of n tracks

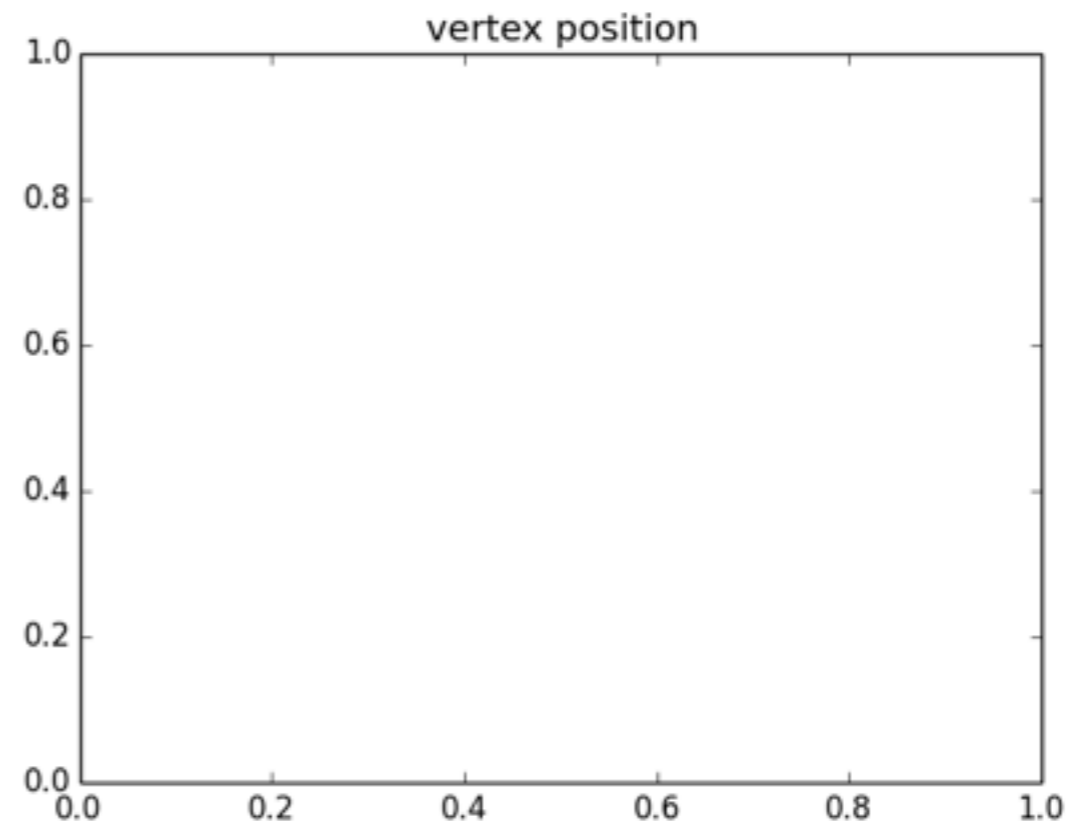
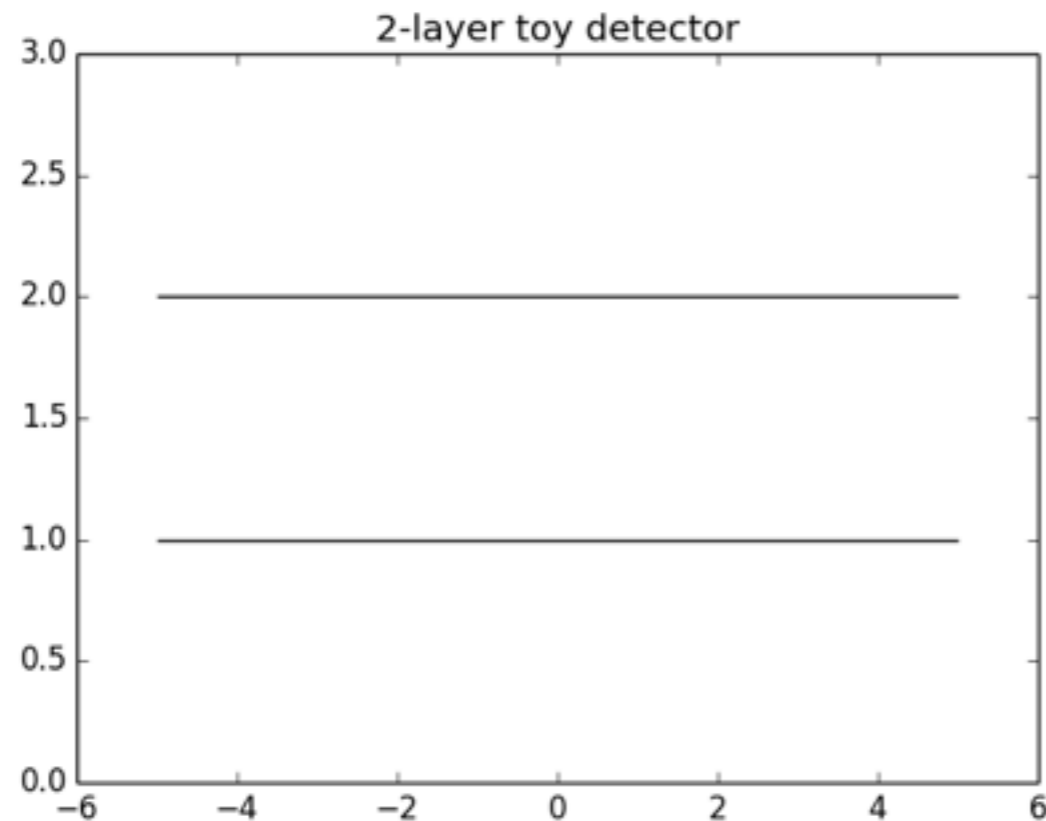
$$q_{i,n} = h_i(v_0 + \delta v_n, p_{i,0} + \delta p_{i,n})$$

with: $\text{cov}(q_{i,n}) = B_i W_i B_i^T + V_i^B G_i A_i C_n A_i^T G_i V_i^B$ and $V_i^B = V_i - B_i W_i B_i^T$



MC Toy: vertex seed finding

- ▶ What about the initial seed ?



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

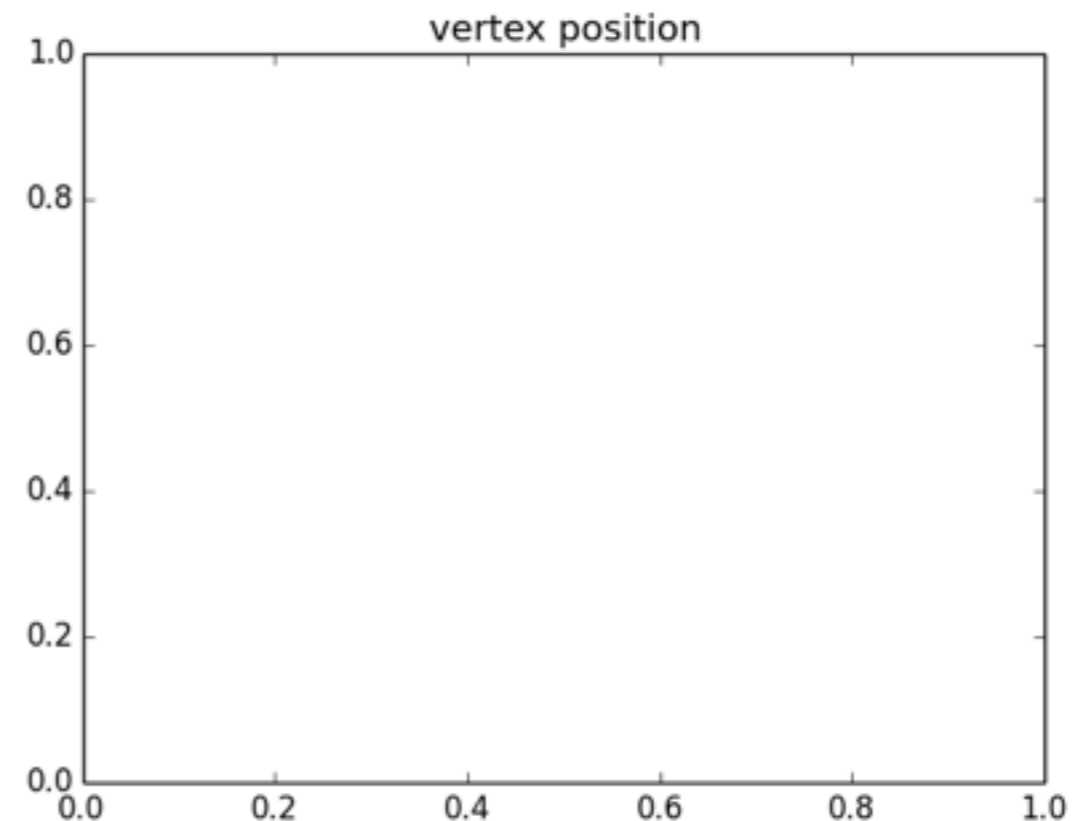
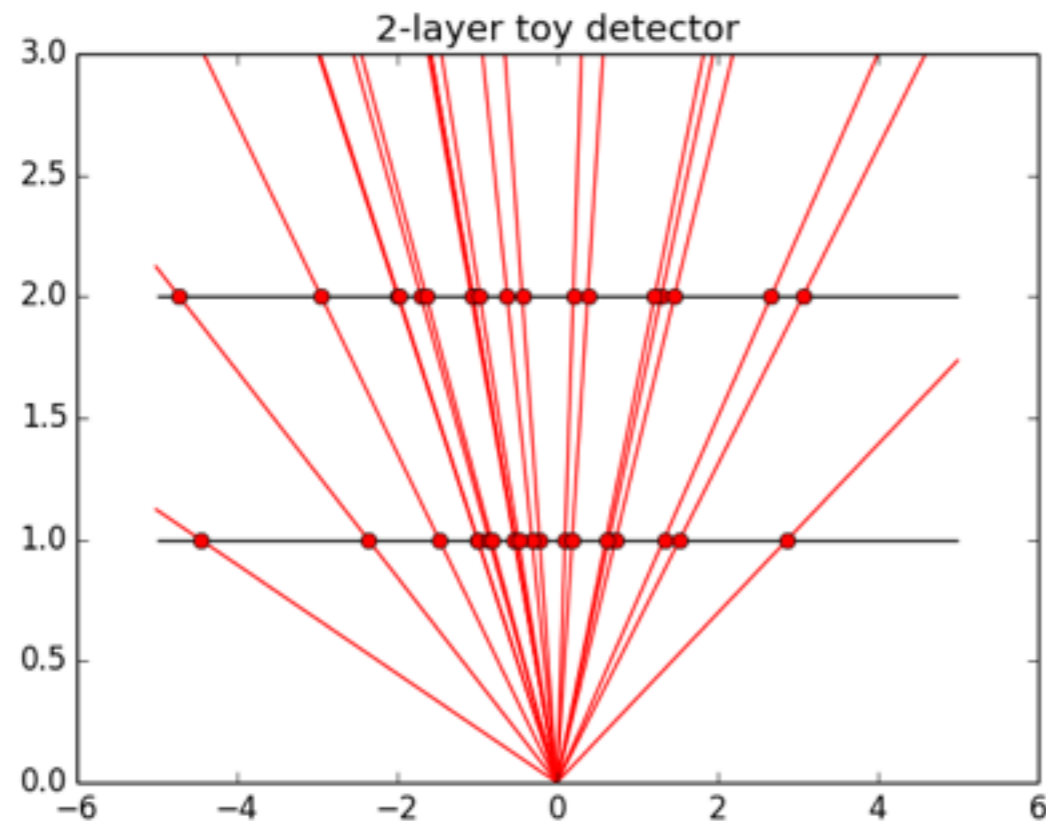
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=1, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding

- ▶ What about the initial seed ?



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

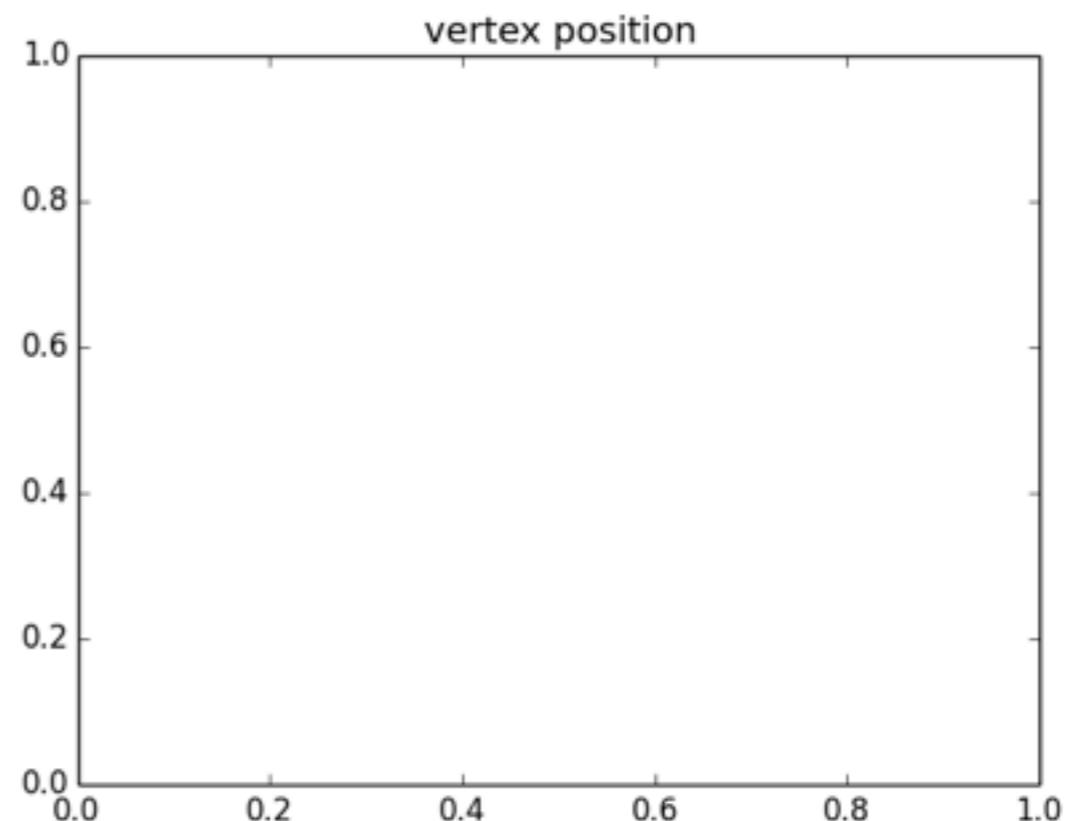
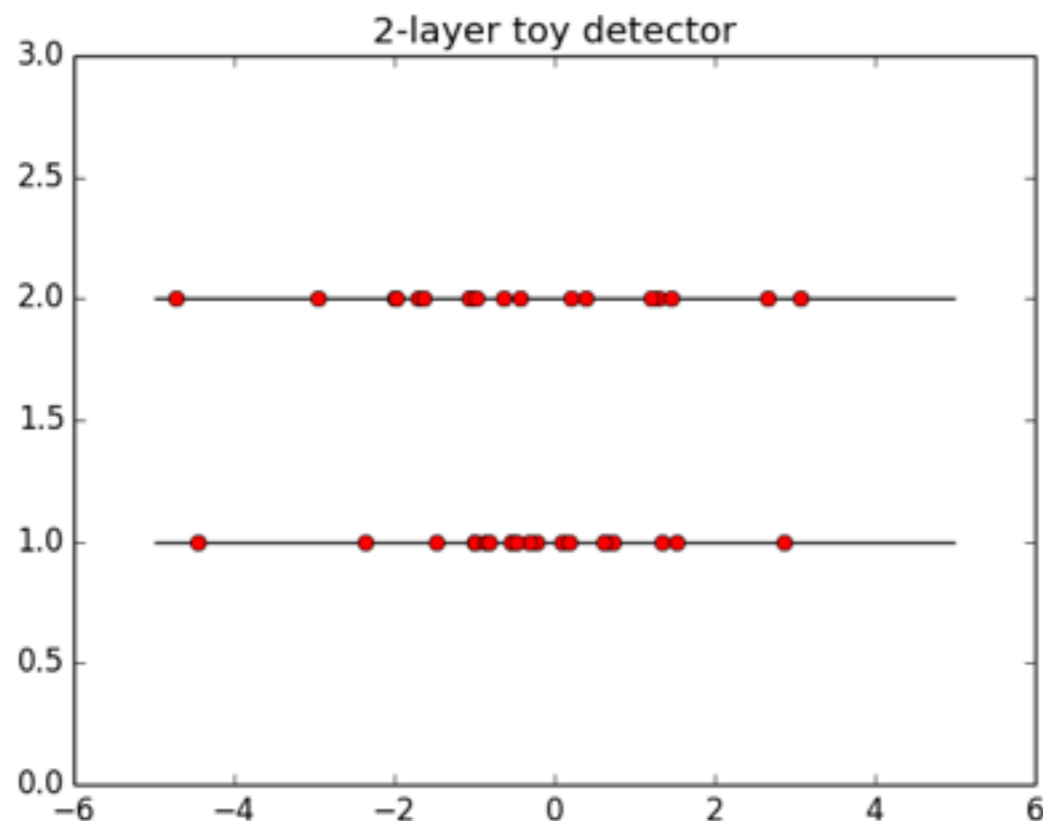
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=1, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding

- ▶ What about the initial seed ?



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

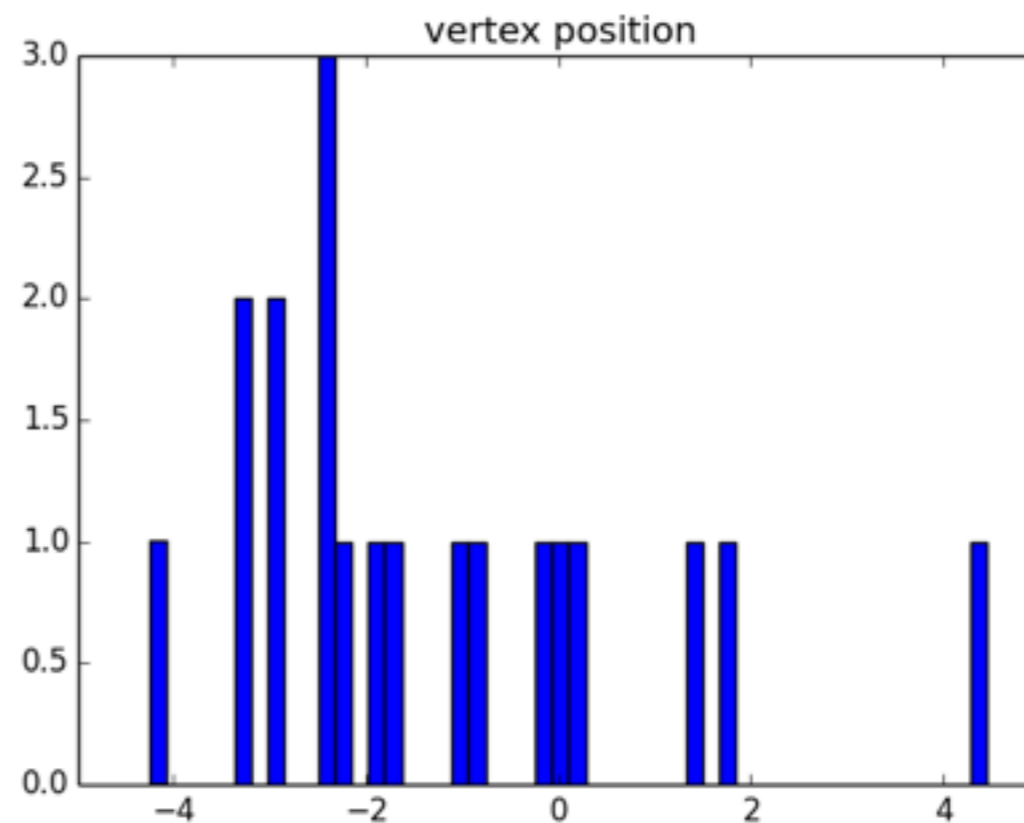
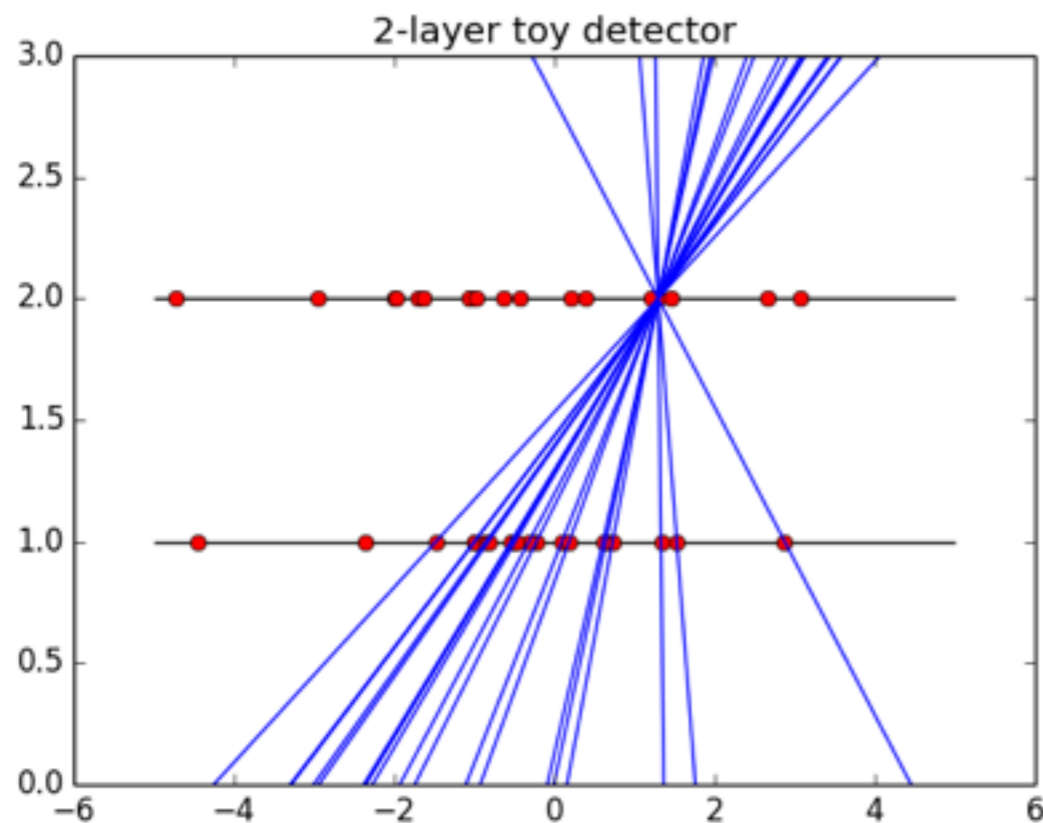
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=1, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding

- ▶ What about the initial seed ?



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

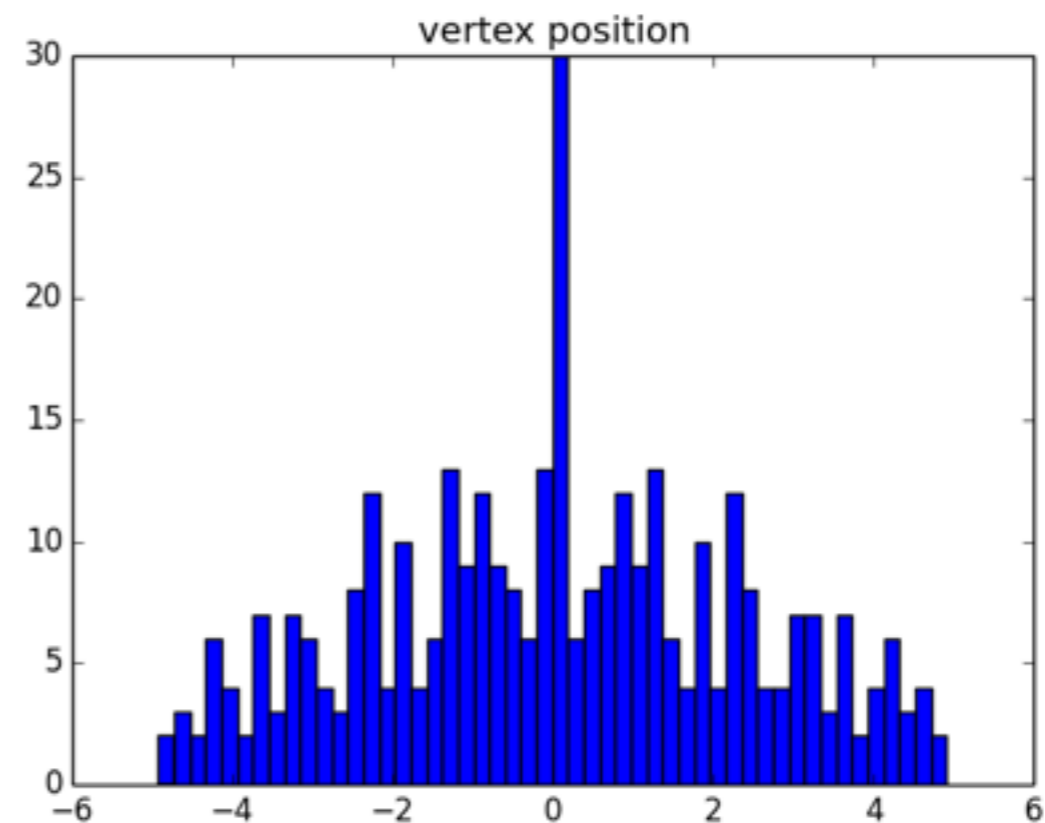
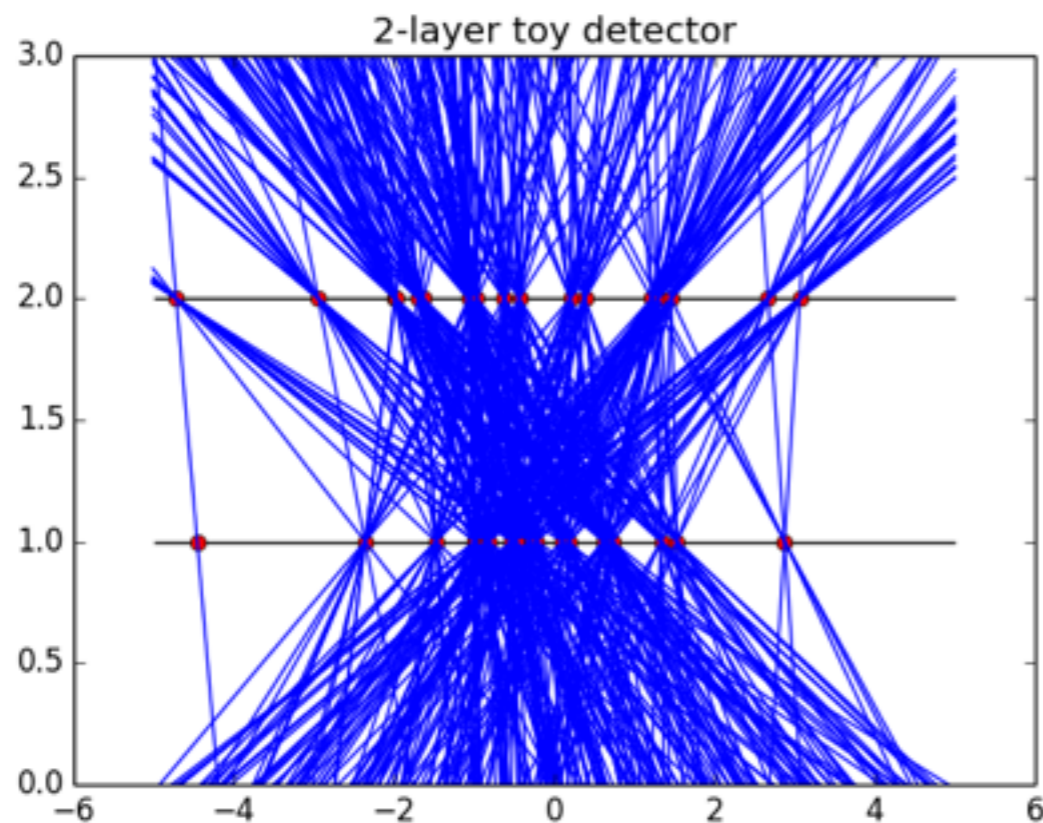
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=1, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```


MC Toy: vertex seed finding

- ▶ What about the initial seed ?



