Towards model-independent event generation for BSM physics with GENIE

John Plows

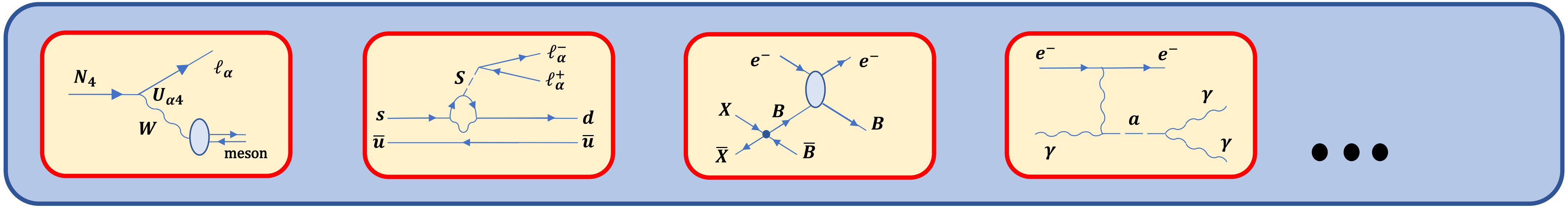
NuFact 2024 - WG5 session: Neutrinos Beyond PMNS

16/Sep/2024

UNIVERSITY OF LIVERPOOL

The problem stated

- Neutrino near detectors have **large exposures**: many events counted → potential to probe many kinds of BSM physics
  - Heavy Neutral Leptons, Higgs Portal Scalars, Boosted Dark Matter, Axion-Like Particles…

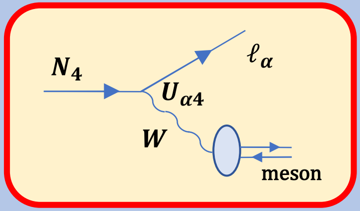


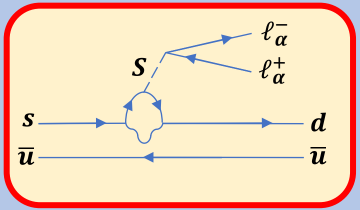
- Neutrino telescopes can also perform searches due to **large size / large number of scattering centres**

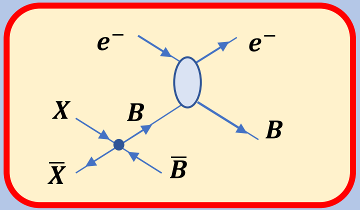
1. How can we maximise the potential for BSM searches?
2. What **inputs** do analyses need to search for anomalies?
3. What **outputs** does theory need to constrain BSM landscape?

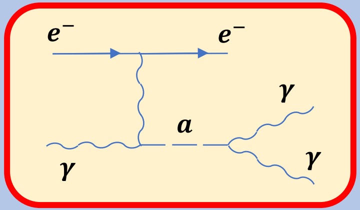
The problem stated

- There are many, **many** models available to look for
- It is not efficient to generate MC samples for all of them…









• • •

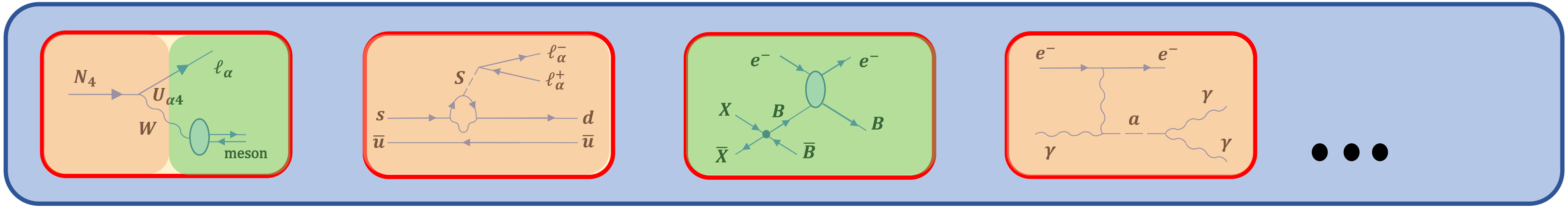
- Different models may **predict similar final states** but with **model-dependent kinematics** ⇒ optimisation problem!
  - Search for **anomalies**, not models (e.g. low-energy excess in LSND / MiniBooNE / MicroBooNE)

JHEP **08** (2023) 092, JHEP **01** (2024) 134, Phys. Rev. **D 106** (2022) 092006