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=1, number=25, seed=12345)
```

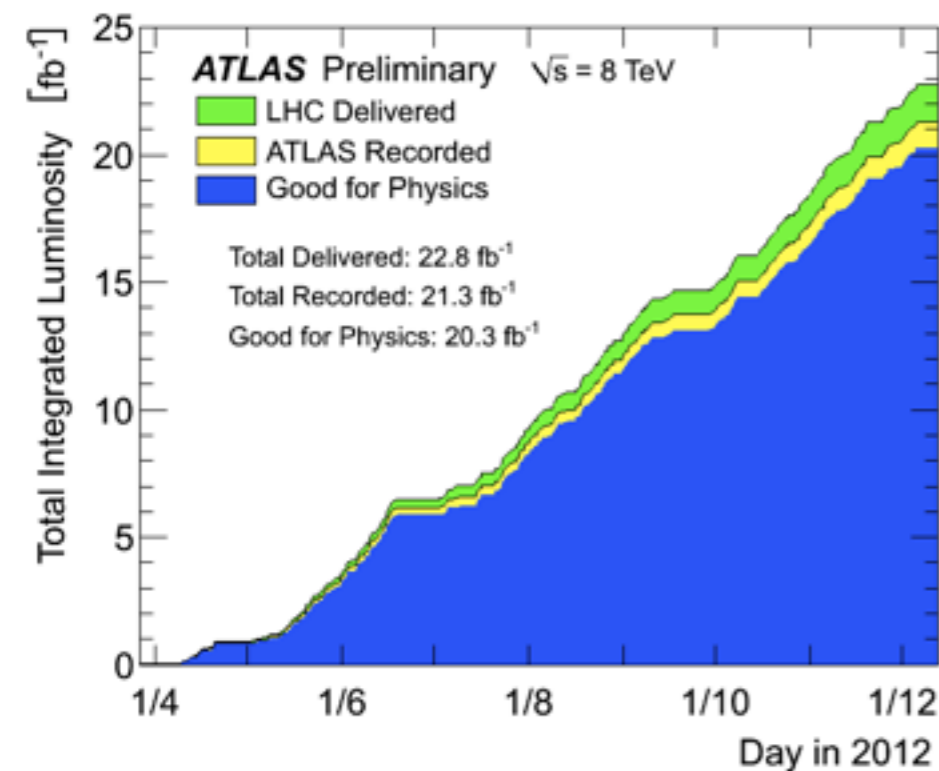
```
In [3]: findVertices(fig, plots, hits)
```

Enemy No. 3: **pile-up**

- ▶ To maximise the physics potential, LHC was running (and will keep on running) with multiple instantaneous collisions: pile-up

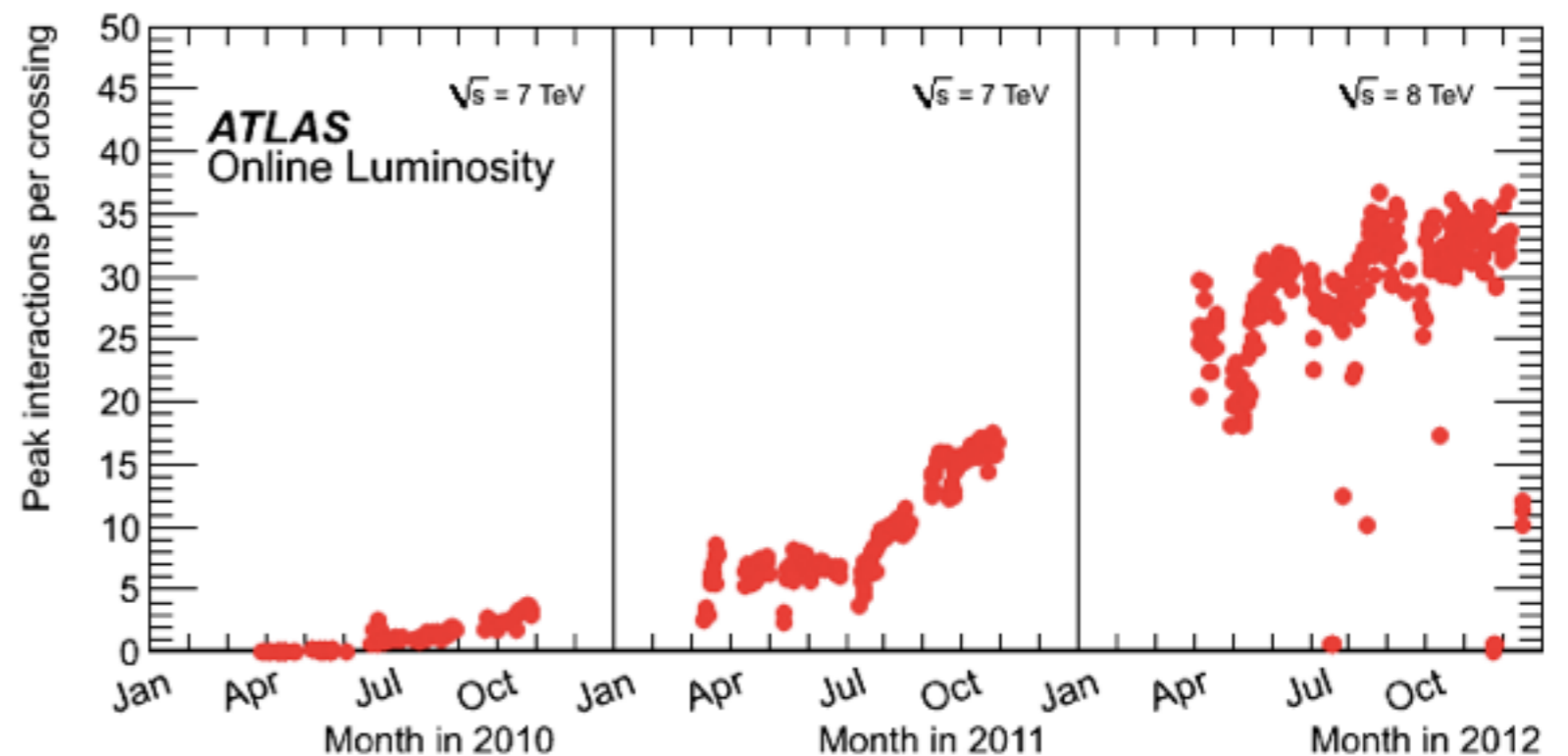
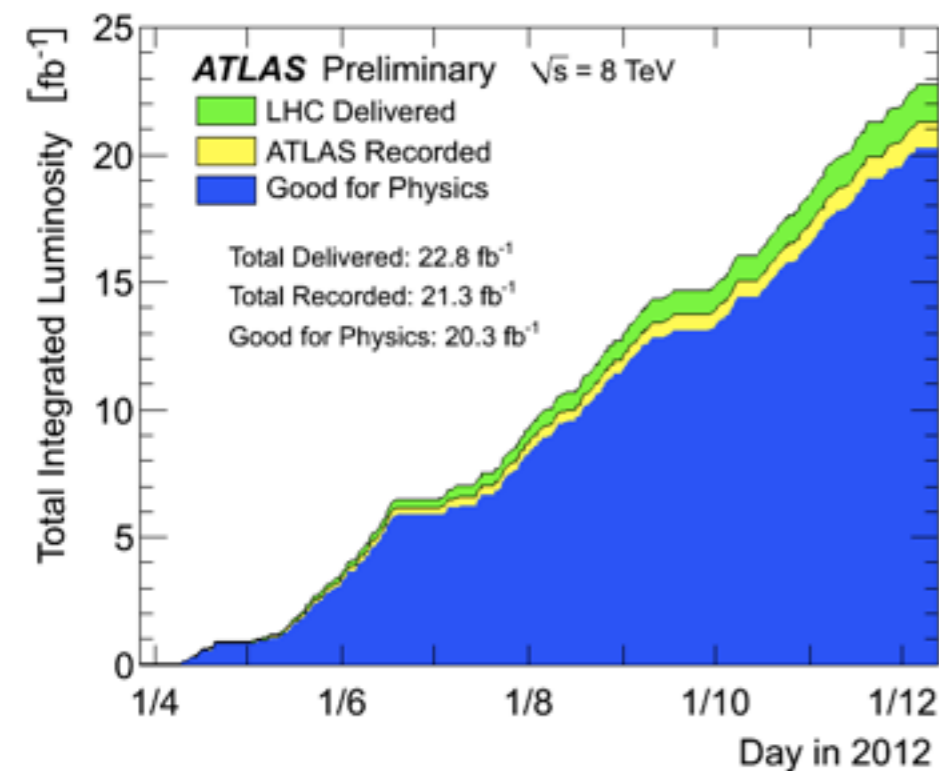
Enemy No. 3: **pile-up**

- ▶ To maximise the physics potential, LHC was running (and will keep on running) with multiple instantaneous collisions: pile-up



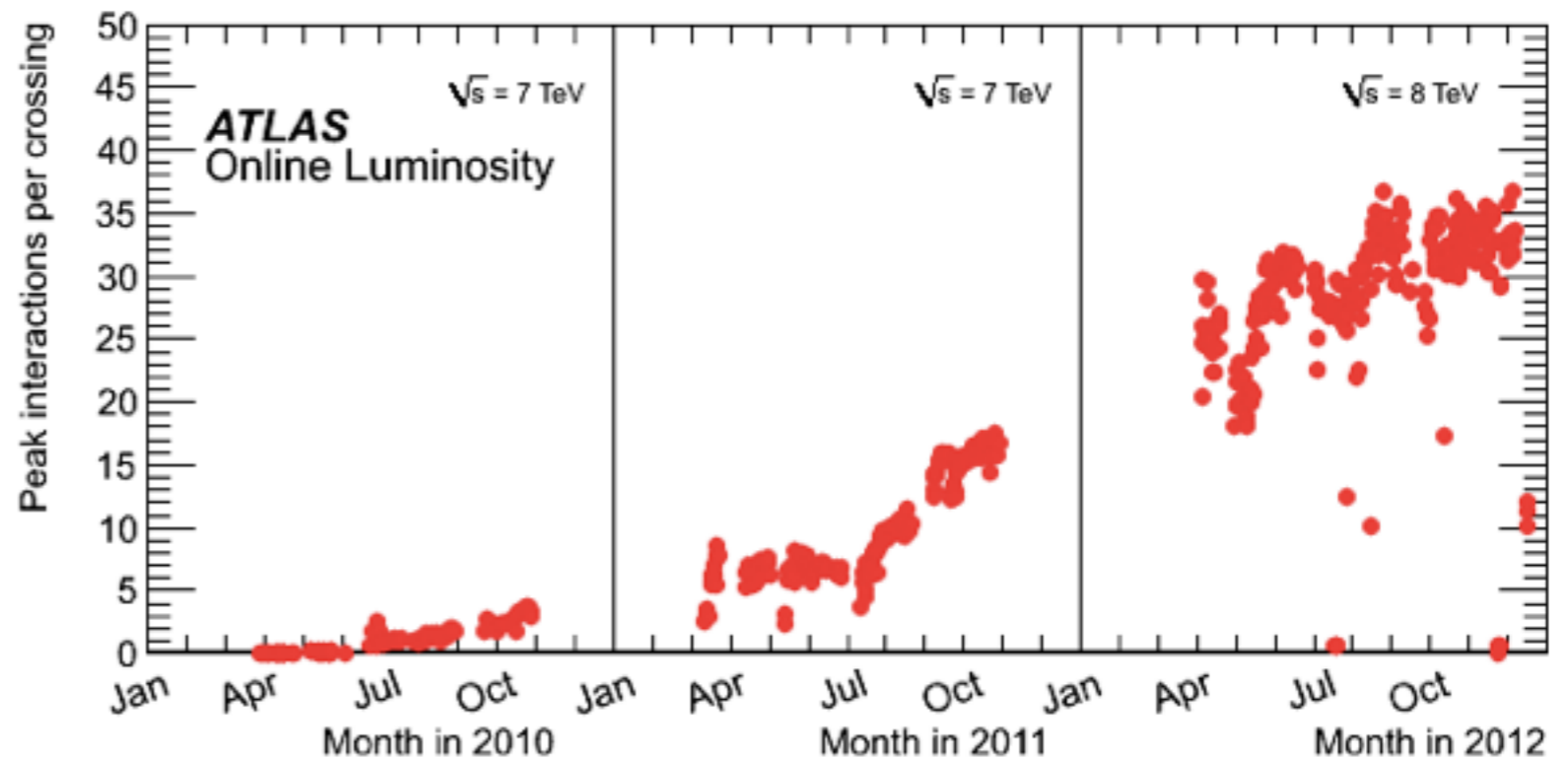
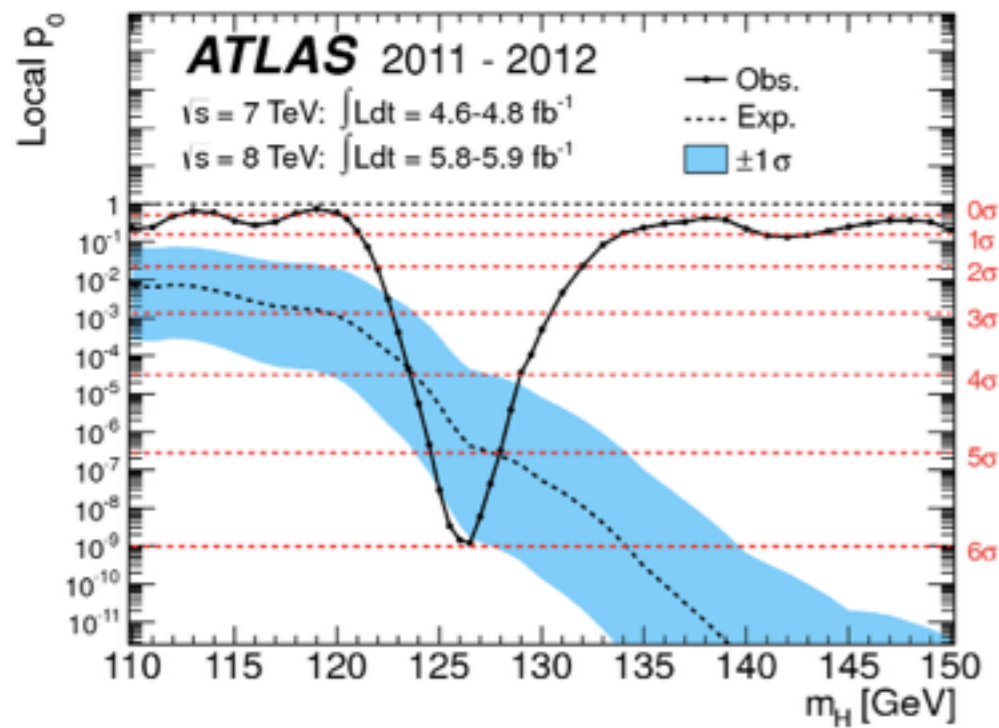
Enemy No. 3: **pile-up**

- ▶ To maximise the physics potential, LHC was running (and will keep on running) with multiple instantaneous collisions: pile-up



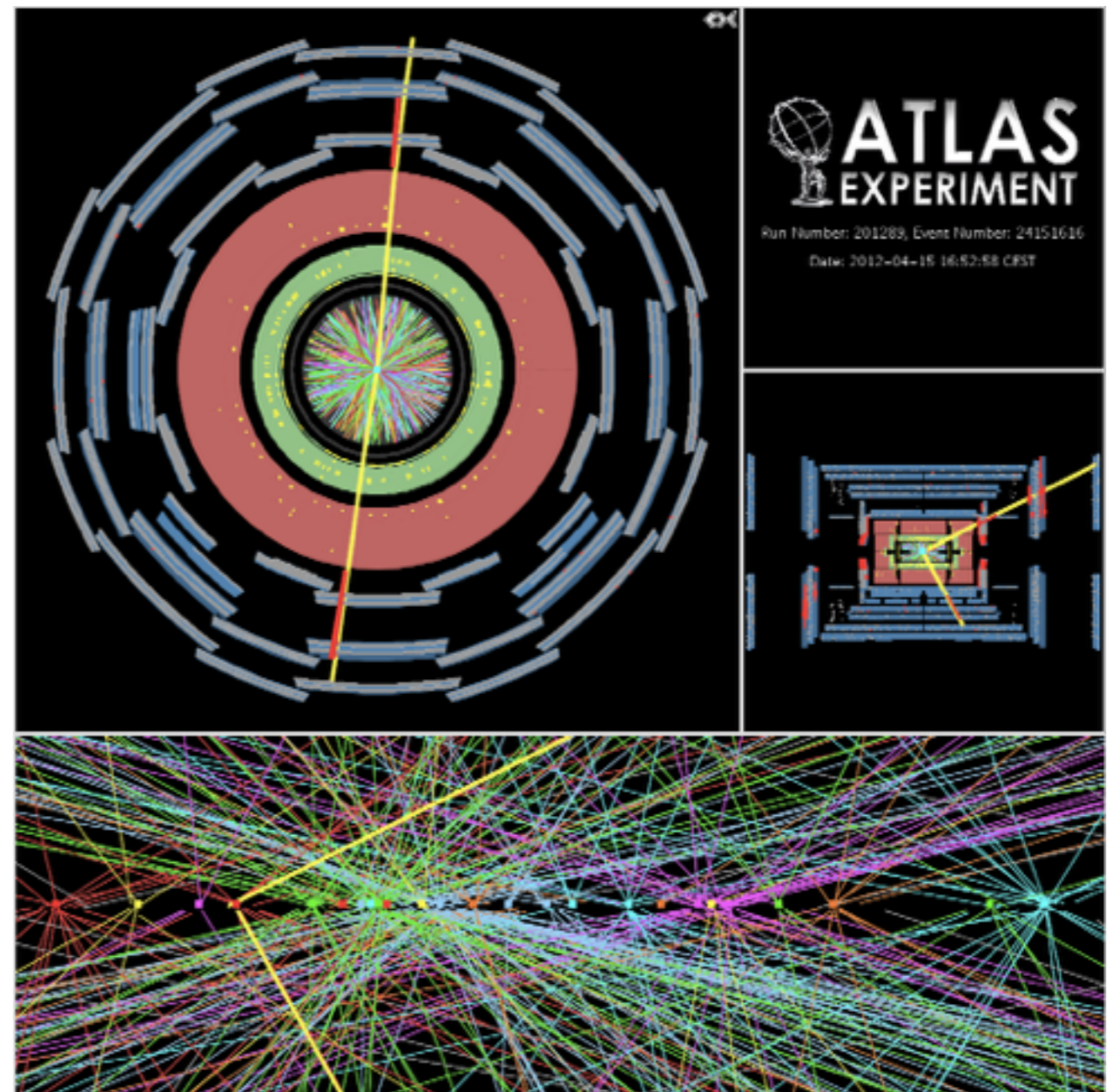
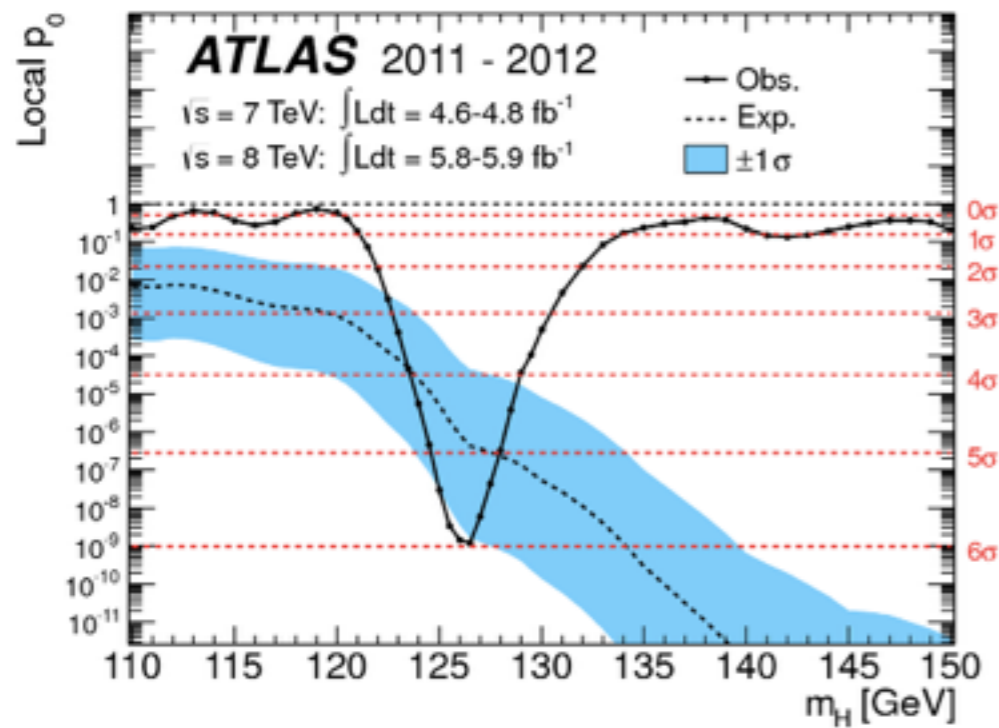
Enemy No. 3: **pile-up**

- ▶ To maximise the physics potential, LHC was running (and will keep on running) with multiple instantaneous collisions: pile-up



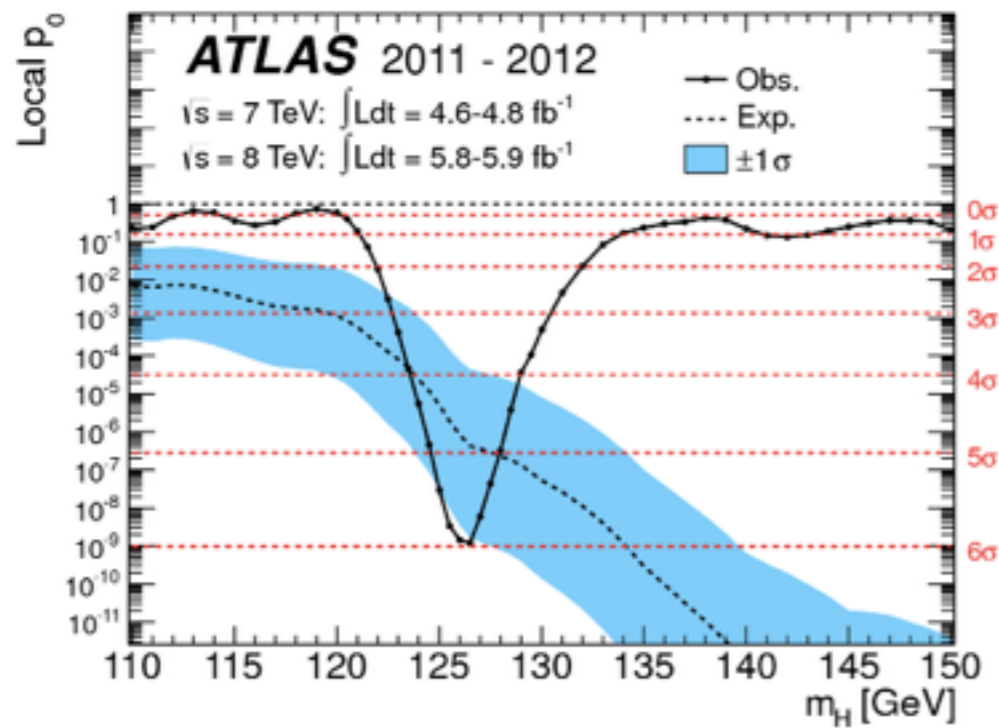
Enemy No. 3: **pile-up**

- ▶ To maximise the physics potential, LHC was running (and will keep on running) with multiple instantaneous collisions: pile-up

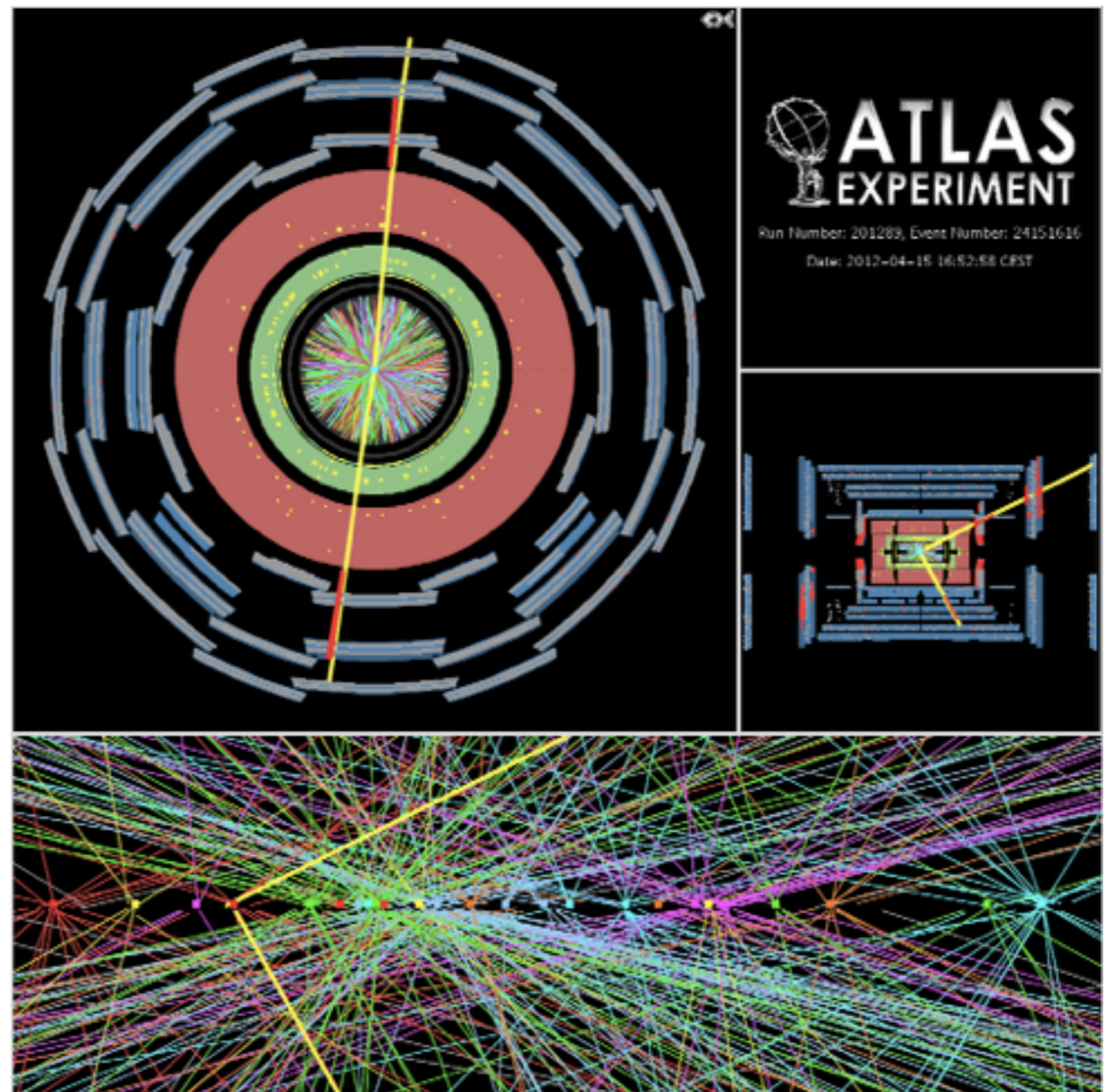


Enemy No. 3: **pile-up**

- ▶ To maximise the physics potential, LHC was running (and will keep on running) with multiple instantaneous collisions: pile-up



- ▶ More “fun” for track reconstruction

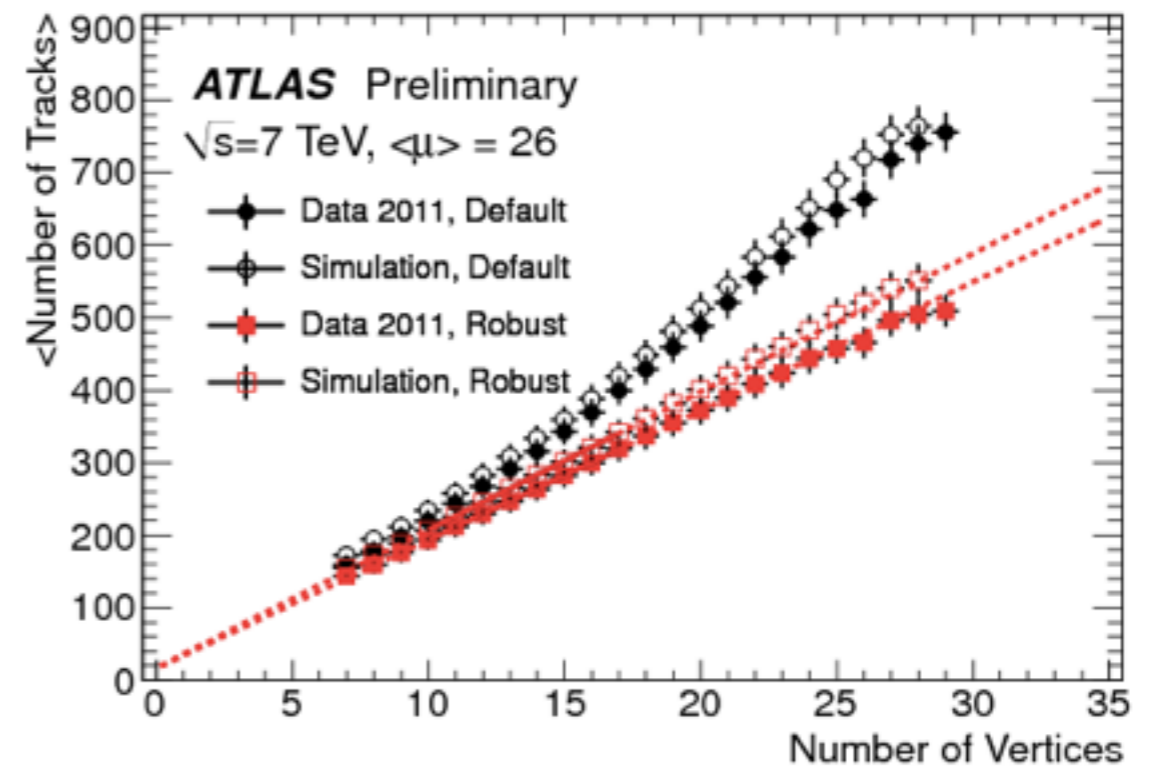


Enemy No. 3: **pile-up**

- ▶ Pile-up let's combinatorics explode
- ▶ This is bad from many angles:
 - CPU time explodes
 - low track fake rate is at risk
(make more stringent requirements)
 - track finding becomes more difficult
(risk of losing tracks)
 - vertex reconstruction suffers from multiple problems:
 - fake vertices
 - merged vertices
 - split vertices

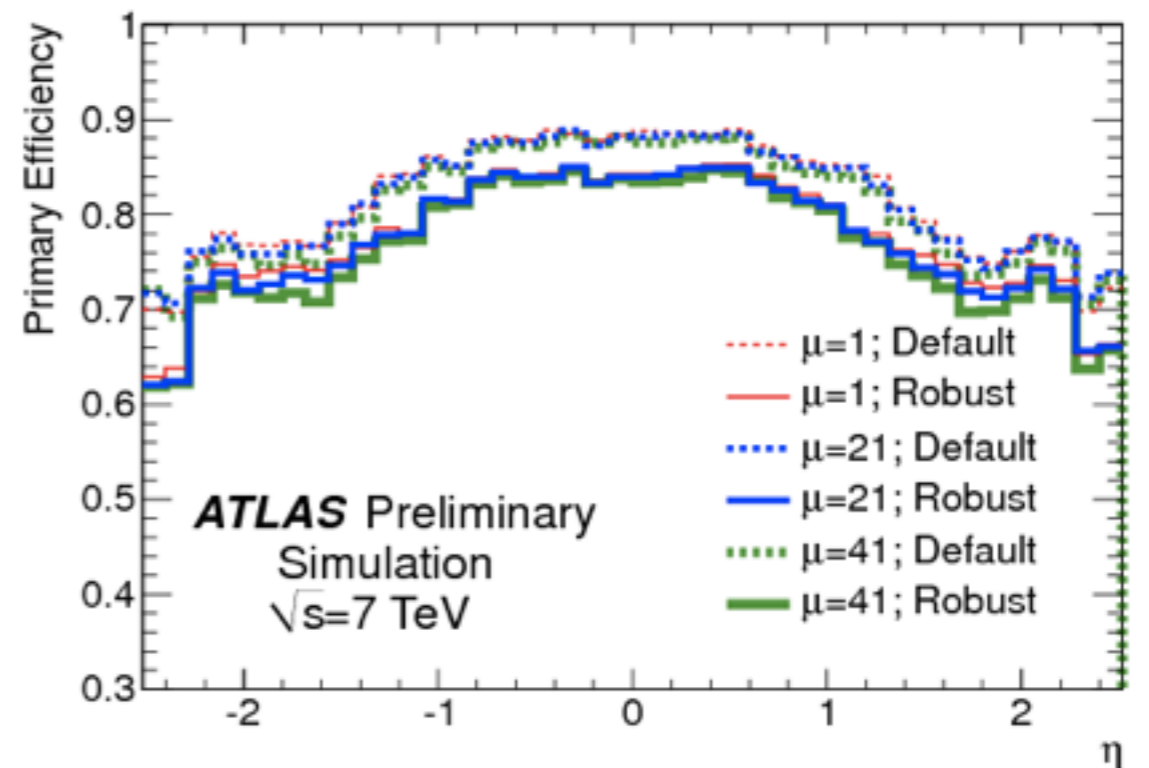
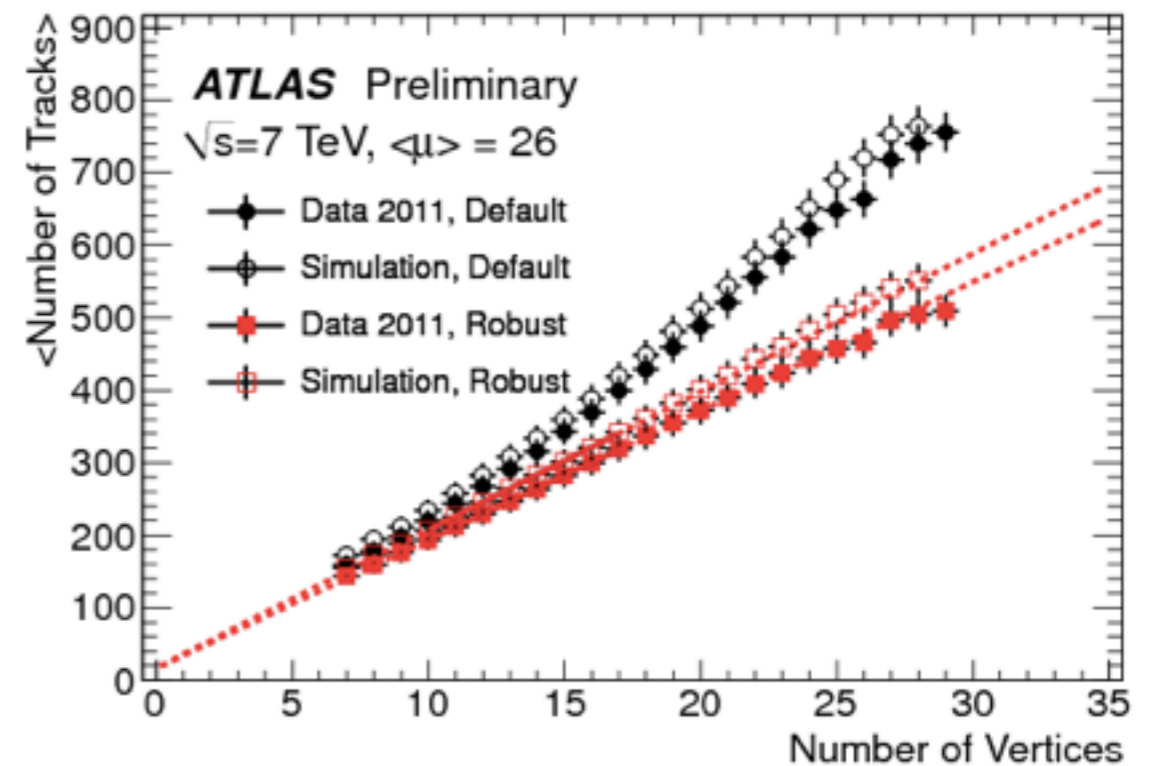
Enemy No. 3: **pile-up**

- ▶ Pile-up let's combinatorics explode
- ▶ This is bad from many angles:
 - CPU time explodes
 - low track fake rate is at risk
(make more stringent requirements)
 - track finding becomes more difficult
(risk of losing tracks)
 - vertex reconstruction suffers from multiple problems:
 - fake vertices
 - merged vertices
 - split vertices



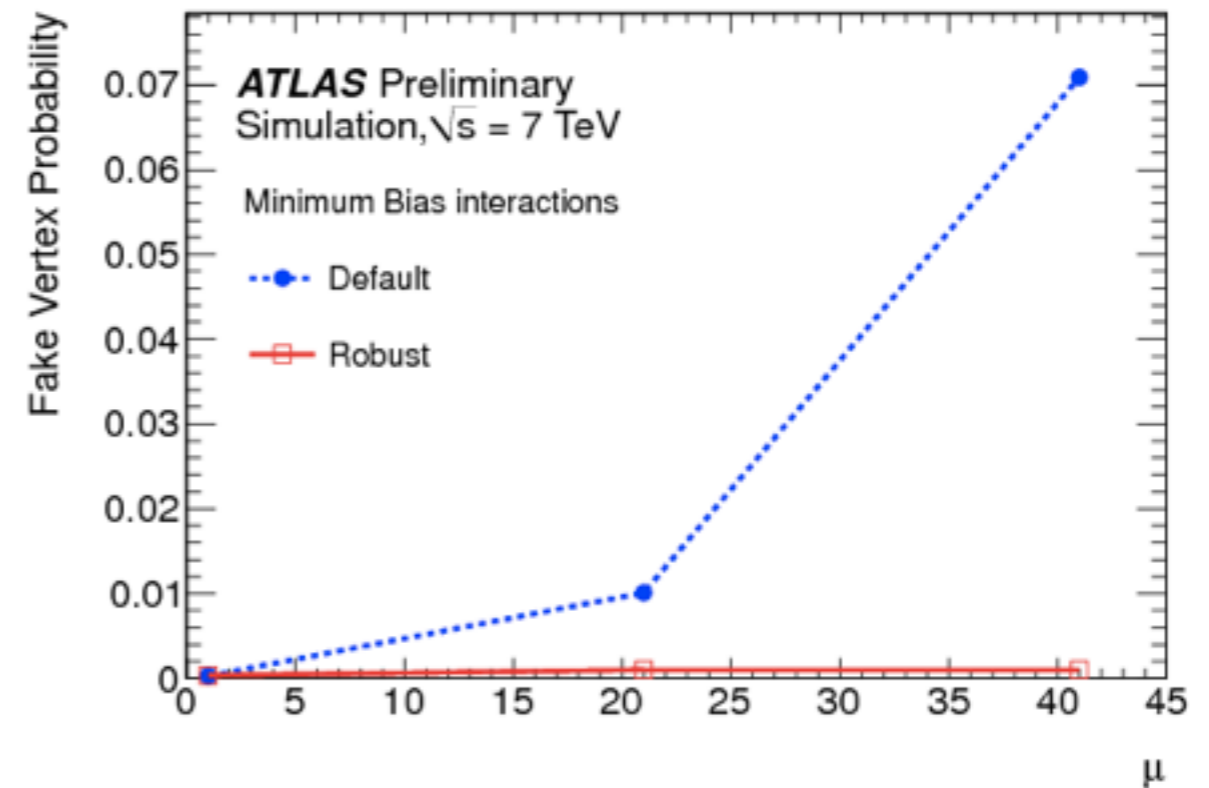
Enemy No. 3: **pile-up**

- ▶ Pile-up let's combinatorics explode
- ▶ This is bad from many angles:
 - CPU time explodes
 - low track fake rate is at risk
(make more stringent requirements)
 - track finding becomes more difficult
(risk of losing tracks)
 - vertex reconstruction suffers from multiple problems:
 - fake vertices
 - merged vertices
 - split vertices



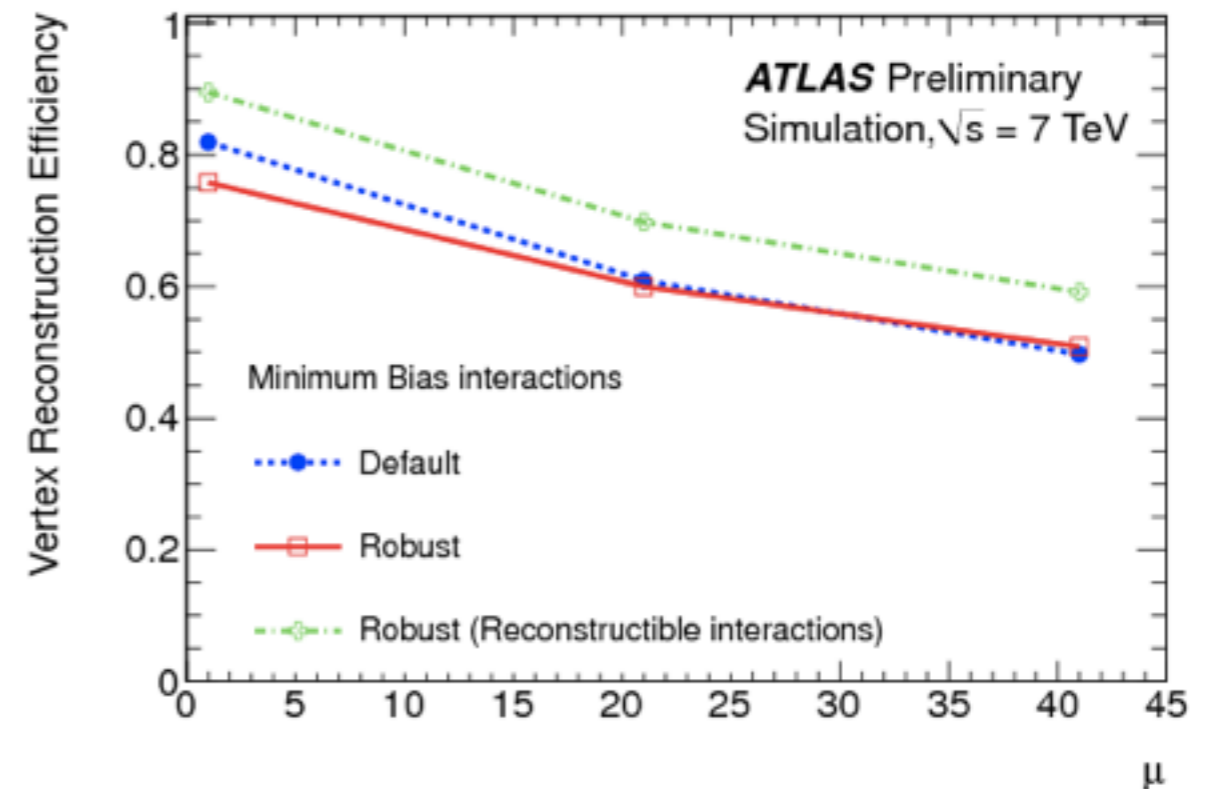
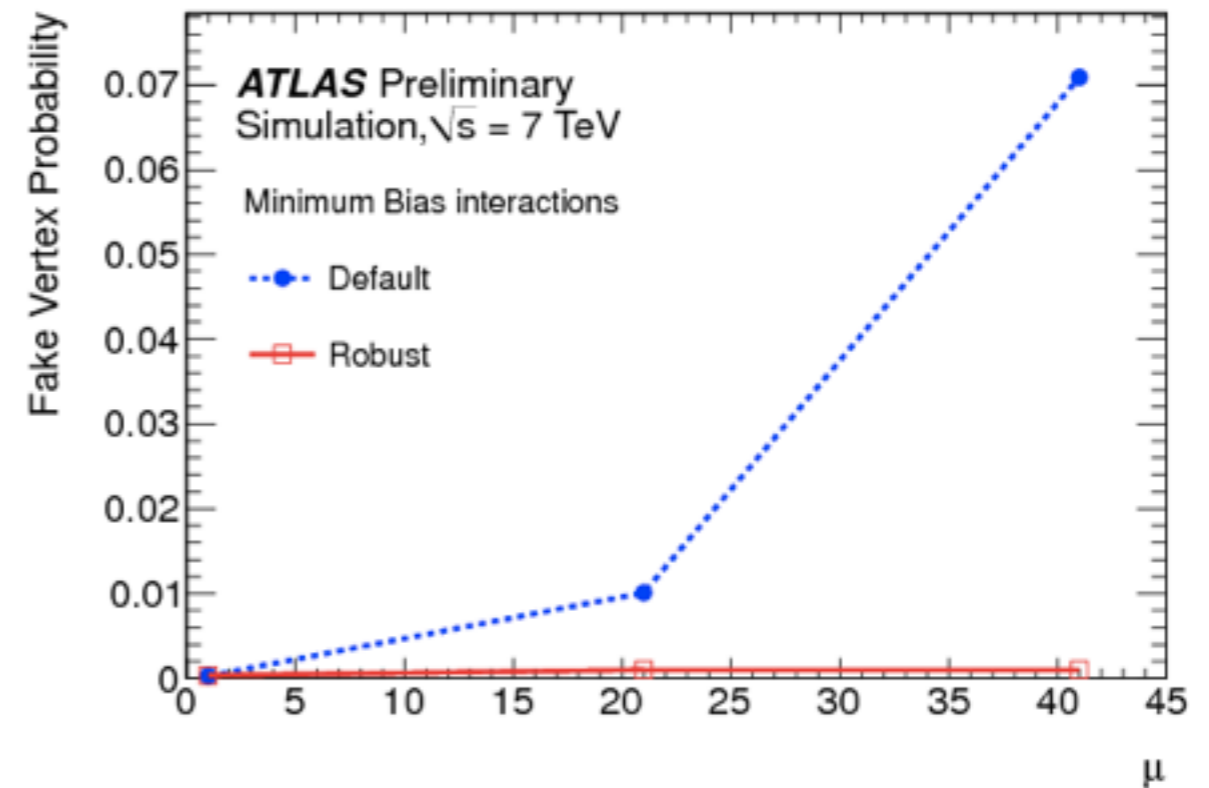
Enemy No. 3: **pile-up**

- ▶ Pile-up let's combinatorics explode
- ▶ This is bad from many angles:
 - CPU time explodes
 - low track fake rate is at risk
(make more stringent requirements)
 - track finding becomes more difficult
(risk of losing tracks)
 - vertex reconstruction suffers from multiple problems:
 - fake vertices
 - merged vertices
 - split vertices



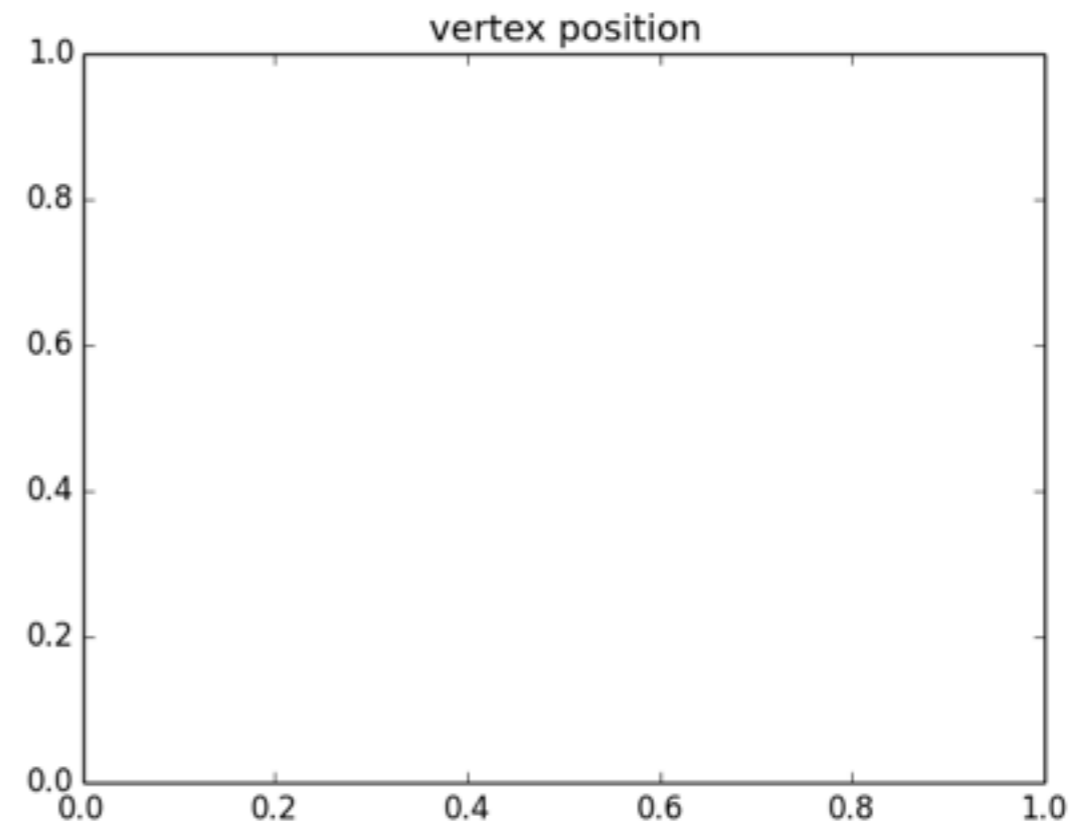
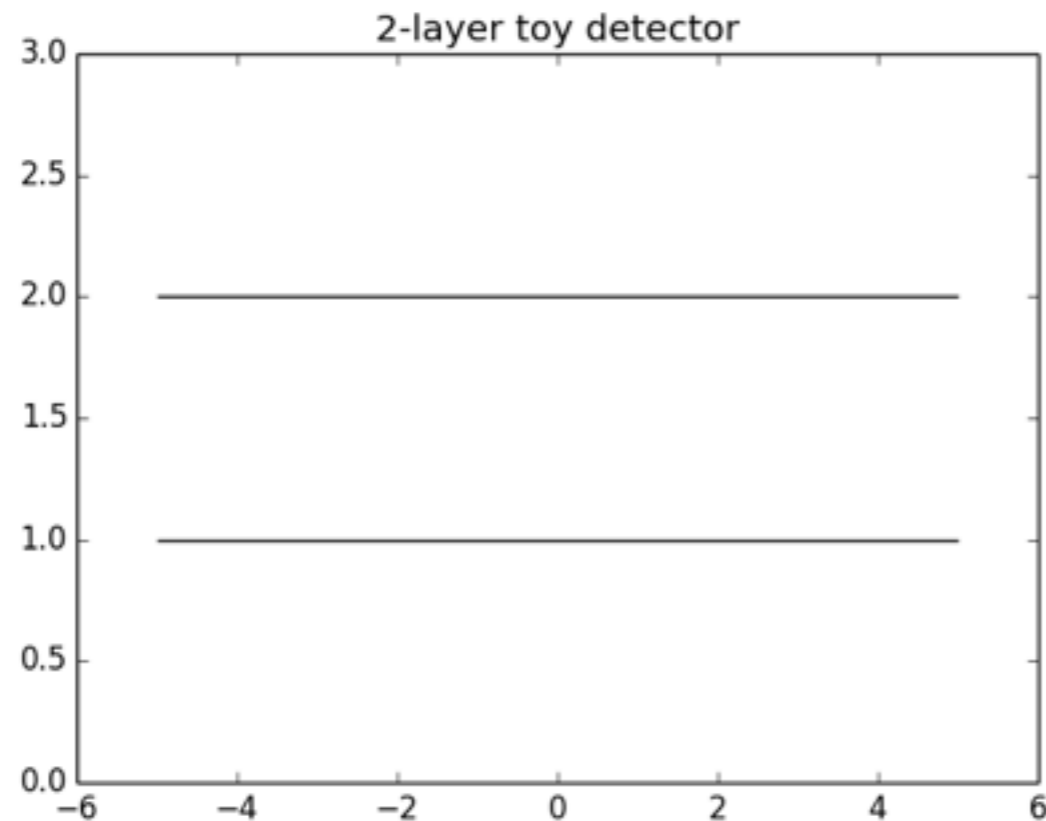
Enemy No. 3: **pile-up**

- ▶ Pile-up let's combinatorics explode
- ▶ This is bad from many angles:
 - CPU time explodes
 - low track fake rate is at risk (make more stringent requirements)
 - track finding becomes more difficult (risk of losing tracks)
 - vertex reconstruction suffers from multiple problems:
 - fake vertices
 - merged vertices
 - split vertices



MC Toy: vertex seed finding with pile-up

- ▶ Our collider switched to pile-up mode



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

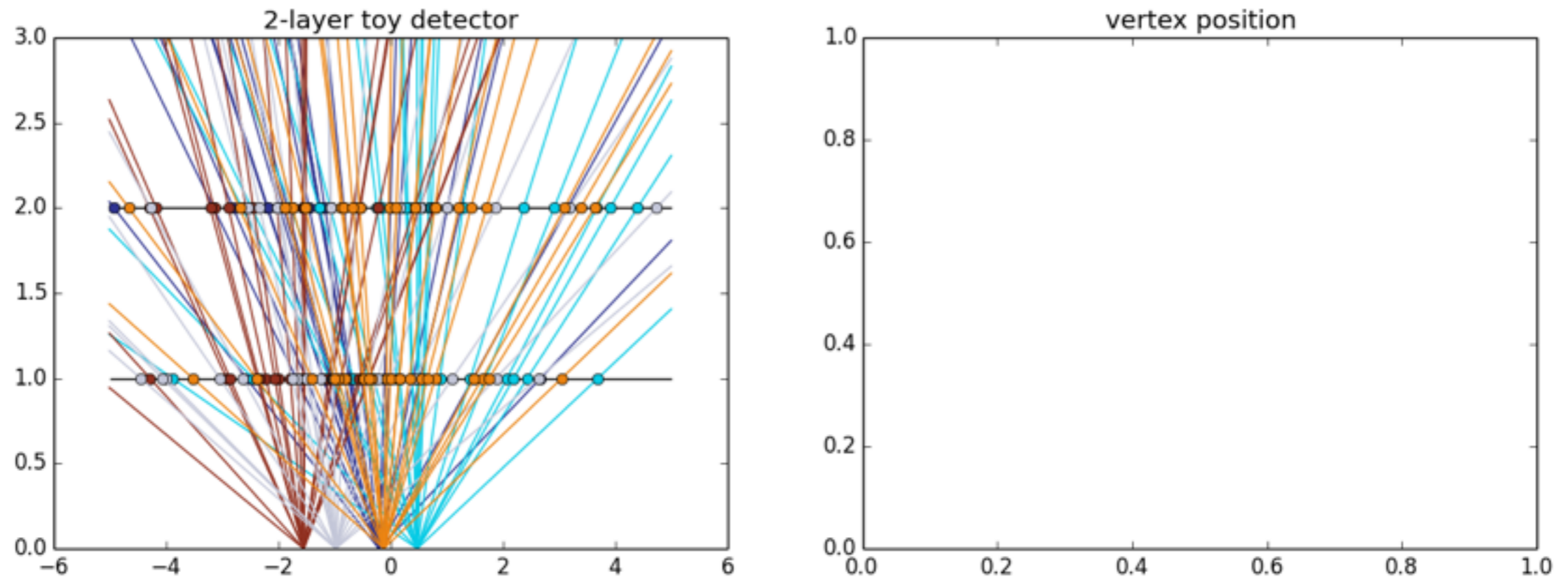
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=1, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding with pile-up

- ▶ Our collider switched to pile-up mode



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

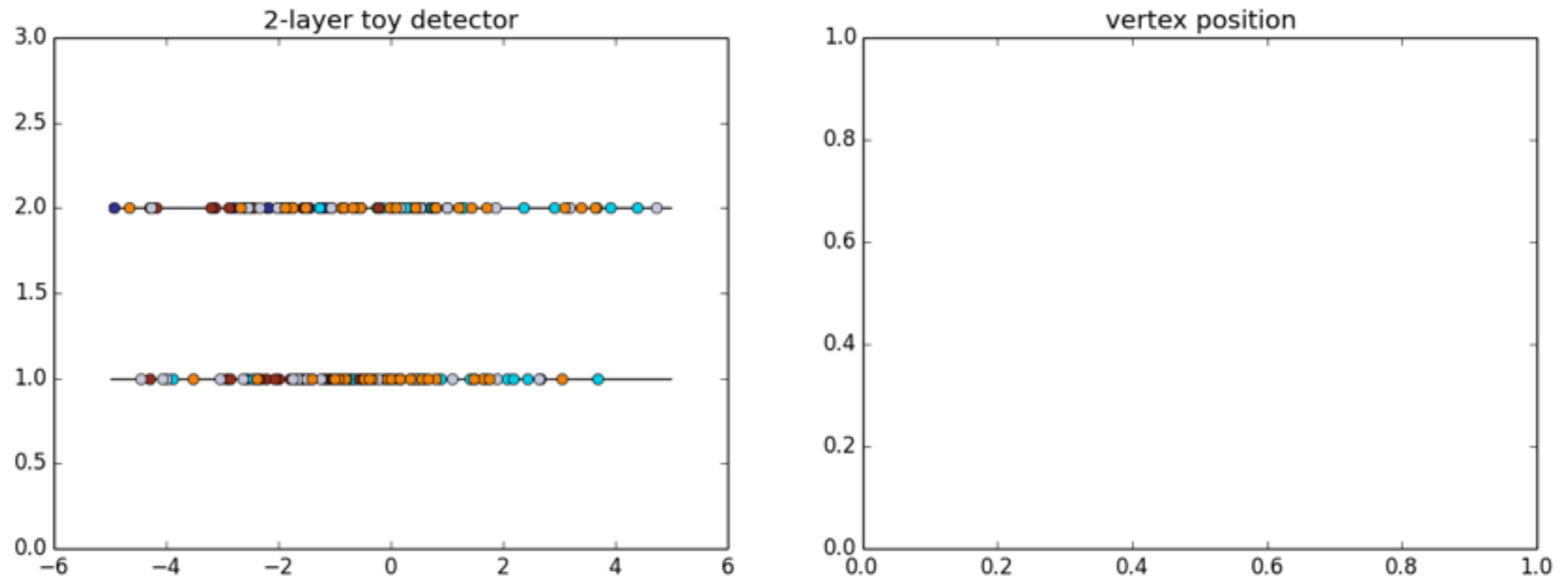
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=1, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding with pile-up

- ▶ Our collider switched to pile-up mode



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

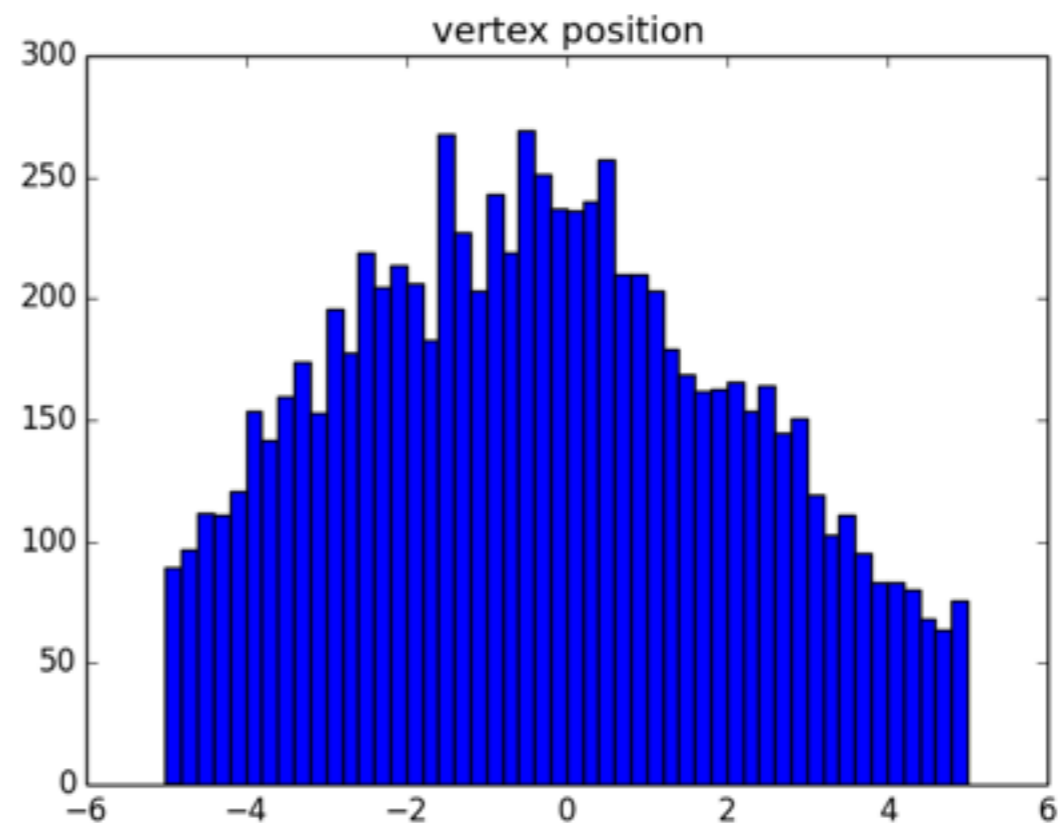
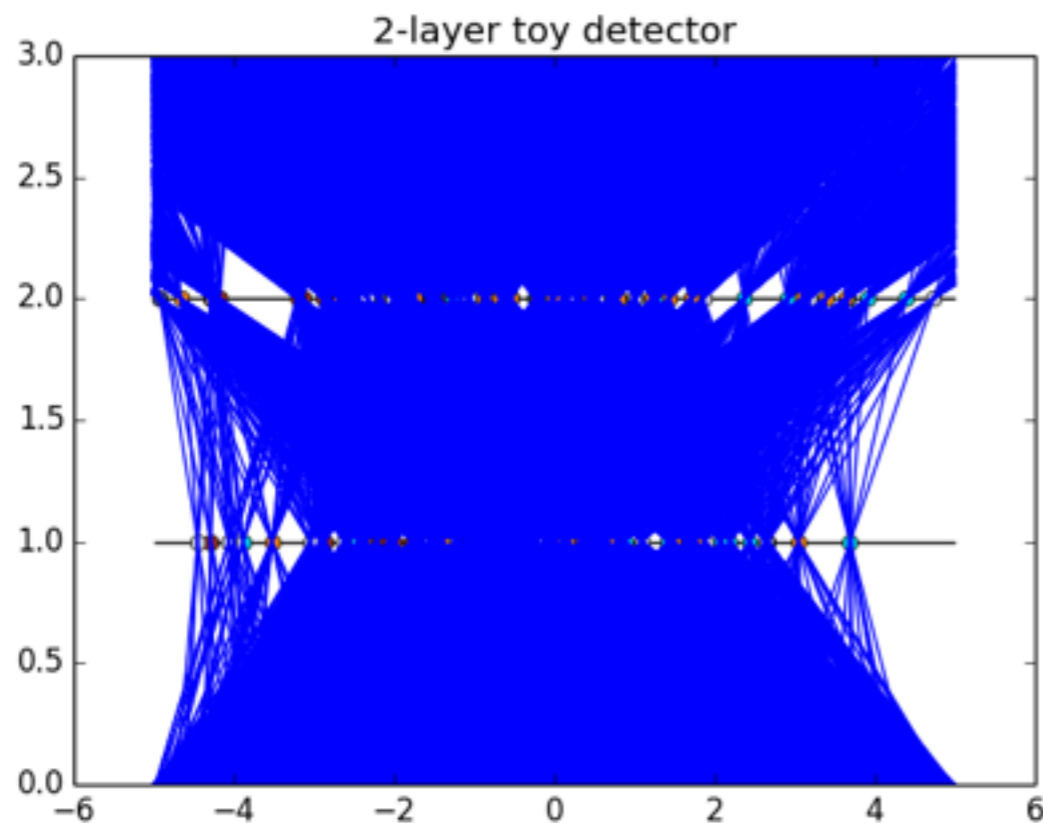
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=1, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding with pile-up

- ▶ Our collider switched to pile-up mode



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

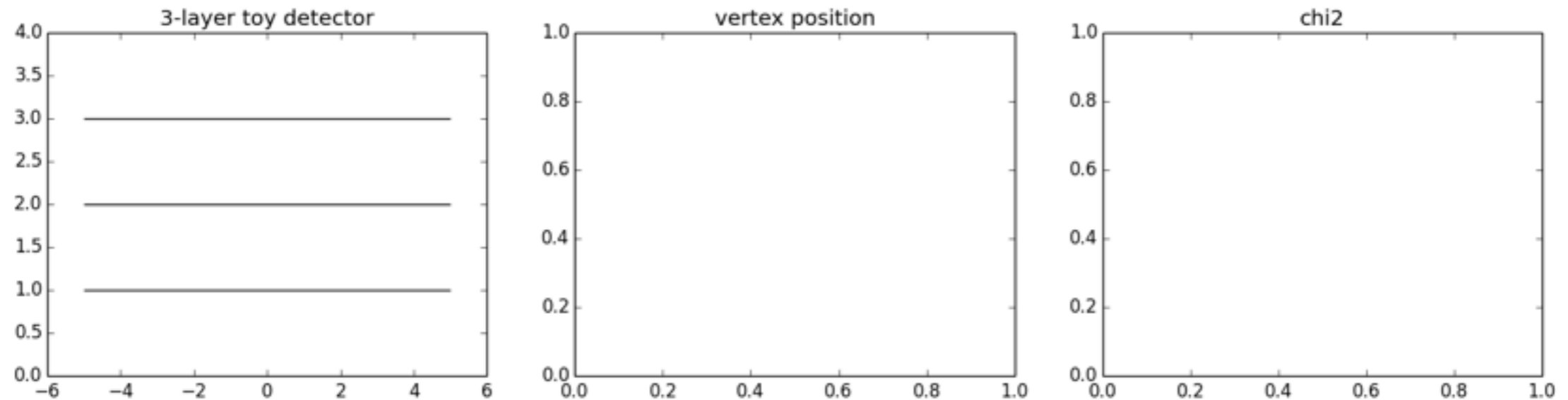
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=1, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```


MC Toy: vertex seed finding in pile-up

- ▶ We need a more complicated detector



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

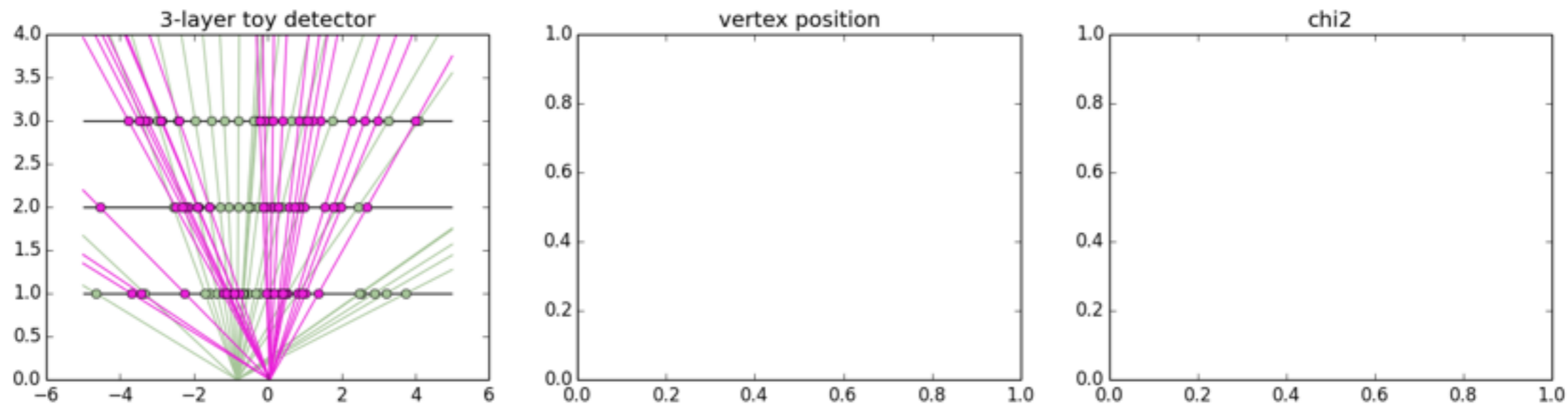
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=5, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding in pile-up

- ▶ We need a more complicated detector



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

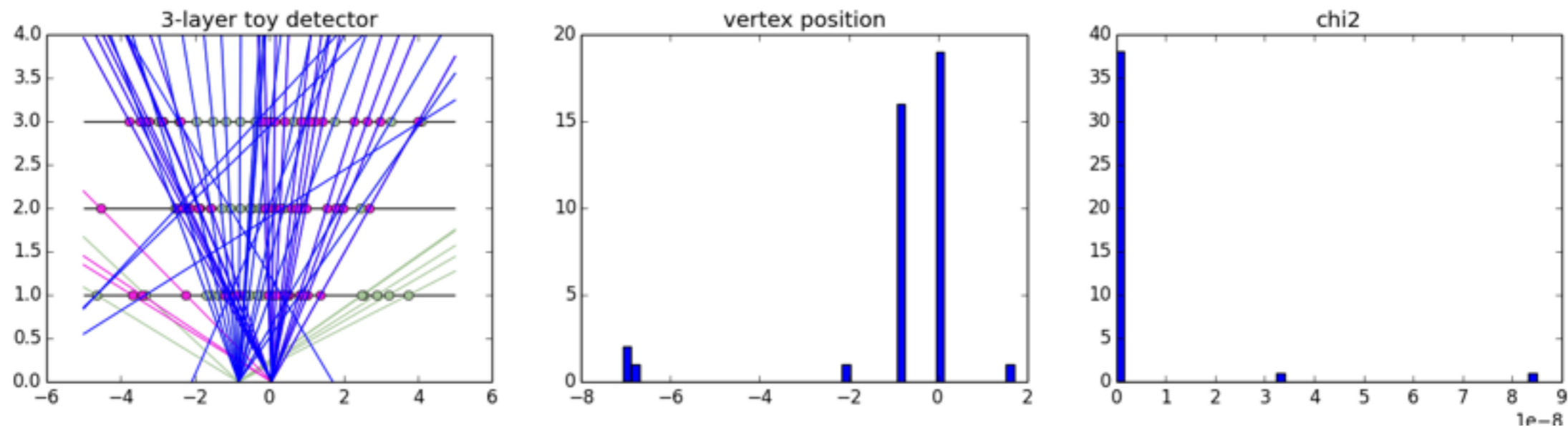
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=5, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding in pile-up

- ▶ We need a more complicated detector



```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

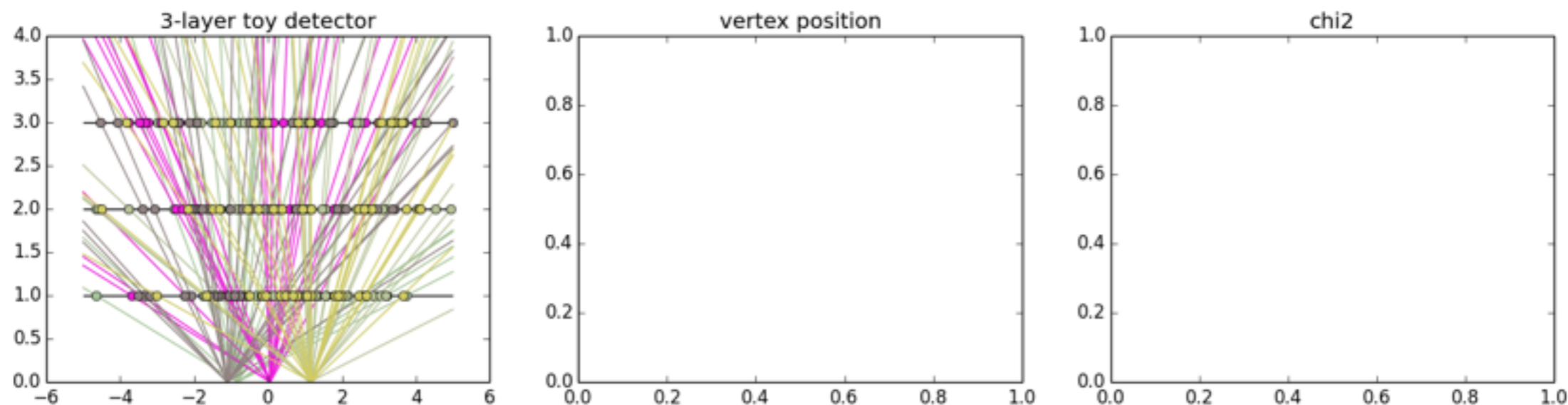
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=5, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding in pile-up

- ▶ We need a more complicated detector



vertex 0 - position is -0.828622777212
vertex 1 - position is 0.0603814871097
vertex 2 - position is 1.12586779511
vertex 3 - position is -1.10335959909
vertex 4 - position is 1.18310584169

```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

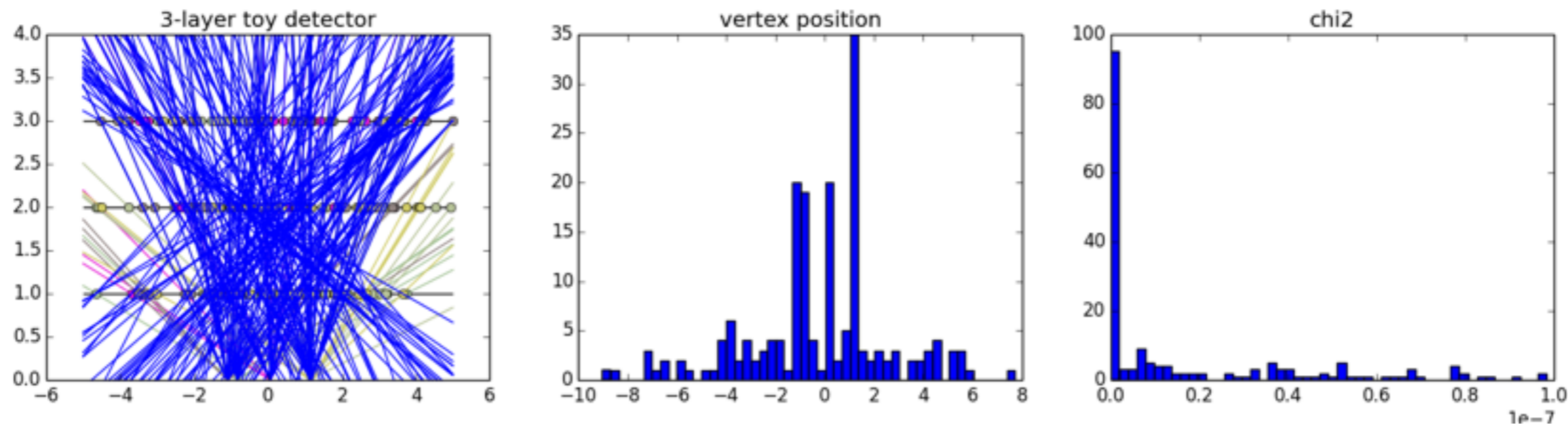
```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=5, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

MC Toy: vertex seed finding in pile-up

- ▶ We need a more complicated detector



vertex 0 - position is -0.828622777212
vertex 1 - position is 0.0603814871097
vertex 2 - position is 1.12586779511
vertex 3 - position is -1.10335959909
vertex 4 - position is 1.18310584169

```
salzburg$ ipython -i --matplotlib=osx VertexFinding.py
```

```
In [1]: fig, plots = buildDetector()
```

```
In [2]: hits = shoot(fig, plots, nvertices=5, number=25, seed=12345)
```

```
In [3]: findVertices(fig, plots, hits)
```

Track reconstruction in analysis

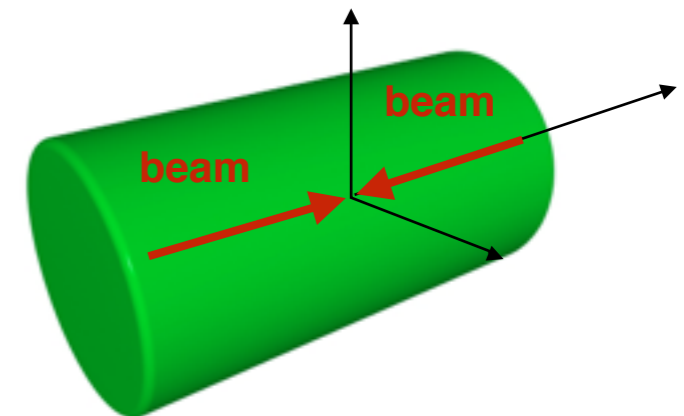
- ▶ The first published analyses of all LHC experiments were tracking based



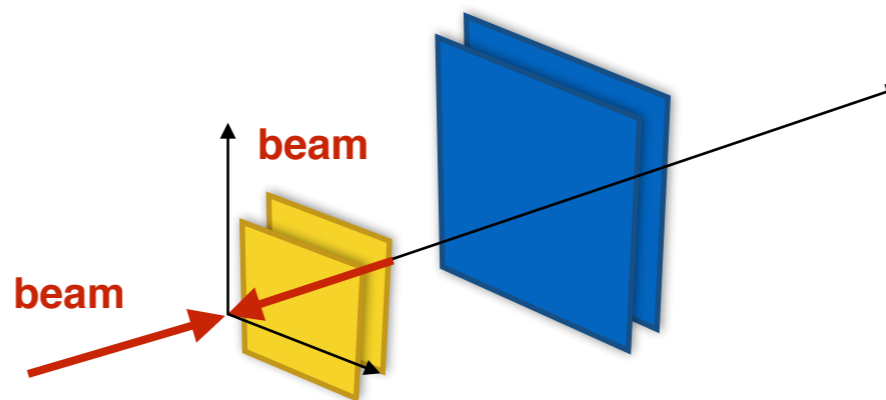
First proton–proton collisions at the LHC as observed with the ALICE detector: measurement of the charged-particle pseudorapidity density at $\sqrt{s}=900$ GeV



Transverse momentum and pseudorapidity distributions of charged hadrons in pp collisions at $\sqrt{s}= 0.9$ and 2.36 TeV



Charged-particle multiplicities in pp interactions at $\sqrt{s} = 900$ GeV measured with the ATLAS detector at the LHC



Prompt K_S^0 production in pp collisions at $\sqrt{s} = 0.9$ TeV



Example: charged particle multiplicity

- ▶ A three-step master plan to a Soft QCD paper :

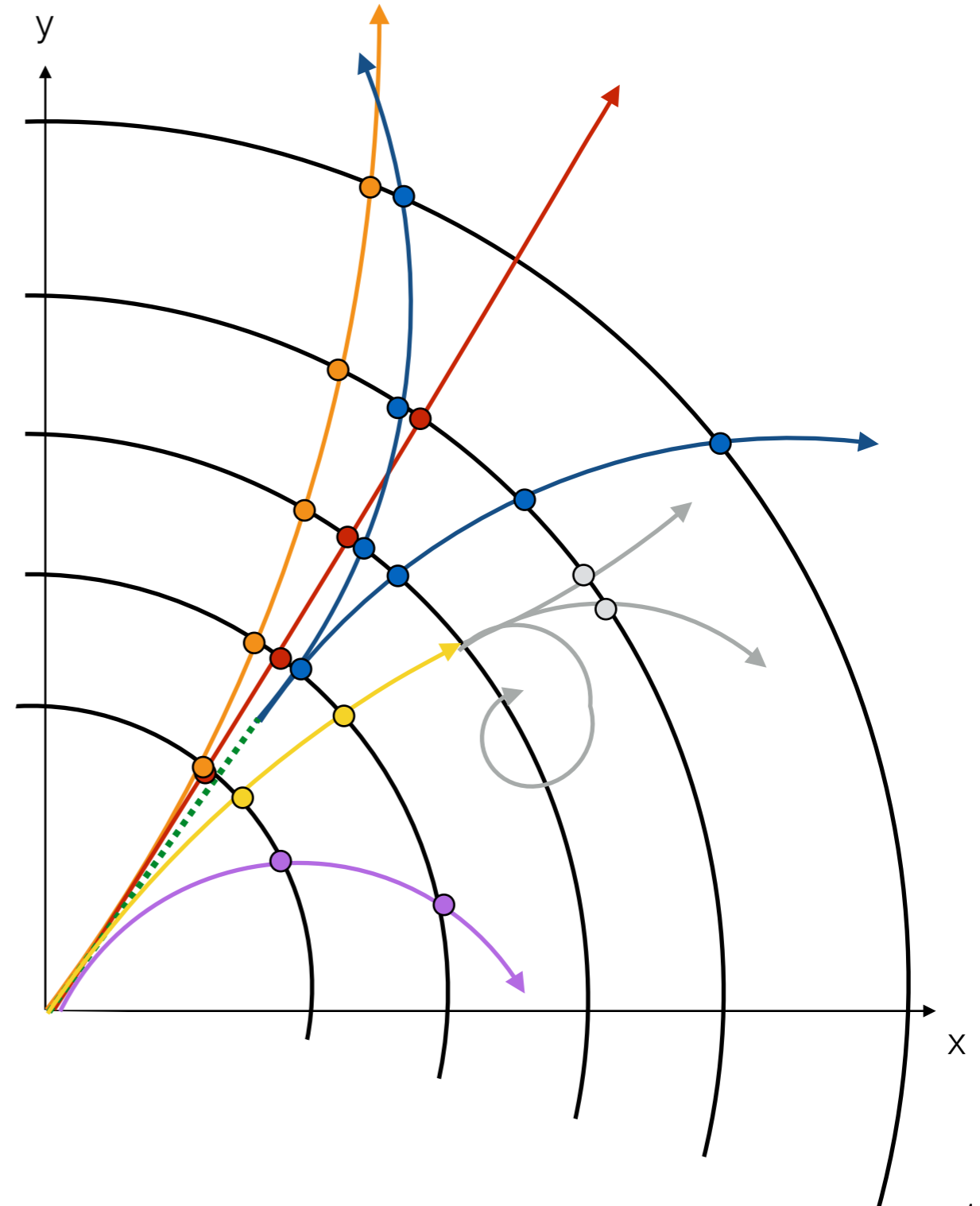
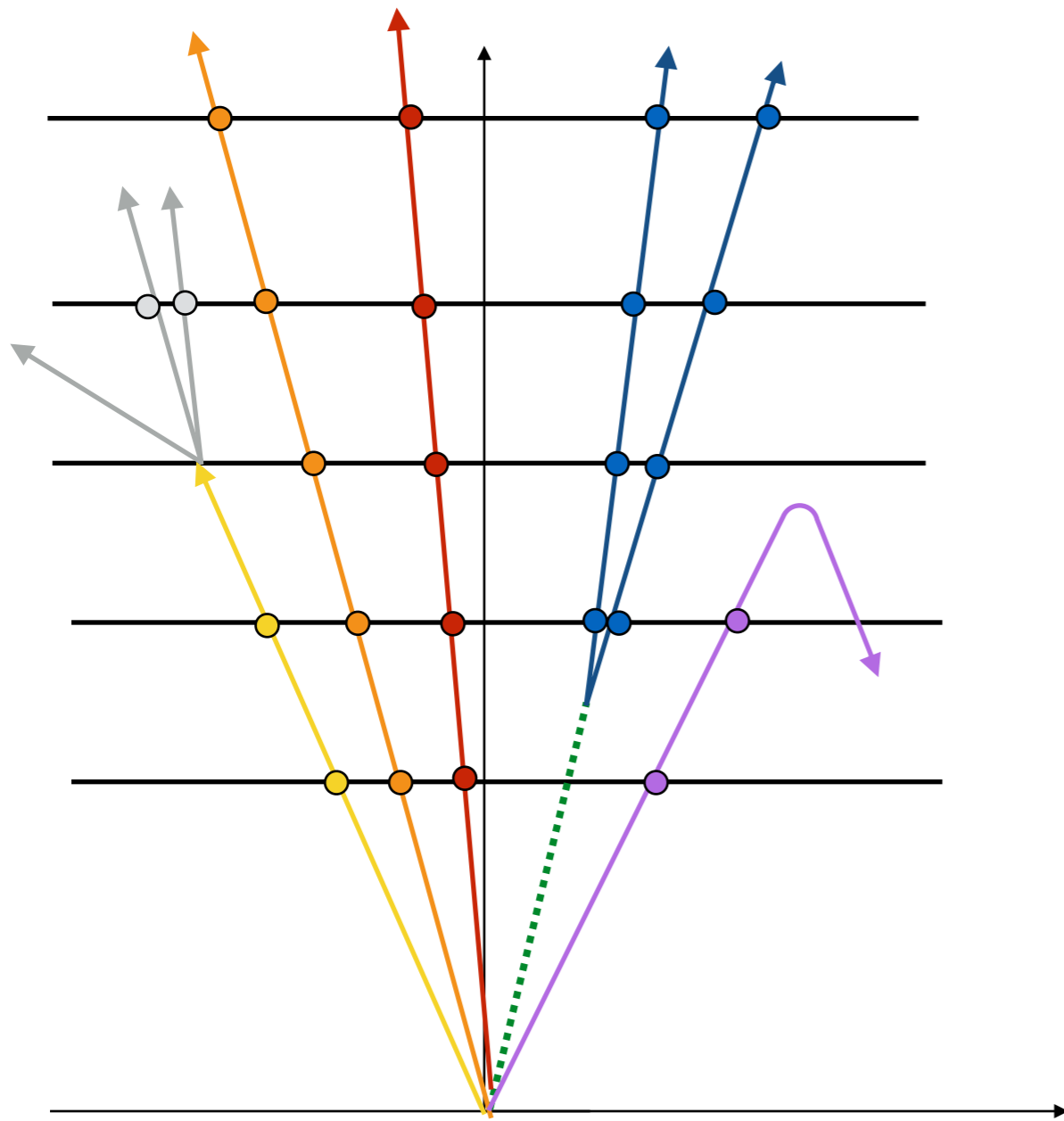
(might be "slightly" simplified)

1. take data from generic p-p collisions without bias towards some signature ("Minimum bias measurement")
2. count how many primary tracks you find (e.g. vs η , vs p_T)
3. correct for tracks you have not found and publish

Let's do it !

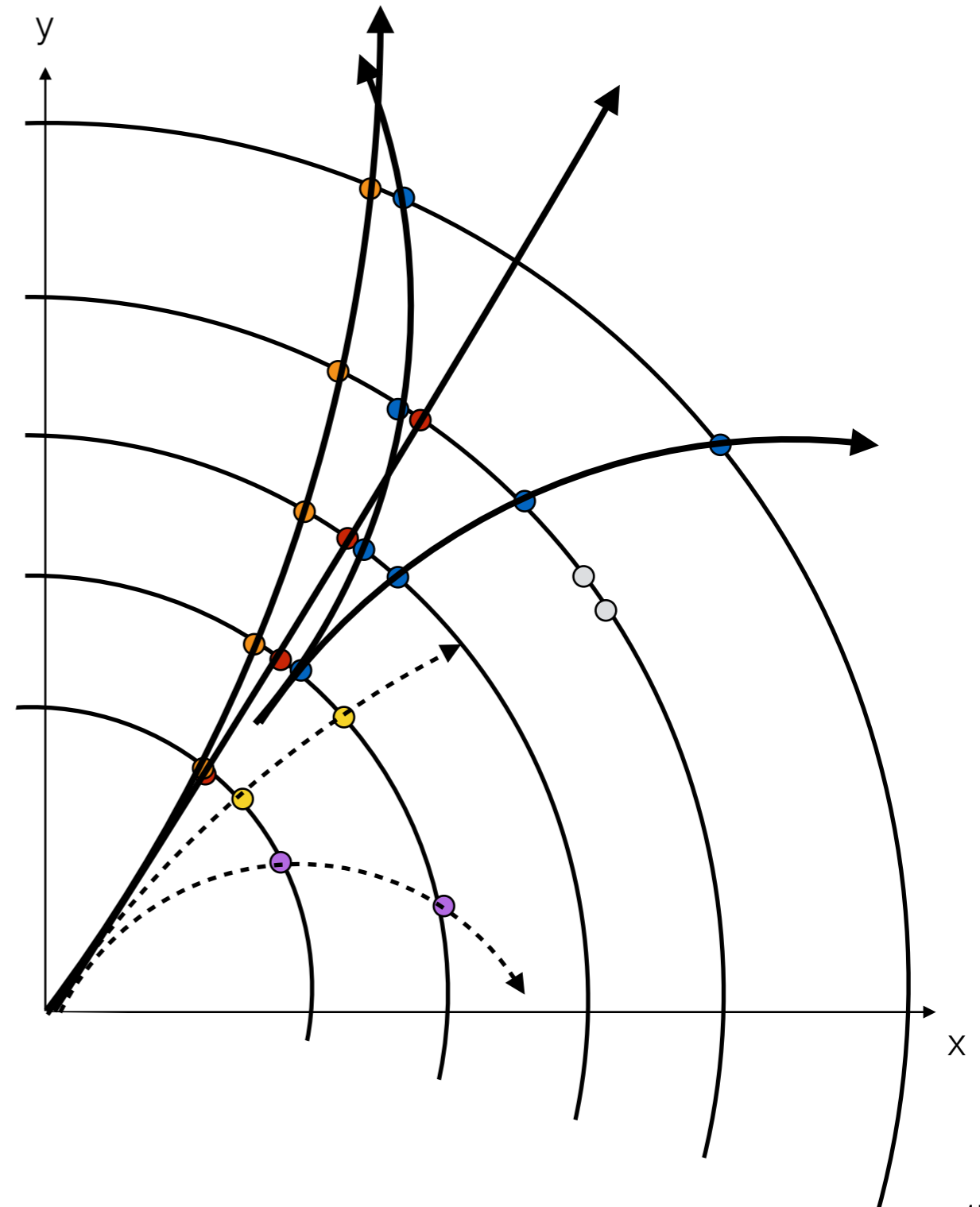
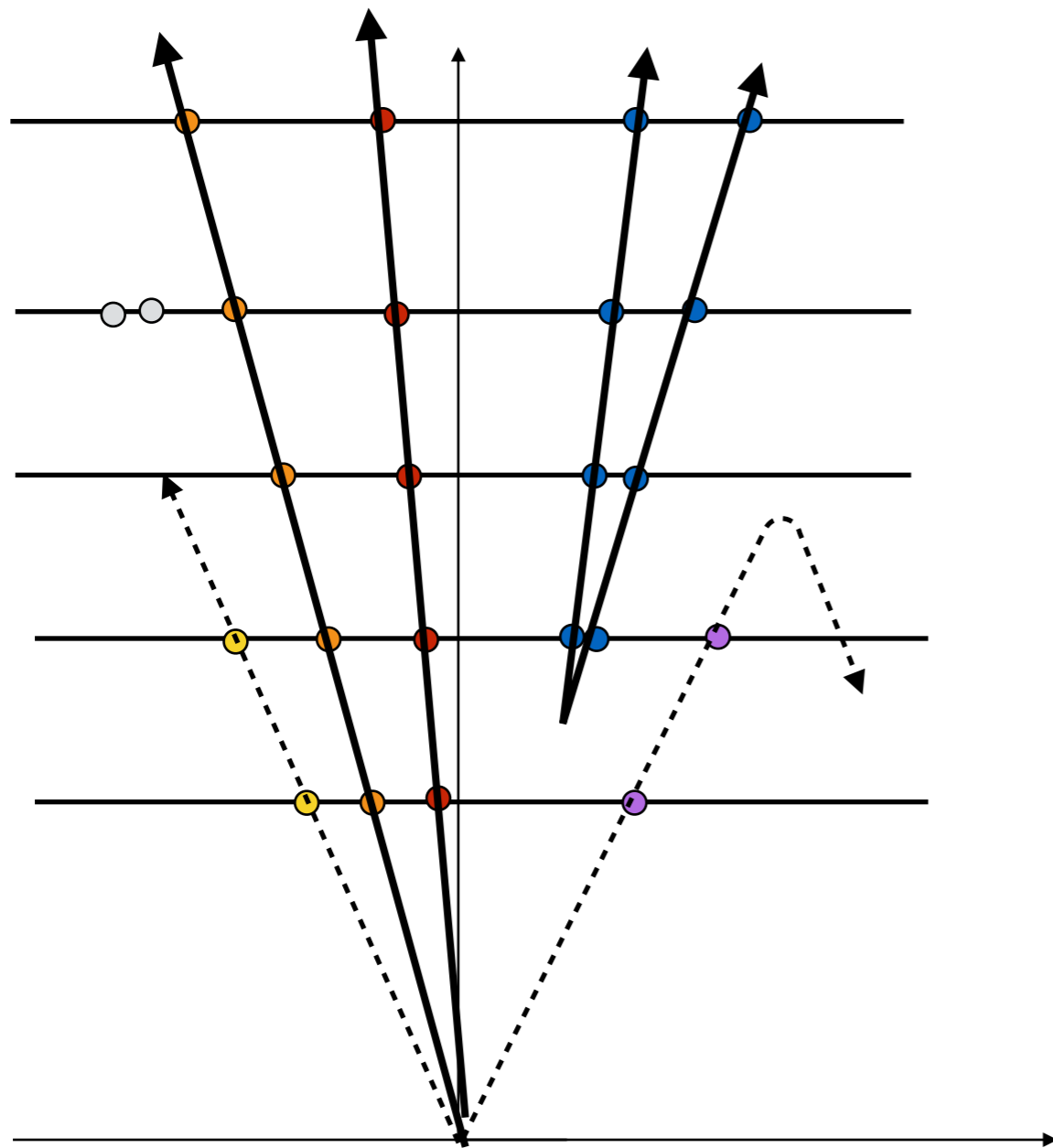
Example: charged particle multiplicity

- ▶ Let's roll the dice ourselves



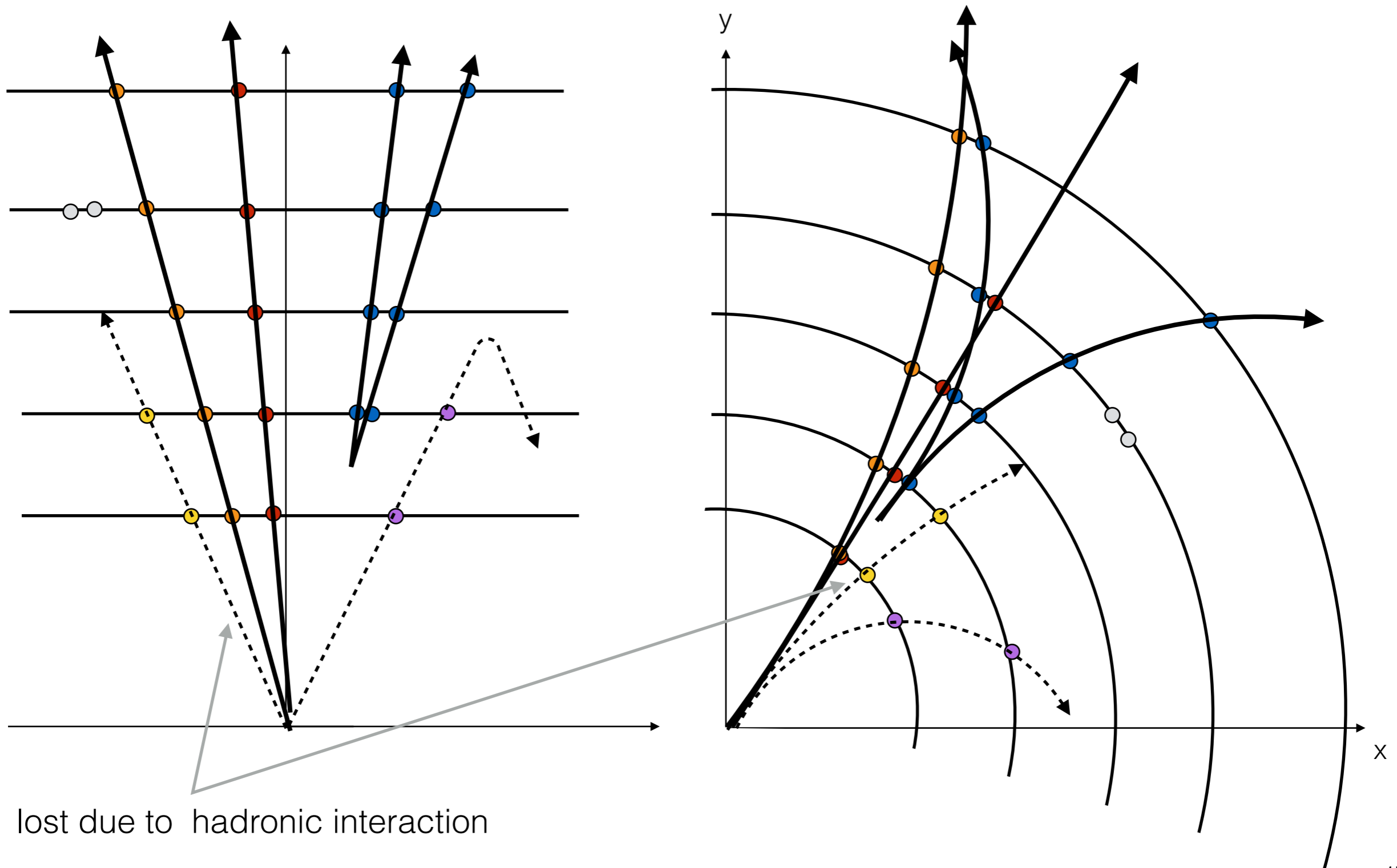
Example: charged particle multiplicity

- ▶ And that's what the track reconstruction gives us



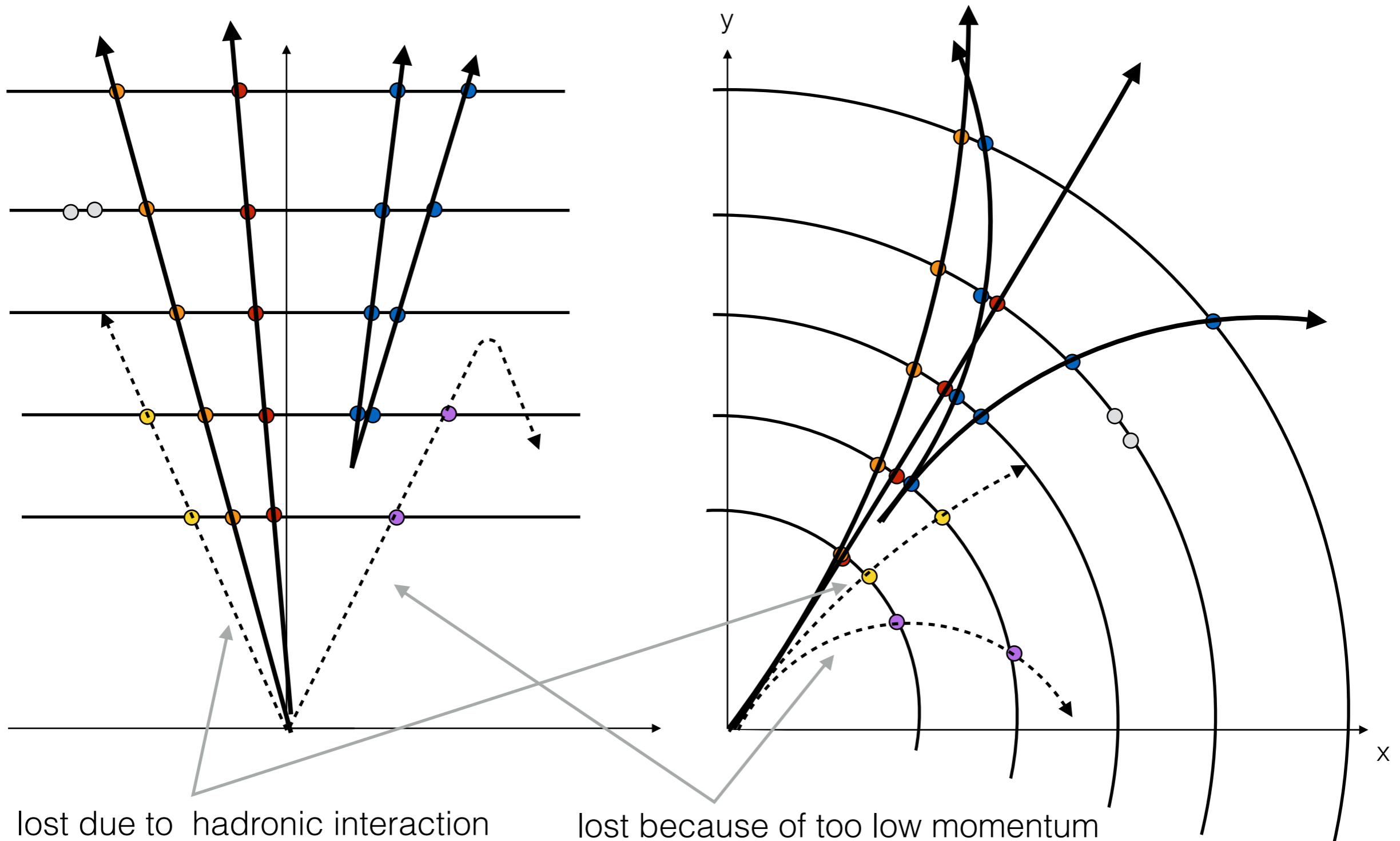
Example: charged particle multiplicity

- ▶ And that's what the track reconstruction gives us



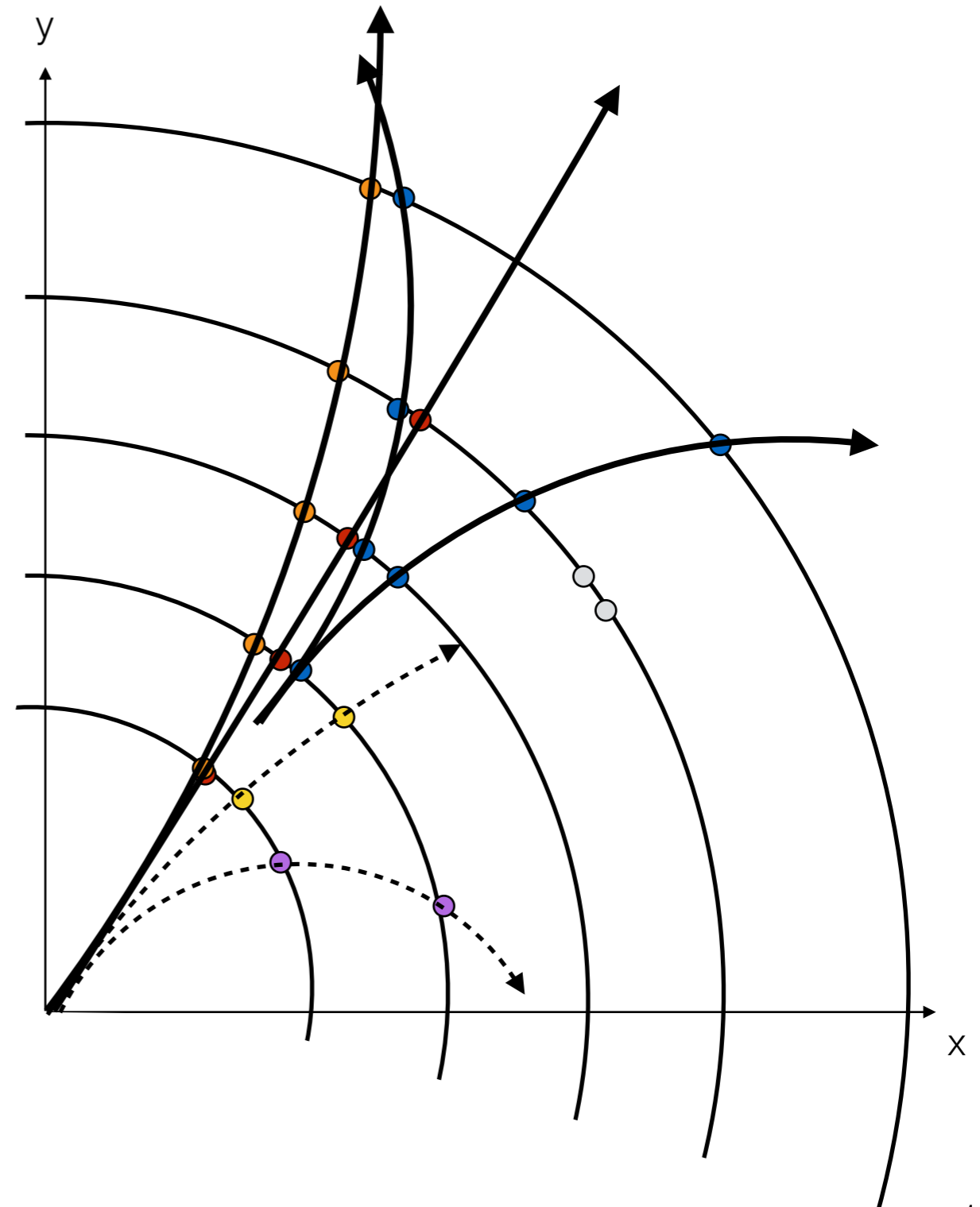
Example: charged particle multiplicity

- ▶ And that's what the track reconstruction gives us



We need to define a tracking efficiency

- ▶ There is **no** such thing as a tracking efficiency
 - restricted to phase-space
 - restricted to your definitions of a track (requirements and cuts)
 - different for different particles
- ▶ For checking the algorithmic performance
 - define a “reconstructable track” in Monte Carlo and check how often you can find it
 - can we do that in data ?



Tracking efficiency

- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates
 - for electrons also bremsstrahlung needs to be considered
- ▶ Tracking efficiency for hadrons is usually taken from Monte Carlo
 - defined by :

$$\mathcal{E} = \frac{\text{Number of found tracks*and matched to a truth particle}}{\text{Number of generated stable charged particles**}}$$

*satisfying my track cuts

**satisfying my phase space

Tracking efficiency

- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates
 - for electrons also bremsstrahlung needs to be considered
- ▶ Tracking efficiency for hadrons is usually taken from Monte Carlo
 - defined by :

$$\mathcal{E} = \frac{\text{Number of found tracks*and matched to a truth particle}}{\text{Number of generated stable charged particles**}}$$

Perfect ! Now we can measure it ...

*satisfying my track cuts

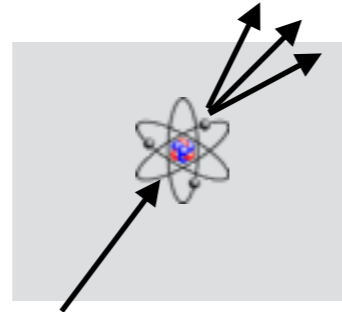
**satisfying my phase space

Tracking efficiency

- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates

Tracking efficiency

- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates



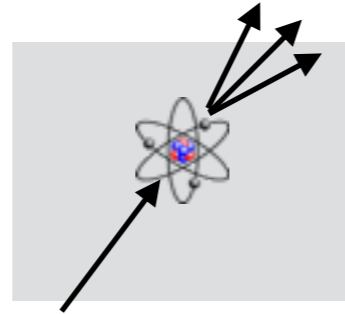
remember ?

“ ... rather constant in p ... ”

“ ... defined by nuclear interaction length Λ_0 ... ”

Tracking efficiency

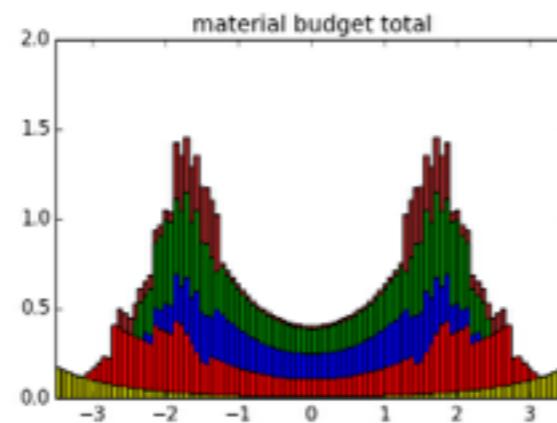
- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates



remember ?

“ ... rather constant in p ... ”

“ ... defined by nuclear interaction length Λ_0 ... ”

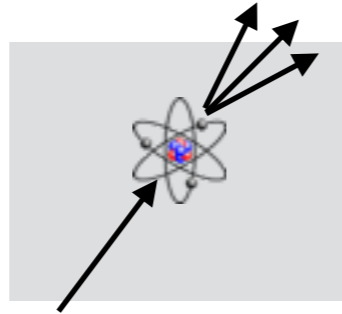


remember ?

our built-up material map ...

Tracking efficiency

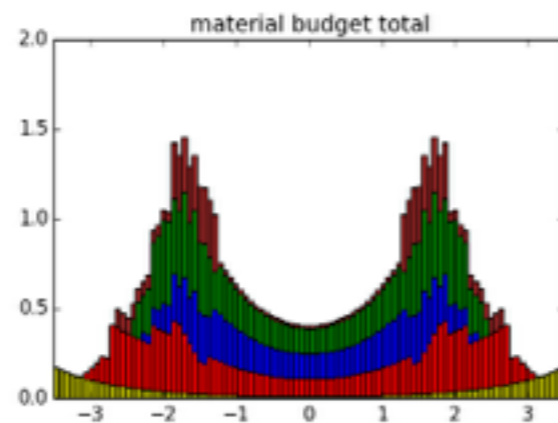
- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates



remember ?

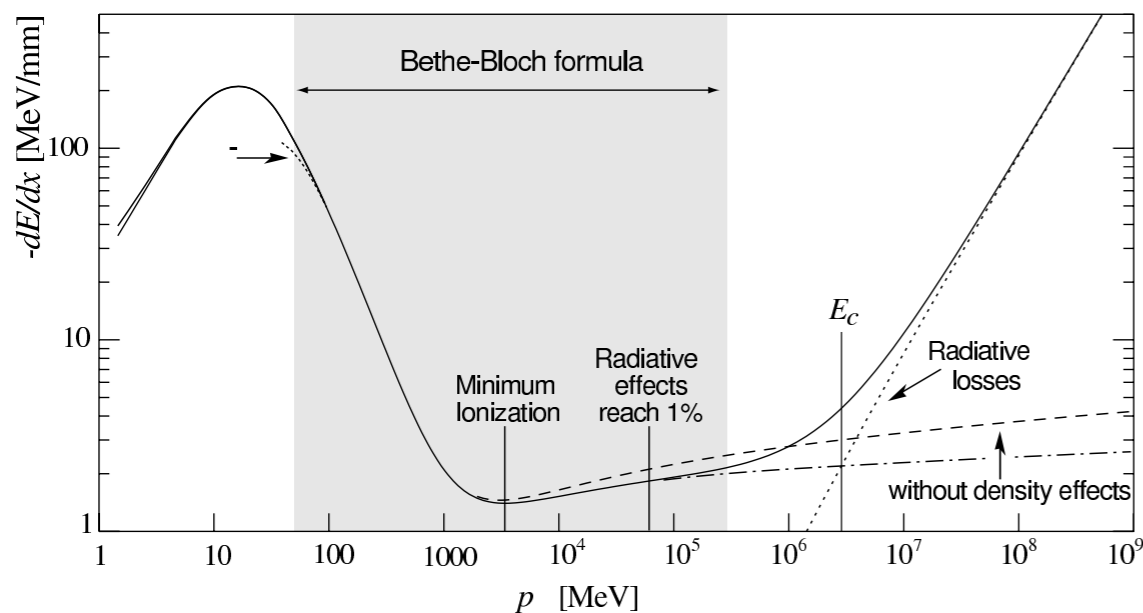
“ ... rather constant in p ... ”

“ ... defined by nuclear interaction length Λ_0 ... ”



remember ?

our built-up material map ...

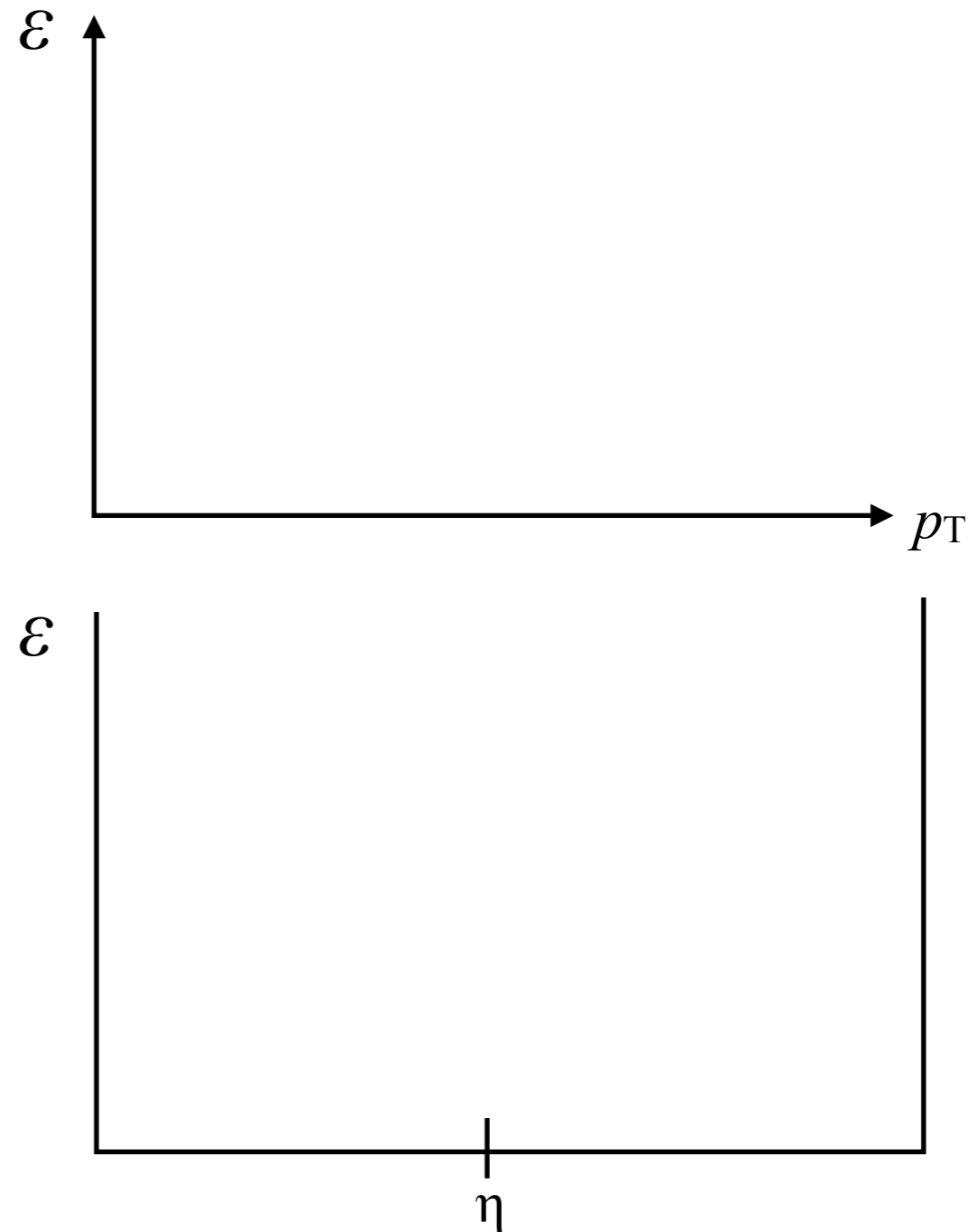
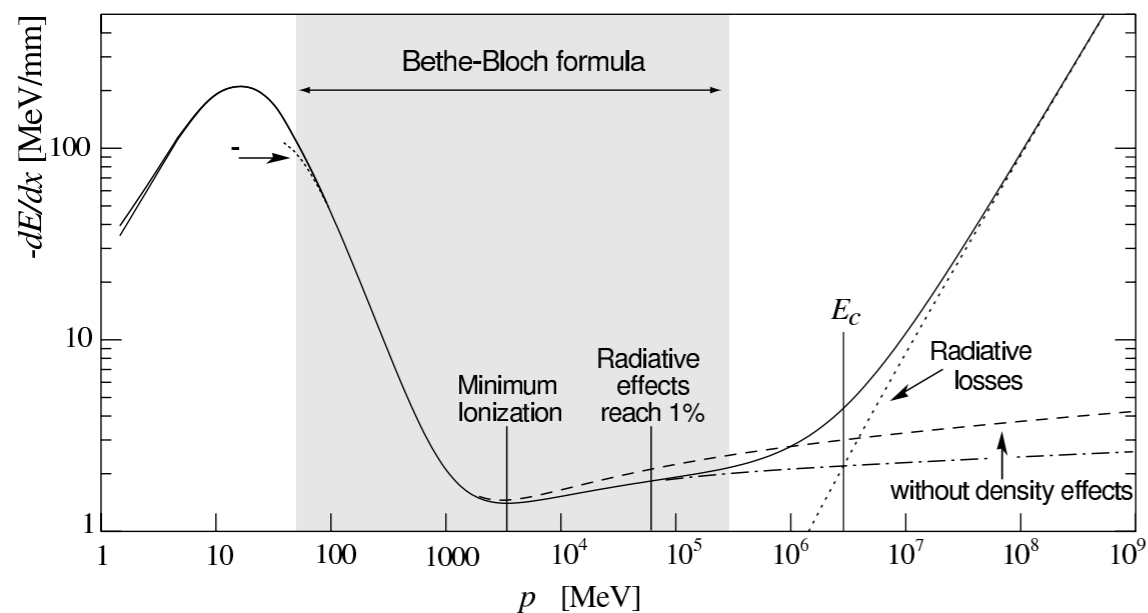
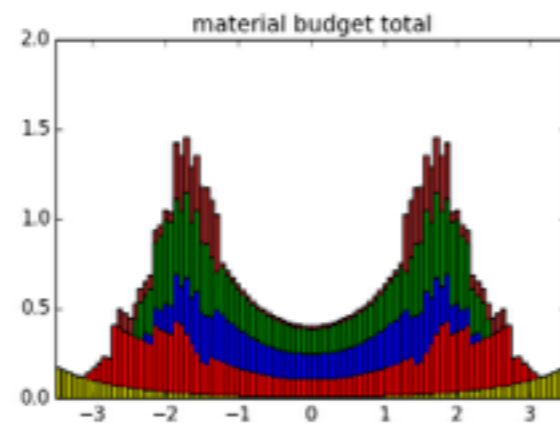
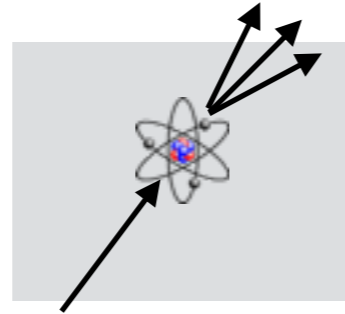


remember ?

both ionisation loss and multiple scattering act most at low p_T in the minimum bias regime

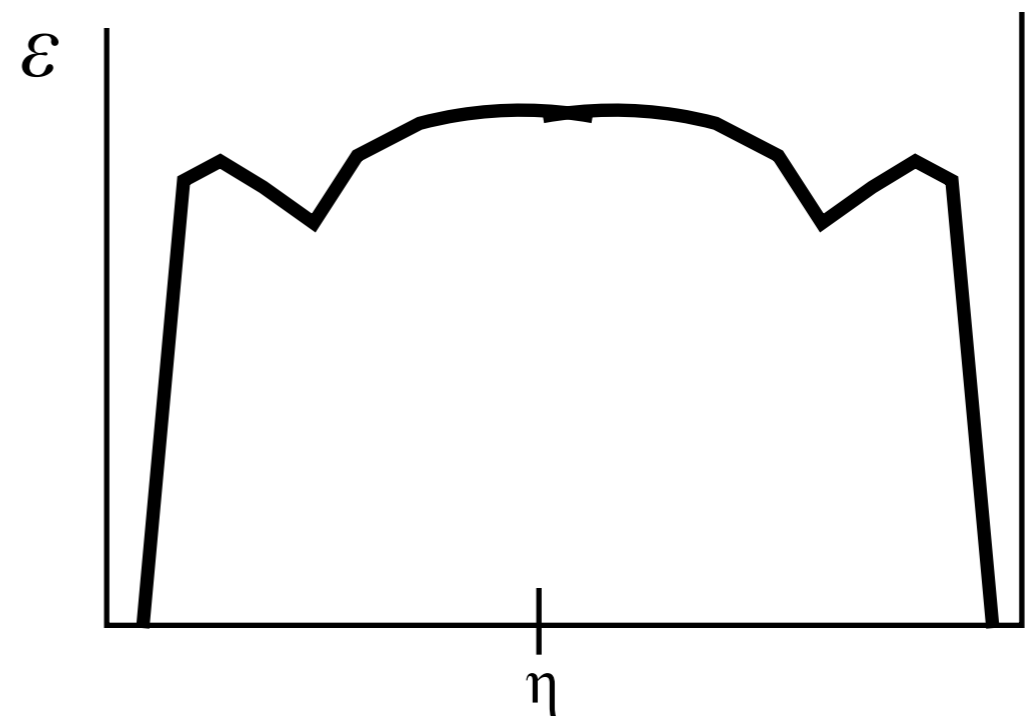
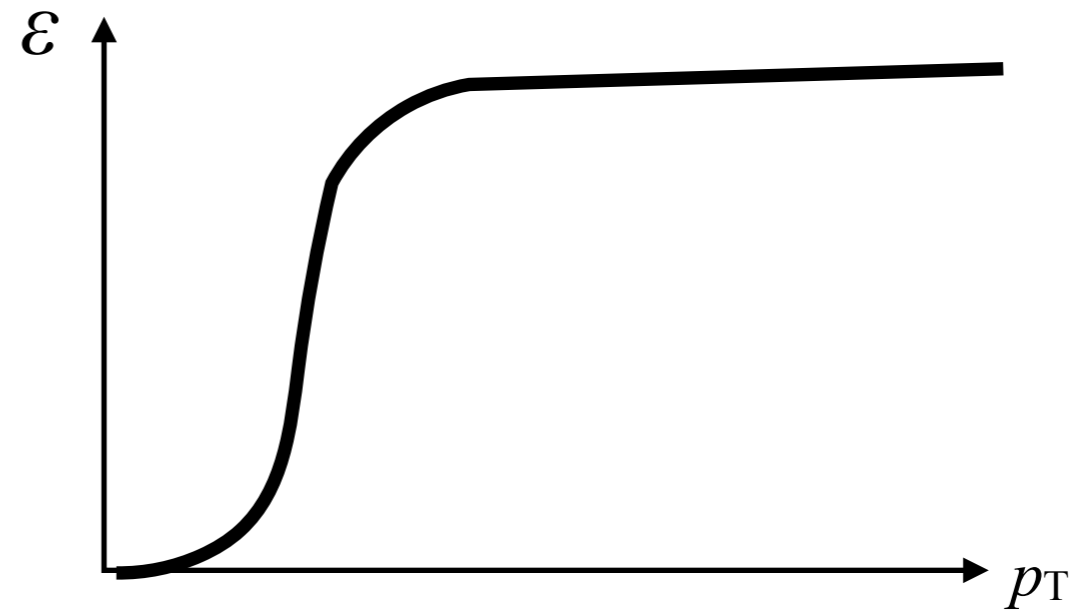
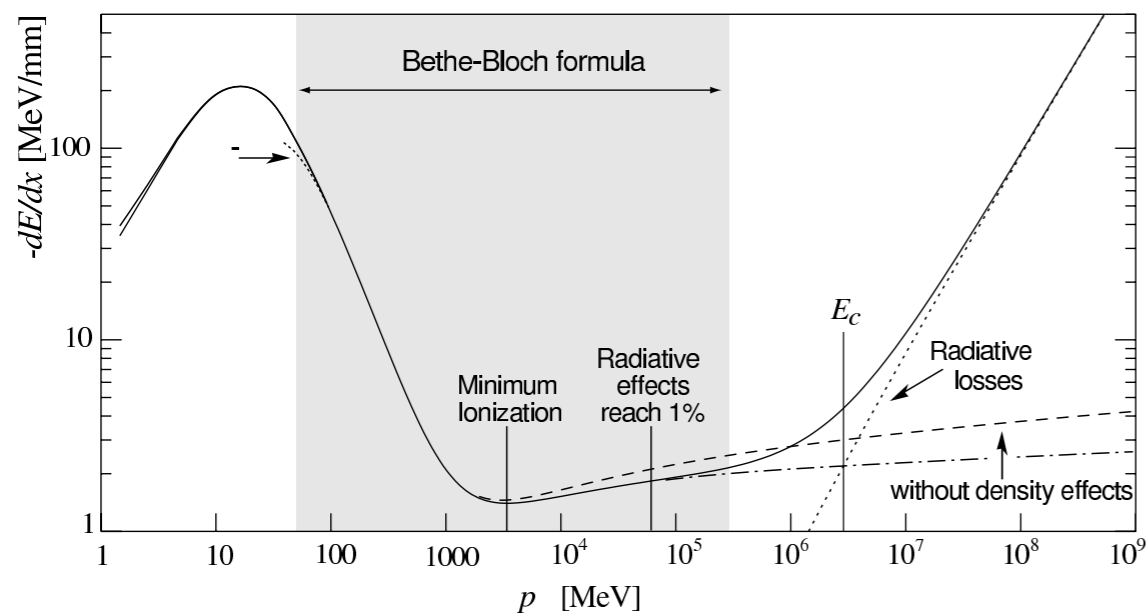
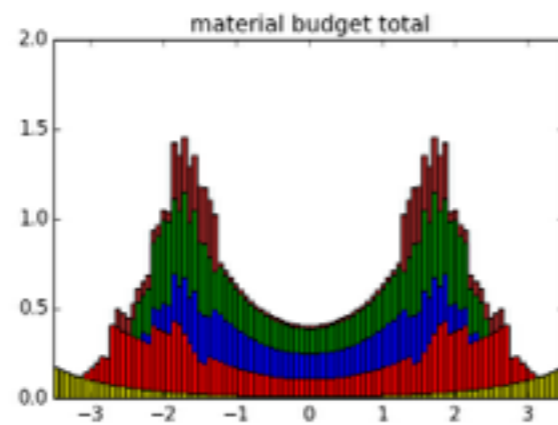
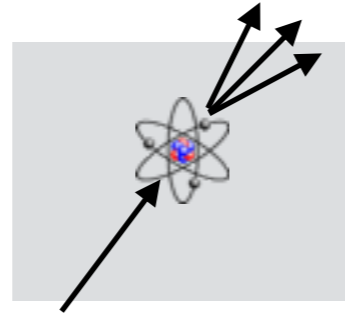
Tracking efficiency

- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates



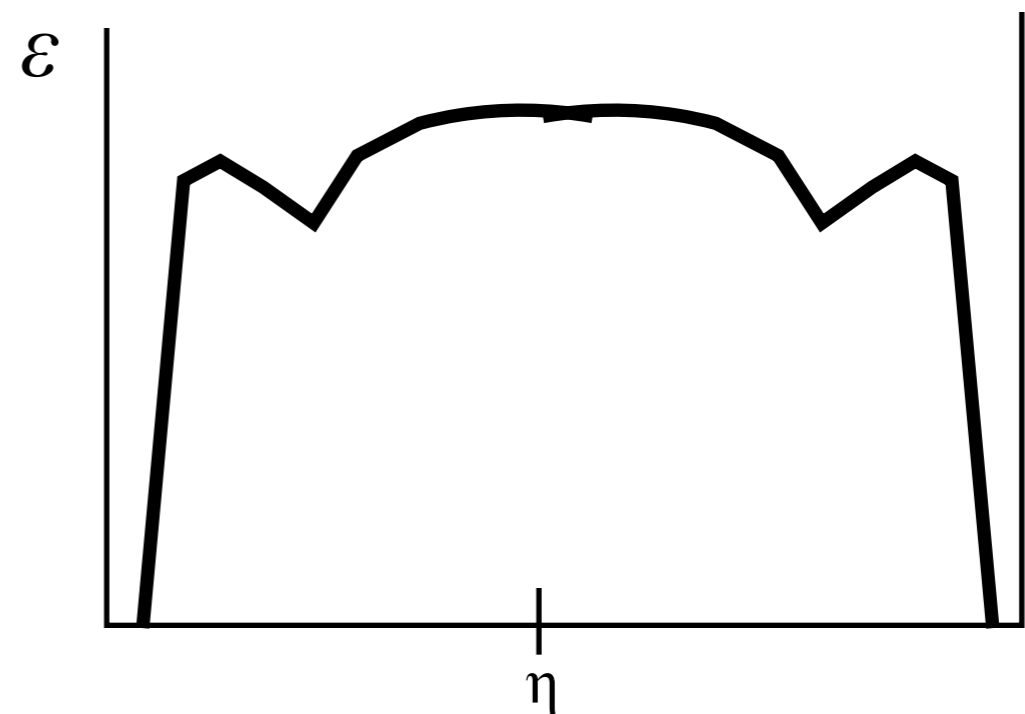
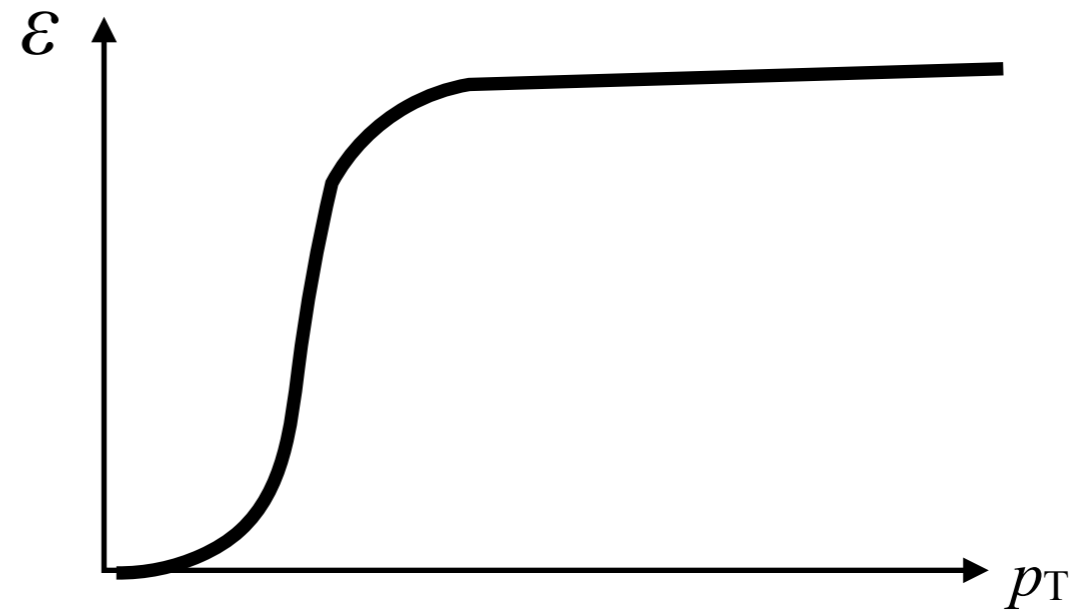
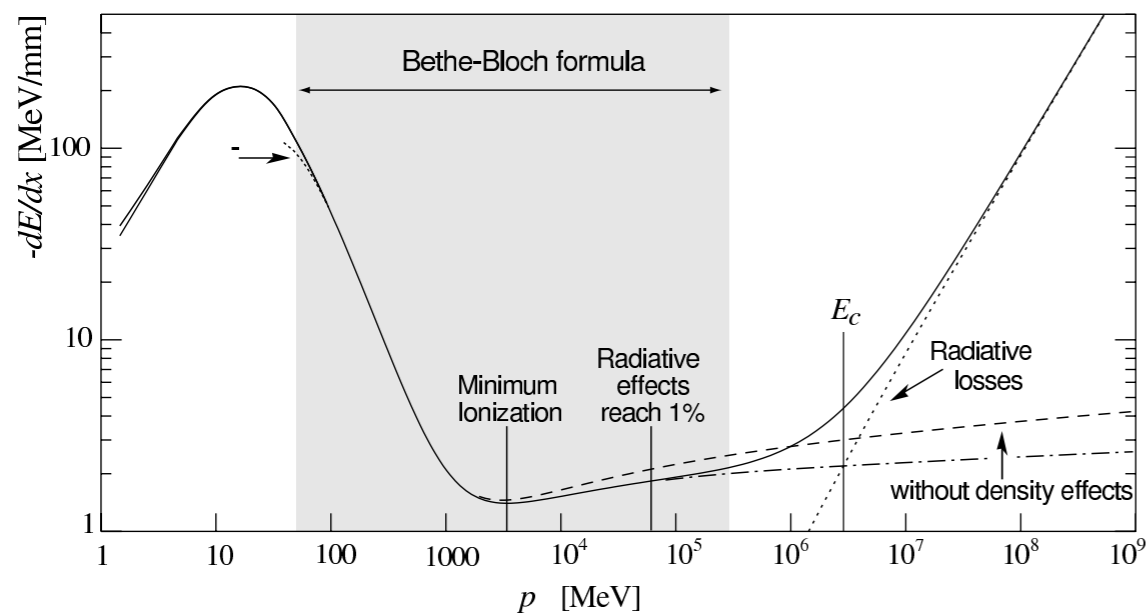
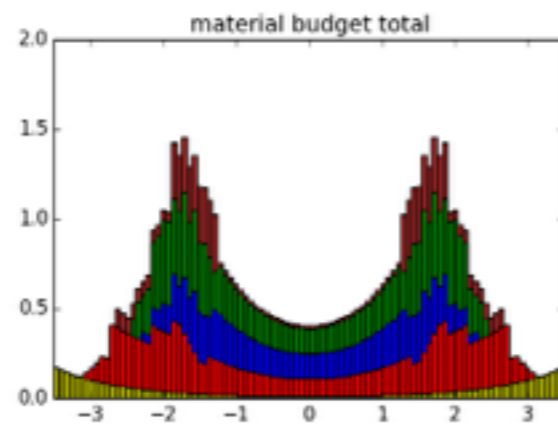
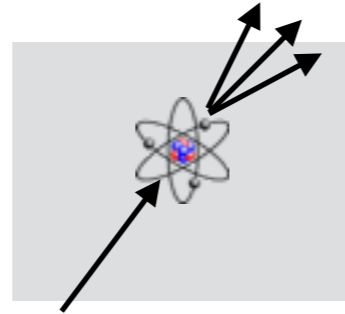
Tracking efficiency

- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates



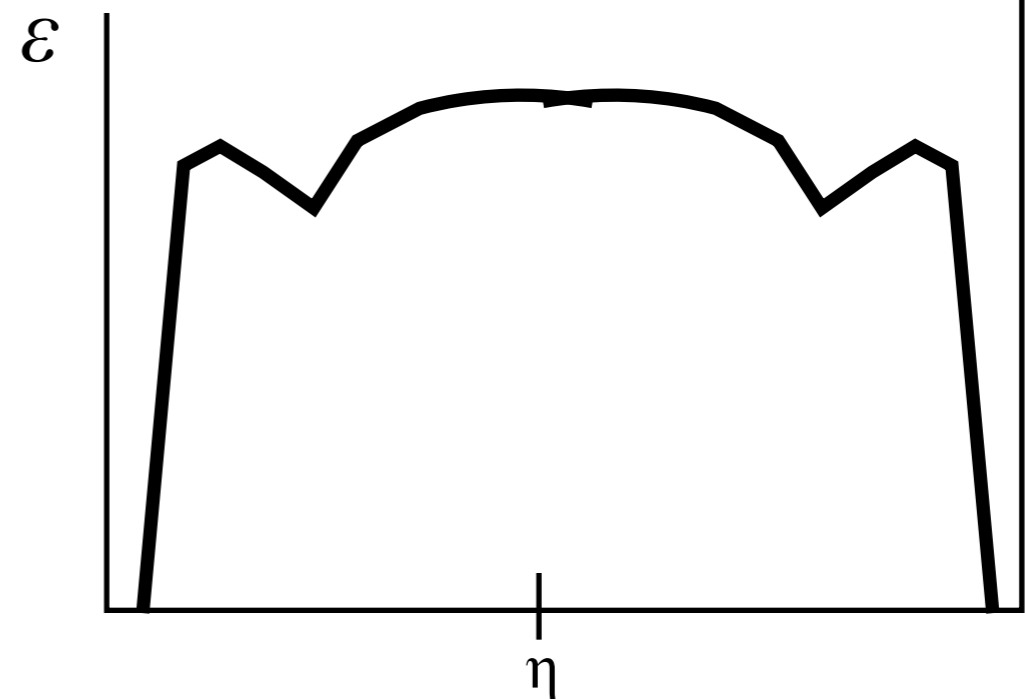
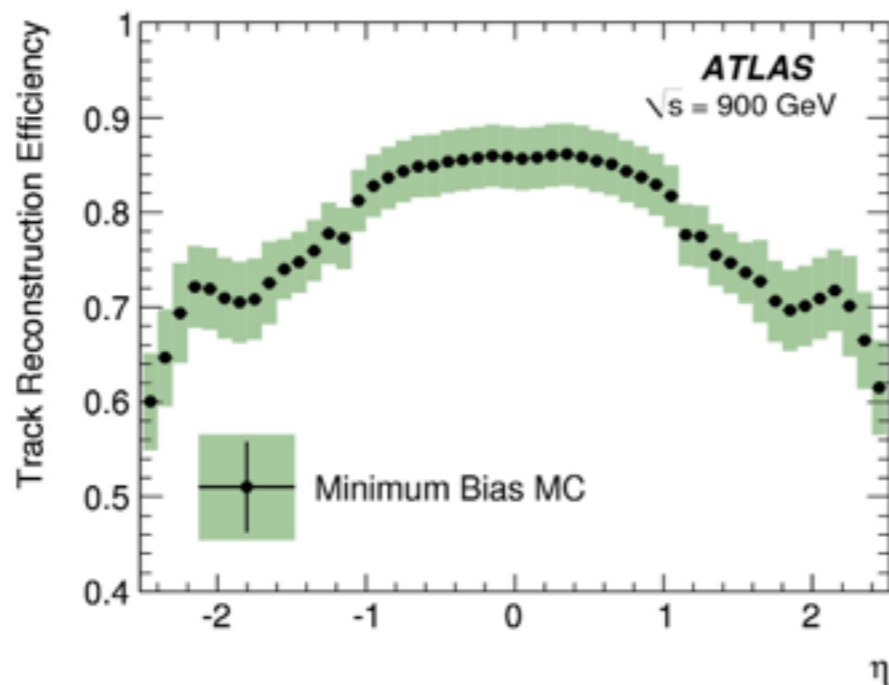
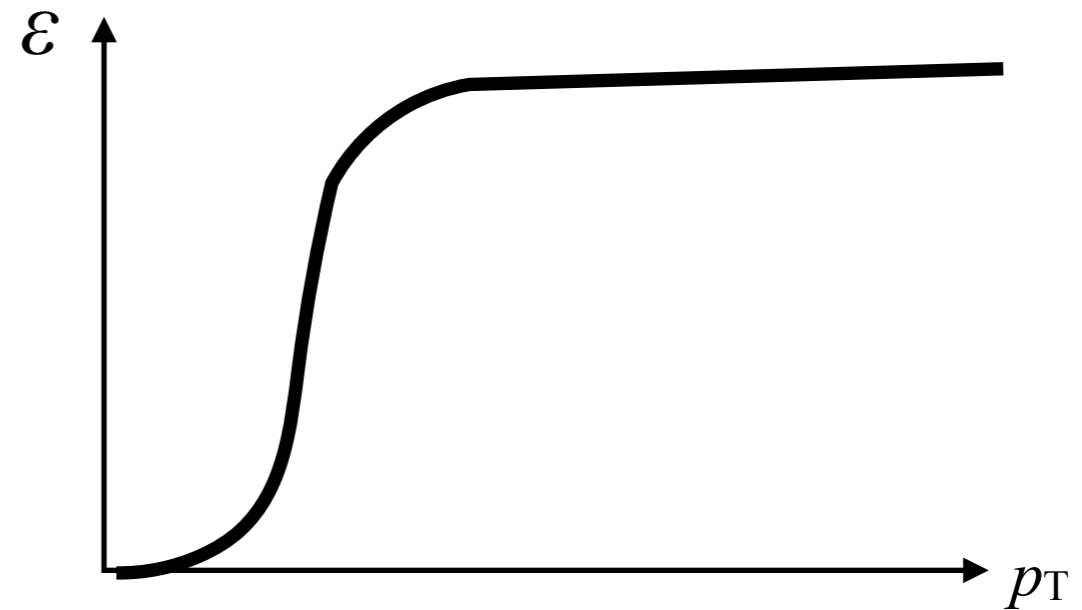
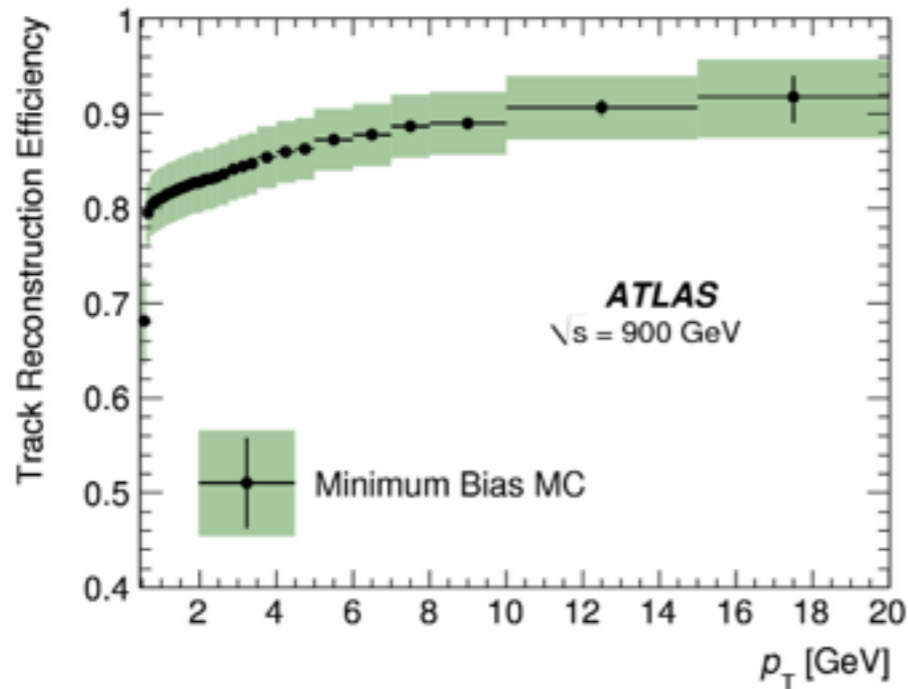
Tracking efficiency

- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates



Tracking efficiency

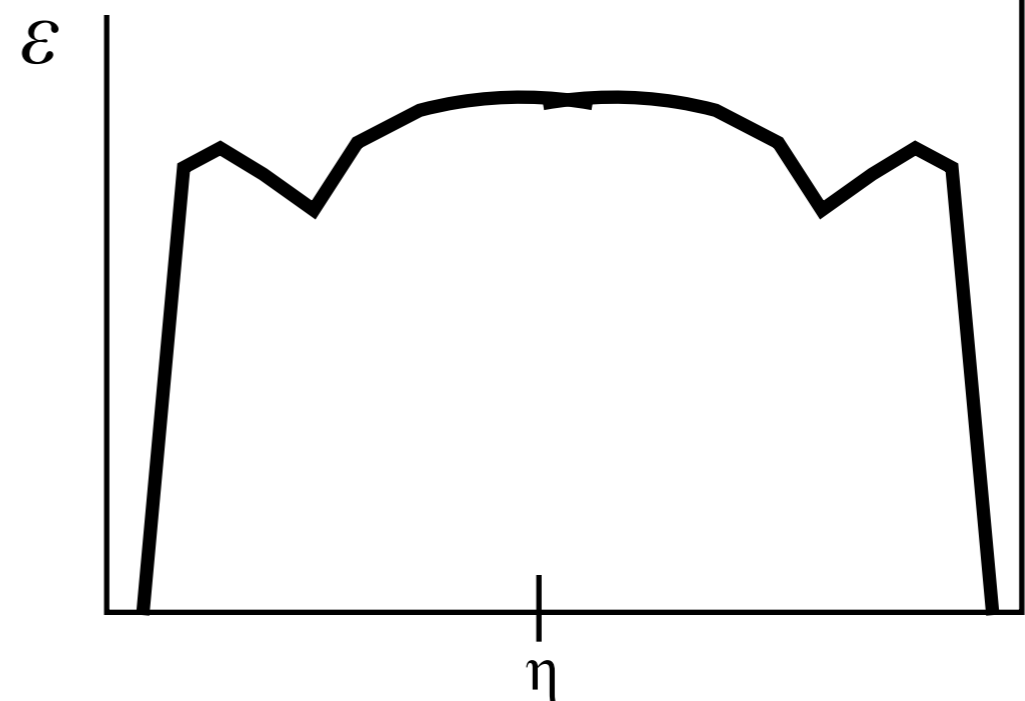
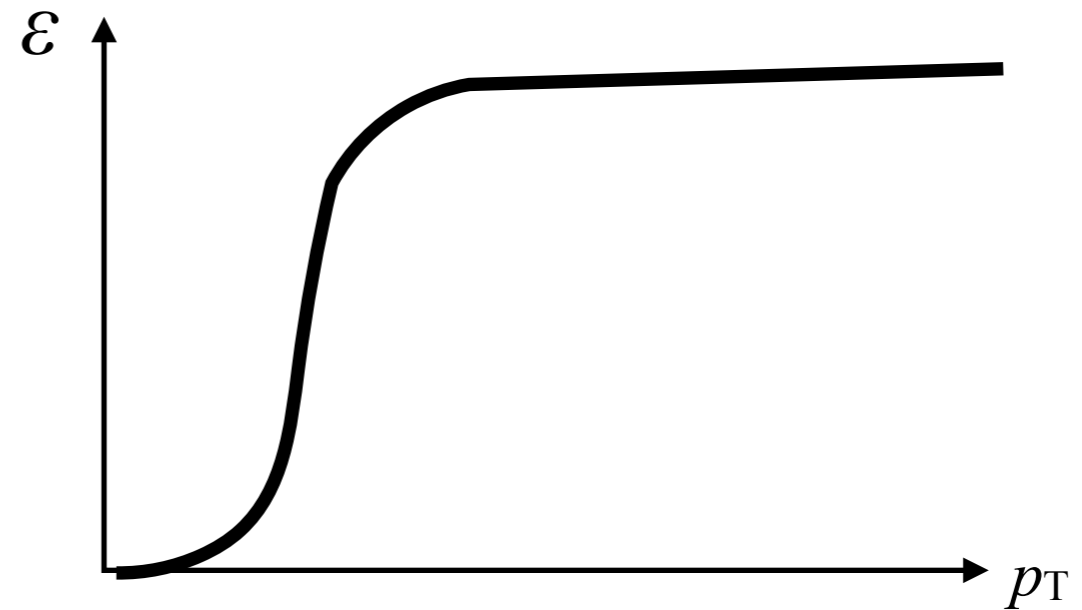
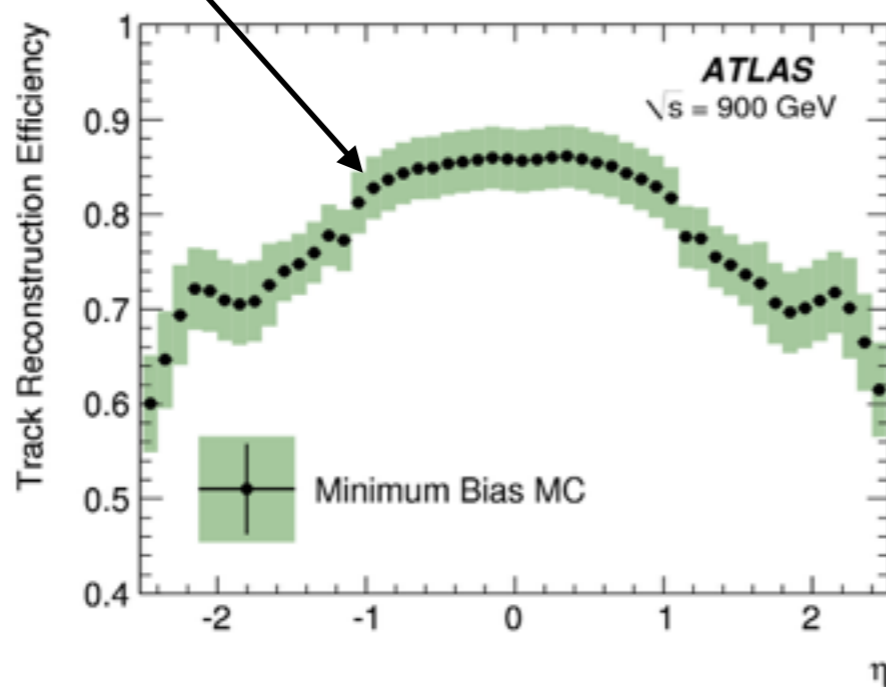
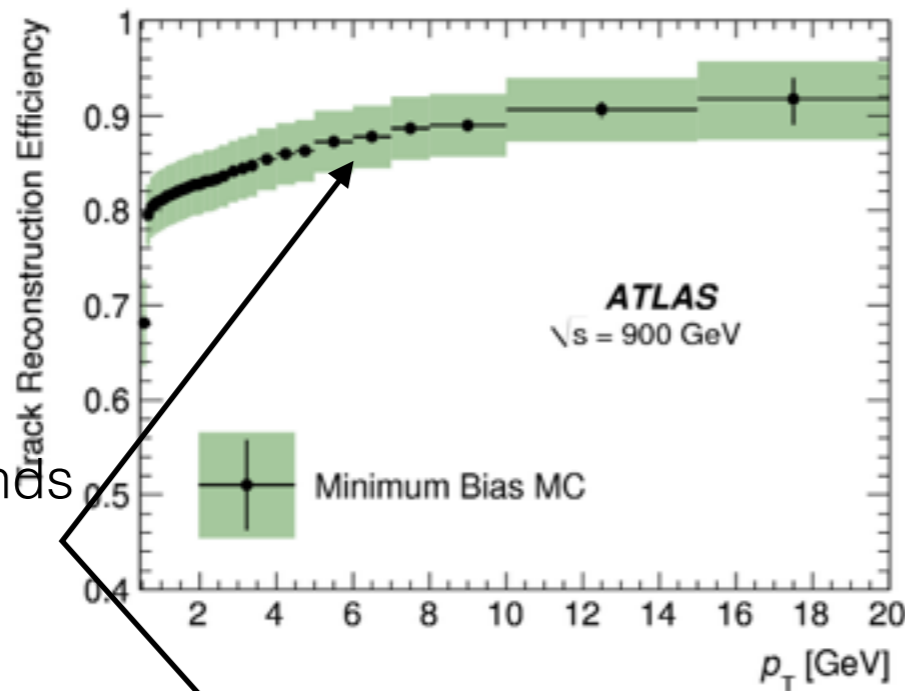
- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates



Tracking efficiency

- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates

where do
the error bands
come from ?

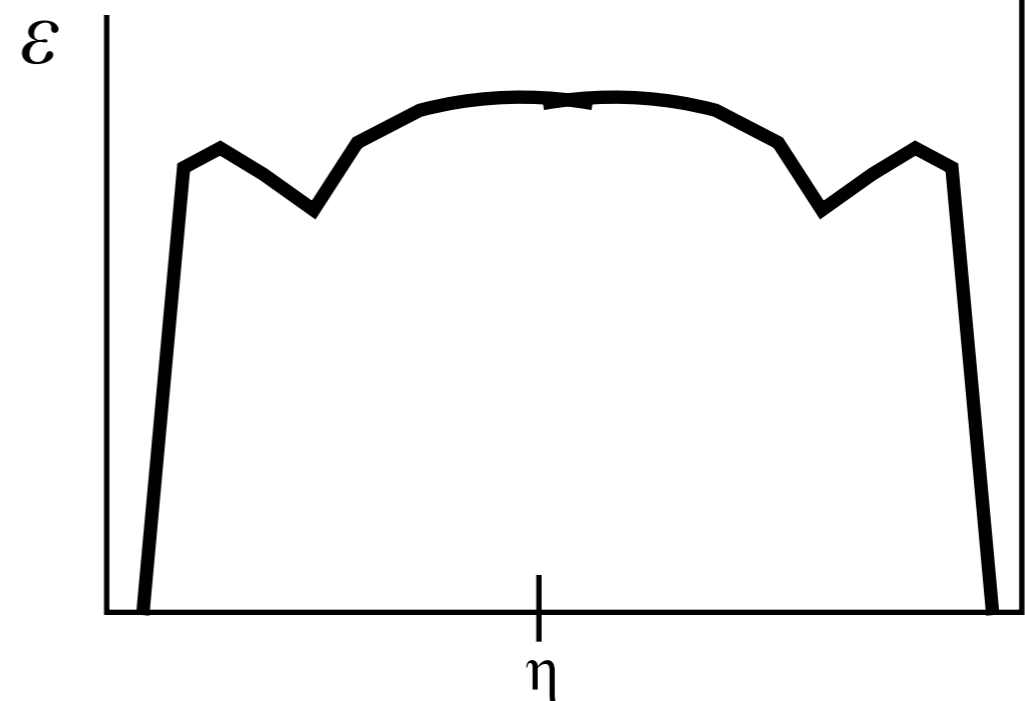
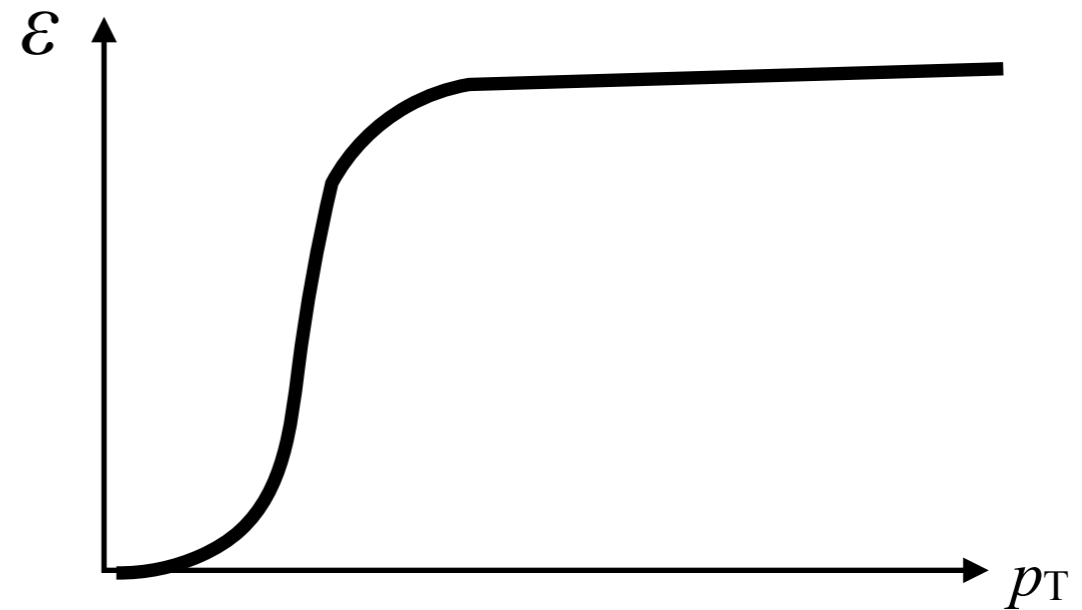
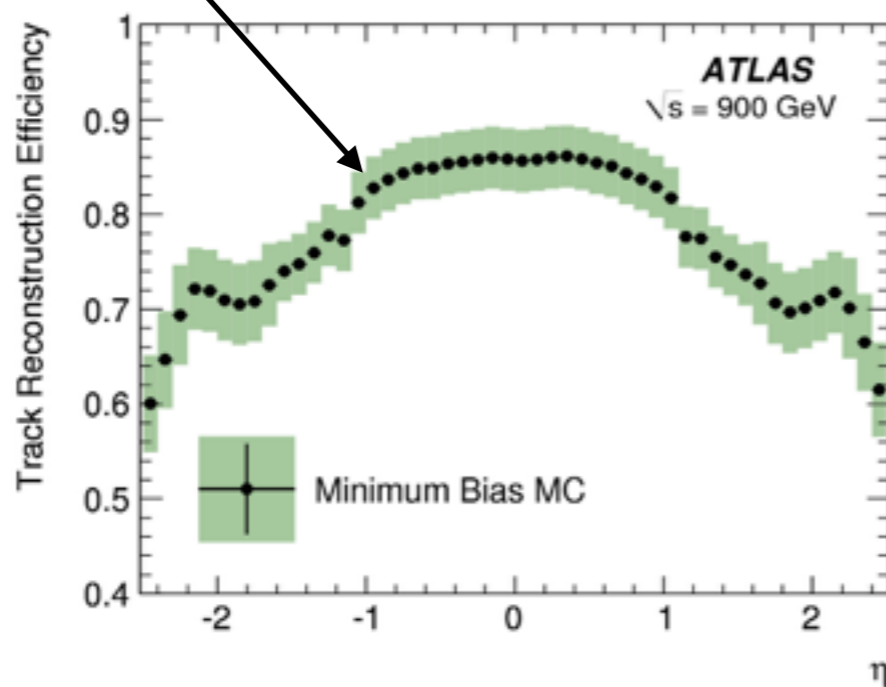
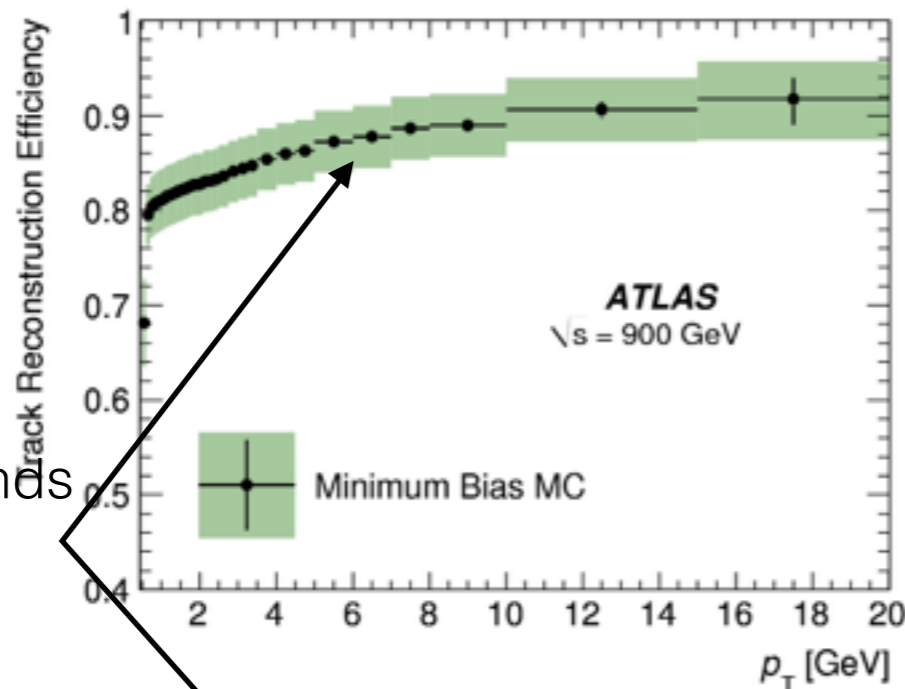


Tracking efficiency

- ▶ Material is the main source of inefficiency in the tracker
 - majority of particles are hadrons: nuclear interaction dominates

where do
the error bands
come from ?

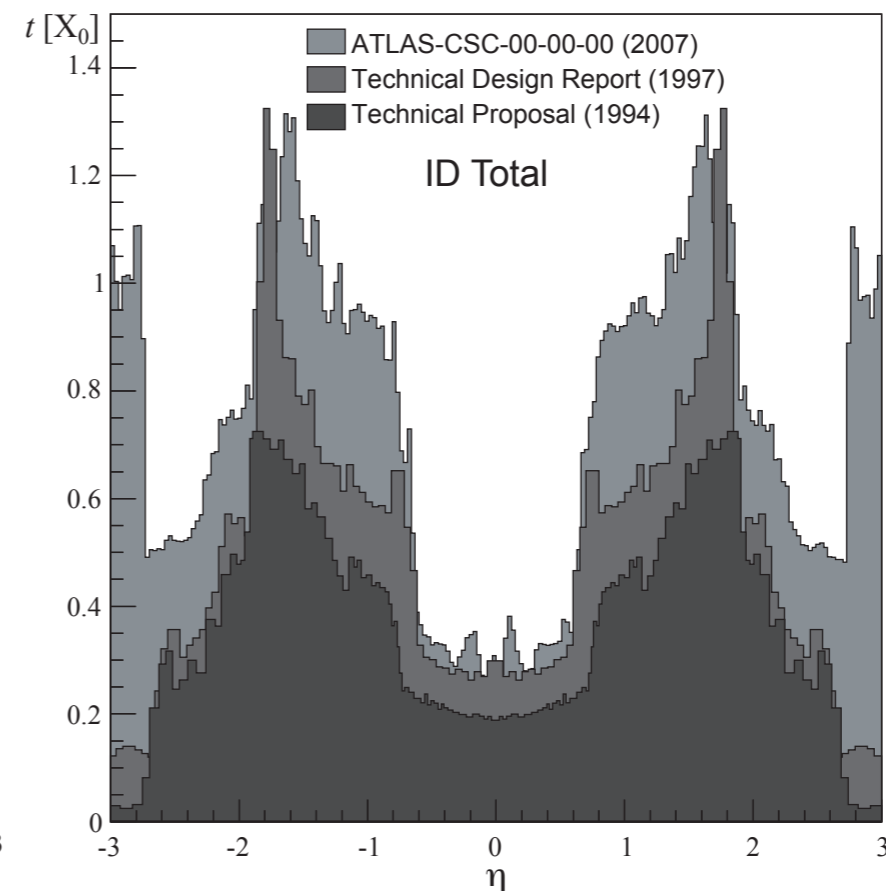
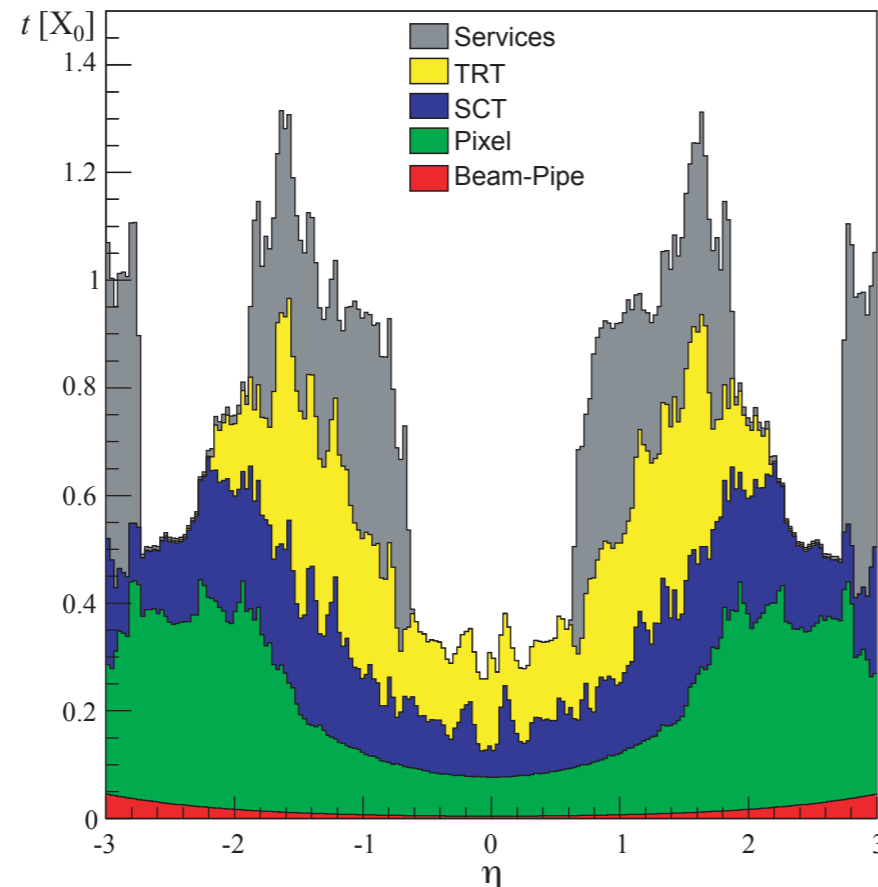
**uncertainty
of material
modelling !**



Measuring your detector material

- ▶ Hundreds of pages of TDRs and technical drawings before detector construction

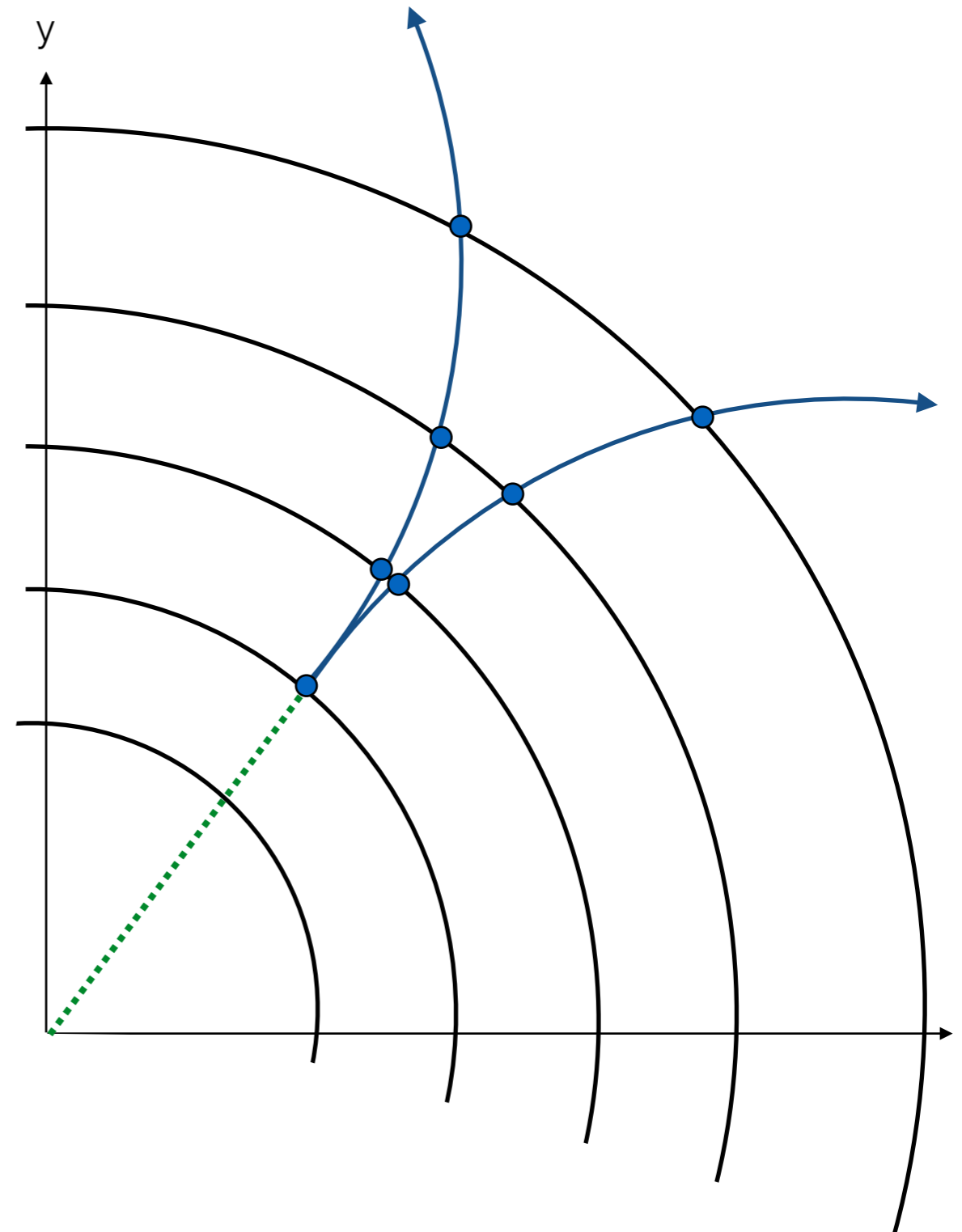
- still ...



- ▶ ATLAS has actually weighted the detector before installing it and compared the weight to the one in simulation
- ▶ Best: measure the material in situ !

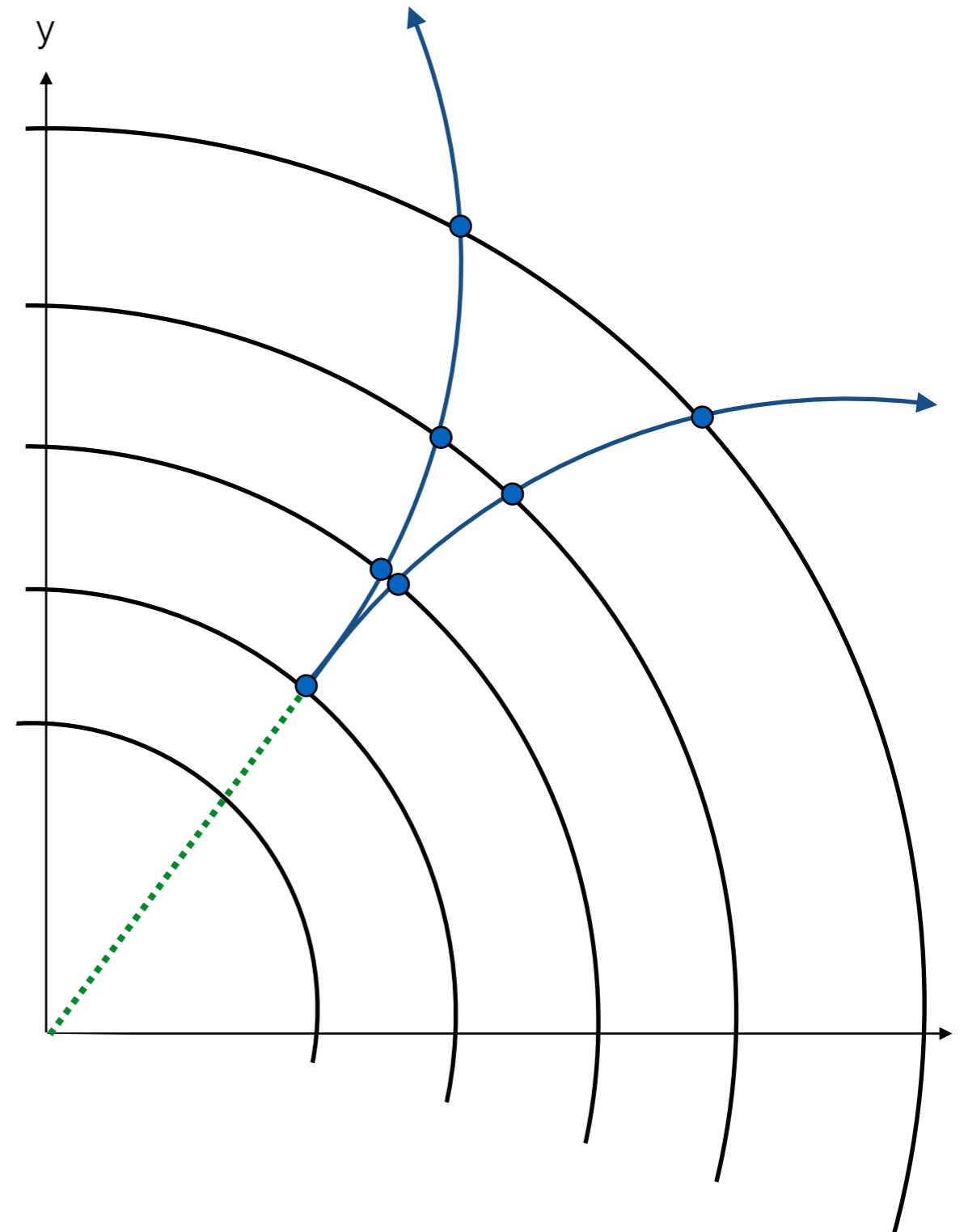
Using photon conversions

- ▶ Photons can convert into e^+e^- when hitting detector material
- ▶ 2-track vertices with electron PID
 - very clear signal
- ▶ Count conversion vertices in data and MC and compare
 - Is this really measuring the material ?
 - Are we just seeing an artefact of non-ideal simulation (cross-section ...)?