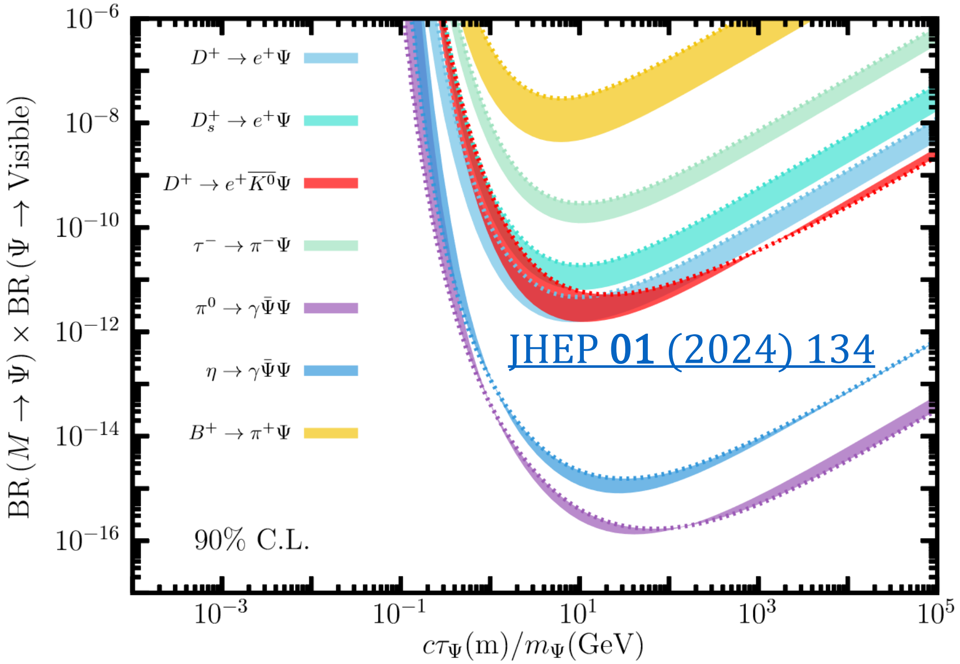
- Solution: **implement model-agnostic frameworks** for event generation
  - “Factor out” the model dependence: simulate signatures in a detector (formulate cuts, obtain efficiencies, reduce background) and weight events according to a model

What do we want to simulate?

- Particles may generally be **stable** or **unstable**
  - Stable particles scatter
  - Unstable particles decay