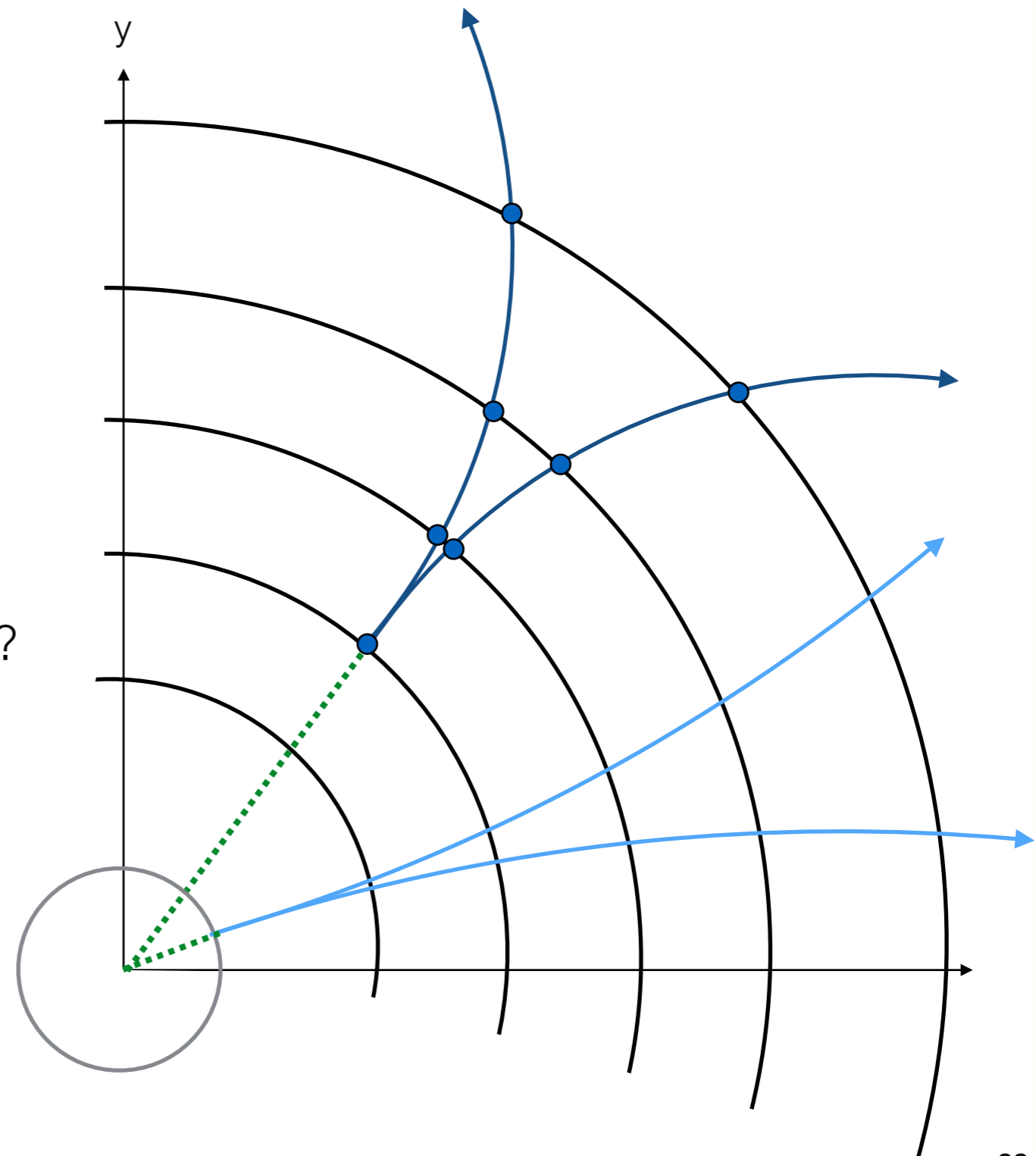
Using photon conversions

- ▶ Photons can convert into e^+e^- when hitting detector material
- ▶ 2-track vertices with electron PID
 - very clear signal
- ▶ Count conversion vertices in data and MC and compare
 - Is this really measuring the material ?
 - Are we just seeing an artefact of non-ideal simulation (cross-section ...)?
- ▶ Try to find a standard candle, i.e. a piece of material you know very well in reality and simulation



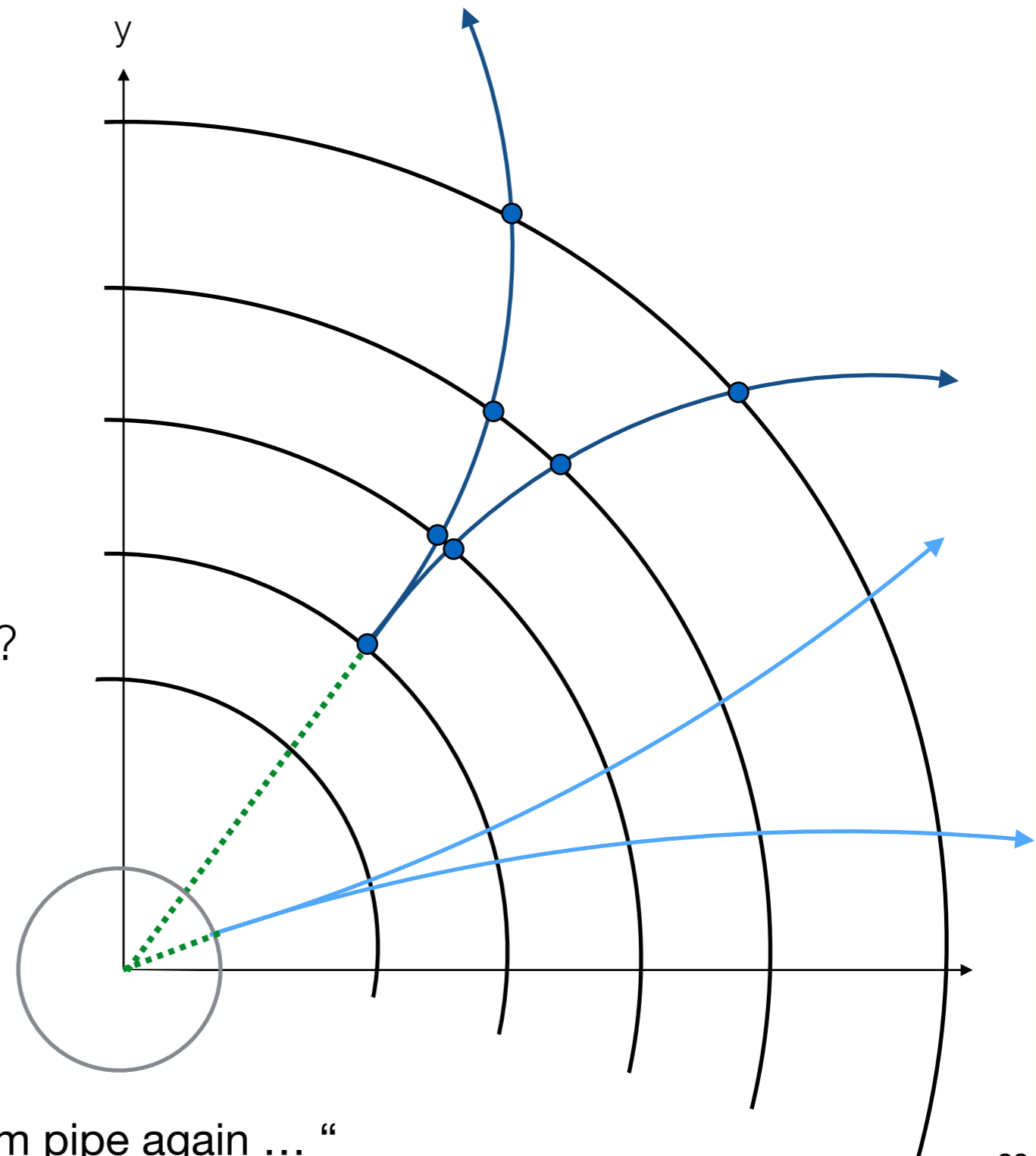
Using photon conversions

- ▶ Photons can convert into e^+e^- when hitting detector material
- ▶ 2-track vertices with electron PID
 - very clear signal
- ▶ Count conversion vertices in data and MC and compare
 - Is this really measuring the material ?
 - Are we just seeing an artefact of non-ideal simulation (cross-section ...) ?
- ▶ Try to find a standard candle, i.e. a piece of material you know very well in reality and simulation



Using photon conversions

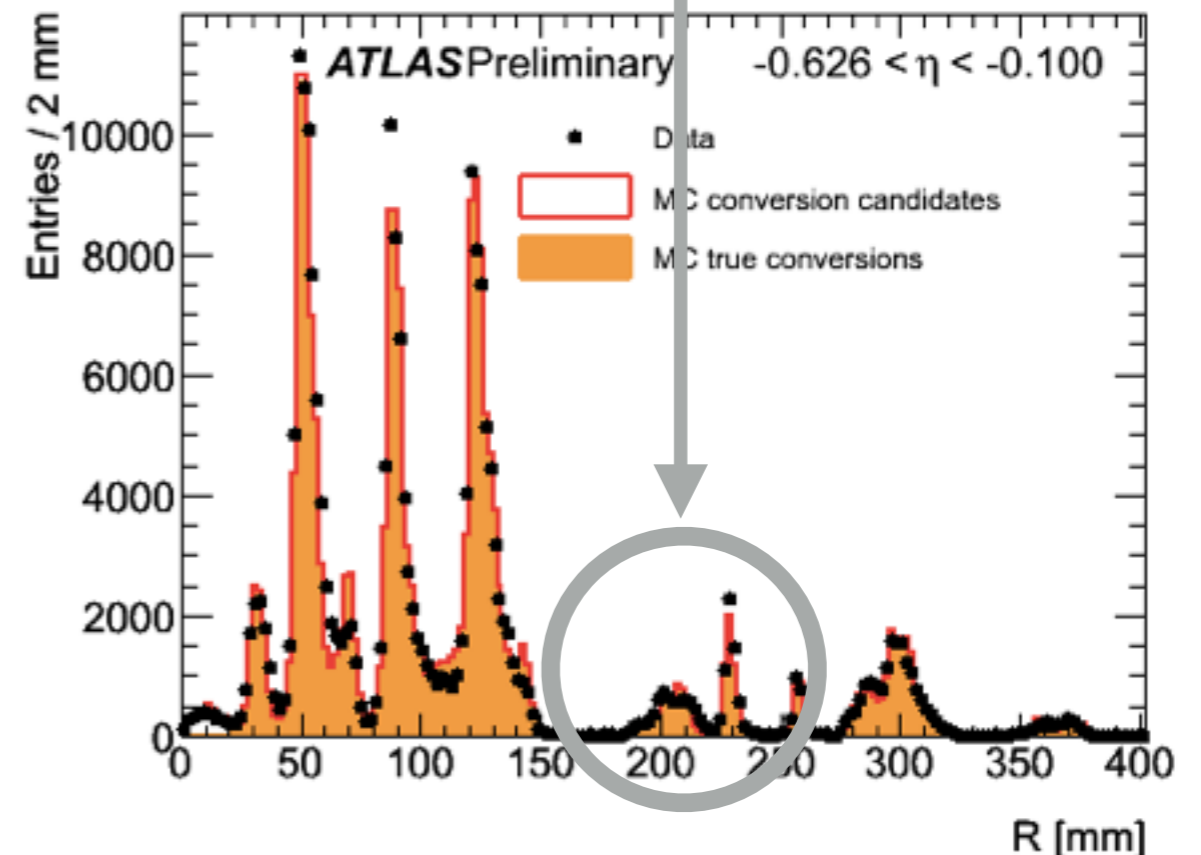
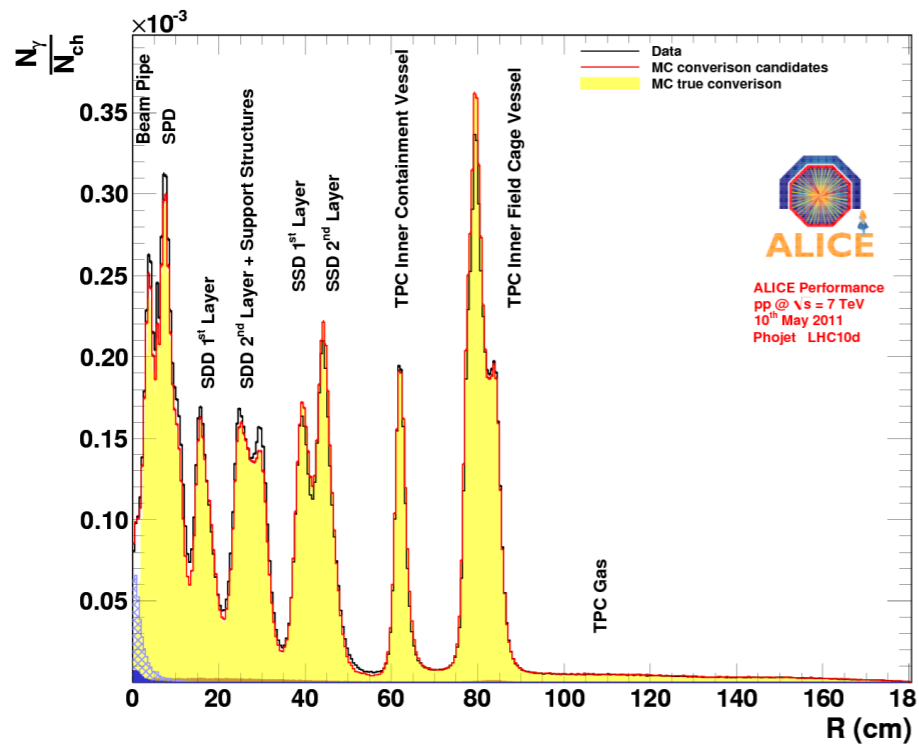
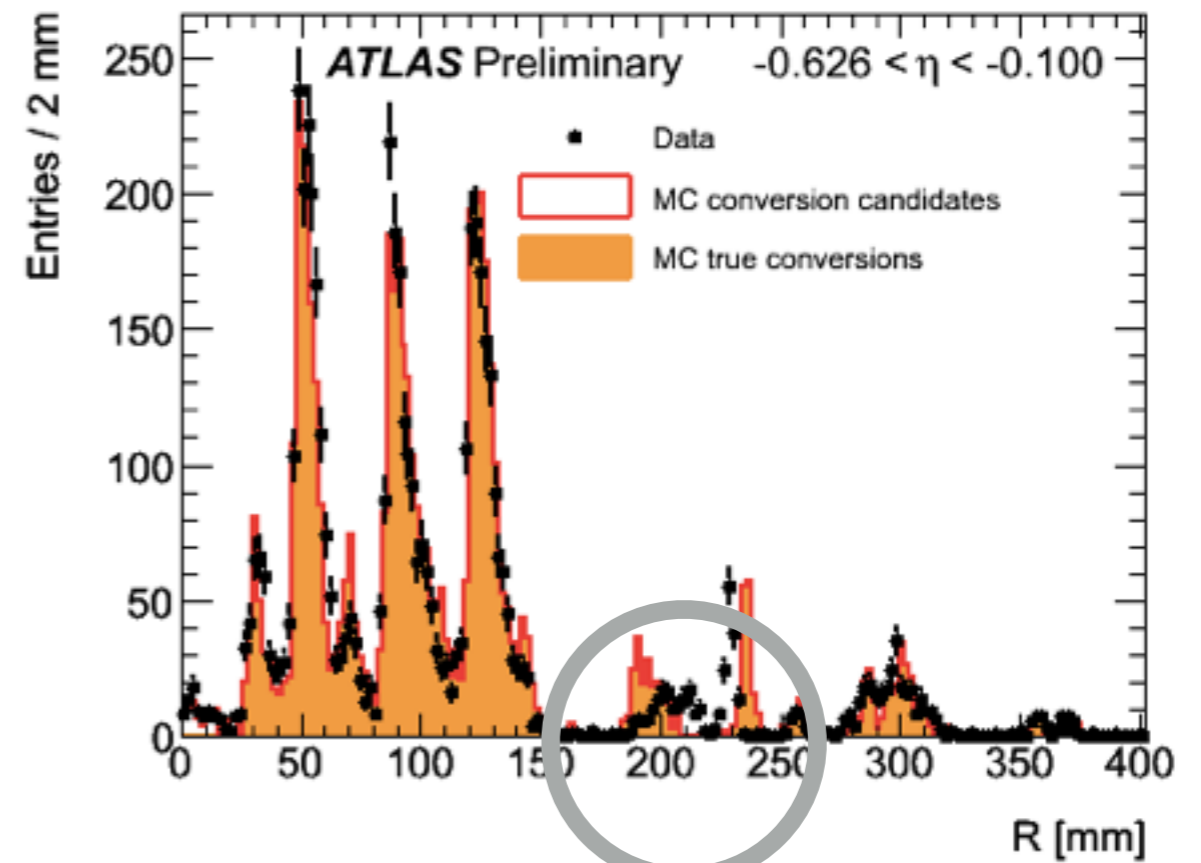
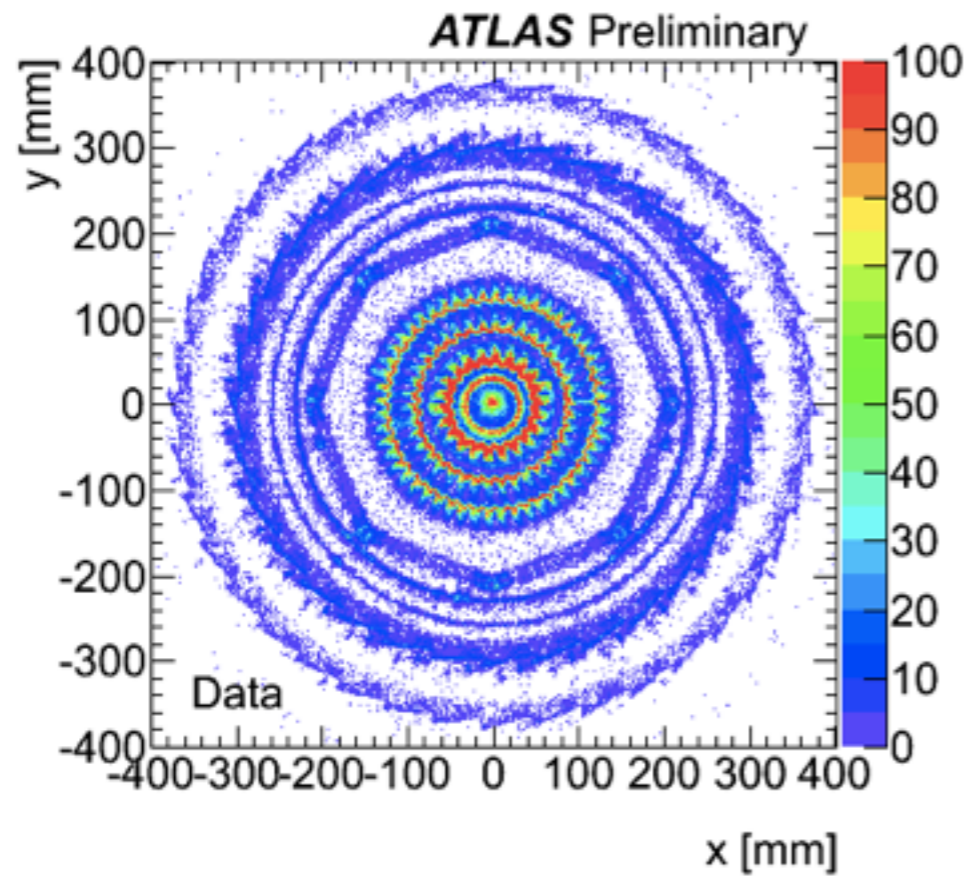
- ▶ Photons can convert into e^+e^- when hitting detector material
- ▶ 2-track vertices with electron PID
 - very clear signal
- ▶ Count conversion vertices in data and MC and compare
 - Is this really measuring the material ?
 - Are we just seeing an artefact of non-ideal simulation (cross-section ...) ?
- ▶ Try to find a standard candle, i.e. a piece of material you know very well in reality and simulation



remember ?

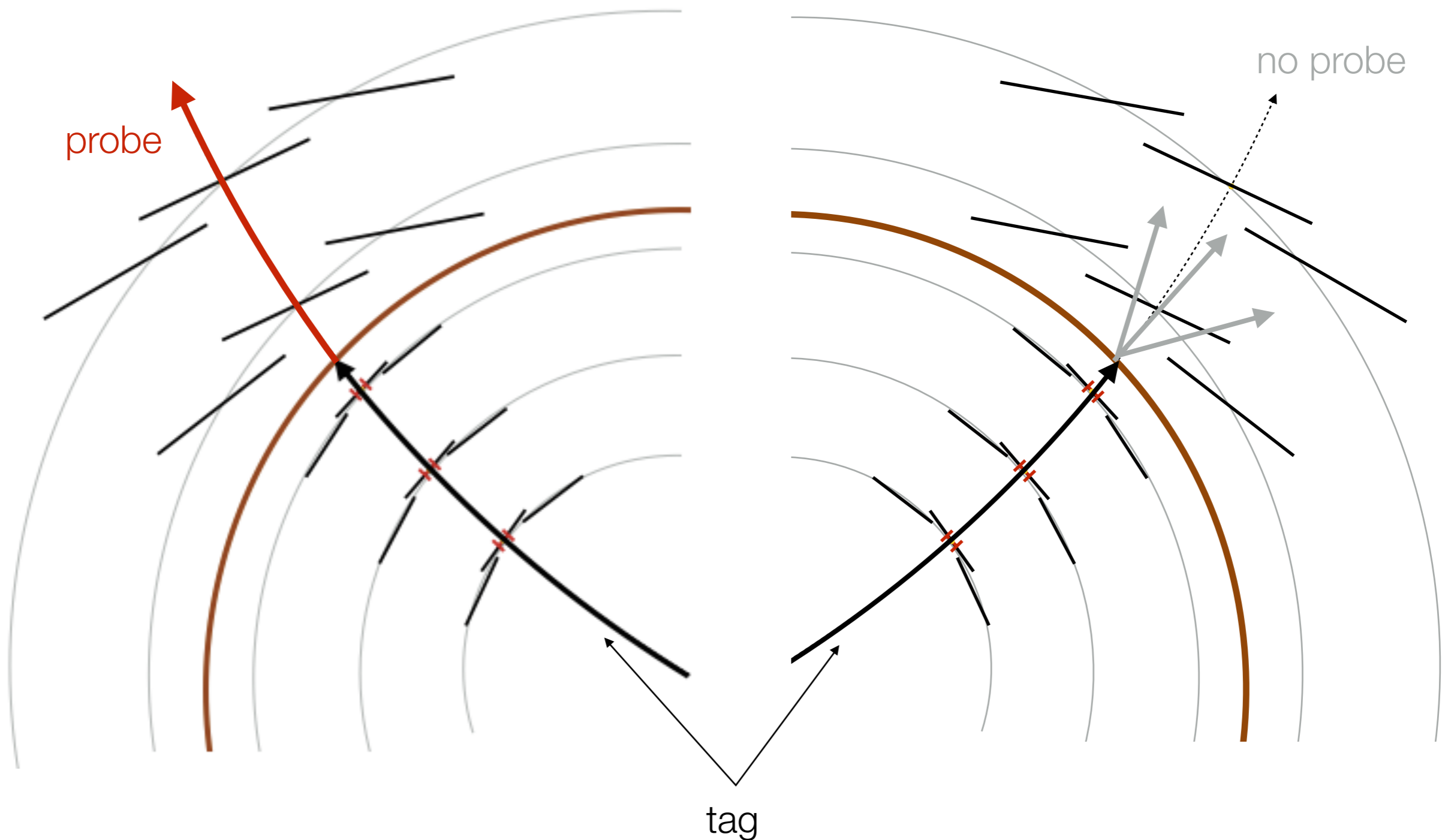
“... ATLAS is trying to measure the old beam pipe again ... “

Using photon conversions



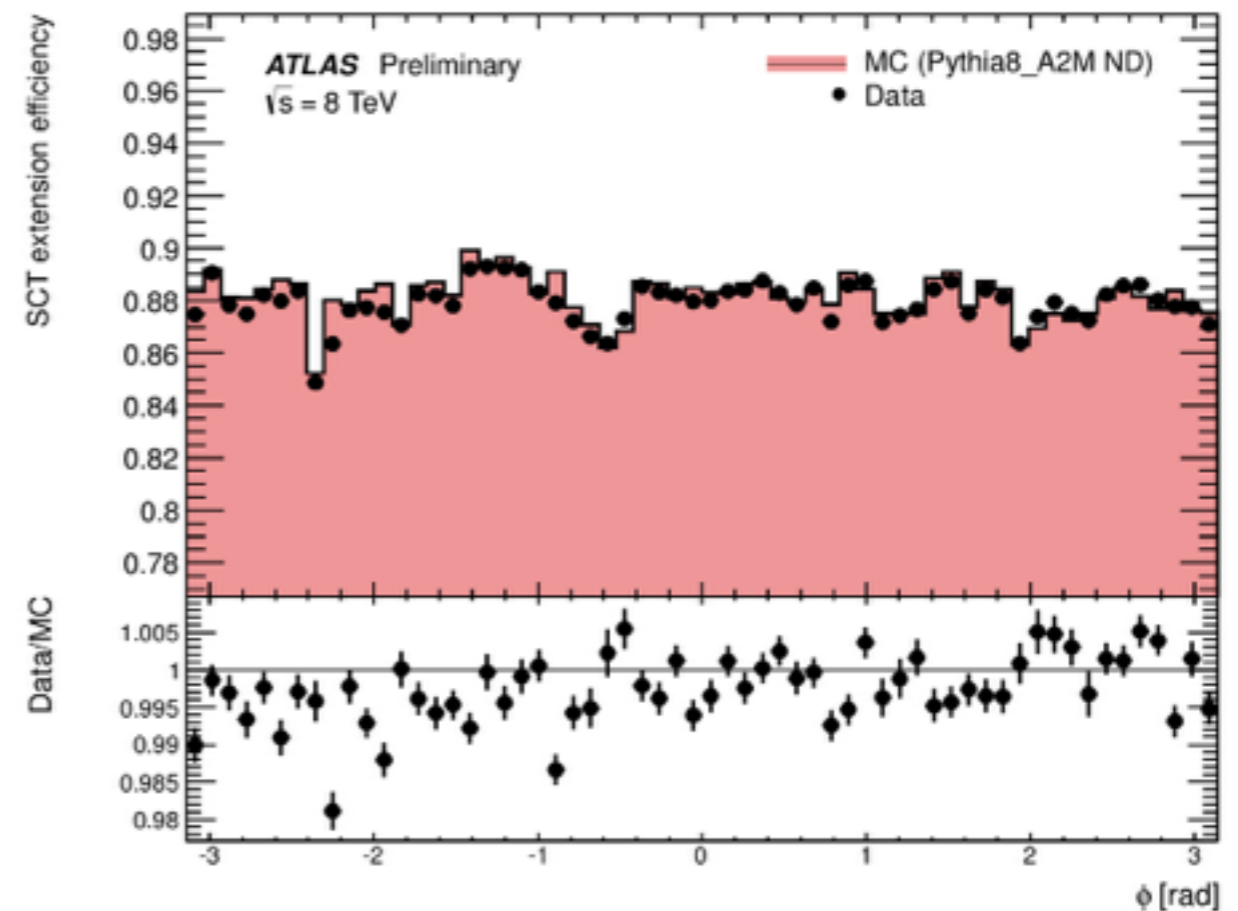
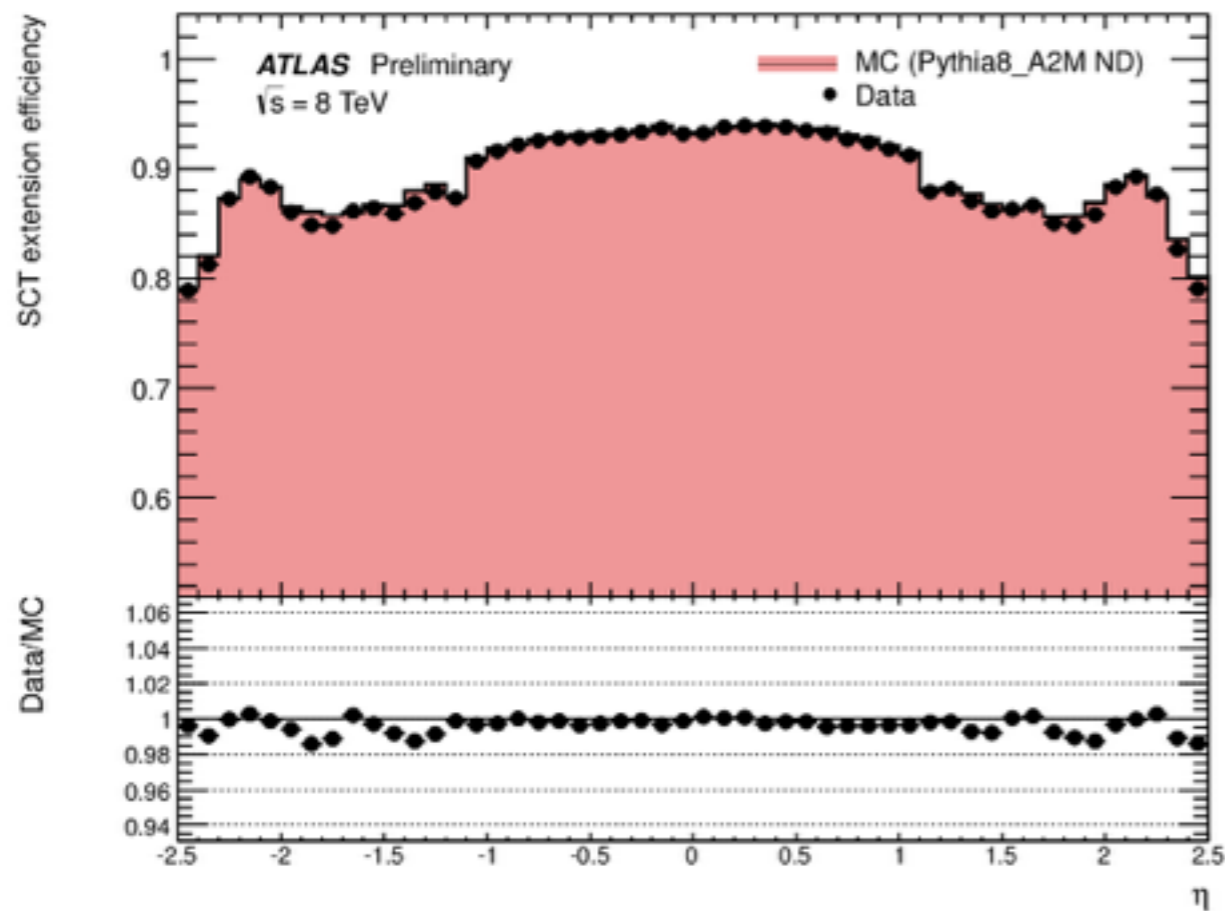
Using hadronic Interactions

- ▶ The straight-forward way
 - measure how often tracks “survive” when passing material



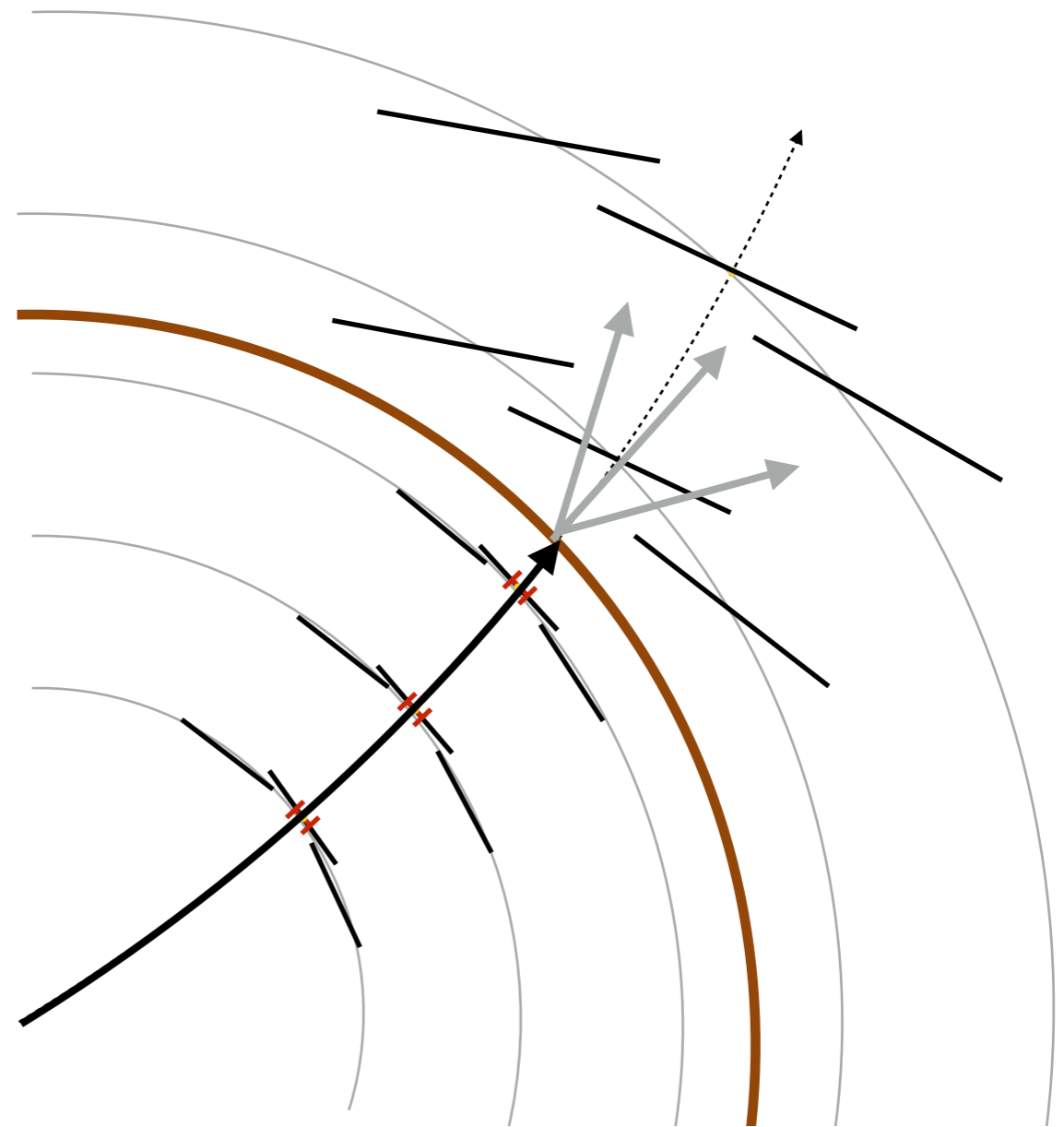
Using hadronic Interactions

- ▶ The straight-forward way
 - measure how often tracks “survive” when passing material



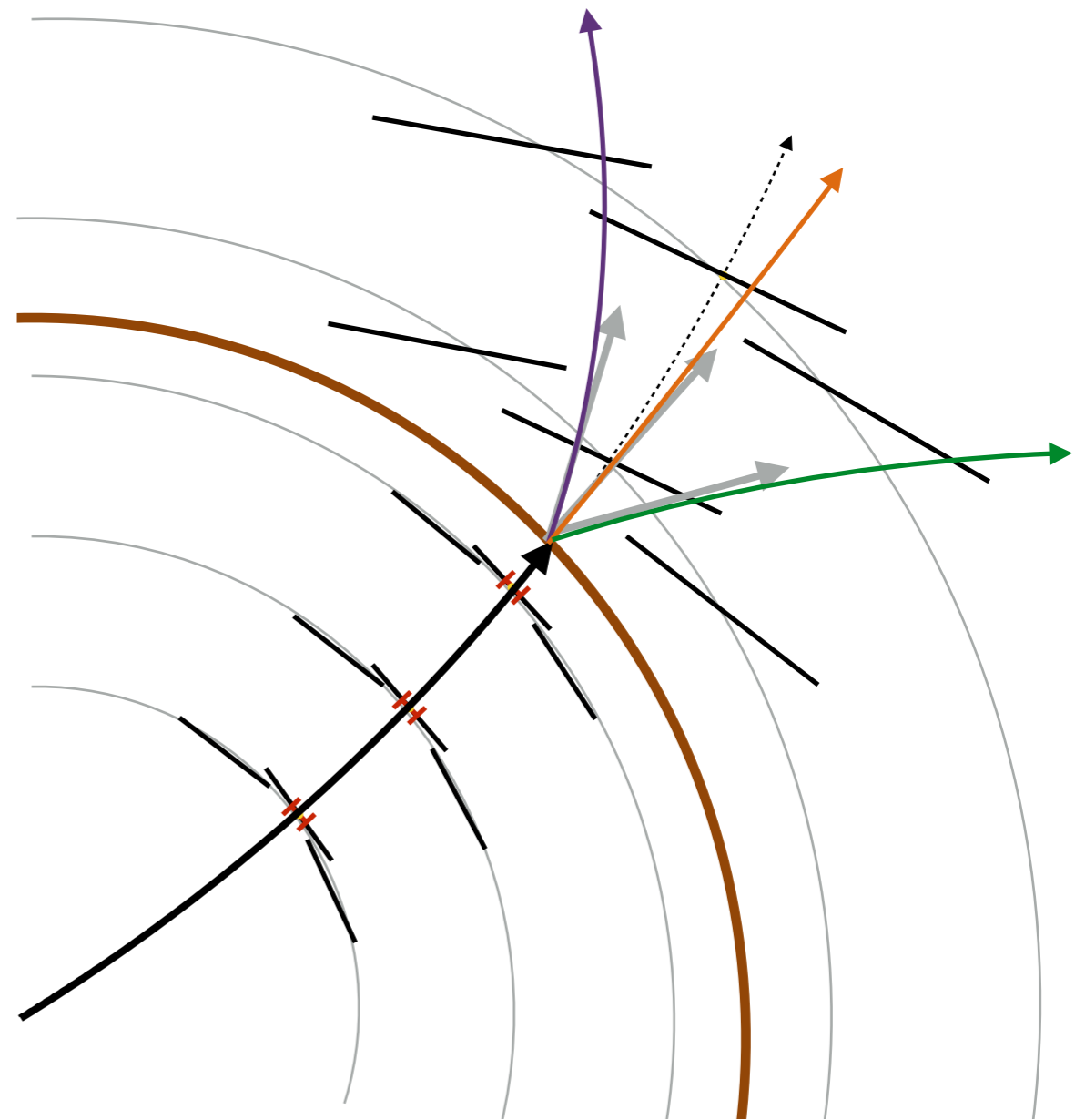
Using hadronic Interactions

- ▶ The more sophisticated way
 - assume we can configure our track reconstruction to also reconstruct the charged particles coming out of a hadronic interaction
 - let's try to use our vertex reconstruction skills and try to reconstruct the vertices from hadronic interactions



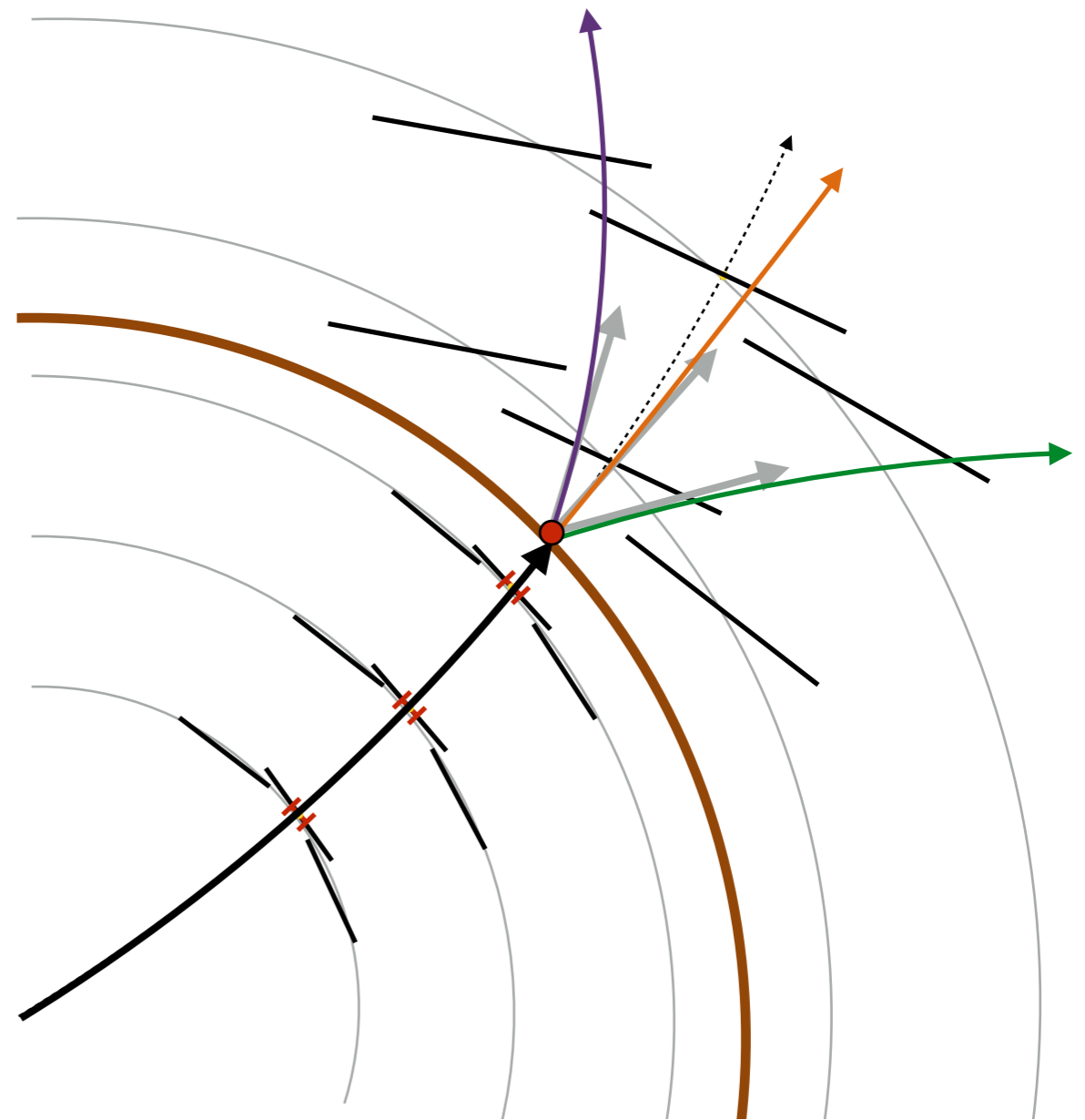
Using hadronic Interactions

- ▶ The more sophisticated way
 - assume we can configure our track reconstruction to also reconstruct the charged particles coming out of a hadronic interaction
 - let's try to use our vertex reconstruction skills and try to reconstruct the vertices from hadronic interactions



Using hadronic Interactions

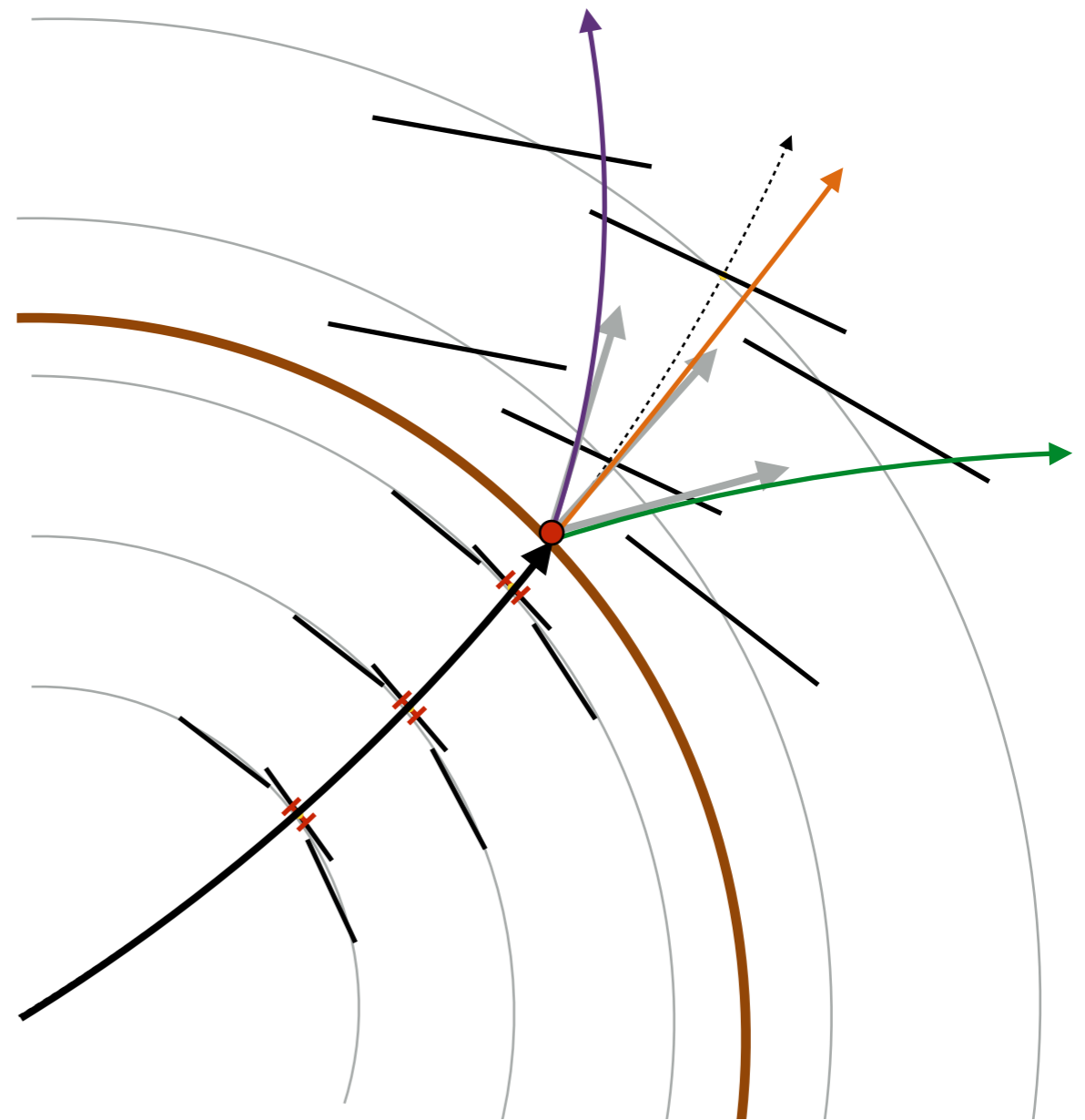
- ▶ The more sophisticated way
 - assume we can configure our track reconstruction to also reconstruct the charged particles coming out of a hadronic interaction
 - let's try to use our vertex reconstruction skills and try to reconstruct the vertices from hadronic interactions



Using hadronic Interactions

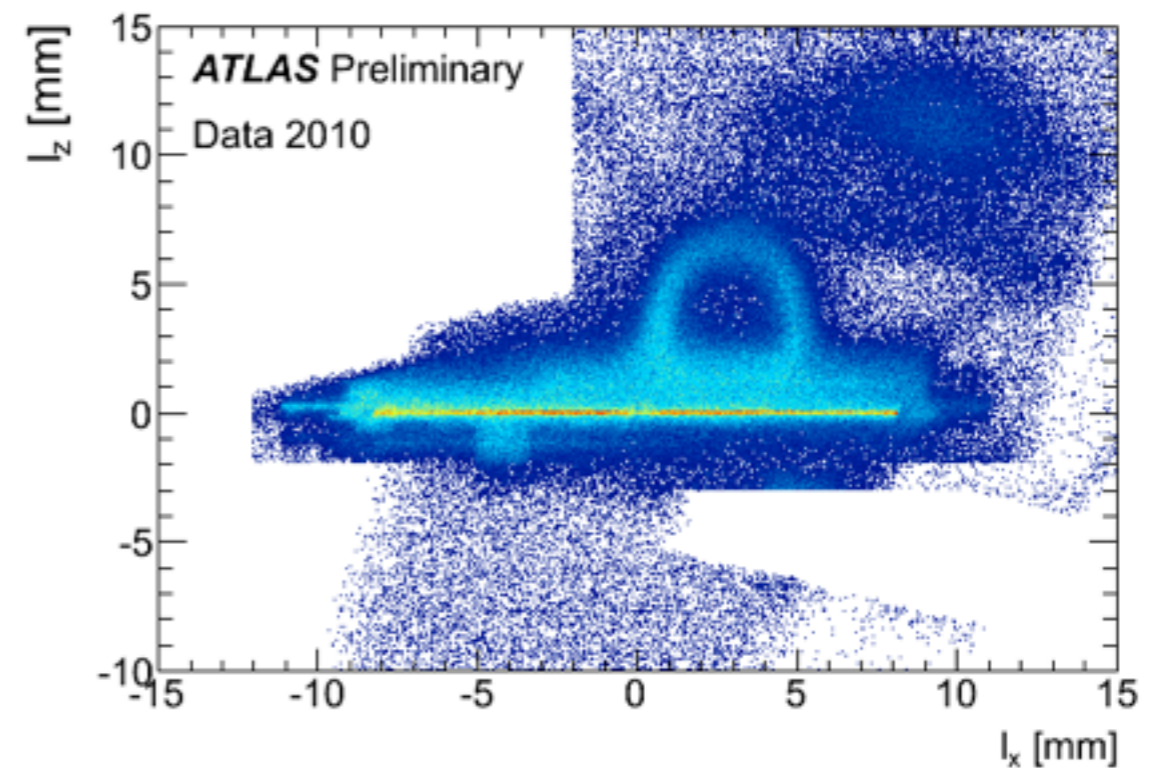
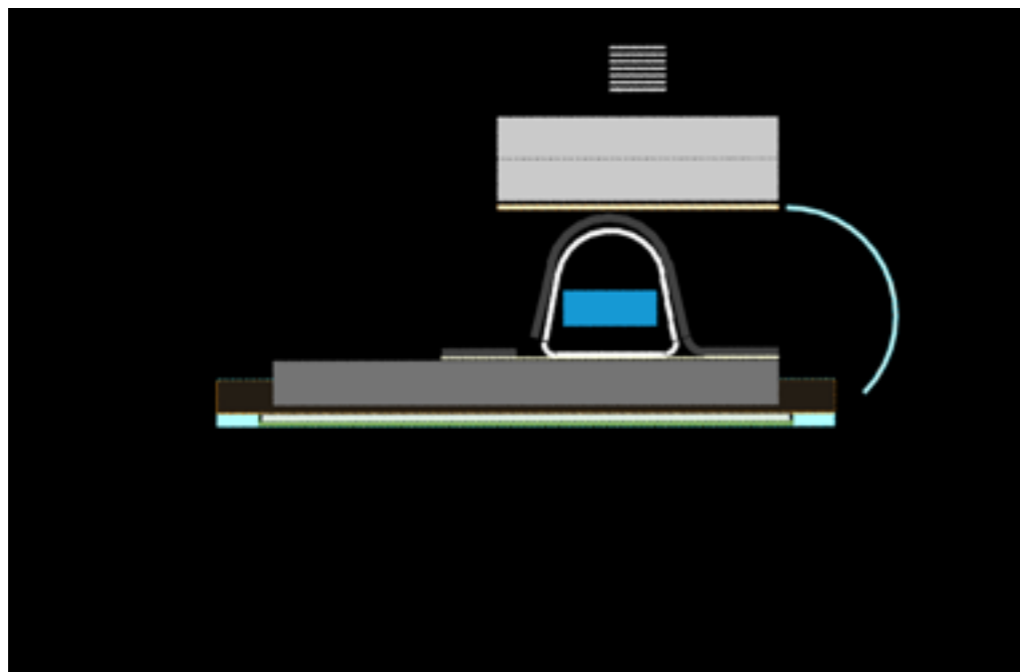
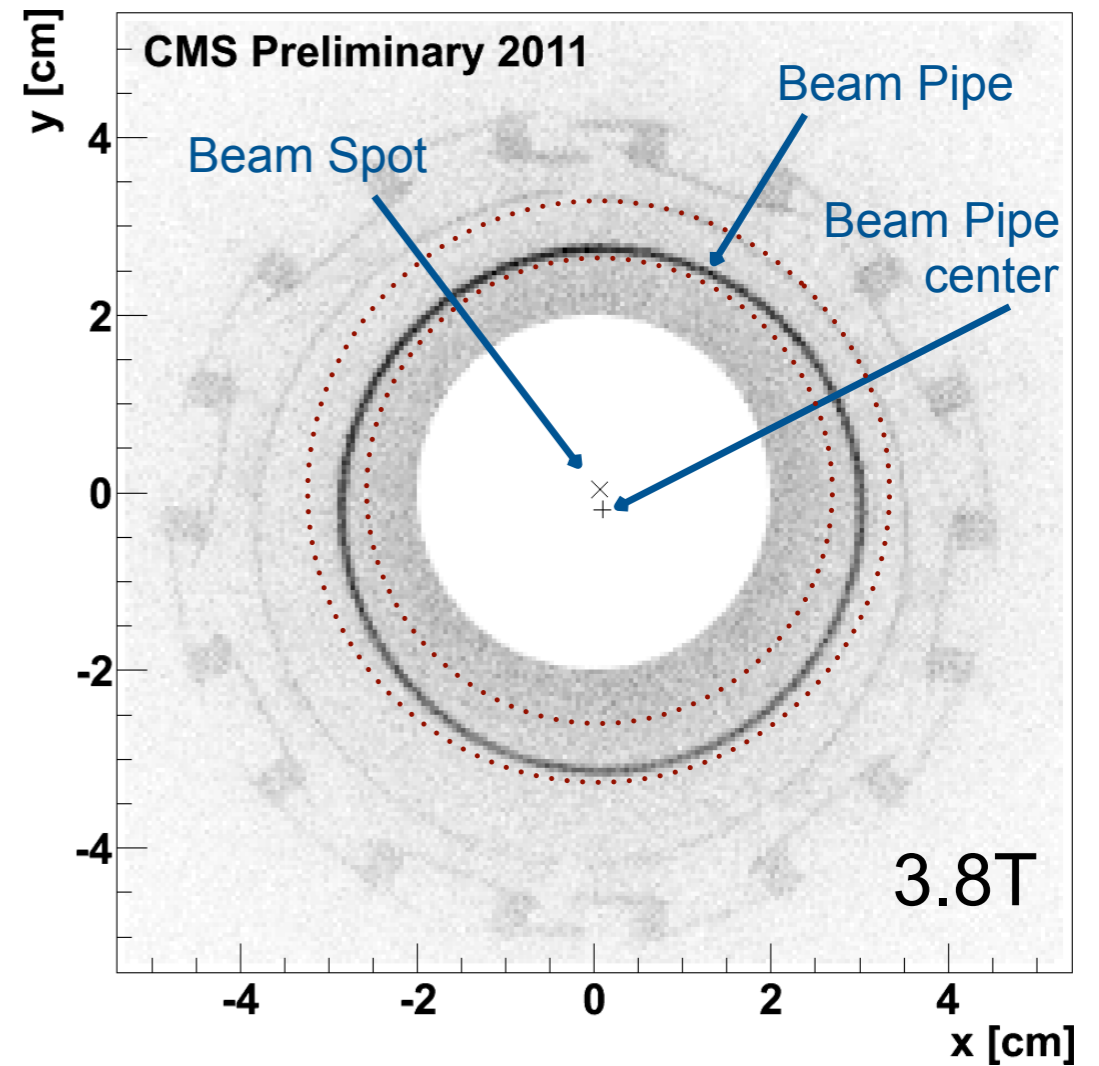
- ▶ The more sophisticated way
 - assume we can configure our track reconstruction to also reconstruct the charged particles coming out of a hadronic interaction
 - let's try to use our vertex reconstruction skills and try to reconstruct the vertices from hadronic interactions

What do you need to be very careful about ?



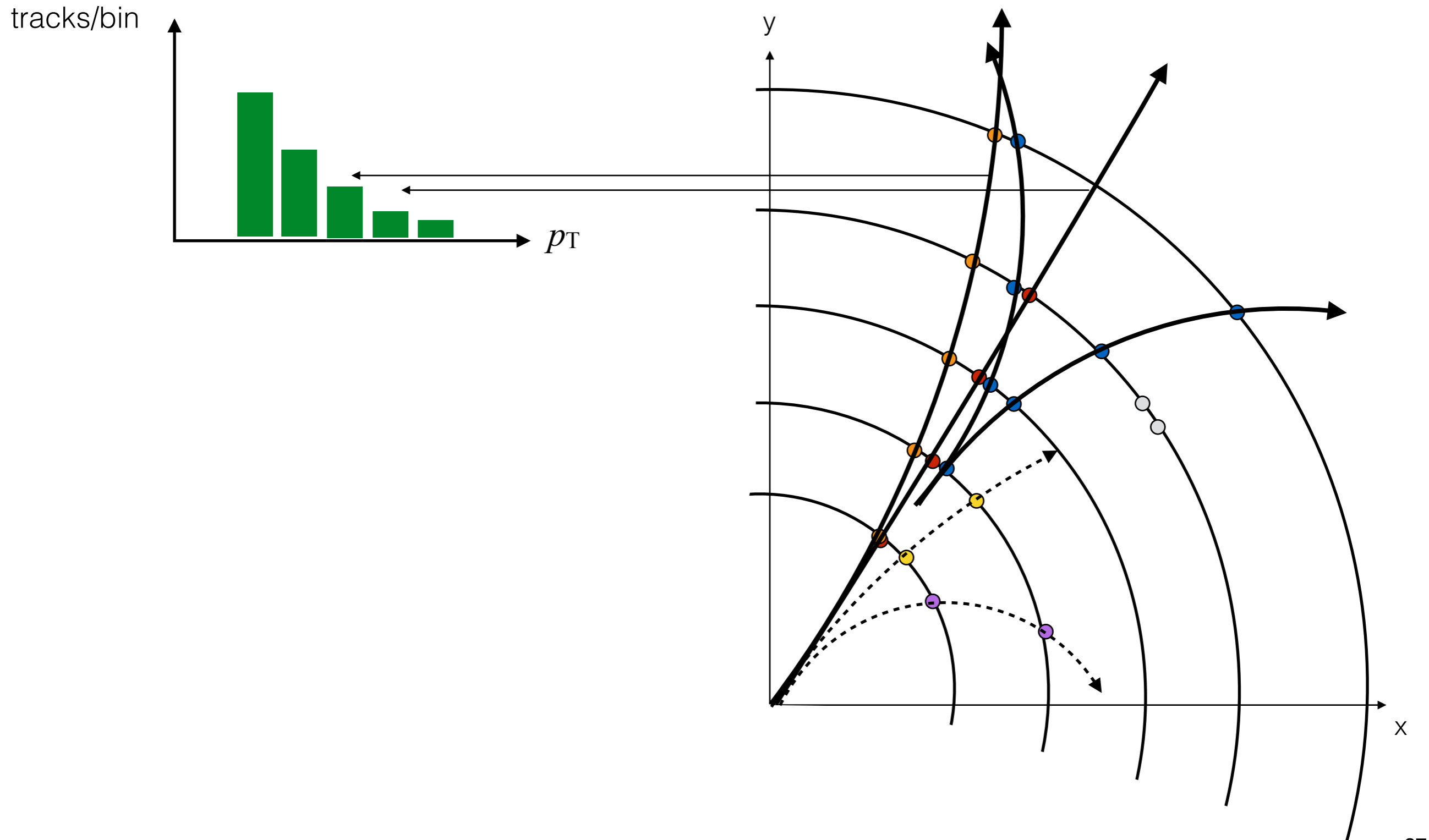
Using hadronic Interactions

- ▶ Both CMS & ATLAS have achieved astonishing results with this technique
 - offset of nominal beam pipe seen for both experiments
 - accuracy of material measurement for ATLAS ~8 %



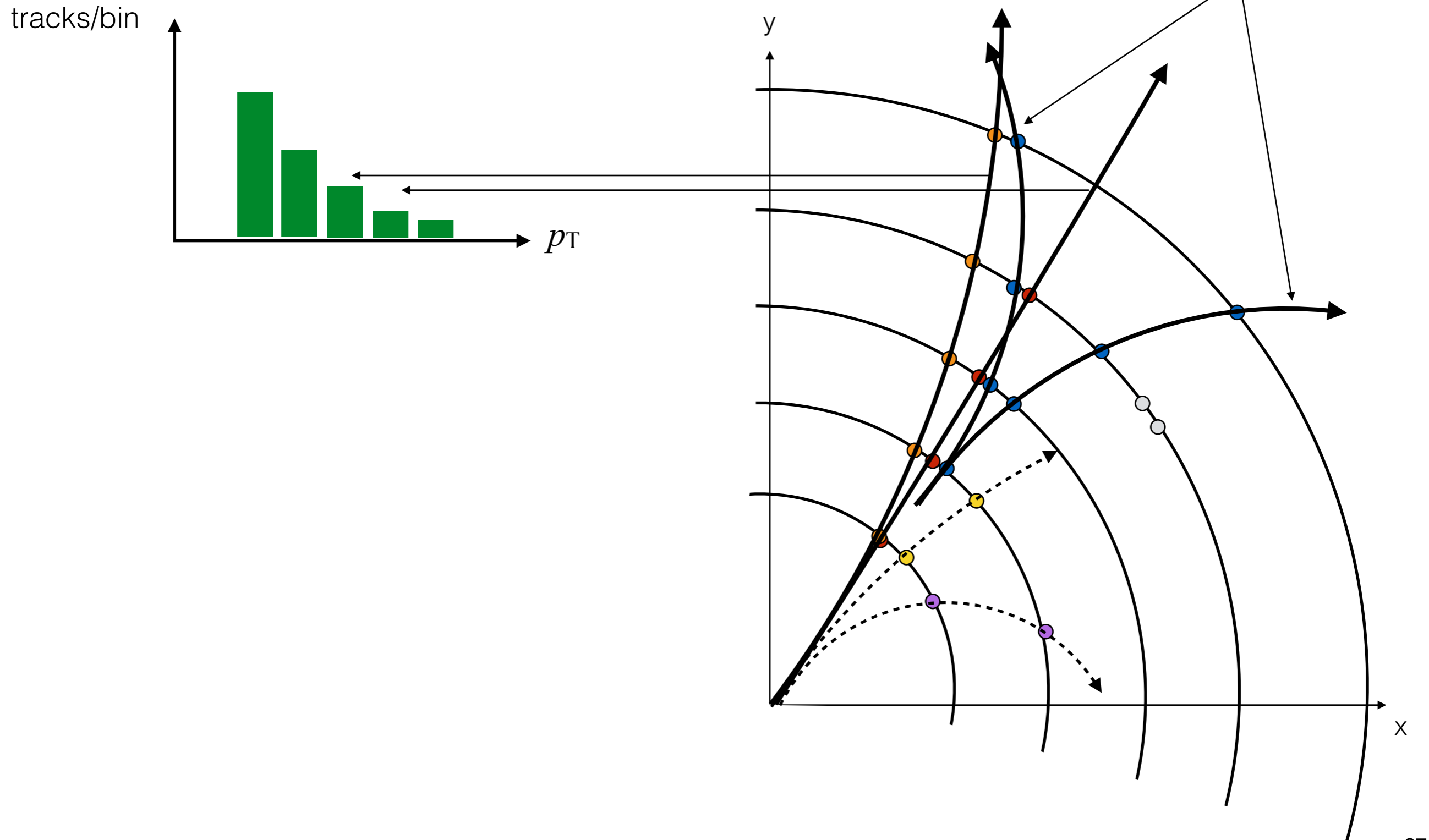
Example: charged particle multiplicity

- ▶ Let's finish the analysis



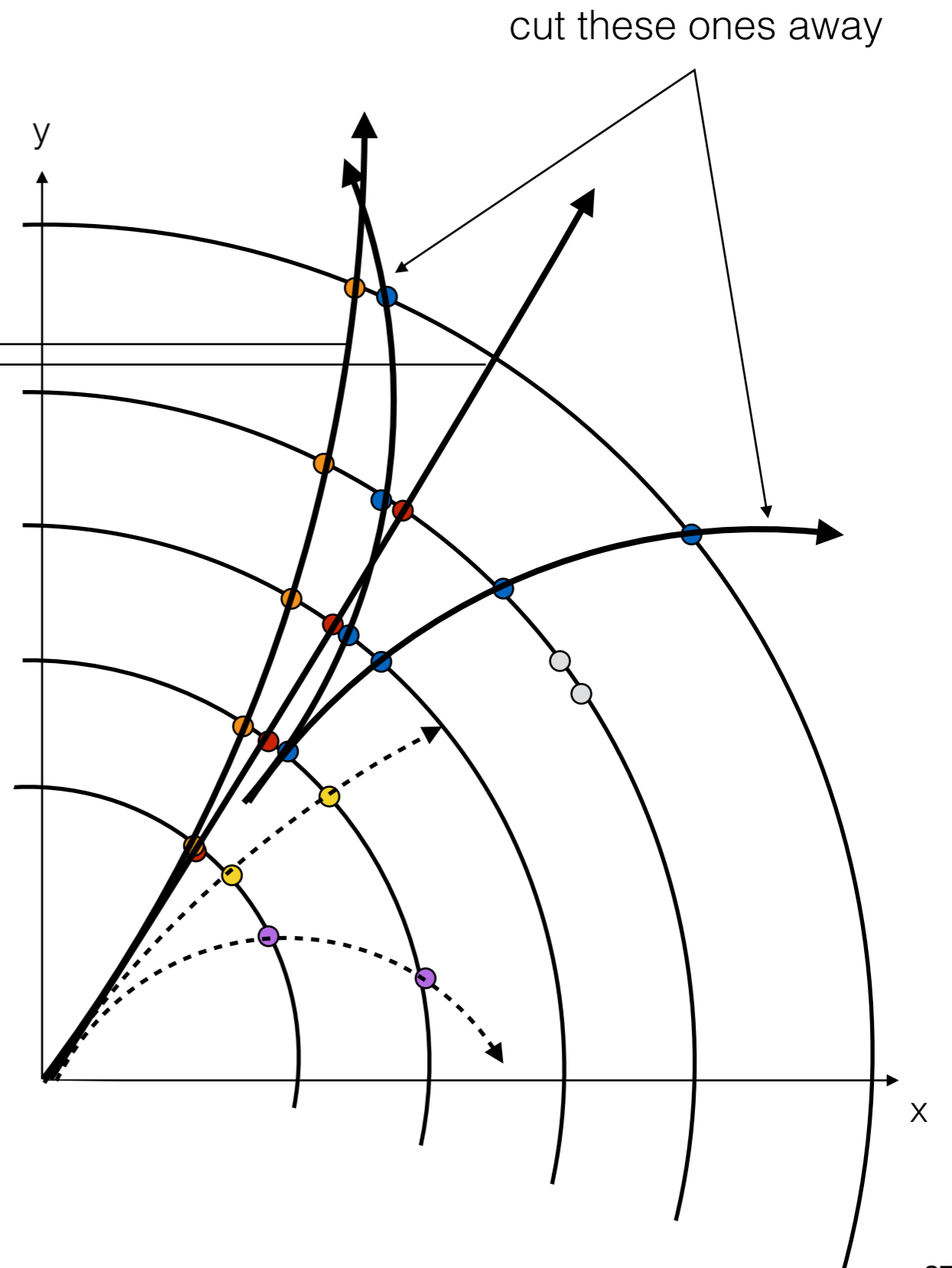
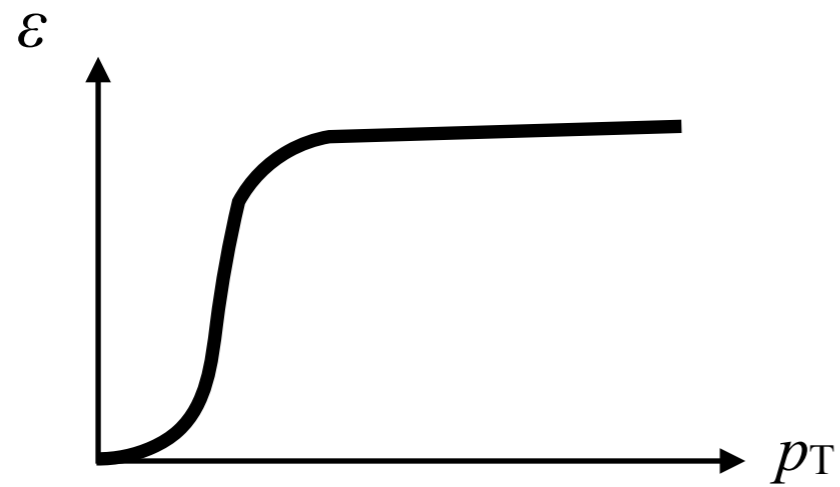
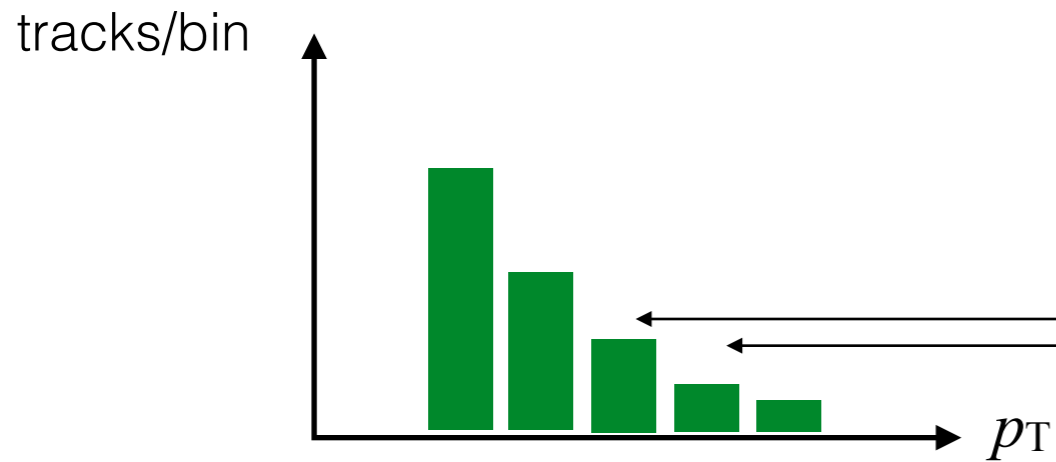
Example: charged particle multiplicity

- ▶ Let's finish the analysis



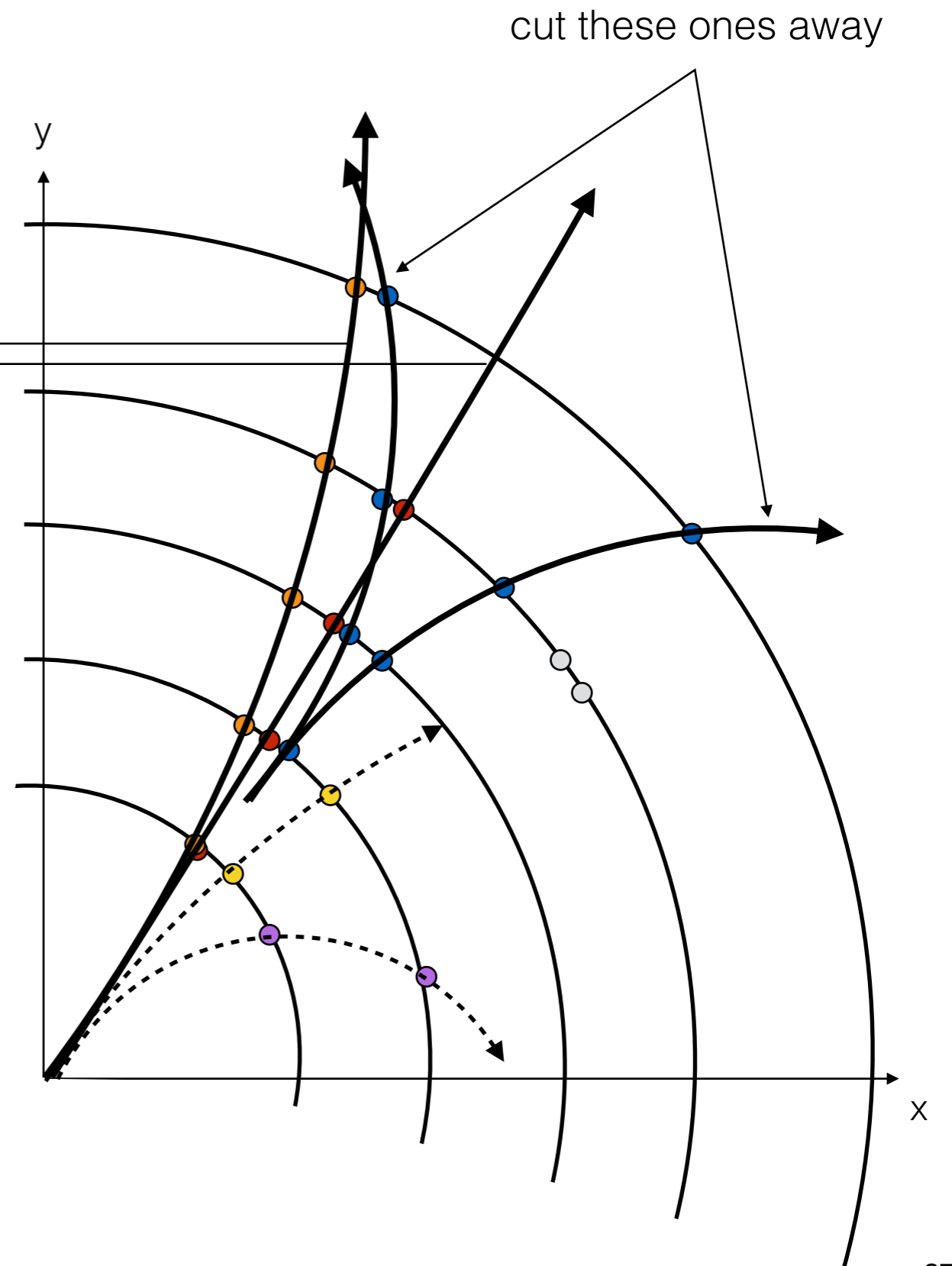
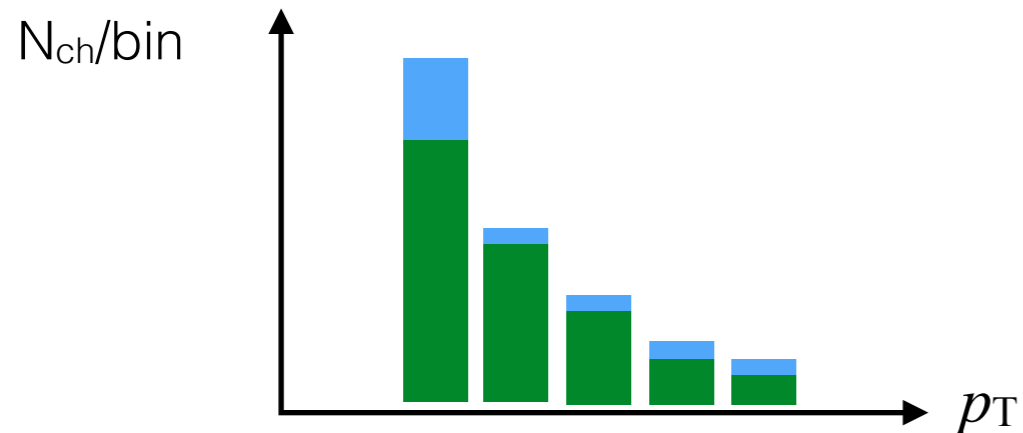
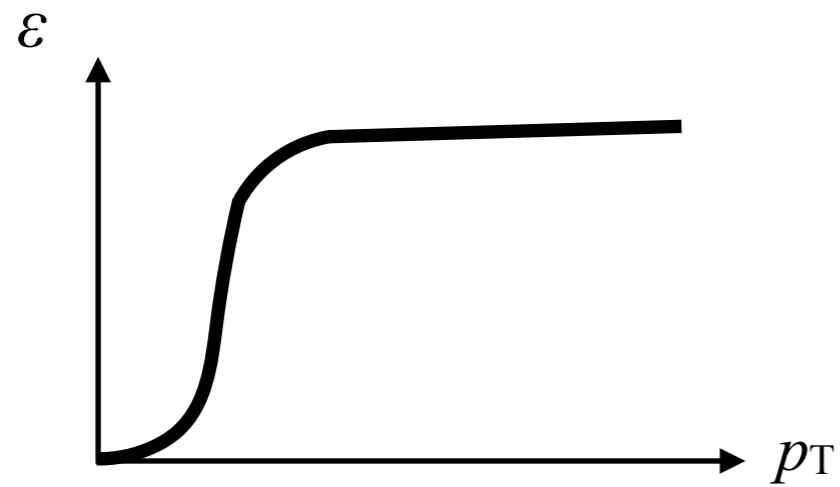
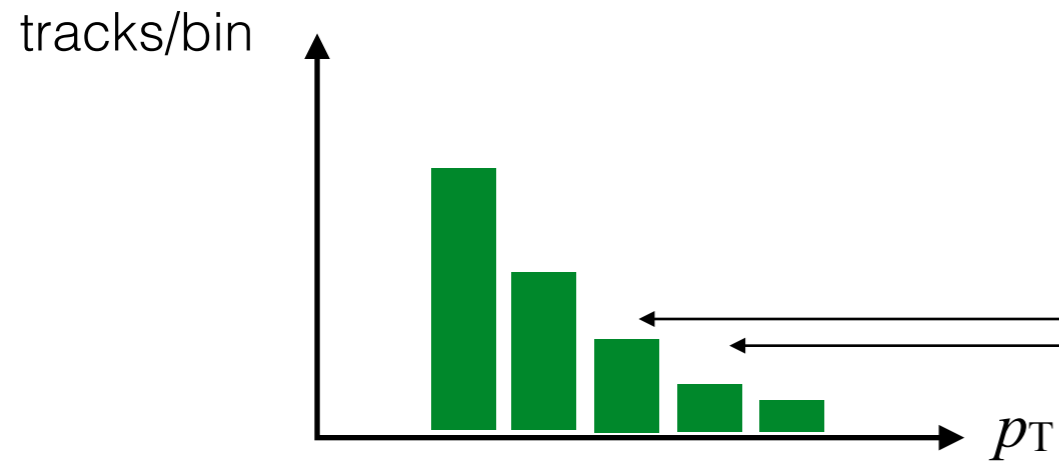
Example: charged particle multiplicity

- ▶ Let's finish the analysis



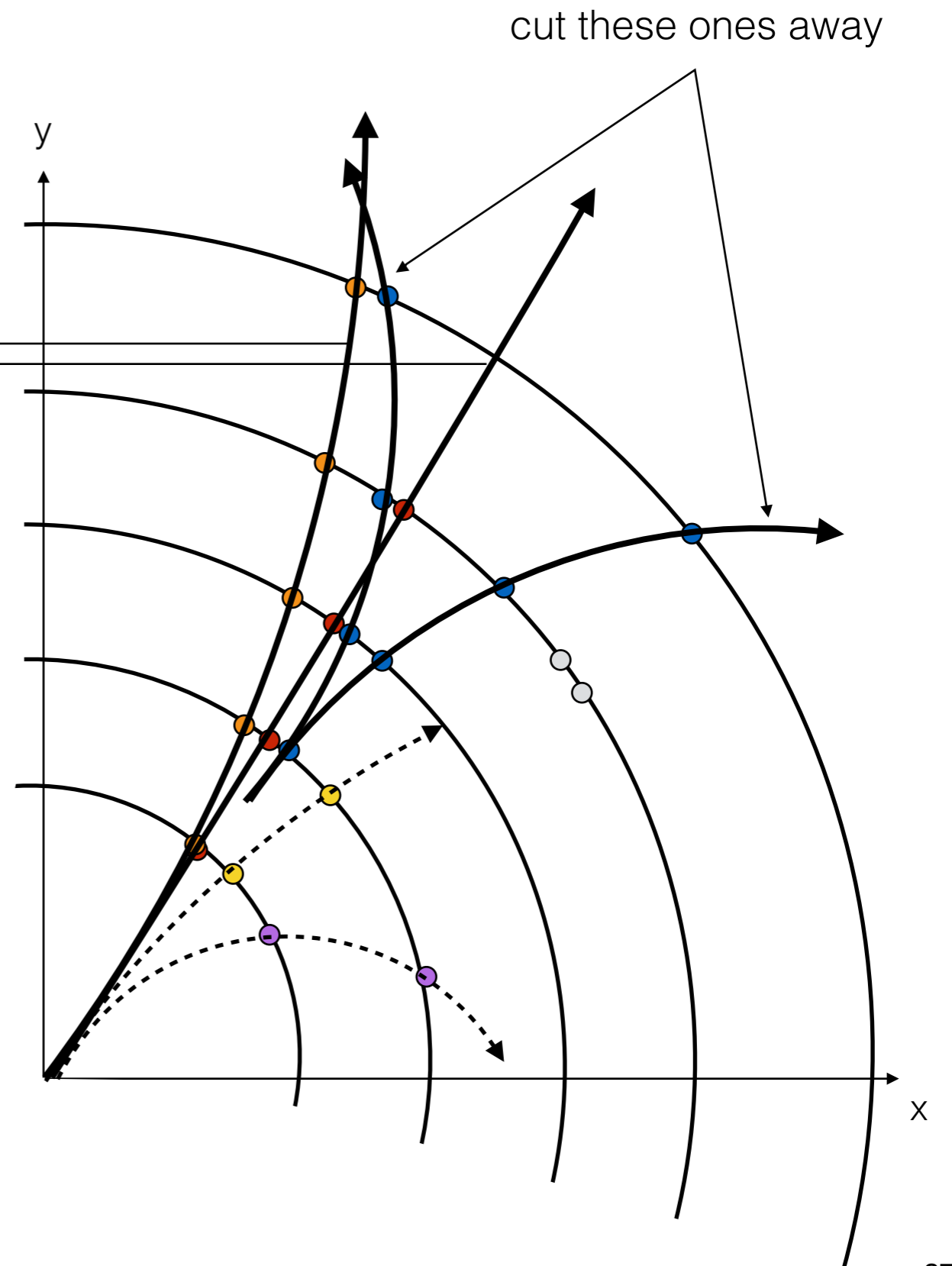
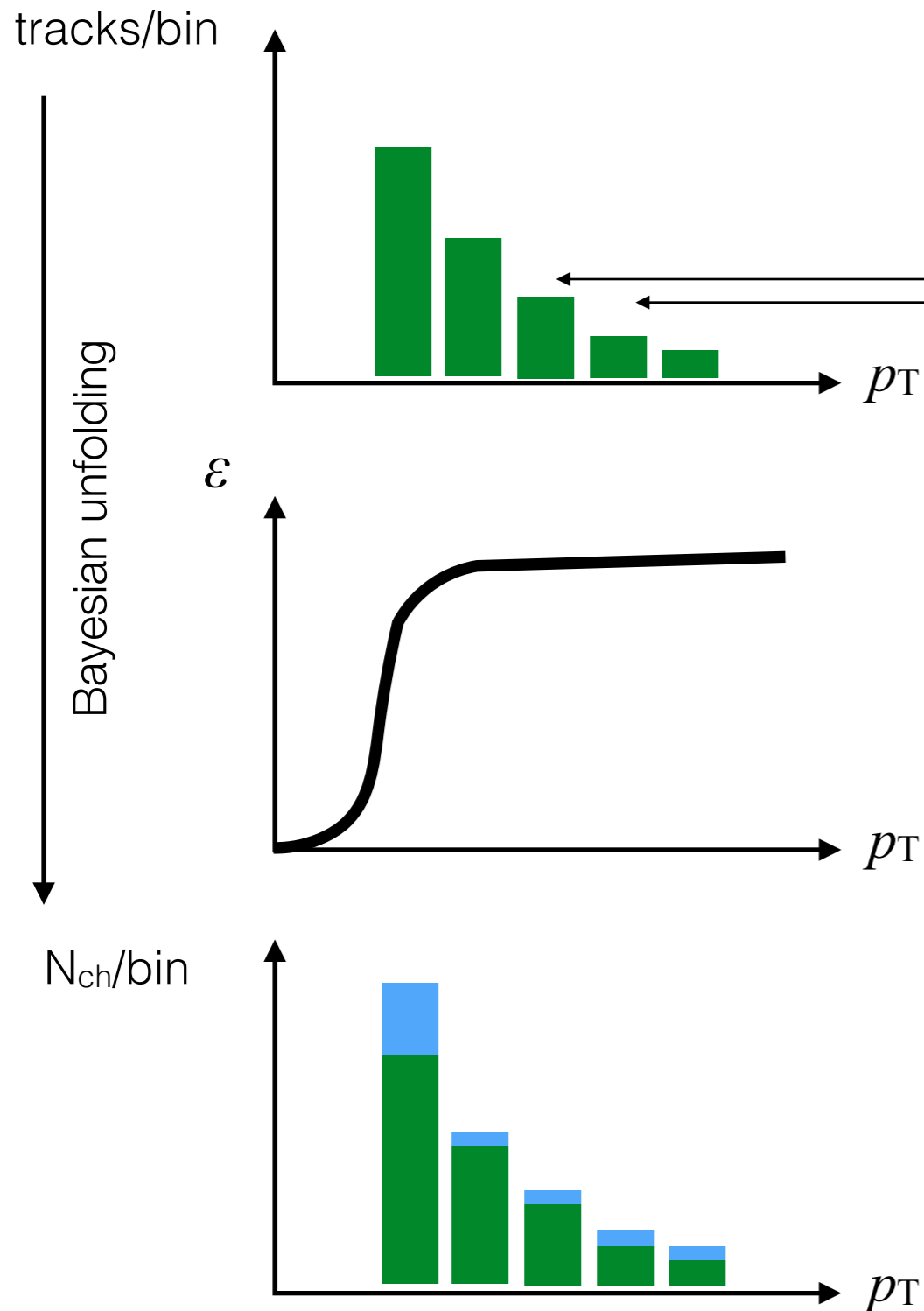
Example: charged particle multiplicity

- ▶ Let's finish the analysis



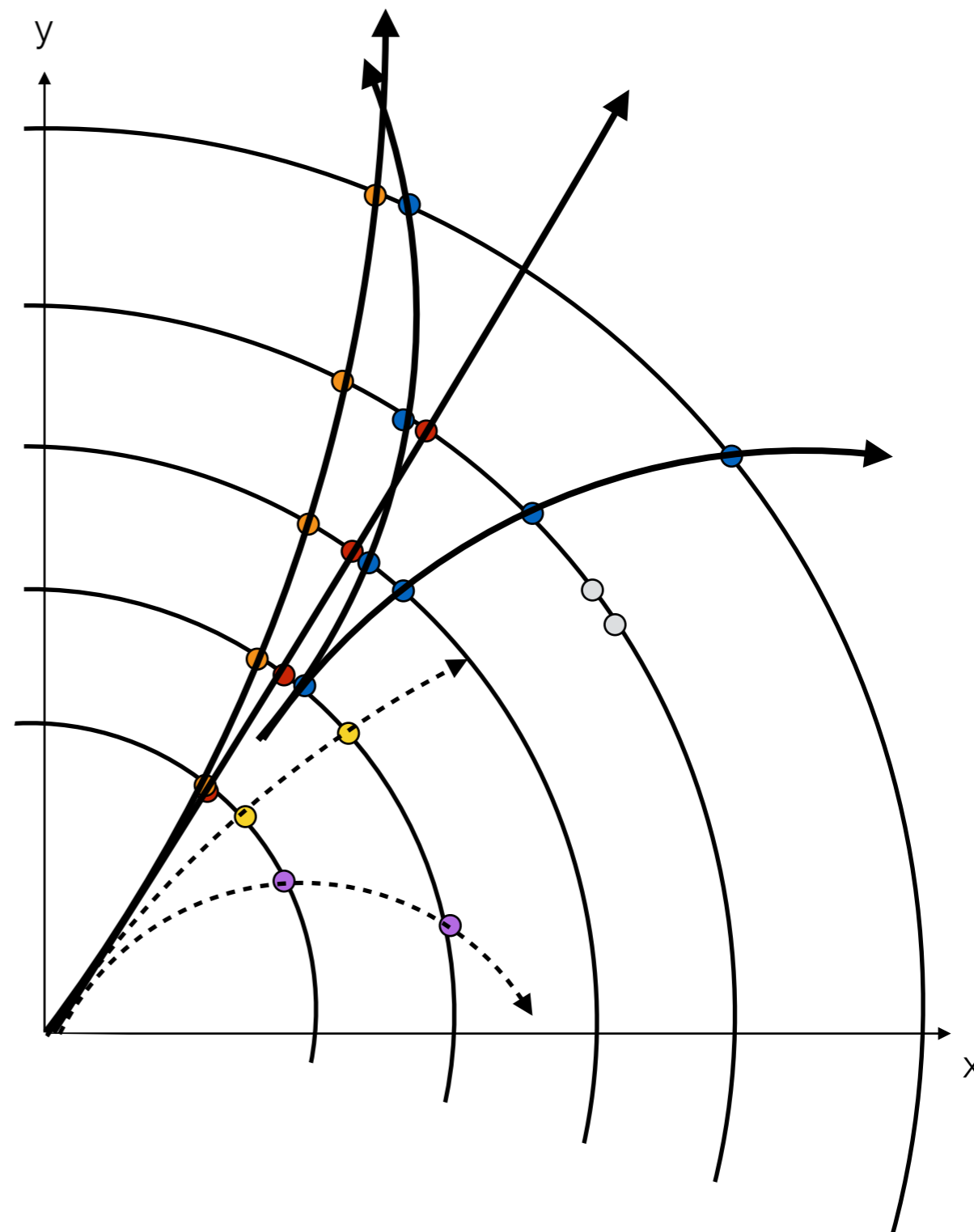
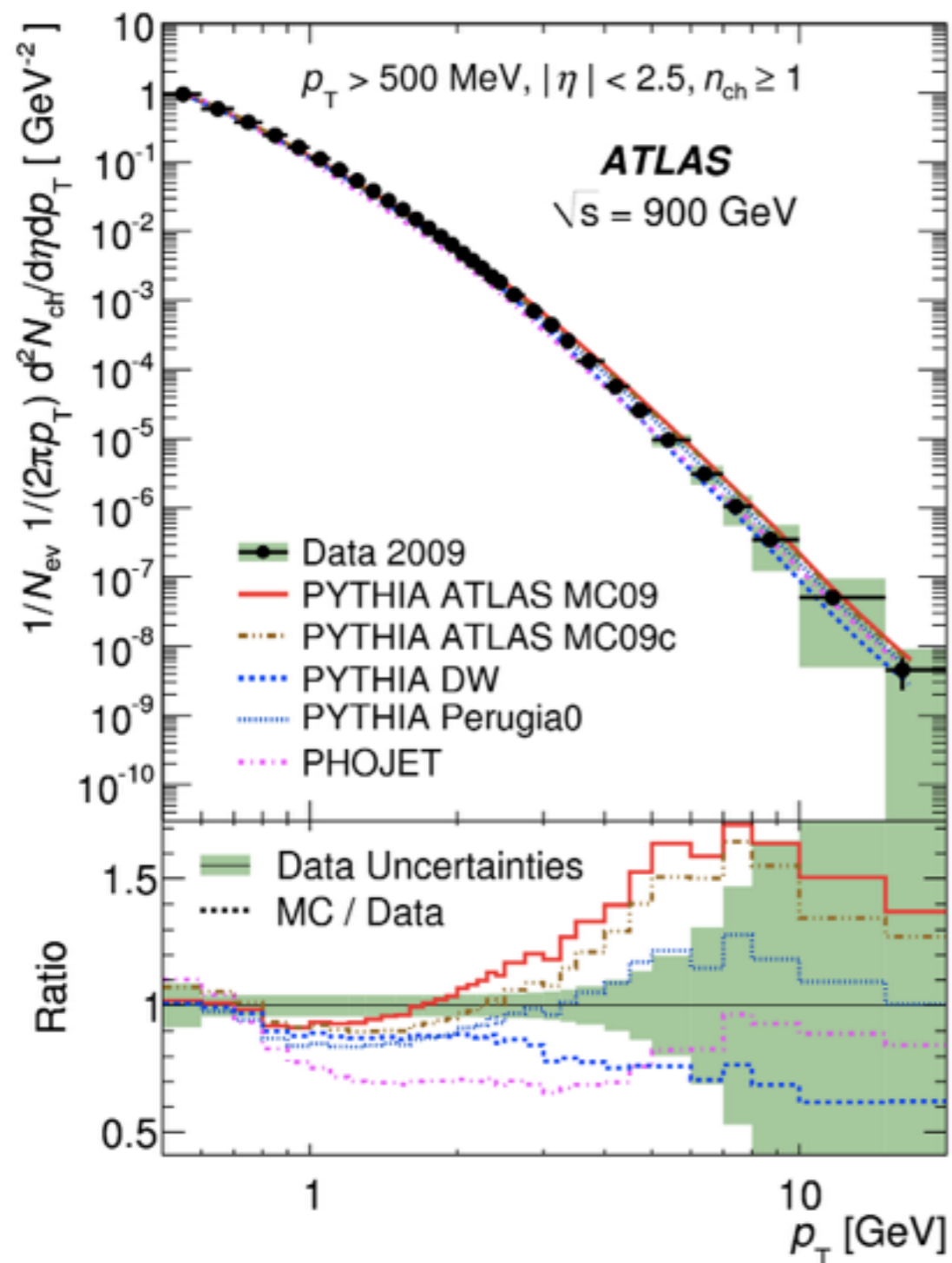
Example: charged particle multiplicity

- ▶ Let's finish the analysis



Example: charged particle multiplicity

- ▶ Let's finish the analysis: done.



Example: Higgs mass from $H \rightarrow ZZ^* \rightarrow 4\mu$

- ▶ A three-step master plan to a Higgs mass paper :
(might be even more "slightly" simplified)

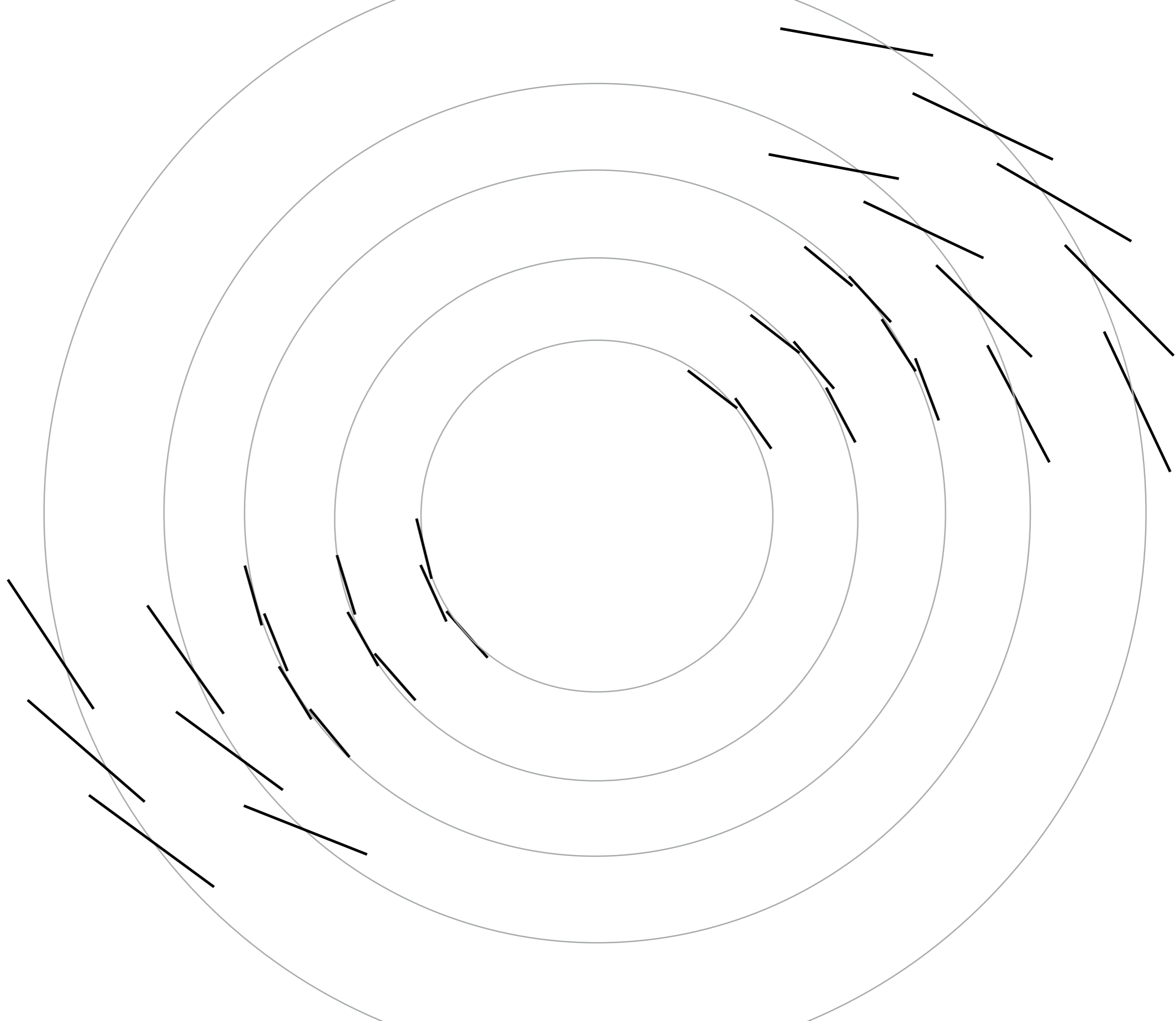
1. take data with some sort of muon trigger and find events with 4 muons in the final state (2 oppositely charged muon pairs)
2. use their track momentum measurement and combine them to a common Higgs mass
3. fit the Higgs mass, write up and publish

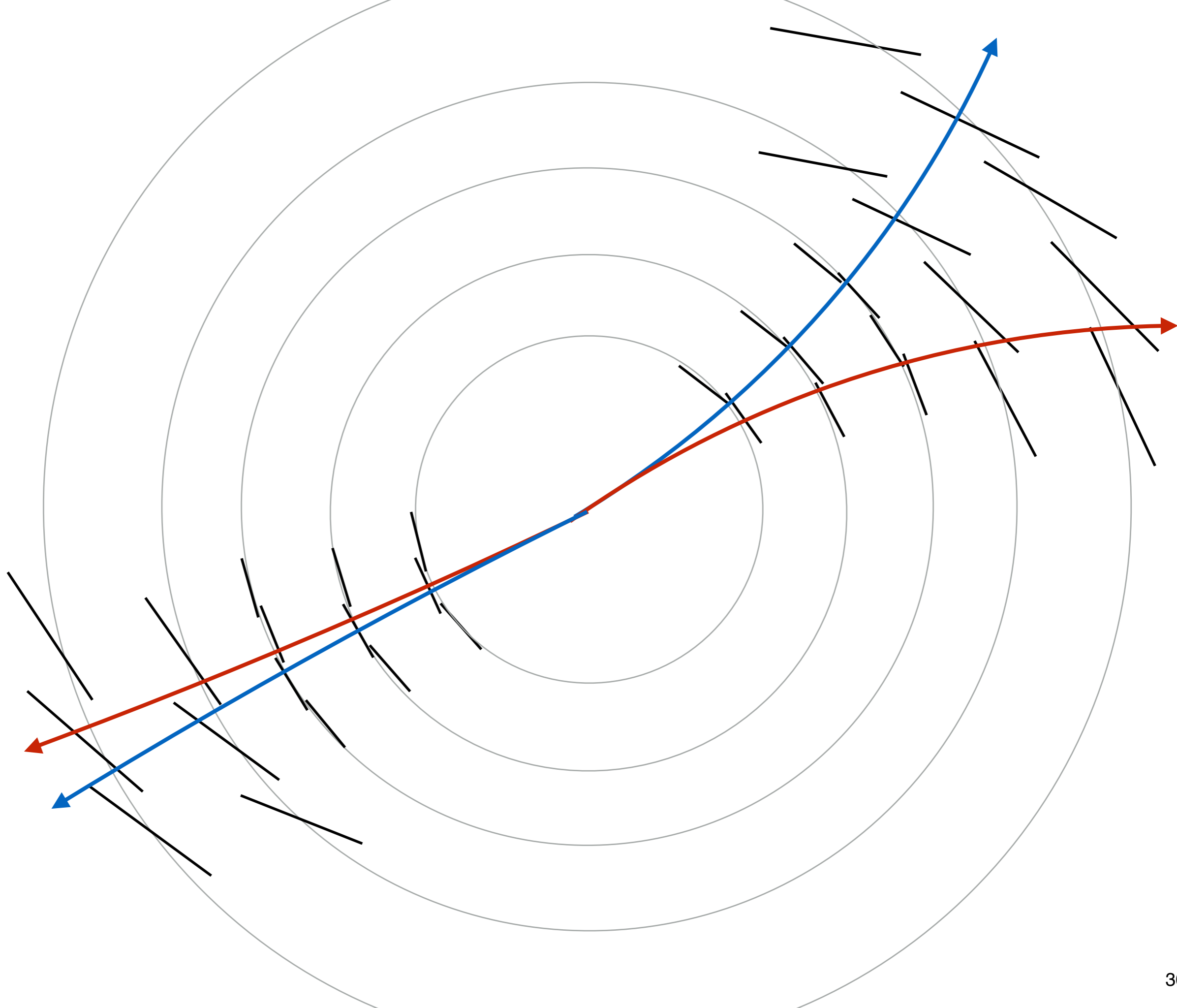
Example: Higgs mass from $H \rightarrow ZZ^* \rightarrow 4\mu$

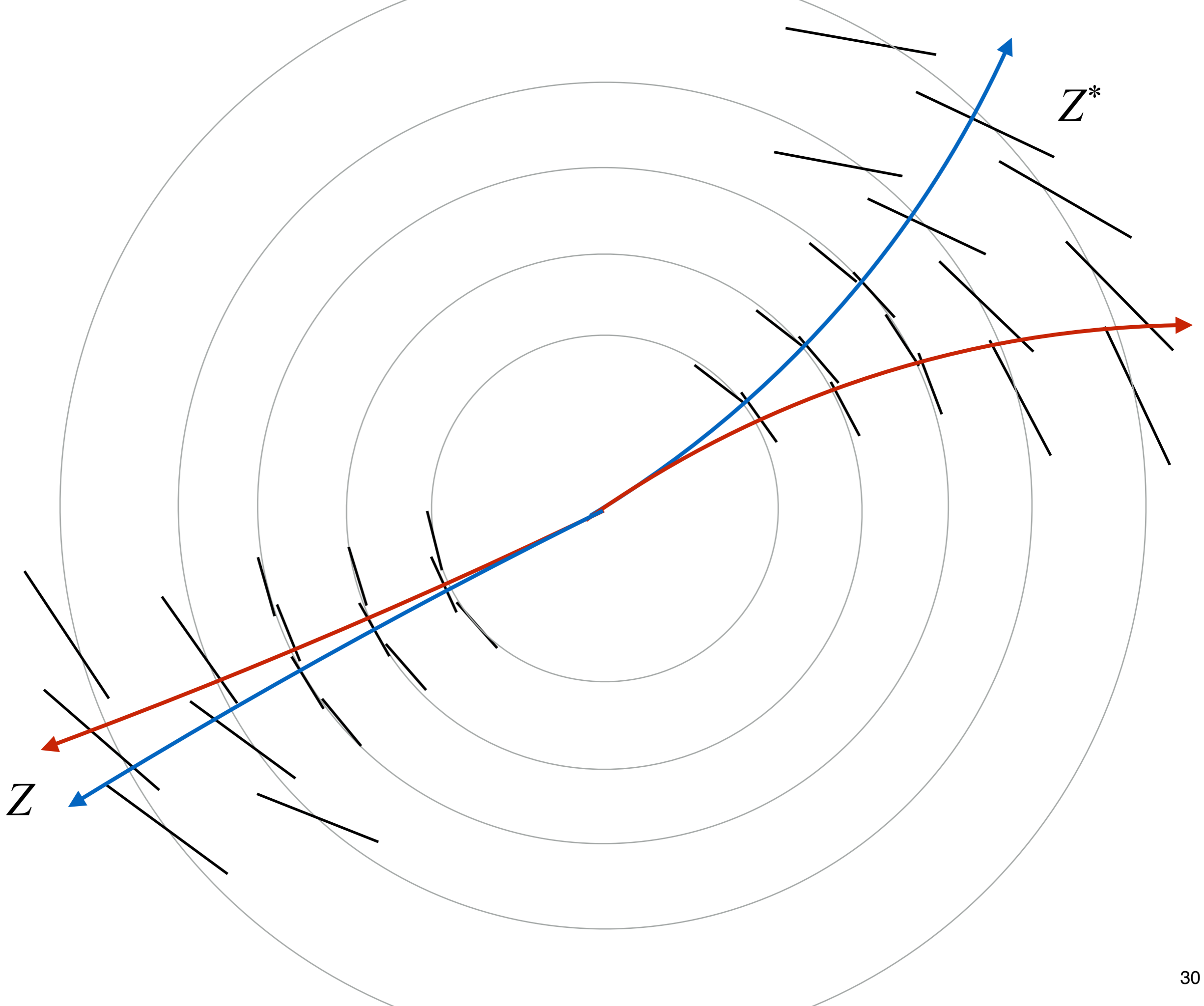
- ▶ A three-step master plan to a Higgs mass paper :
(might be even more "slightly" simplified)

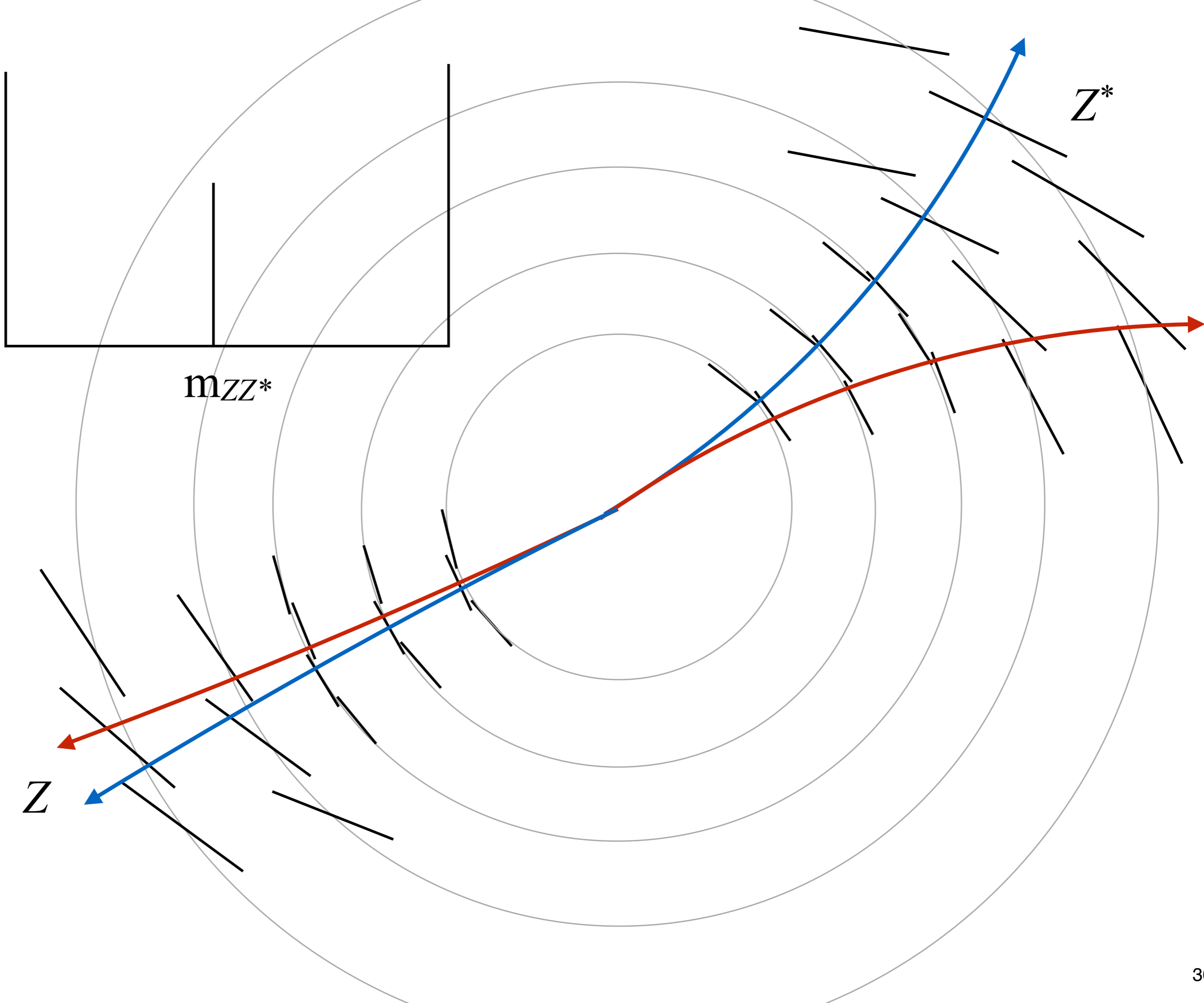
1. take data with some sort of muon trigger and find events with 4 muons in the final state (2 oppositely charged muon pairs)
2. use their track momentum measurement and combine them to a common Higgs mass
3. fit the Higgs mass, write up and publish

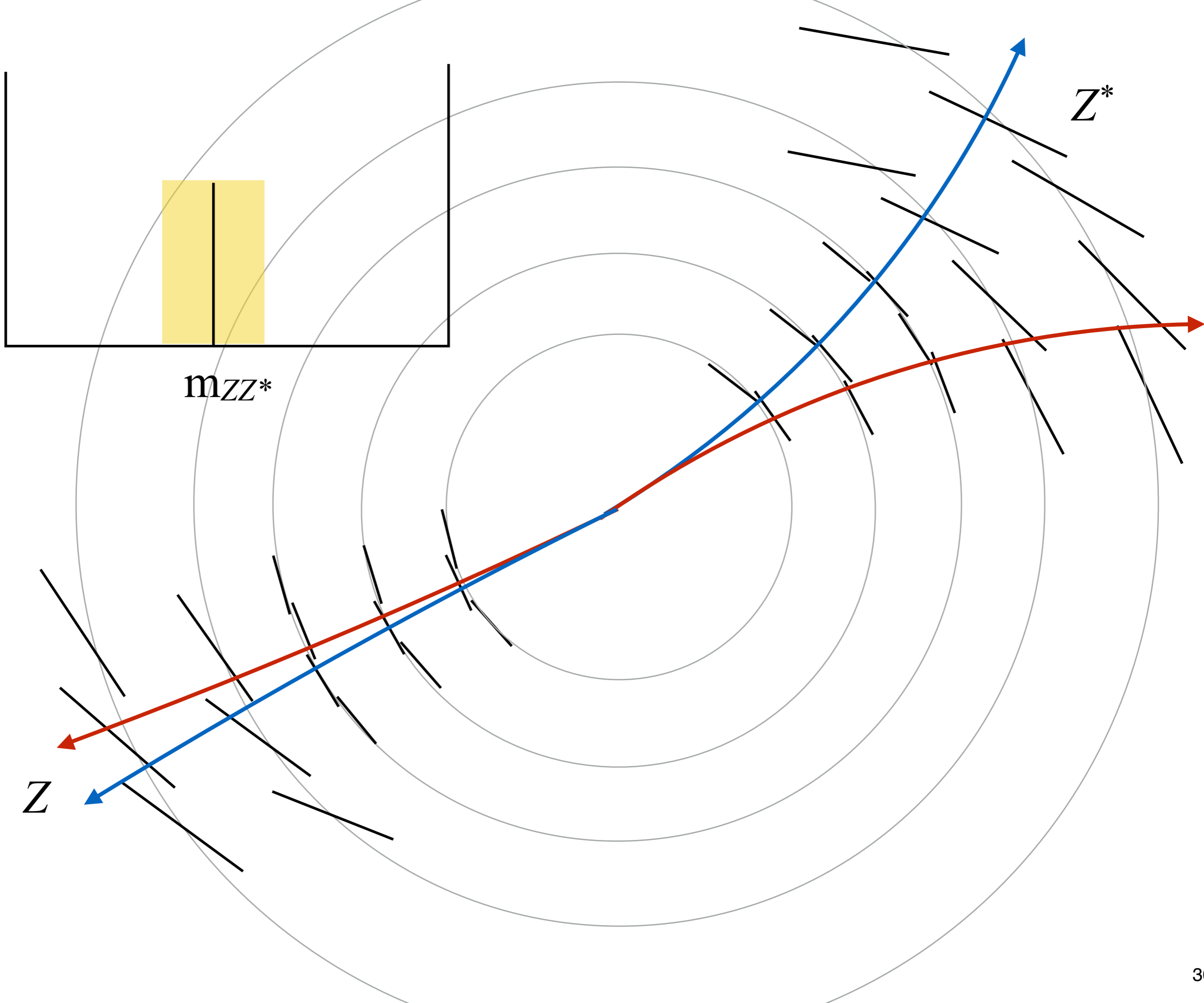
Let's do it !