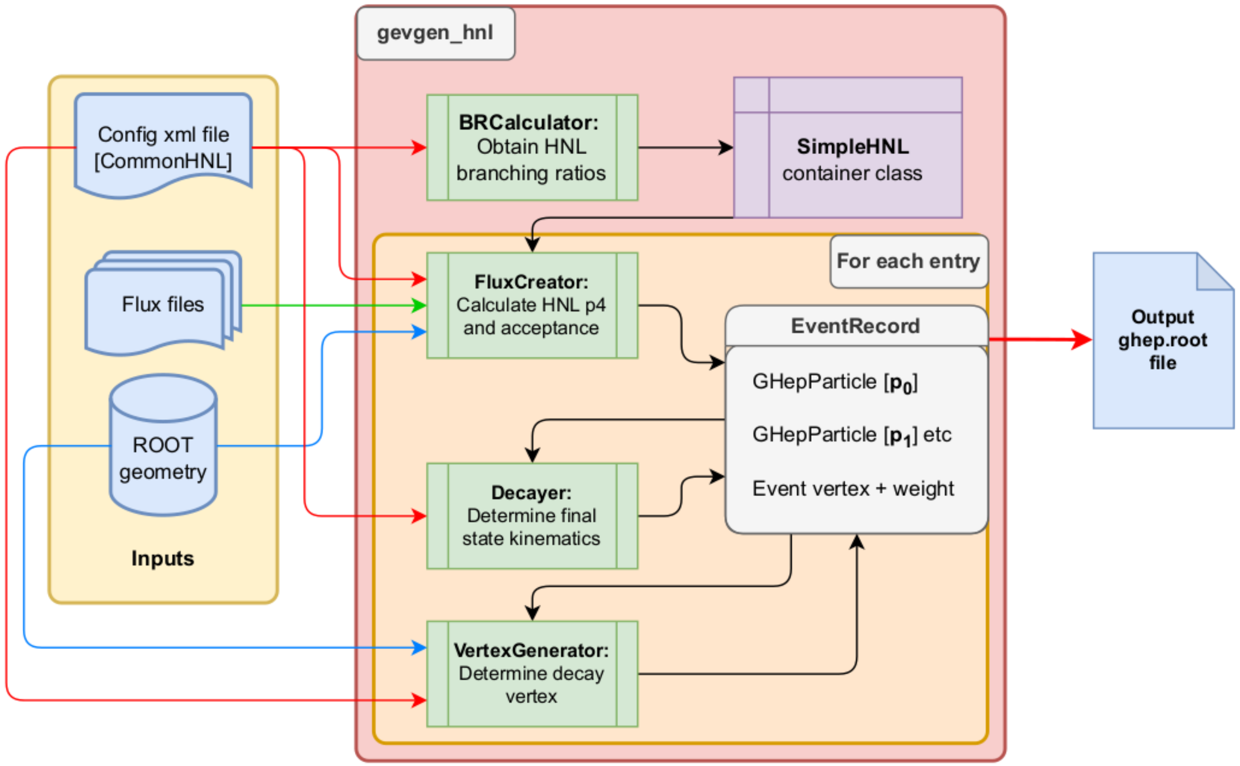


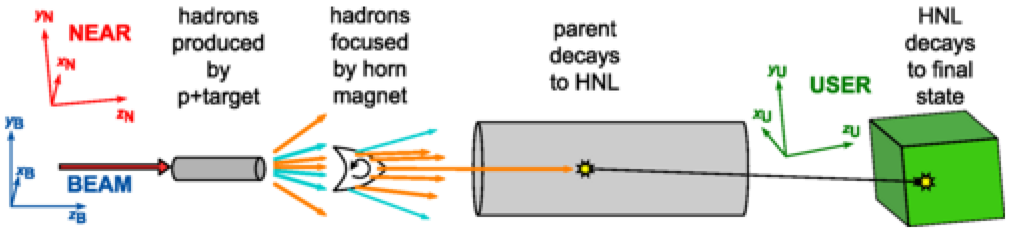
- Can parametrise relevant inputs of both long- and short-lived particles for analyses **but need detailed flux prediction** (kinematics) and **geometry** (spatial / time distribution)
- Focusing on **long-lived particles** (LLPs) first

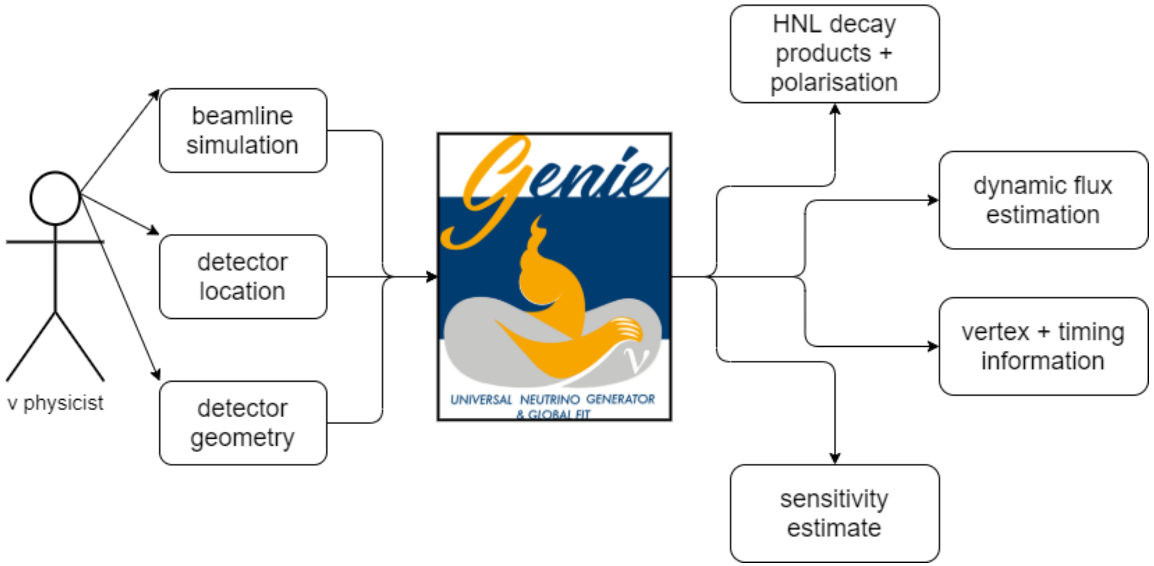


How does this work?

- Similar action to BeamHNL input in current GENIE (Phys. Rev. **D 107** (2023) 055003)
- **Upgrades** to visible-channel construction + geom calculations + output handling