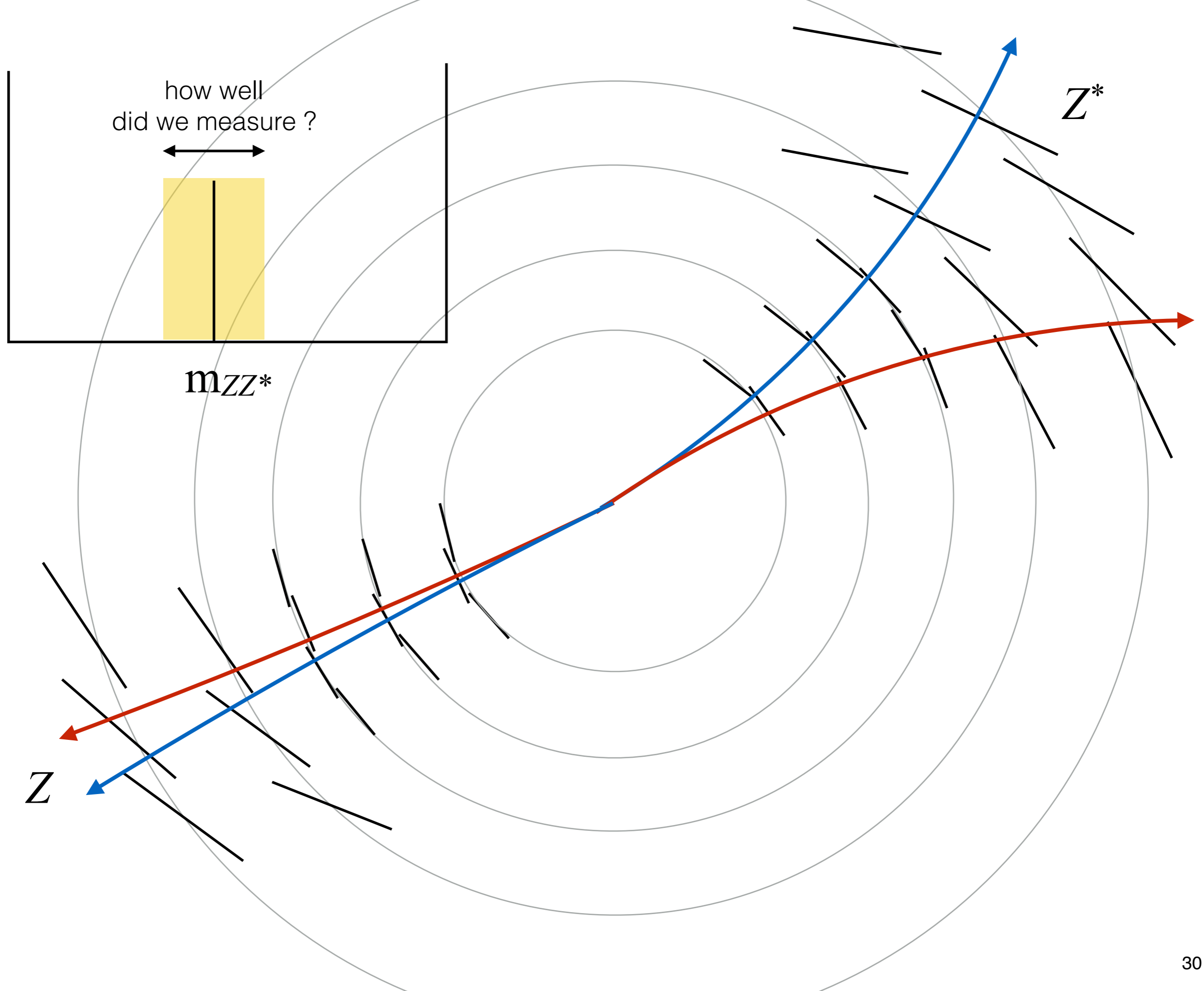


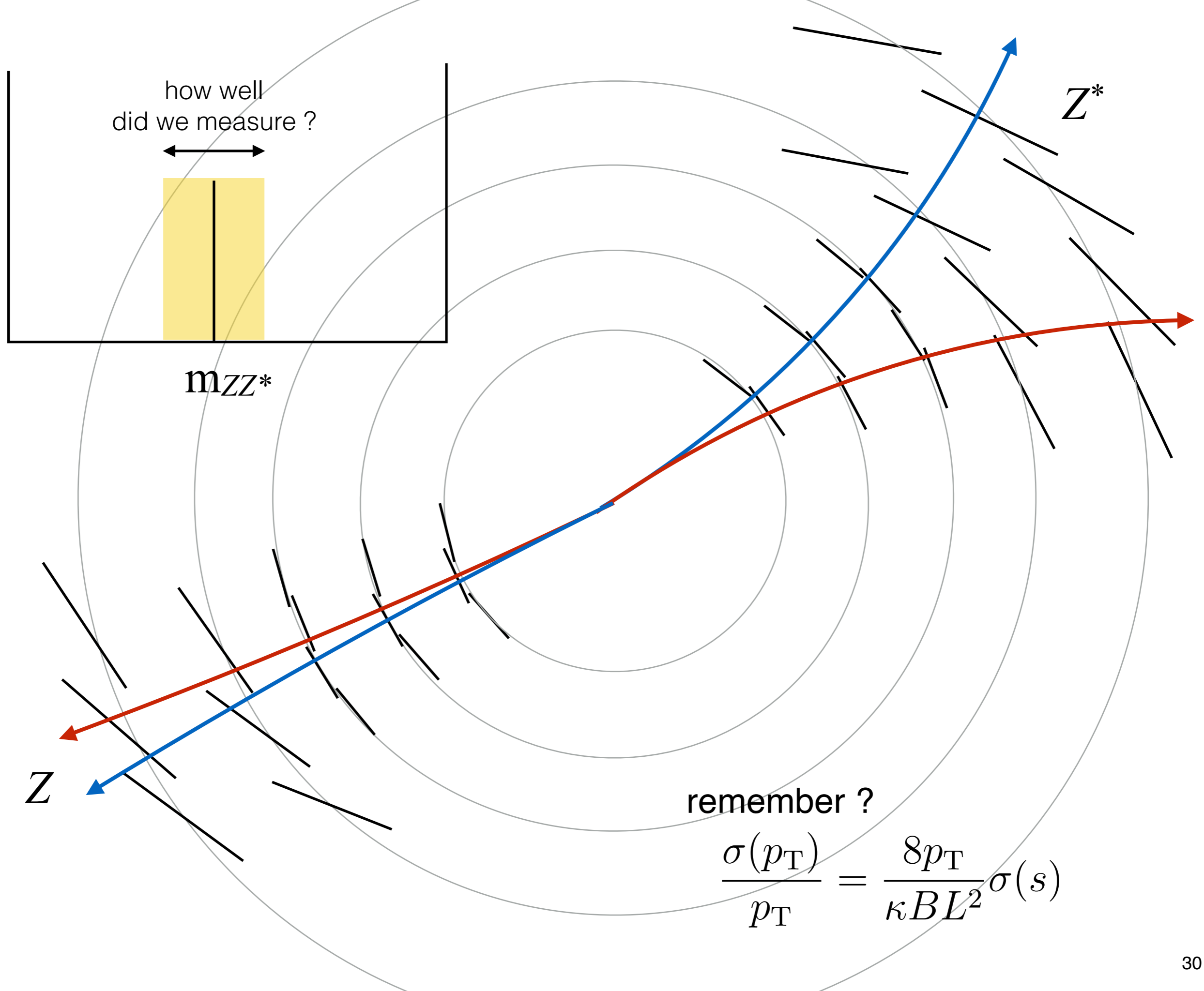




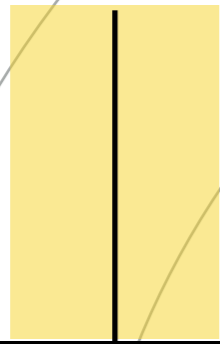








how well
did we measure ?



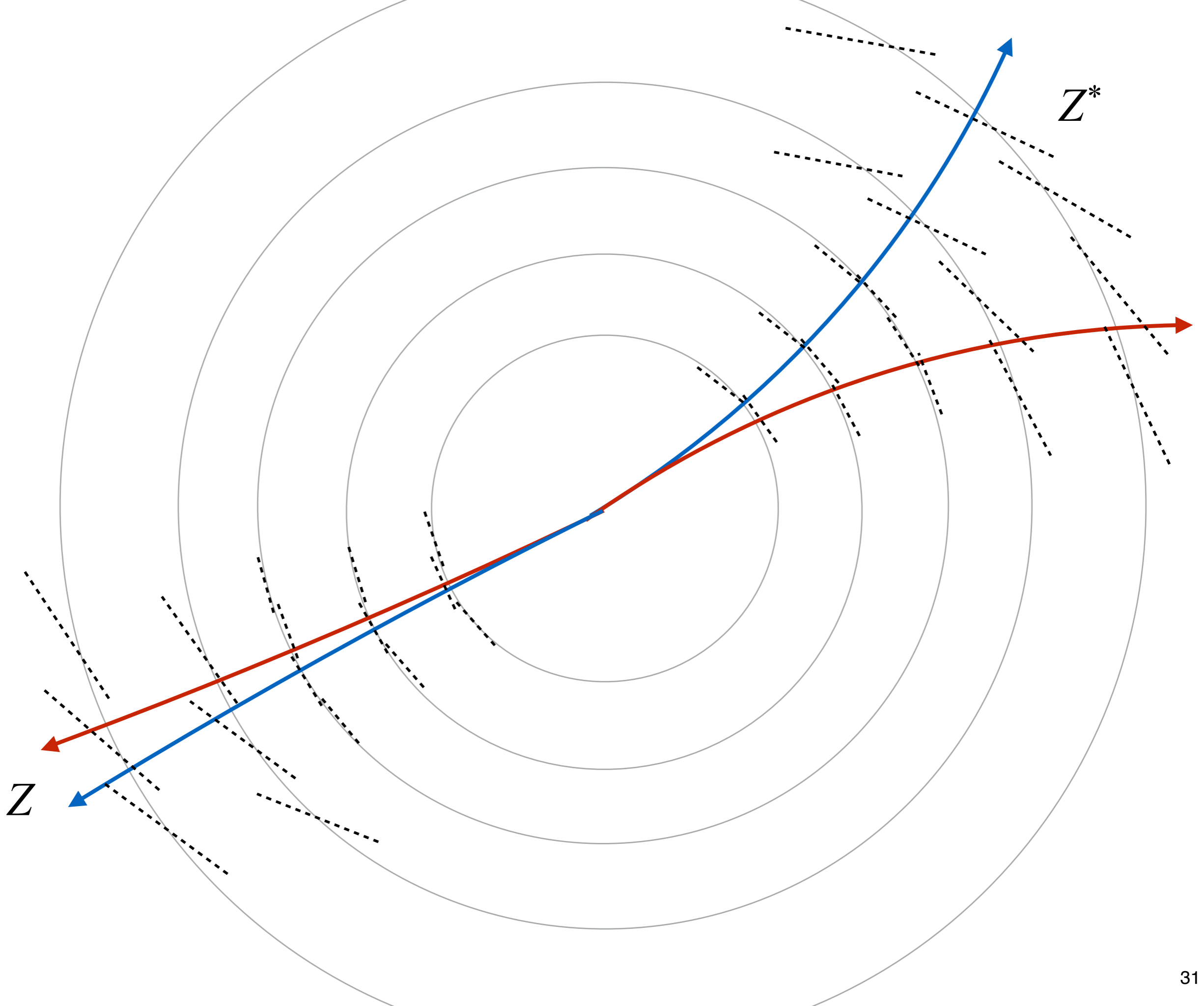
m_{ZZ^*}

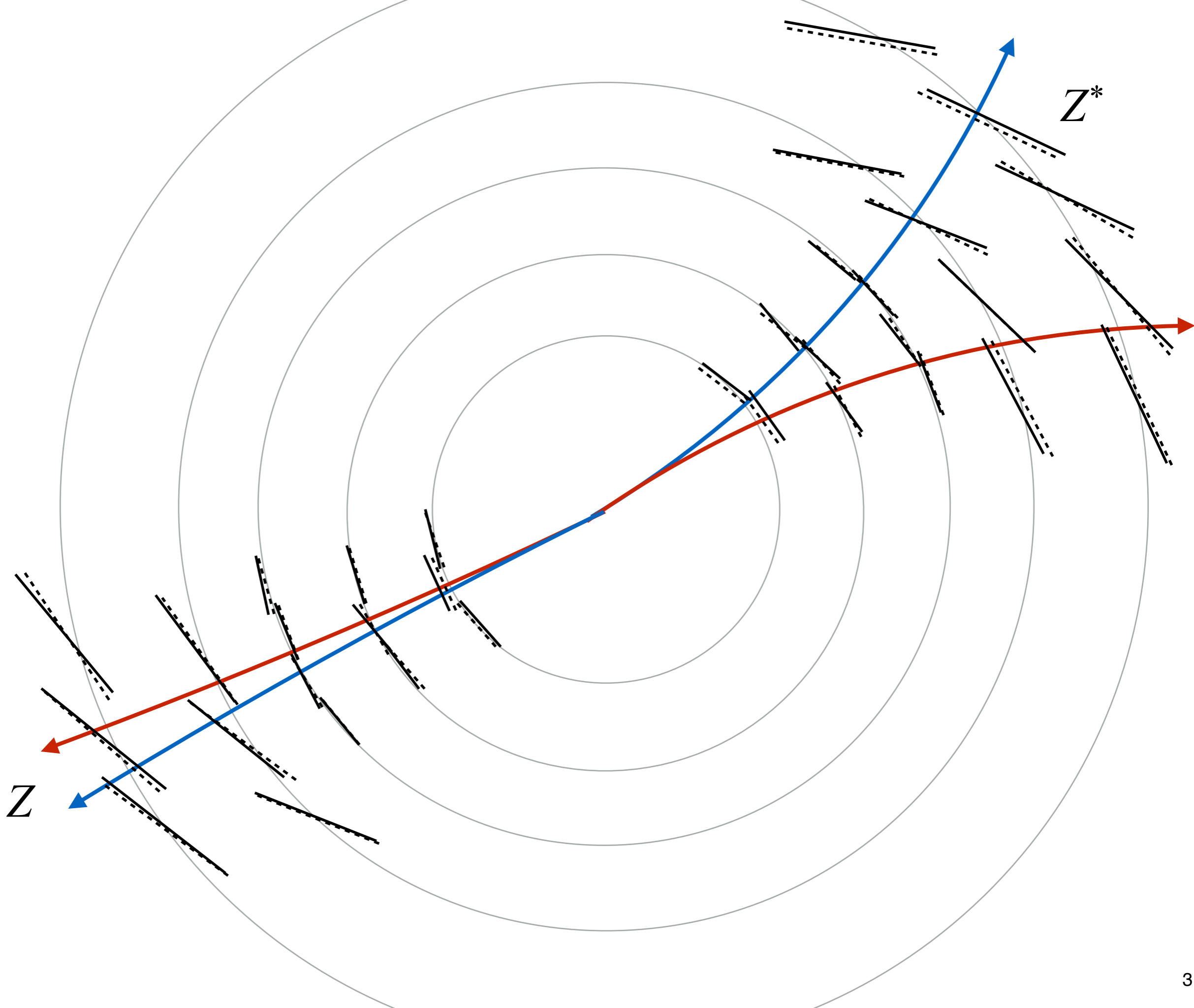
Z^*

Z

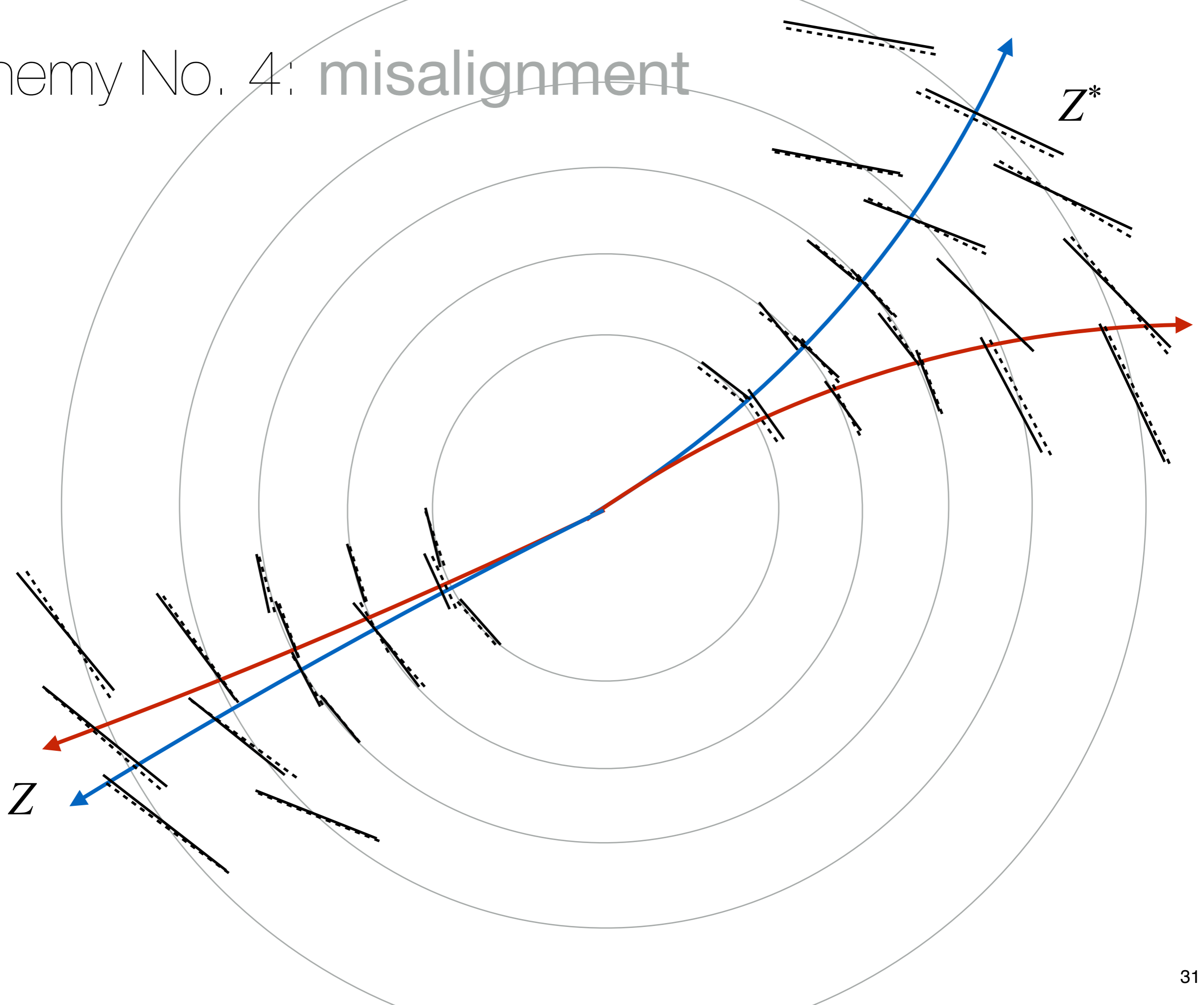
remember ?

$$\frac{\sigma(p_T)}{p_T} = \frac{8p_T}{\kappa B L^2} \sigma(s)$$



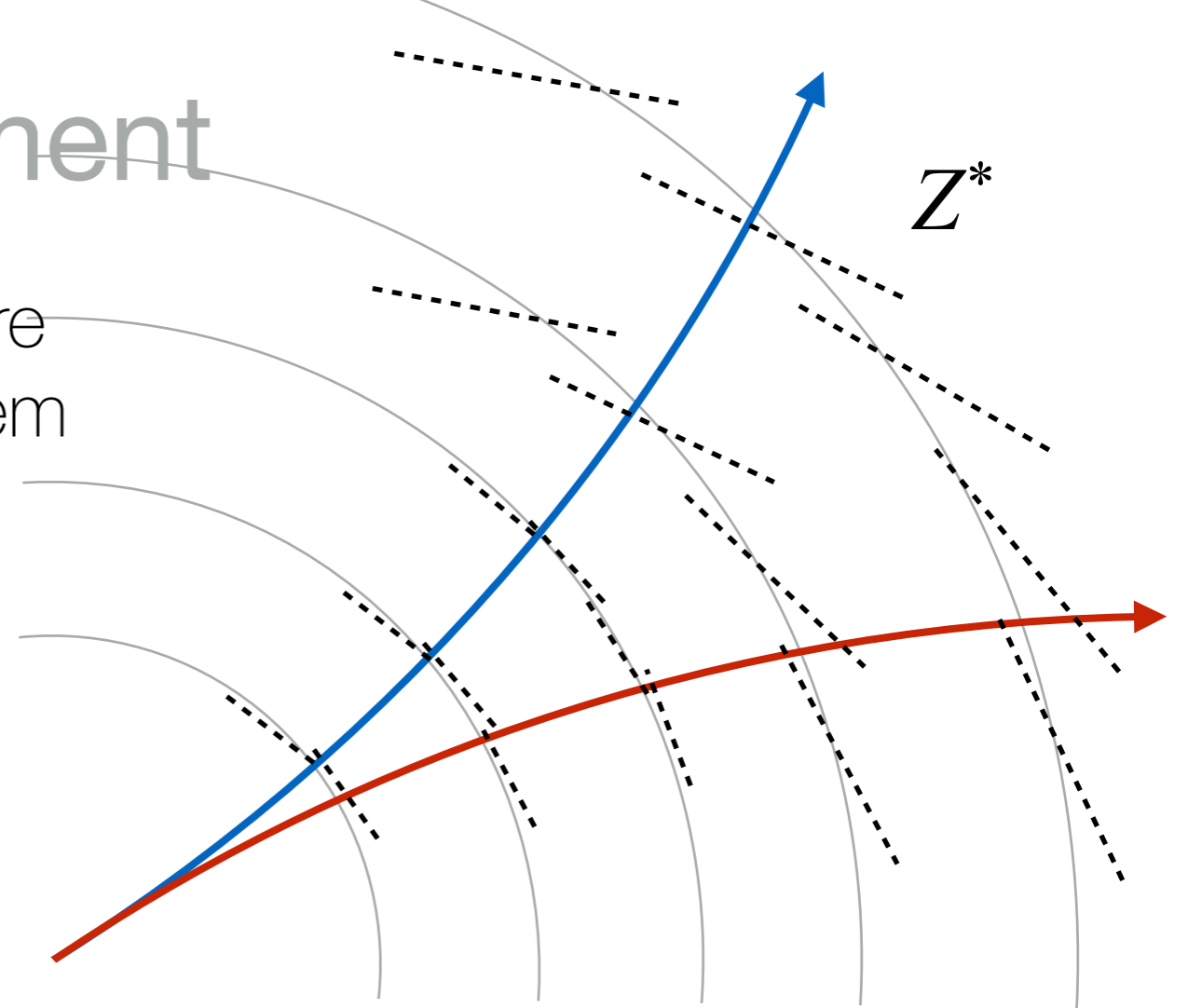


Enemy No. 4: misalignment



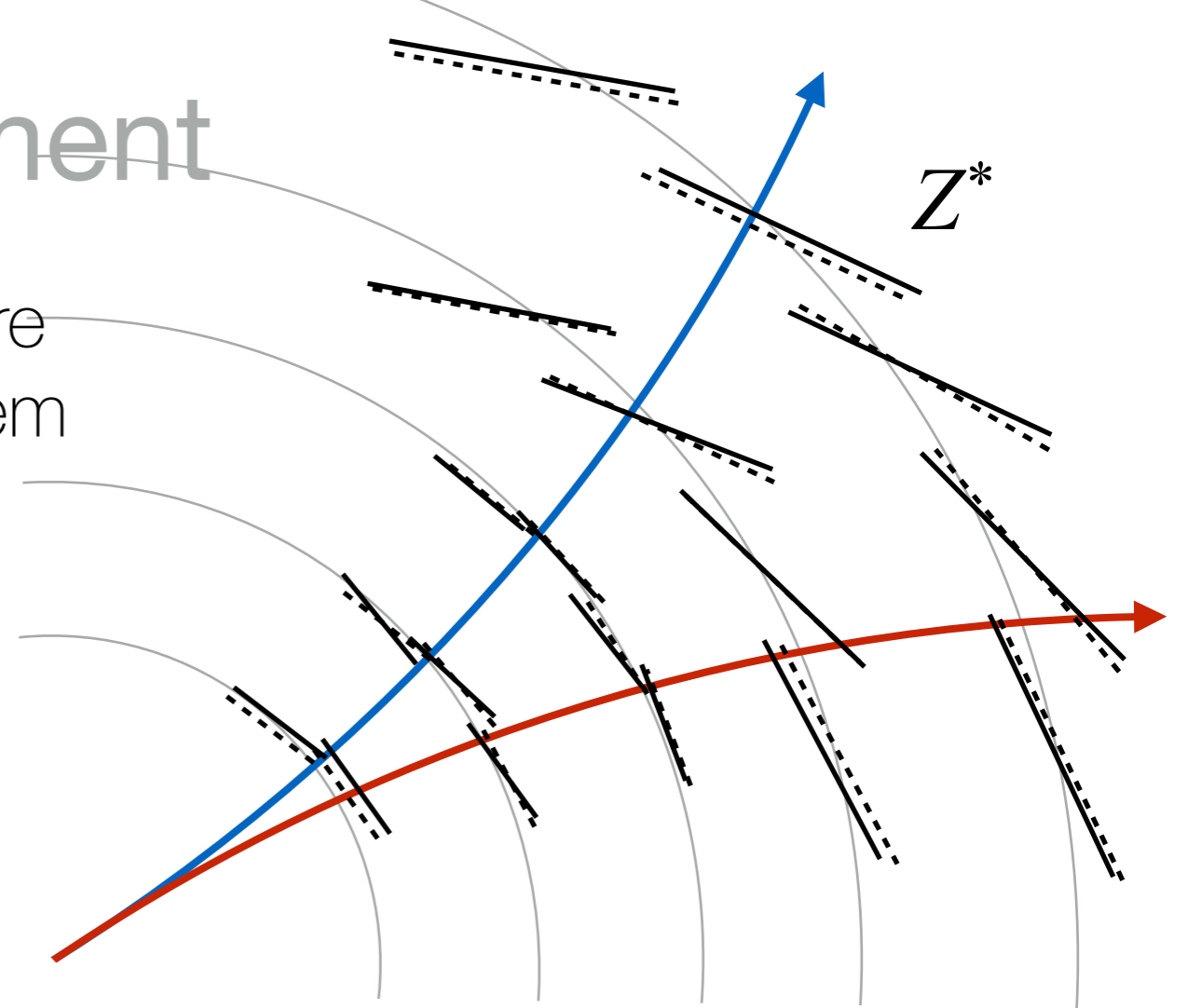
Enemy No. 4: misalignment

- ▶ No one said the detector elements are actually exactly where you expect them
 - you need to find out where they are !
- ▶ Major feature: random module misalignment
 - can be corrected for using alignment algorithms
 - most commonly used:
a global χ^2 minimisation using tracks and varying the module positions
 - many many degrees of freedom (ATLAS: 36 k matrix inversion, CMS even more)
- ▶ It works !



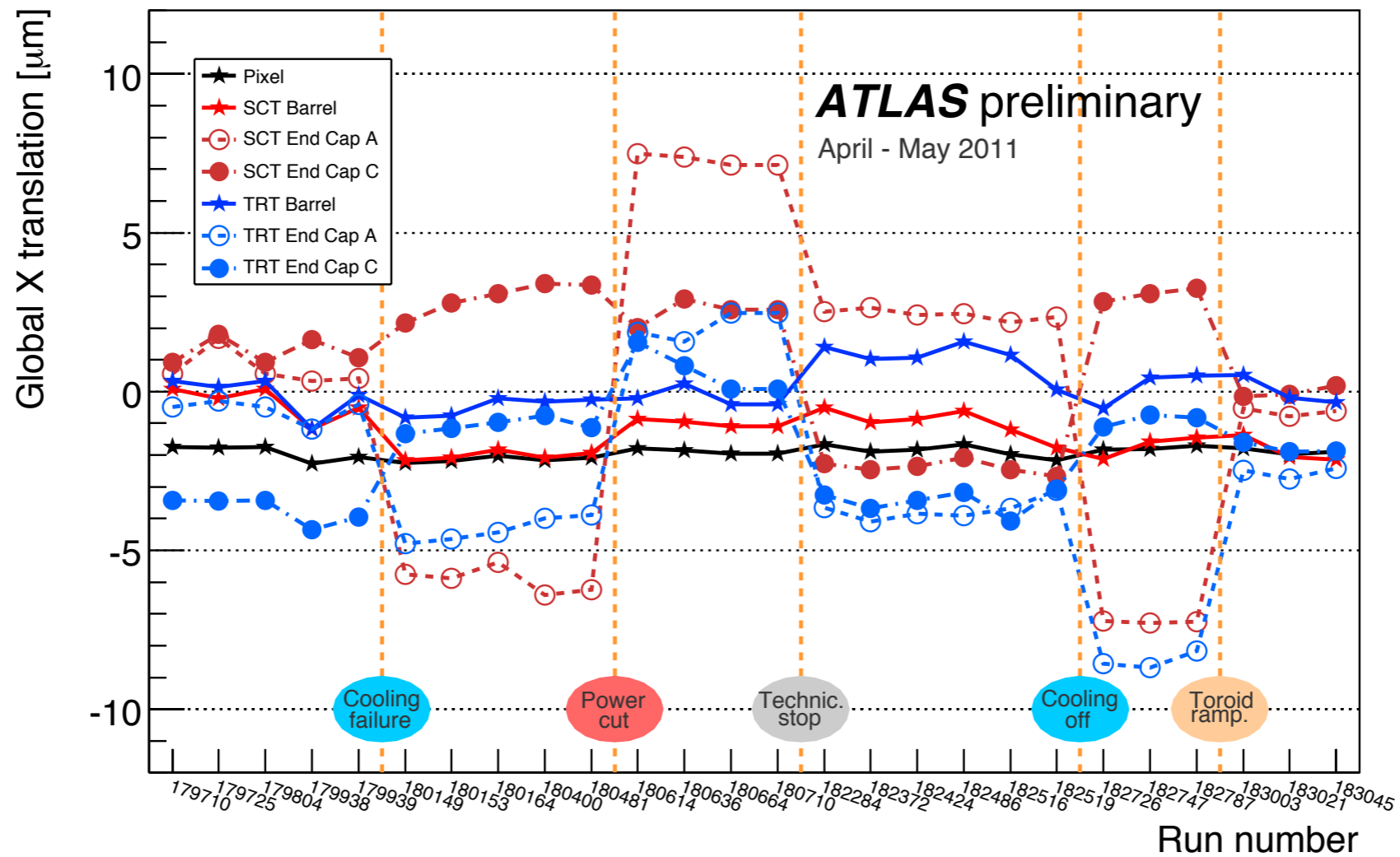
Enemy No. 4: misalignment

- ▶ No one said the detector elements are actually exactly where you expect them
 - you need to find out where they are !
- ▶ Major feature: random module misalignment
 - can be corrected for using alignment algorithms
 - most commonly used:
a global χ^2 minimisation using tracks and varying the module positions
 - many many degrees of freedom (ATLAS: 36 k matrix inversion, CMS even more)
- ▶ It works !



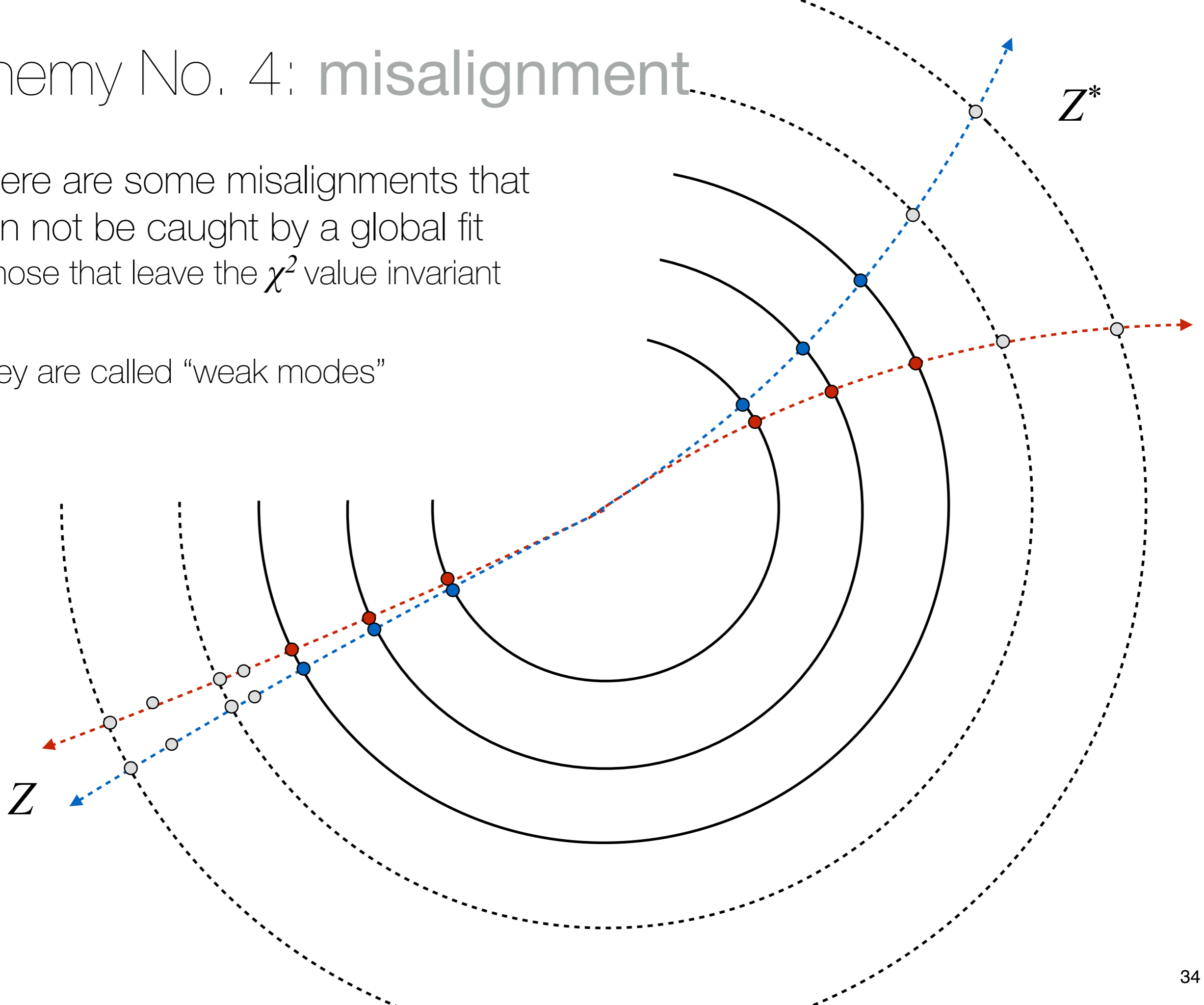
Enemy No. 4: misalignment

Level 1 alignment



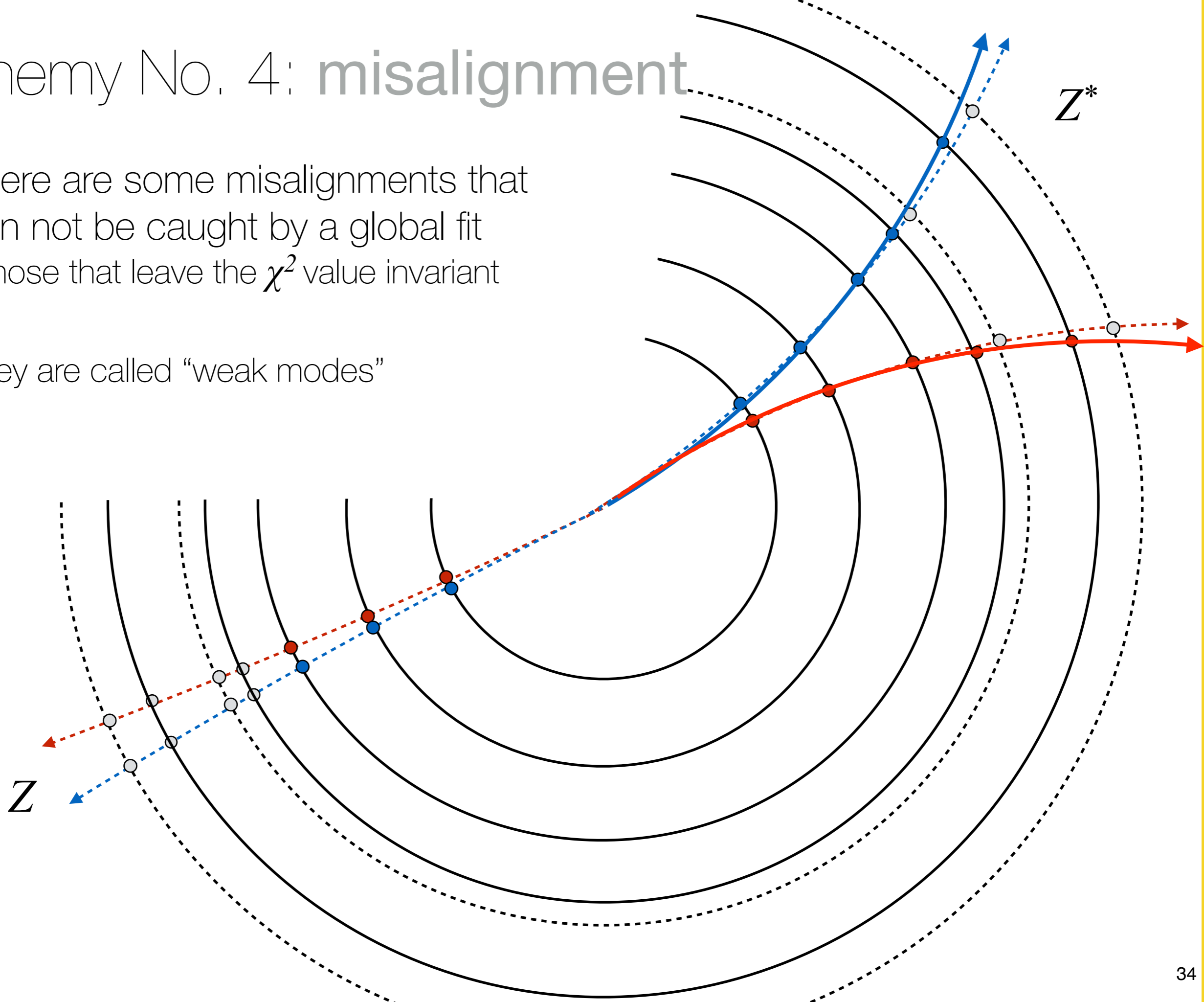
Enemy No. 4: misalignment

- ▶ There are some misalignments that can not be caught by a global fit
 - those that leave the χ^2 value invariant
- ▶ They are called “weak modes”



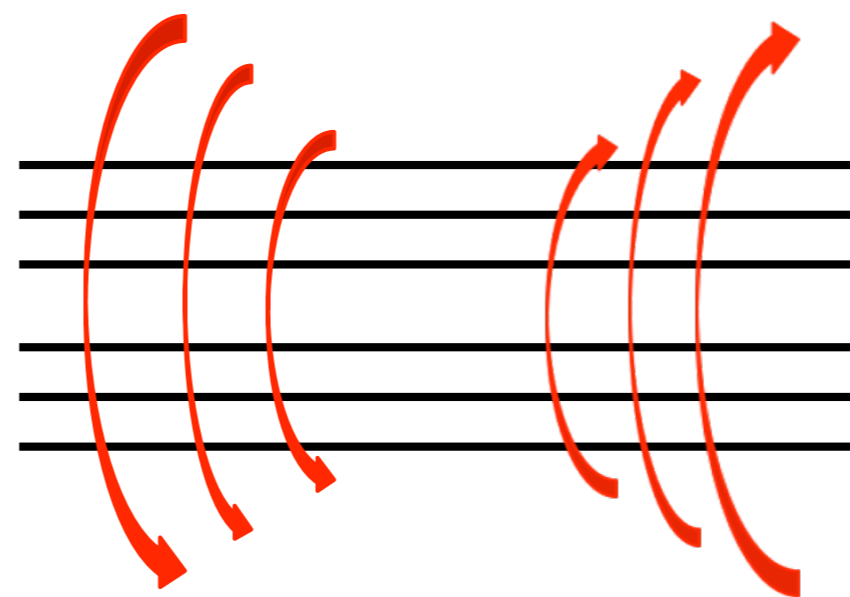
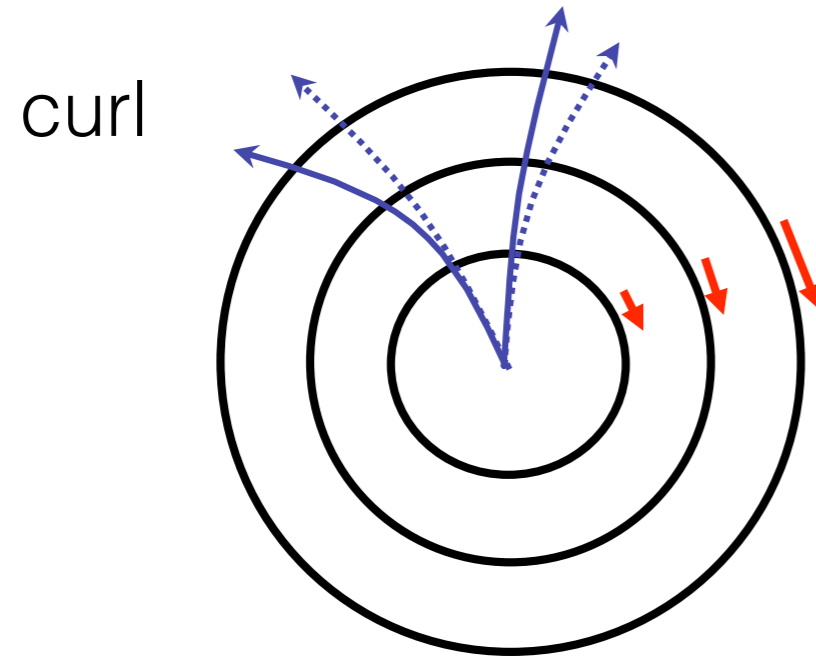
Enemy No. 4: misalignment

- ▶ There are some misalignments that can not be caught by a global fit
 - those that leave the χ^2 value invariant
- ▶ They are called “weak modes”

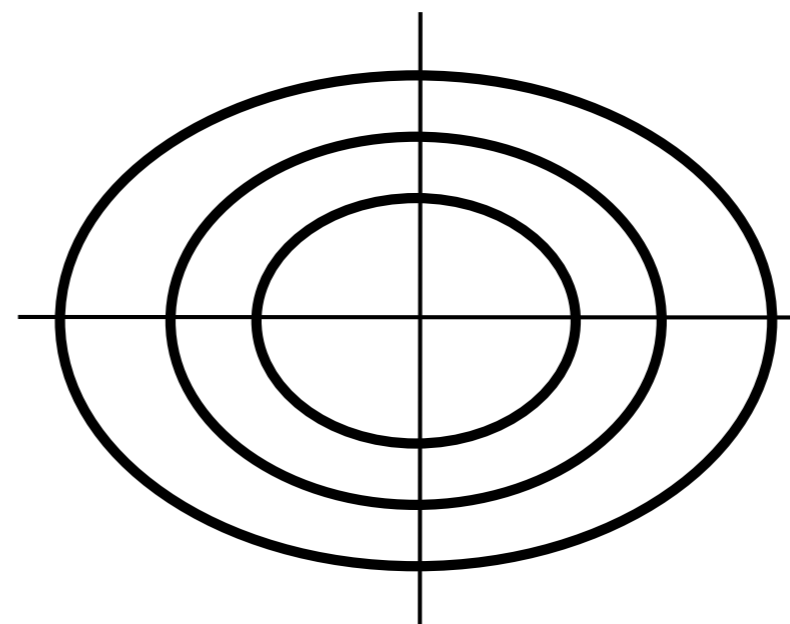
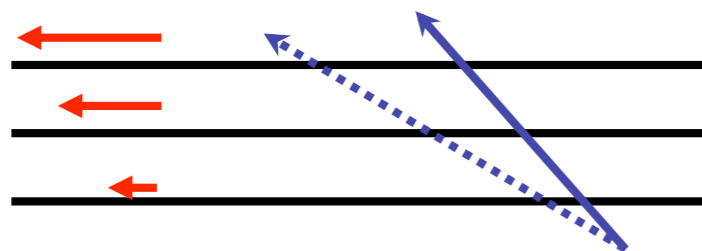


Enemy No. 4: misalignment

- ▶ weak modes that leave the χ^2 of a track invariant

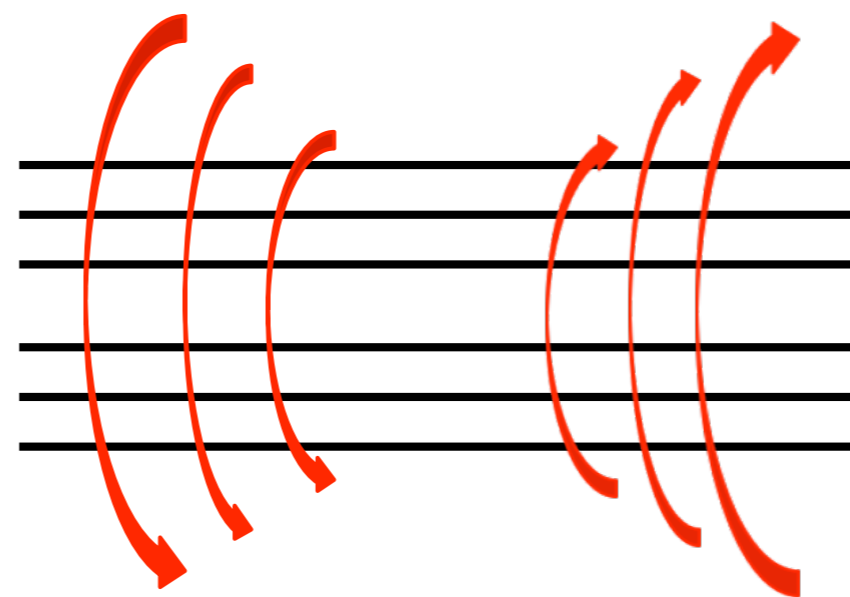
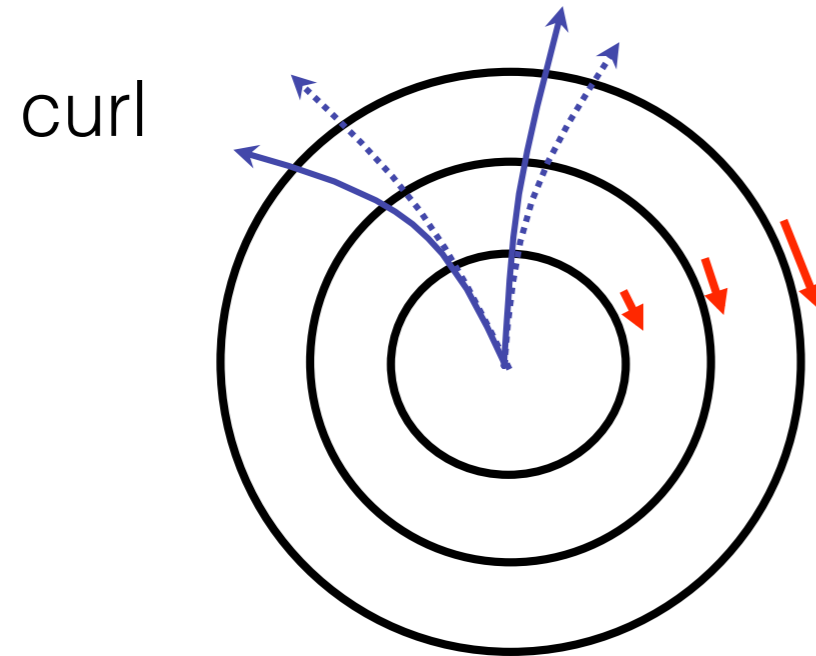


telescope

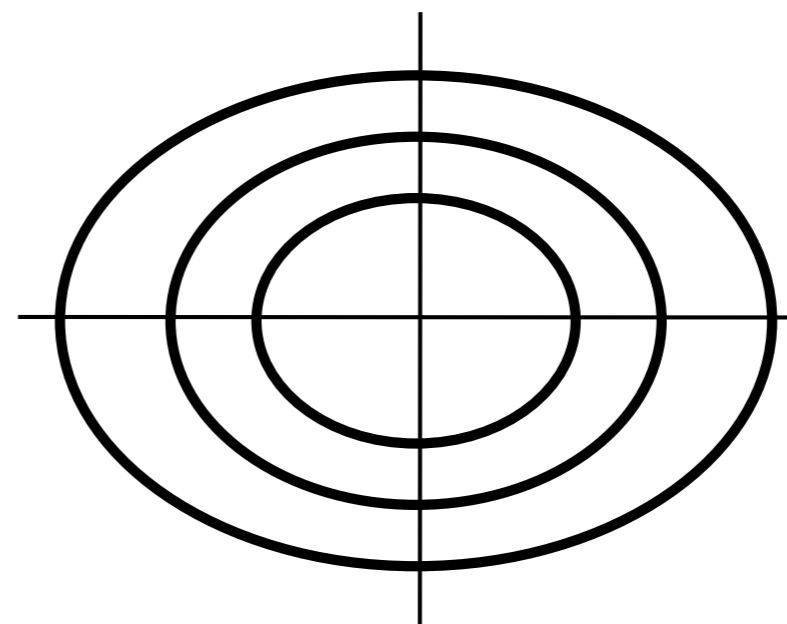
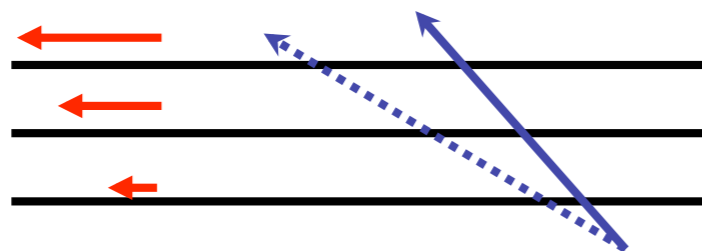


Enemy No. 4: misalignment

- ▶ weak modes that leave the χ^2 of a track invariant



telescope



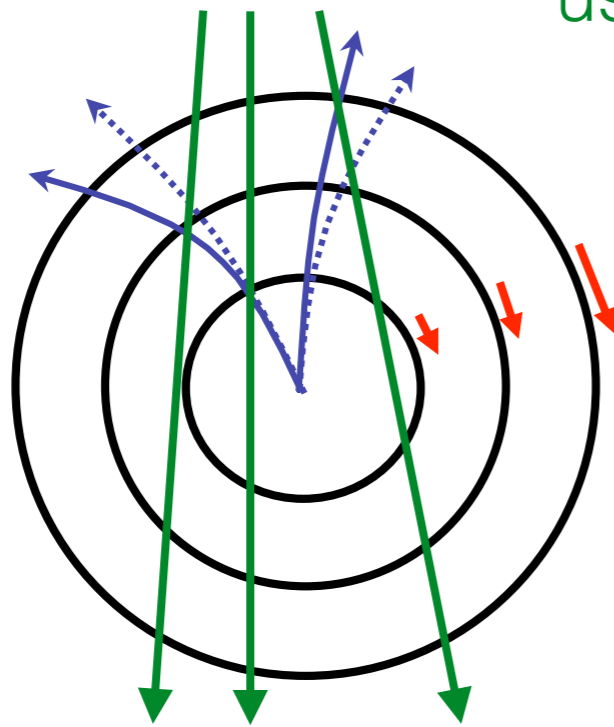
- ▶ What can we do ?