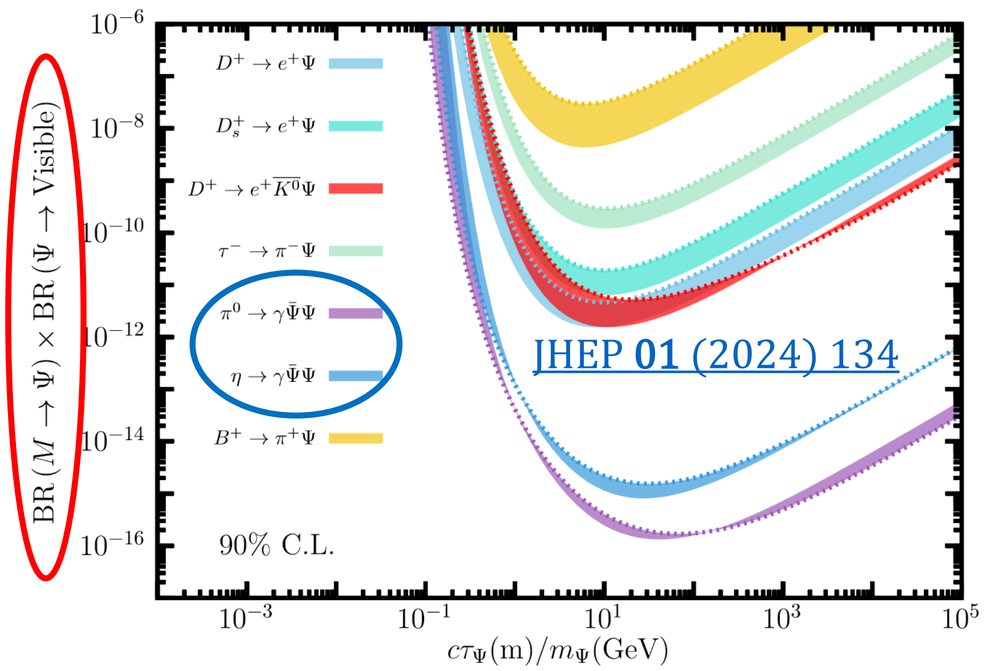
Constructing visible channels

- Considering particle decays
- One wants to parametrise the sensitivity with respect to the product

$$BR(prod)\ x\ BR(decay)$$

- Want to provide user with flexibility to “mix and match” different channels (or come up with new ones!)
- XML-based interface to construct channels

- **TODO**: Implement > 1 LLP in same decay



```xml
<!-- Specify the LLP masses to interpolate scores -->
<param type="vec-double" name="Masses" delim=";">
  0.000; 0.001; 0.005; 0.010; 0.030;
  0.050; 0.075; 0.100; 0.200; 0.300;
  0.500; 0.750; 1.000; 1.250; 1.500
</param>

<!-- D+ -> X + e+ -->
<modeObject name="DPToEP">
  <param type="vec-double" name="Scores" delim=";">
    0.100; 0.101; 0.105; 0.120; 0.160;
    0.200; 0.433; 0.550; 0.550; 0.600;
    0.750; 0.800; 0.850; 0.900; 0.950
  </param>

  <!-- Always specify parent first -->
  <param type="vec-int" name="PDGList" delim=";">
    411 ; 2000022022 ; -11
  </param>
</modeObject>
```

User passes series of `modeObject` s

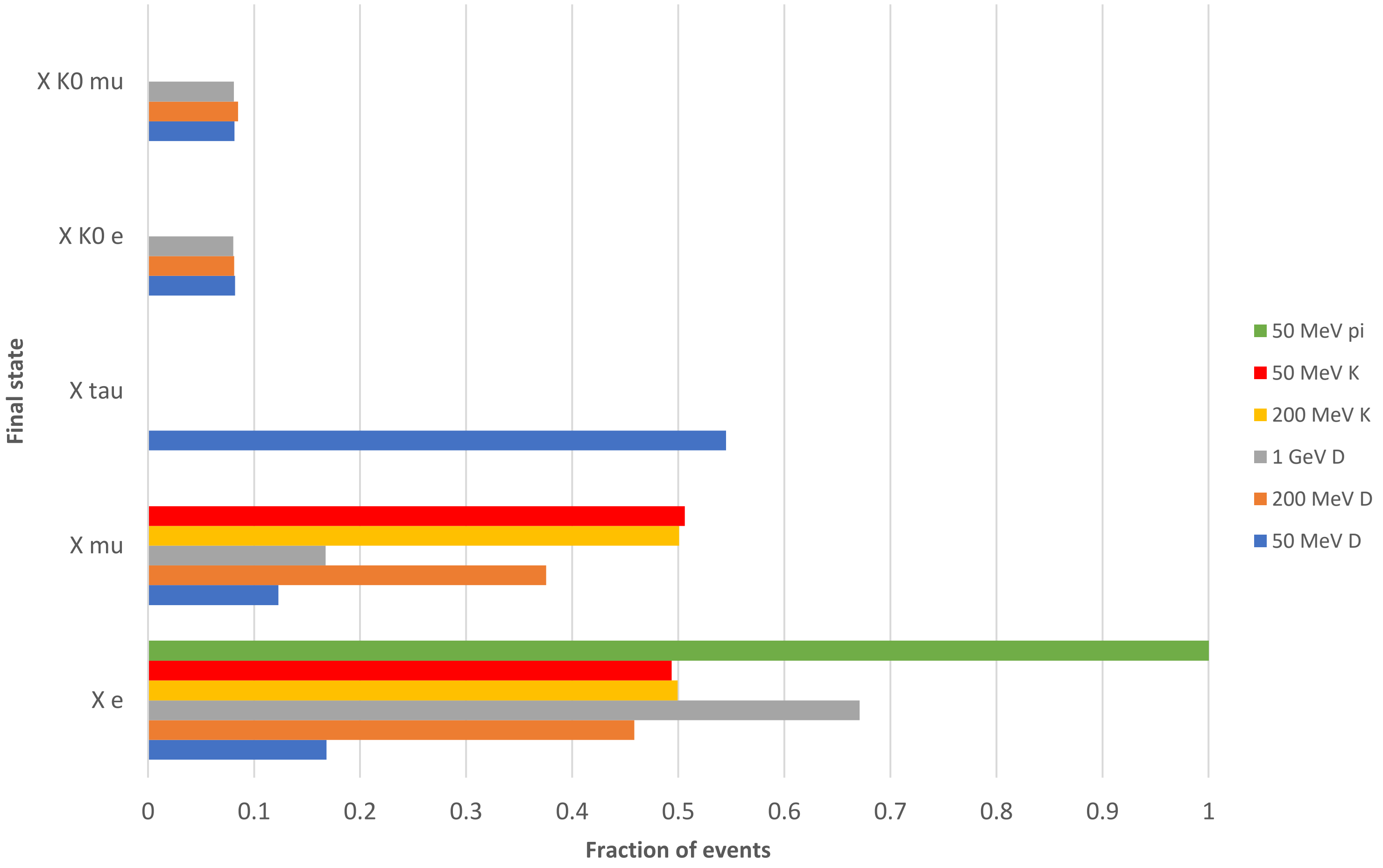
Each object specifies **one decay mode**