Enemy No. 4: misalignment

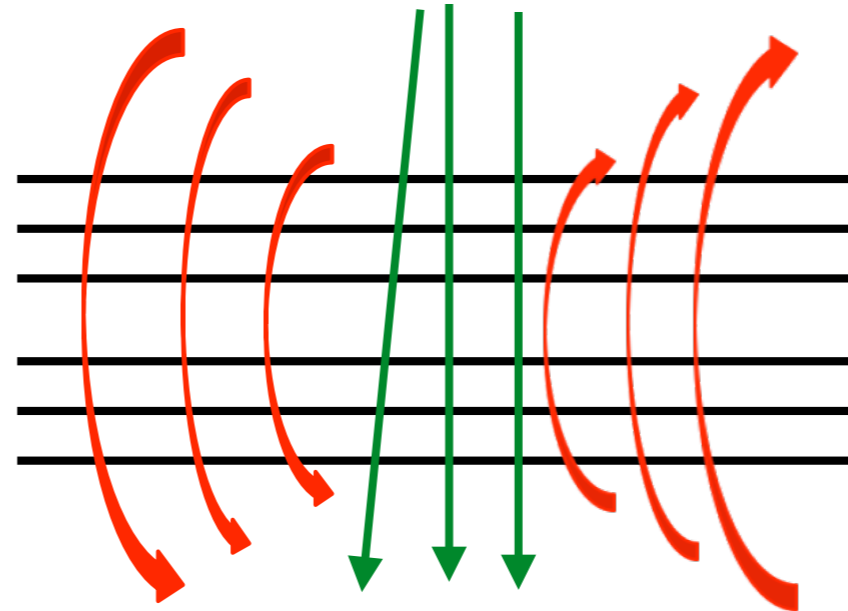
- ▶ weak modes that leave the χ^2 of a track invariant

use cosmic rays

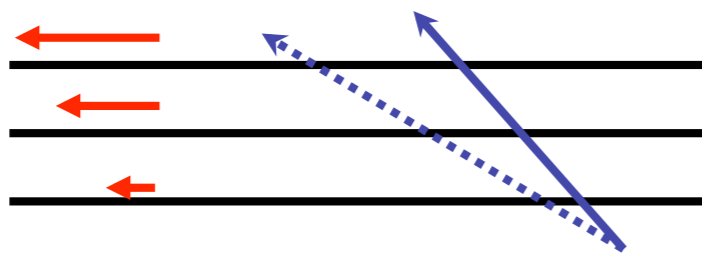
curl



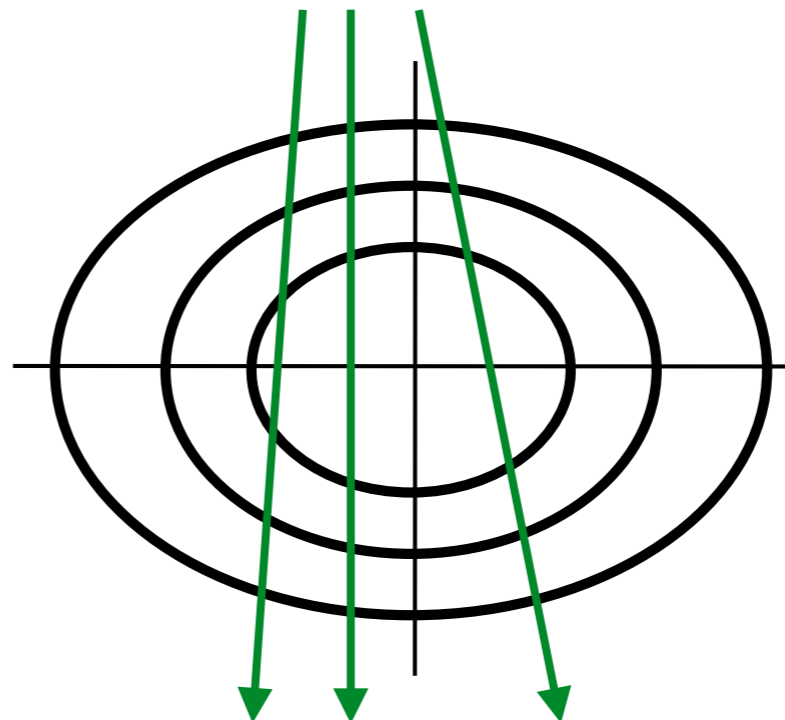
twist



telescope



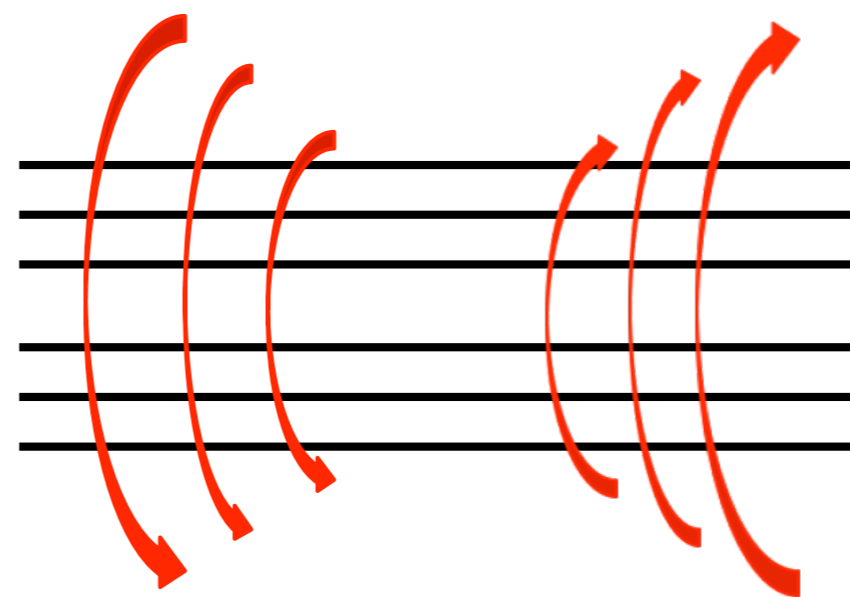
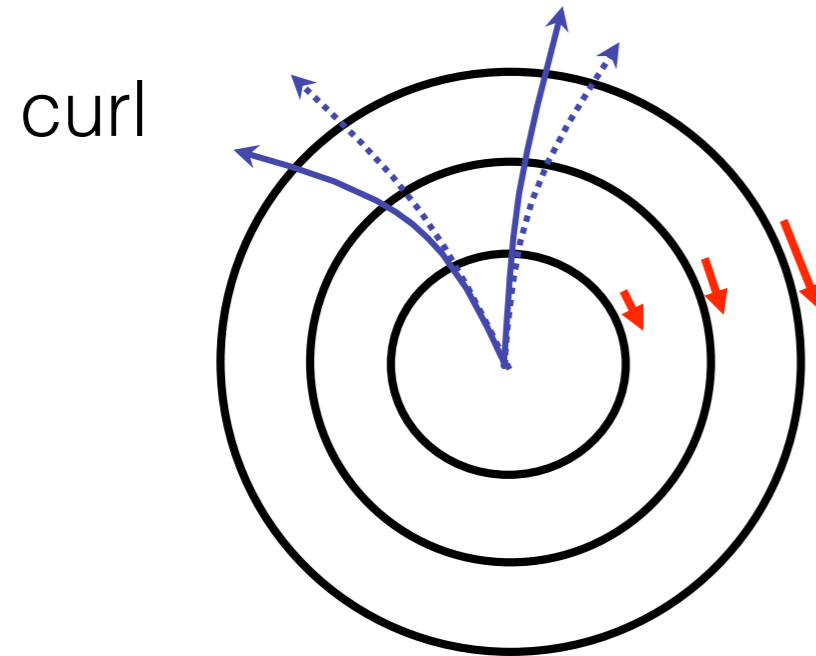
elliptical



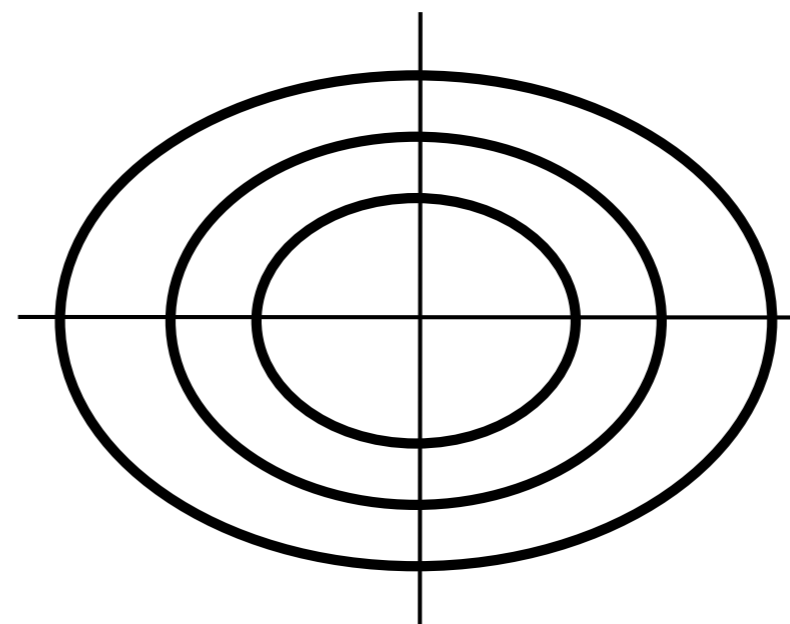
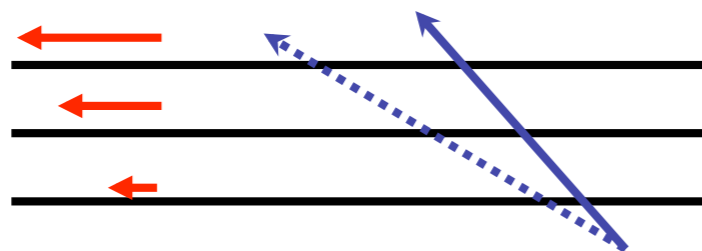
- ▶ What can we do ?

Enemy No. 4: misalignment

- ▶ weak modes that leave the χ^2 of a track invariant



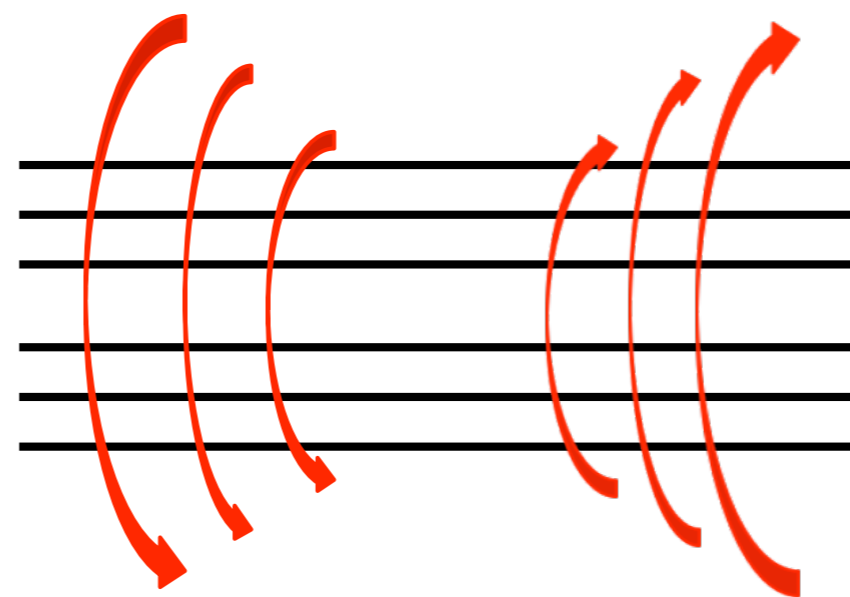
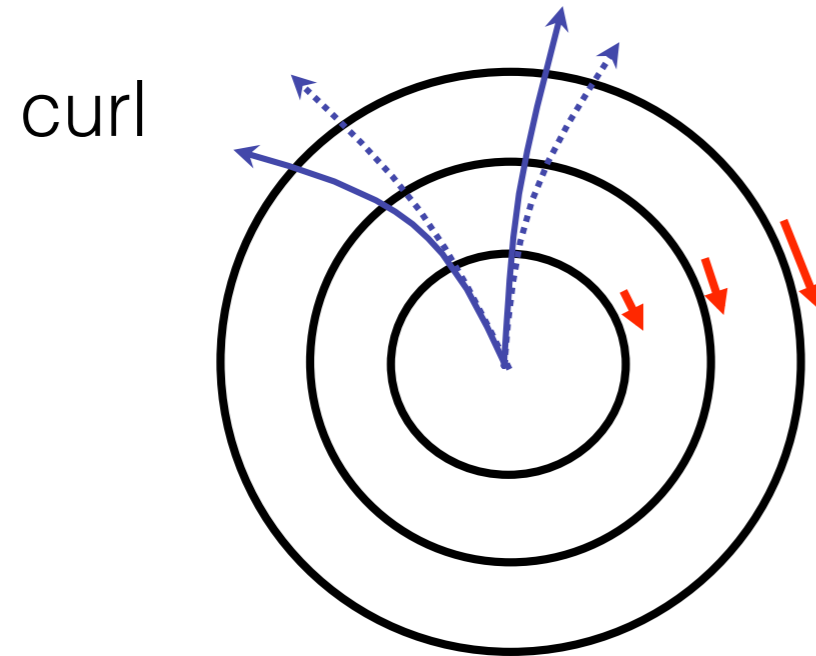
telescope



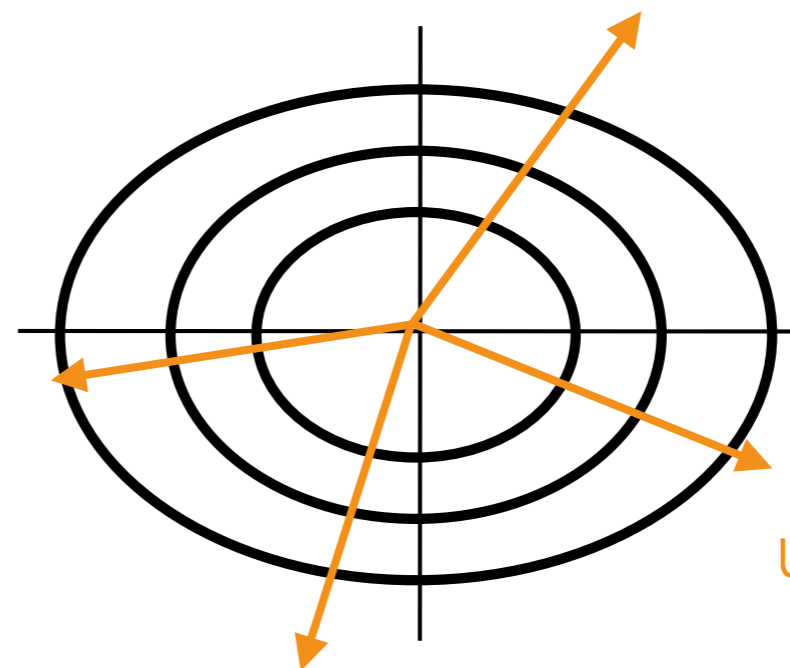
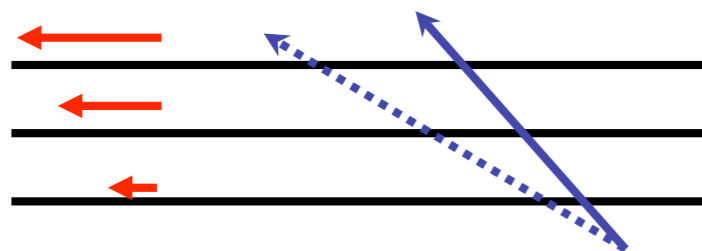
- ▶ What can we do ?

Enemy No. 4: misalignment

- ▶ weak modes that leave the χ^2 of a track invariant



telescope

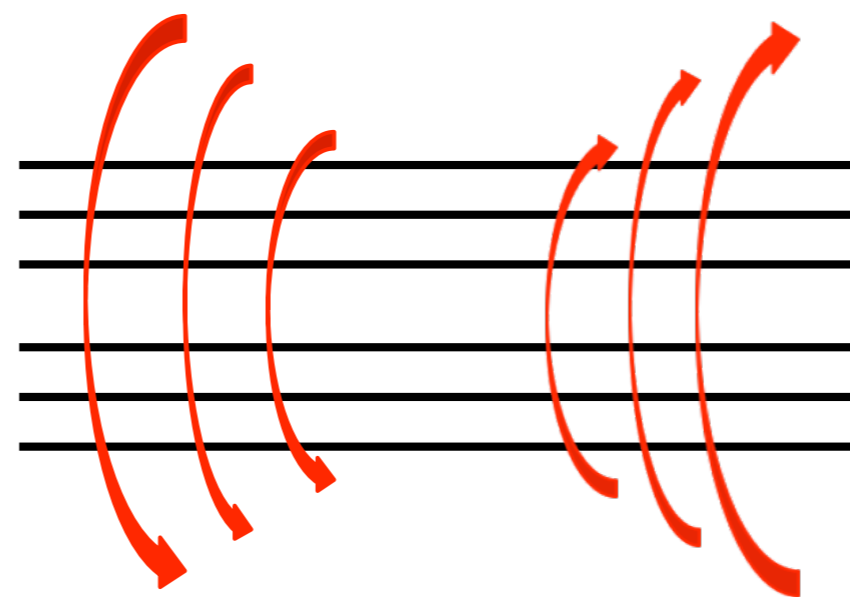
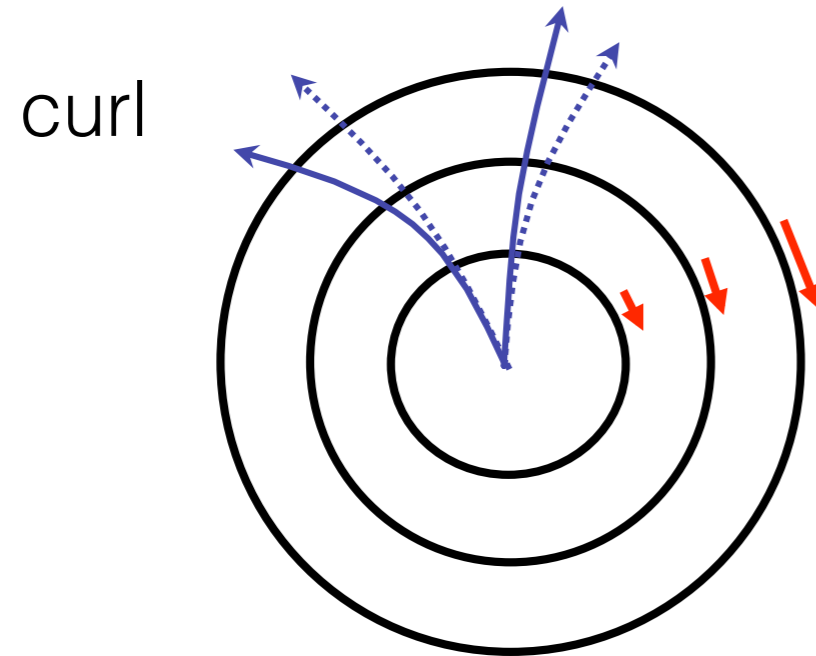


use field-off data

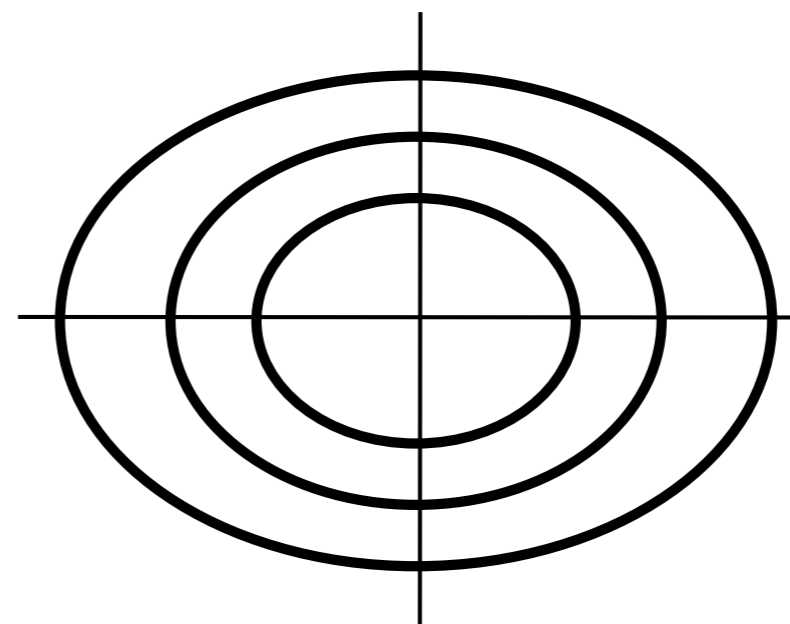
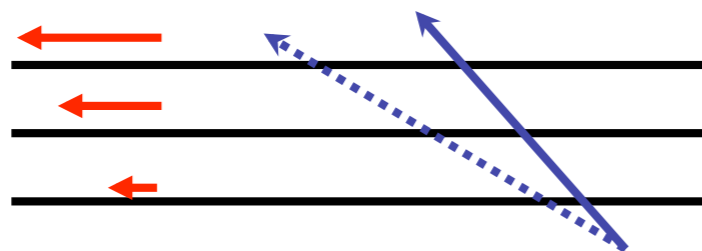
- ▶ What can we do ?

Enemy No. 4: misalignment

- ▶ weak modes that leave the χ^2 of a track invariant



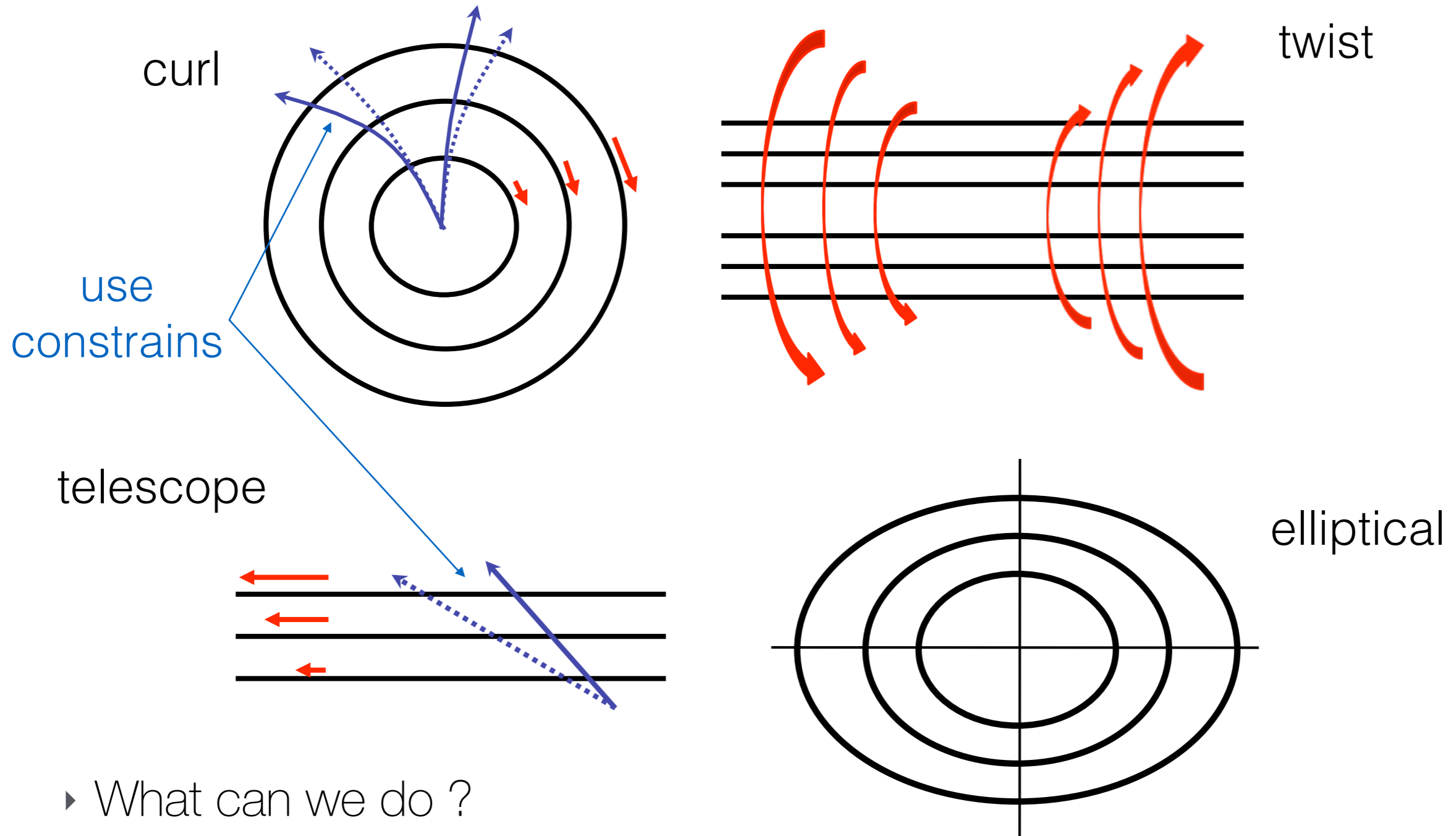
telescope



- ▶ What can we do ?

Enemy No. 4: misalignment

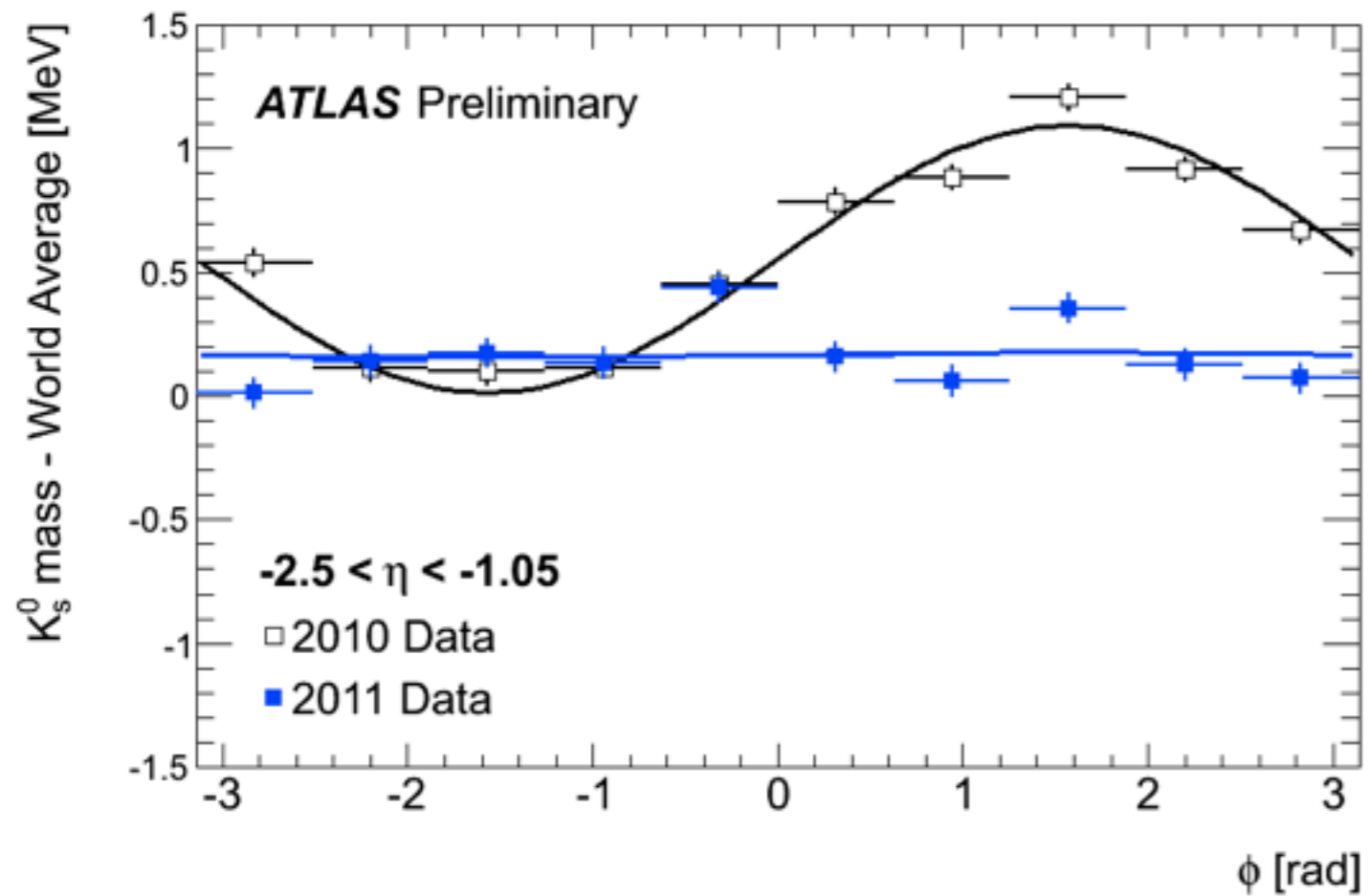
- ▶ weak modes that leave the χ^2 of a track invariant



- ▶ What can we do ?

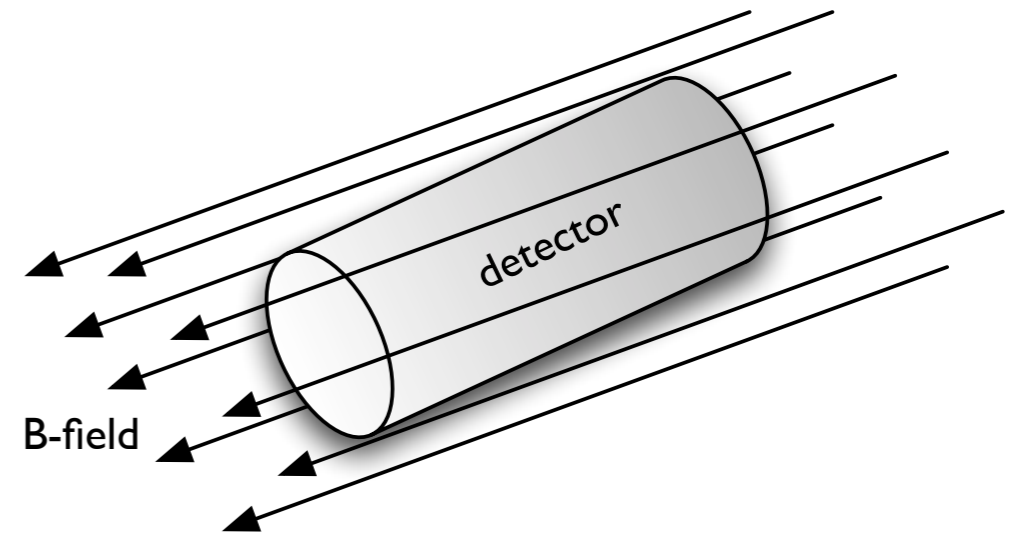
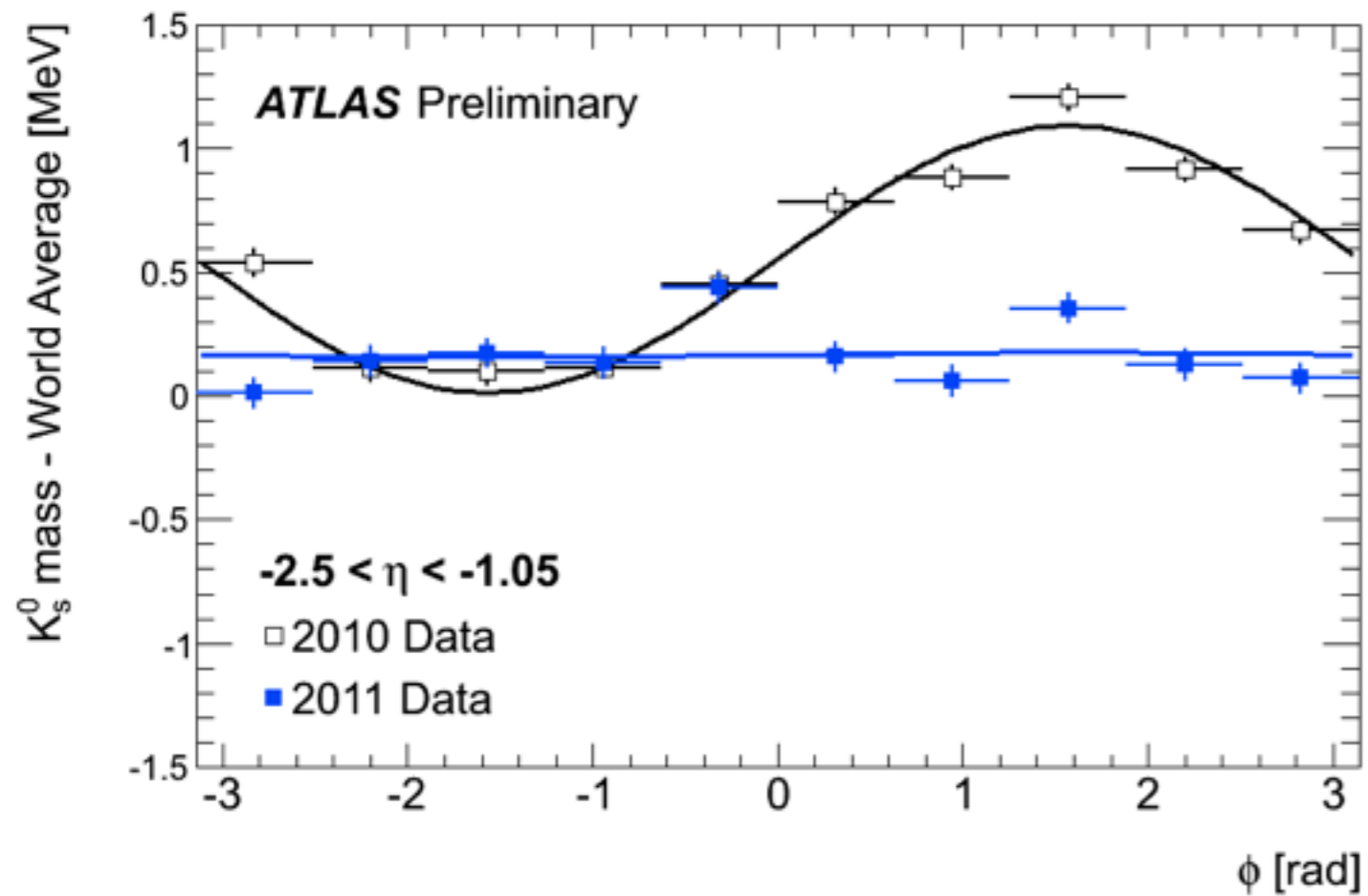
Calibrate your detector

- ▶ use known resonances to calibrate your detector



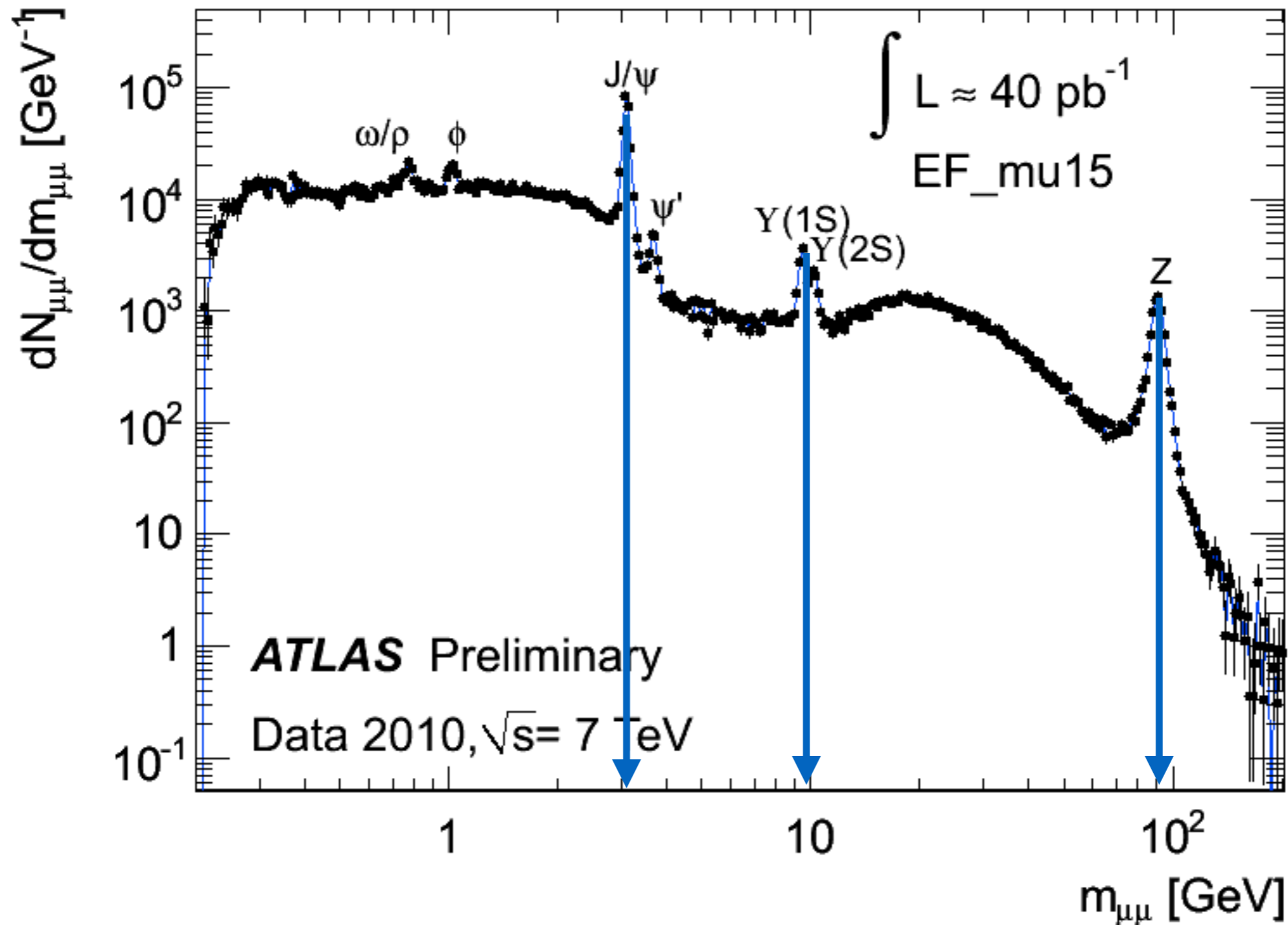
Calibrate your detector

- ▶ use known resonances to calibrate your detector



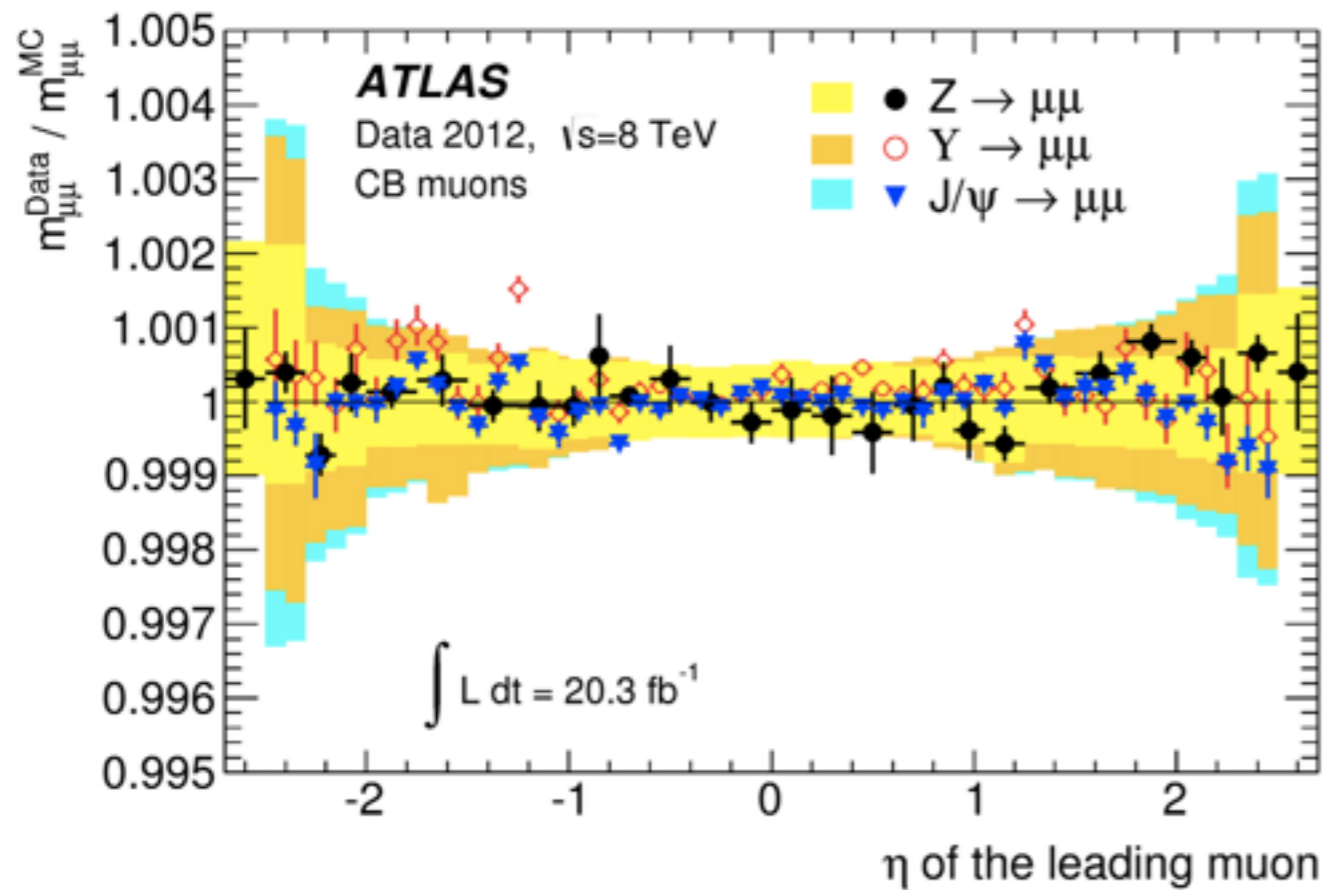
Calibrate your detector

- ▶ use known resonances to calibrate your detector



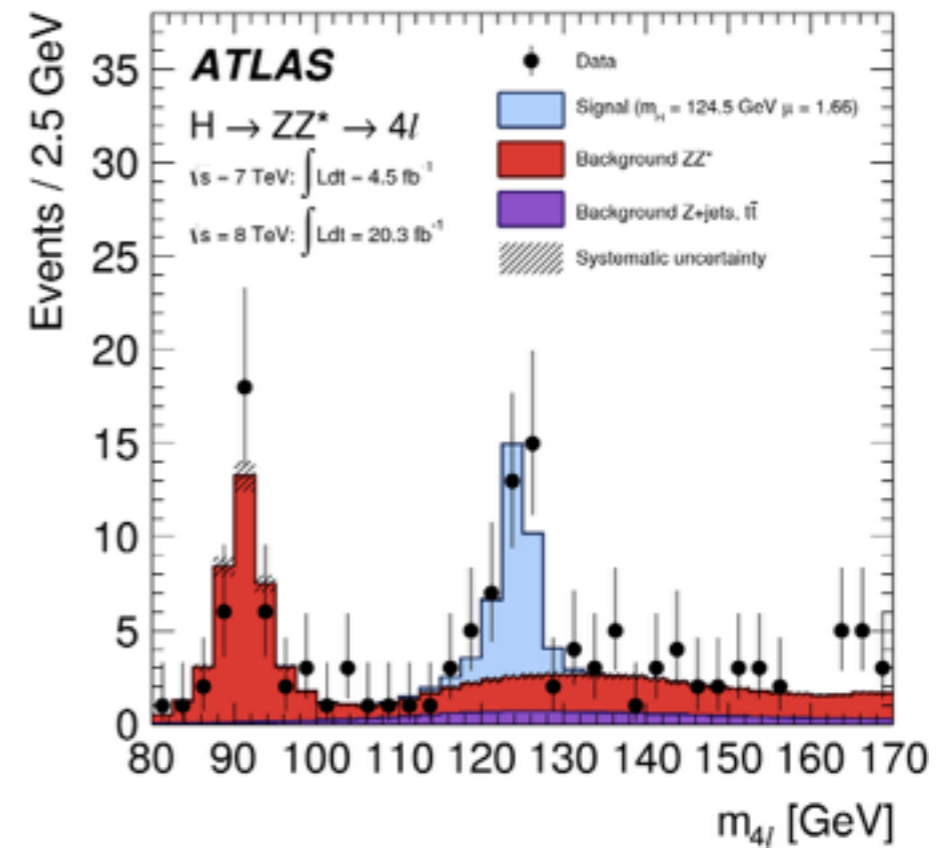
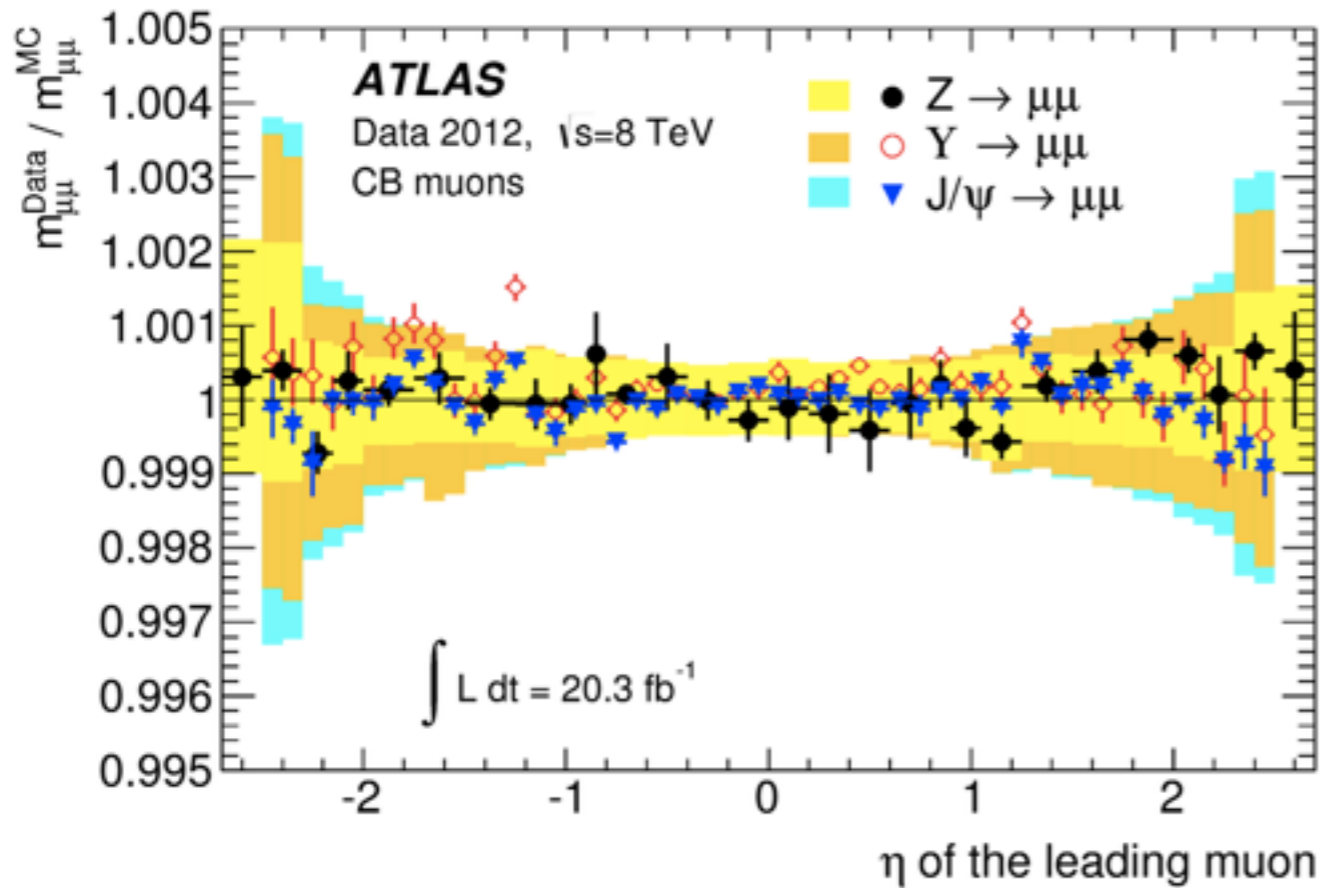
Example: Higgs mass from $H \rightarrow ZZ^* \rightarrow 4\mu$

- ▶ Let's finish our analysis and get out of there ...



Example: Higgs mass from $H \rightarrow ZZ^* \rightarrow 4\mu$

- ▶ Let's finish our analysis and get out of there ...



The take aways

- ▶ Actually, you should fill this page !

What I could not cover ...

- ▶ A lot ...
- ▶ Track/vertex finding: there are way more methods
- ▶ Track fitting: there are way more methods
- ▶ Track reconstruction in very dense environments
 - e.g. in core of very dense jets
- ▶ Combined reconstruction
 - combined μ fitting, decay chain fitting, constraint fitting ...
- ▶ Computing aspect
 - CPU time optimisation (very important for high pile-up)

... and much more

And finally

- ▶ Thanks to you for being such a great audience !
- ▶ To the school organisers and other lecturers to make this such a great school

- ▶ A special thank to Markus Elsing for some of the material.