It consists of **two vectors**:

1. **Scores**: at each reference mass, the score is the conditional probability that a parent of the specified type will decay via this mode
2. **PDG list**: specifies the particles of the decay

File lives in dedicated LLP config directory for ease of access

$$\text{score}\ (K \to Xe) = \text{score}\ (K \to X\mu)$$



D scores:

a/a	50 MeV	200 MeV	1 GeV
$D \to X\ e$	0.2	0.55	0.85
$D \to X\ \mu$	0.15	0.45	0.15
$D \to X\ \tau$	0.65	0	0
$D \to X\ K^0\ e$	0.1	0.1	0.1
$D \to X\ K^0\ \mu$	0.1	0.1	0.1

D conditional probabilities:

a/a	50 MeV	200 MeV	1 GeV
$D \to X\ e$	0.167	0.458	0.708
$D \to X\ \mu$	0.125	0.375	0.125
$D \to X\ \tau$	0.542	0	0
$D \to X\ K^0\ e$	0.083	0.083	0.083
$D \to X\ K^0\ \mu$	0.083	0.083	0.083

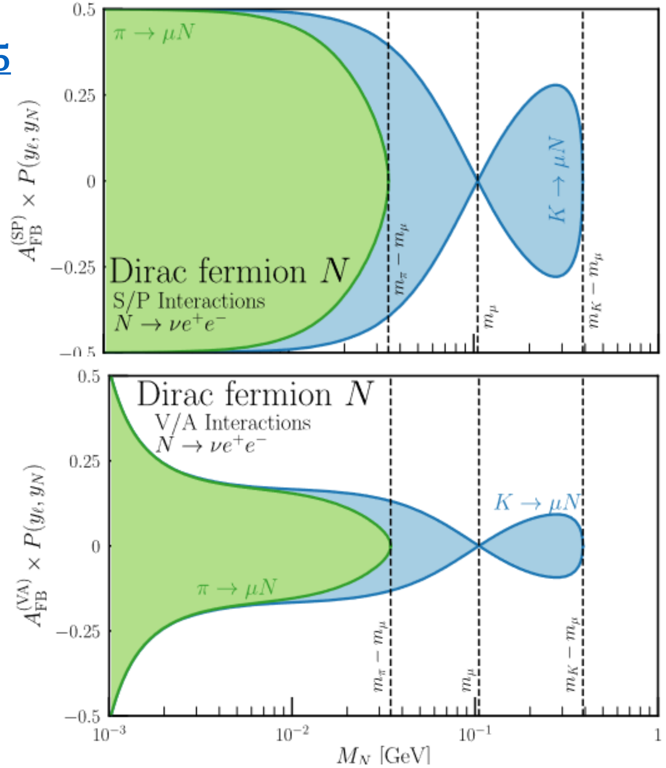
Visible channel kinematics

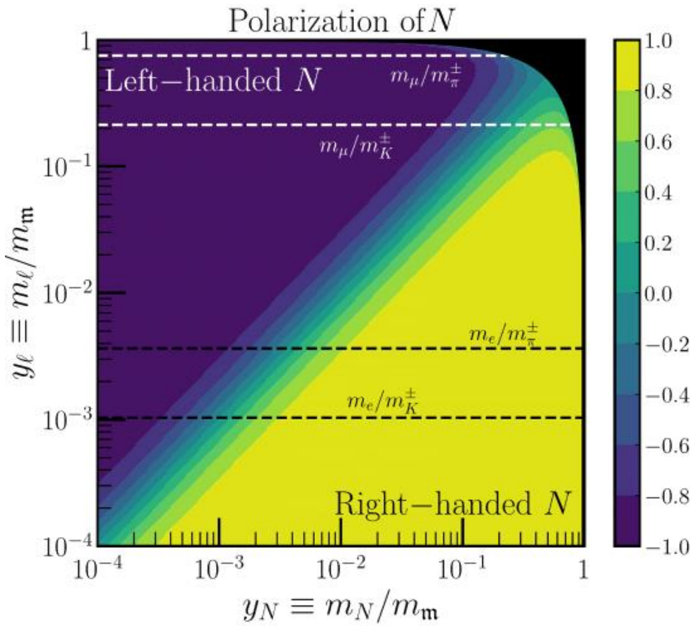
[Phys. Rev. D 105 (2022) 015019]

- Model-independent approach means **we have to scale back assumptions about polarisation** (what is the spin of the particle we are producing / decaying?)
- Can circumvent this by **saving the full particle stack** for LLP production and decay
  - Full information needed for calculation of matrix elements

```
root [1] gRooTracker->GetEntries()
(long long) 1
root [2] gRooTracker->Scan("LLP_production_pdgs:LLP_production_p4xs:LLP_production_p
************************************************************************************
*    Row   * Instance * LLP_produ * LLP_produ * LLP_produ * LLP_produ * LLP_produ *
************************************************************************************
*        0 *        0 *       411 *         0 *         0 *         0 *   1.86966 *
*        0 *        1 * 2.000e+09 * 0.0090112 * -0.037748 * 0.6606357 * 1.1991437 *
*        0 *        2 *       -13 * -0.009011 * 0.0377484 * -0.660635 * 0.6701562 *
************************************************************************************
(long long) 3
```

$D^+ \to X + \mu^+$ in D rest frame

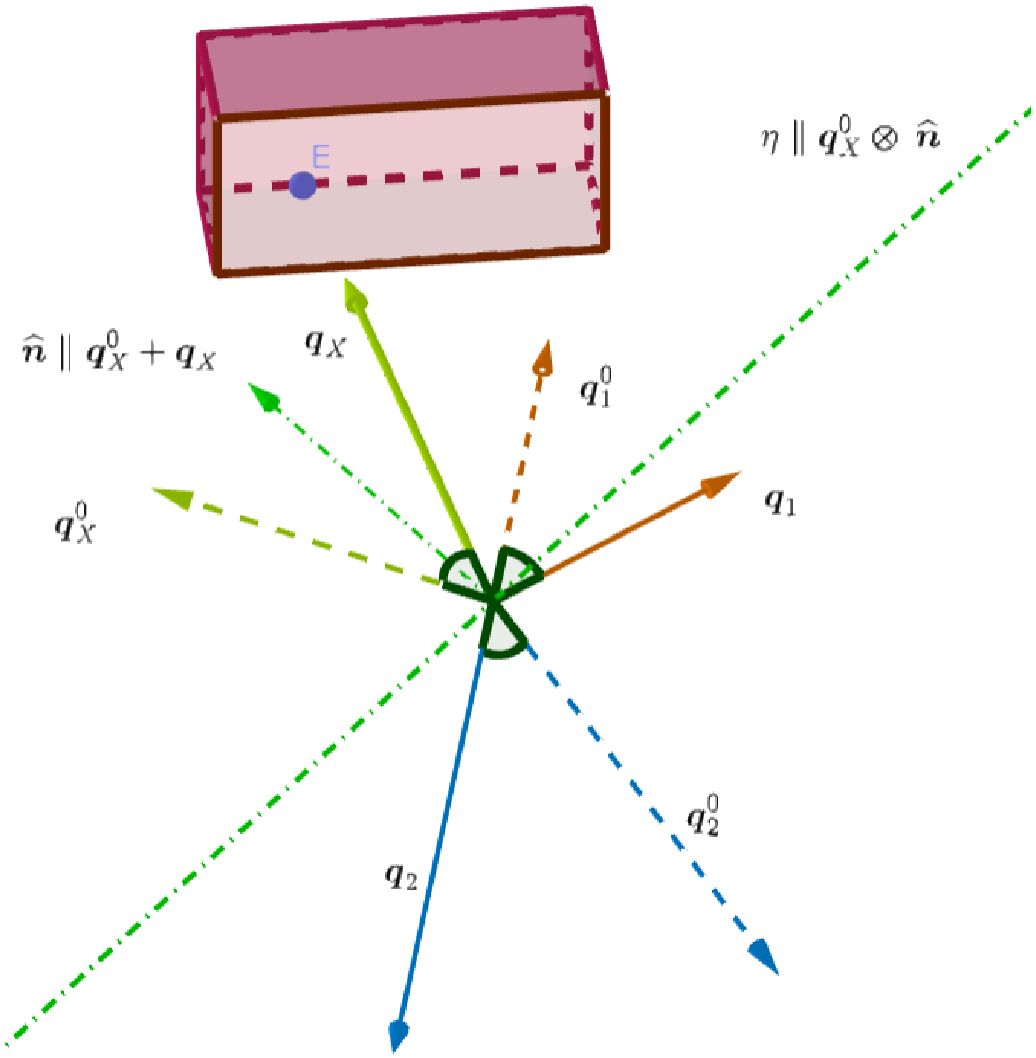




Applying the acceptance constraint

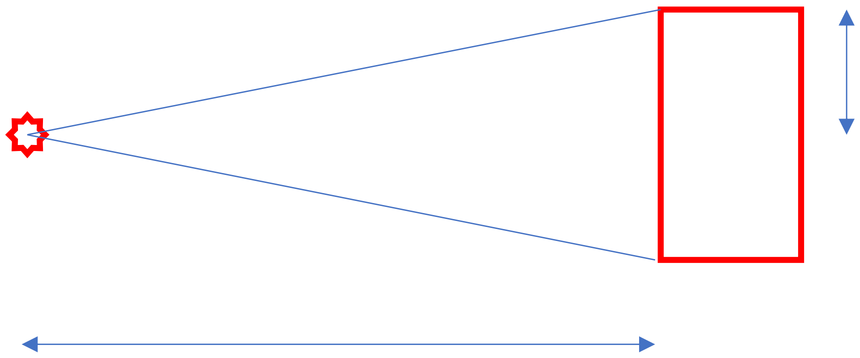
- The LLP production system **such that the LLP is accepted** should be saved
  - This is not a random decay… It is constrained by acceptance

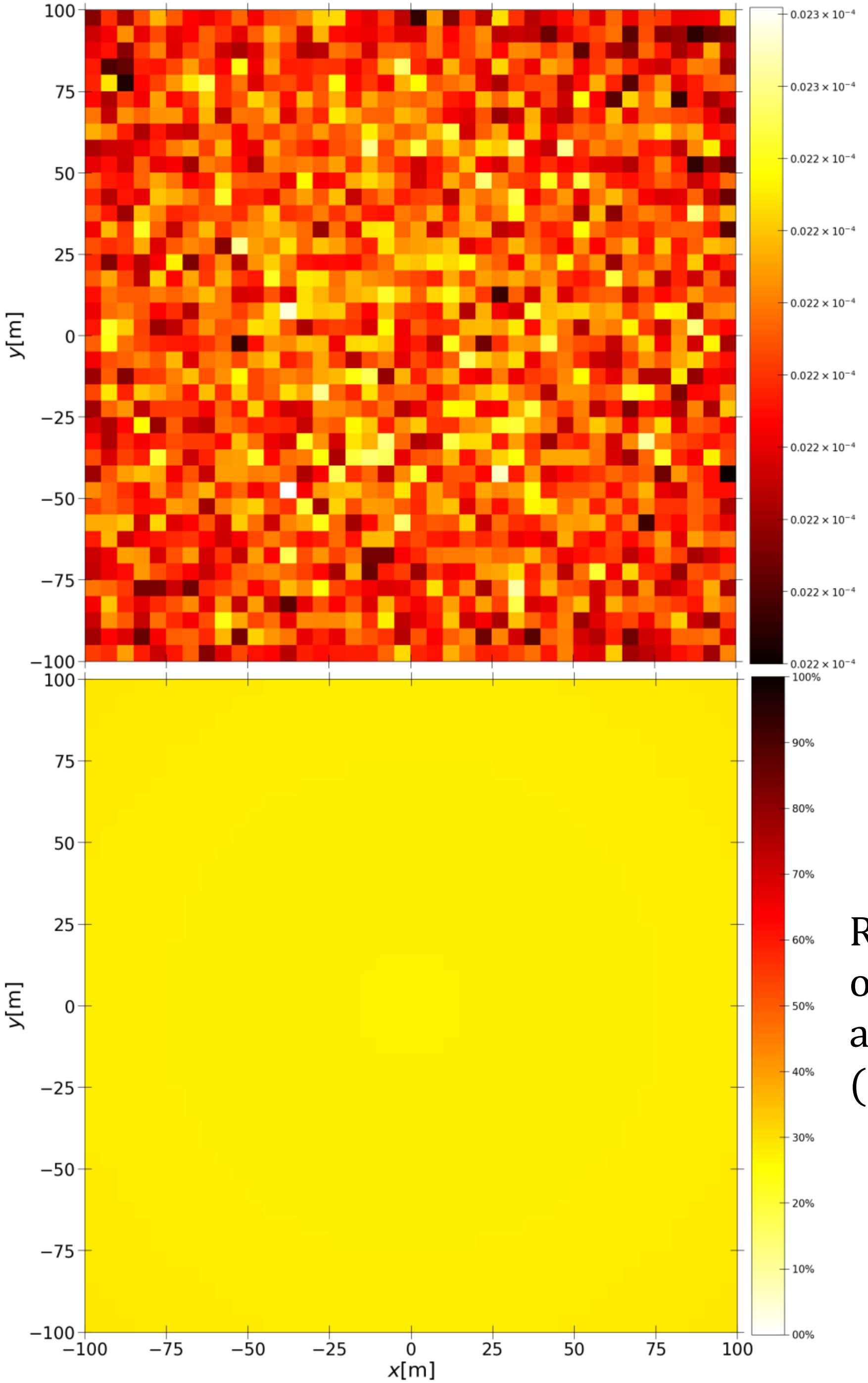
- Solution:
  - First throw random phase-space decay to get energies of channel in parent rest frame and save random momentum $\boldsymbol{q}_X^0$
  - Apply geometrical constraints to get lab-frame $\boldsymbol{p}_X$
  - Boost back by $\Lambda(\boldsymbol{\beta}_{\text{parent}})$ to obtain rest-frame $\boldsymbol{q}_X$
  - Calculate quaternion ⇔ rotate $\boldsymbol{q}_X^0 \leftrightarrow \boldsymbol{q}_X$ and apply to all the particles in the system



Geometry calculations

- Important to know **how large angular acceptance of detector is**
  - Normally handled by “small angle approximation”
  - Assume detector has small transverse size compared to baseline
  - Assume front face of detector parallel to z axis



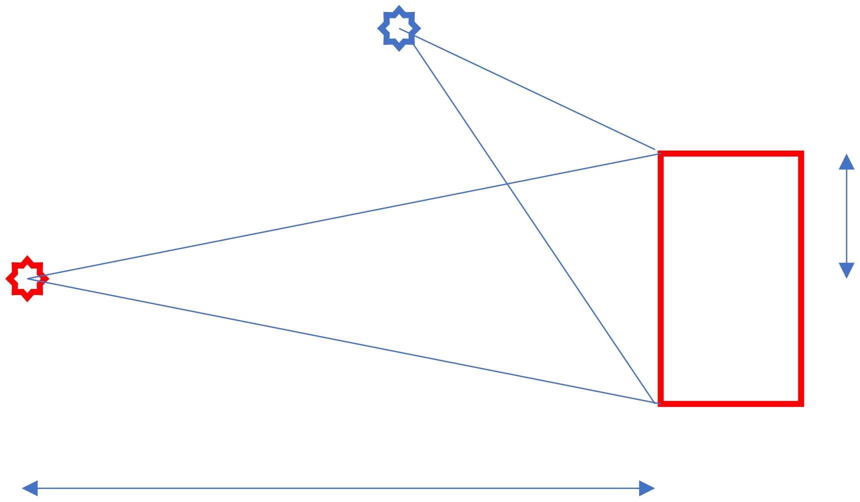


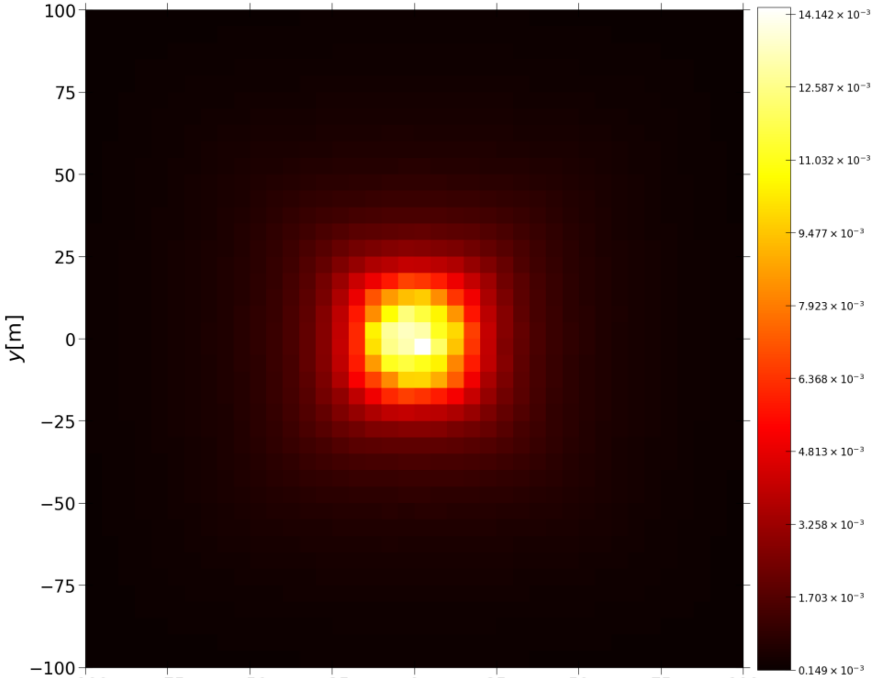
Angular acceptance / 4π $(0.9 \leq d\ [km] \leq 1)$

Relative difference of small-angle from analytical calc $(0.9 \leq d\ [km] \leq 1)$

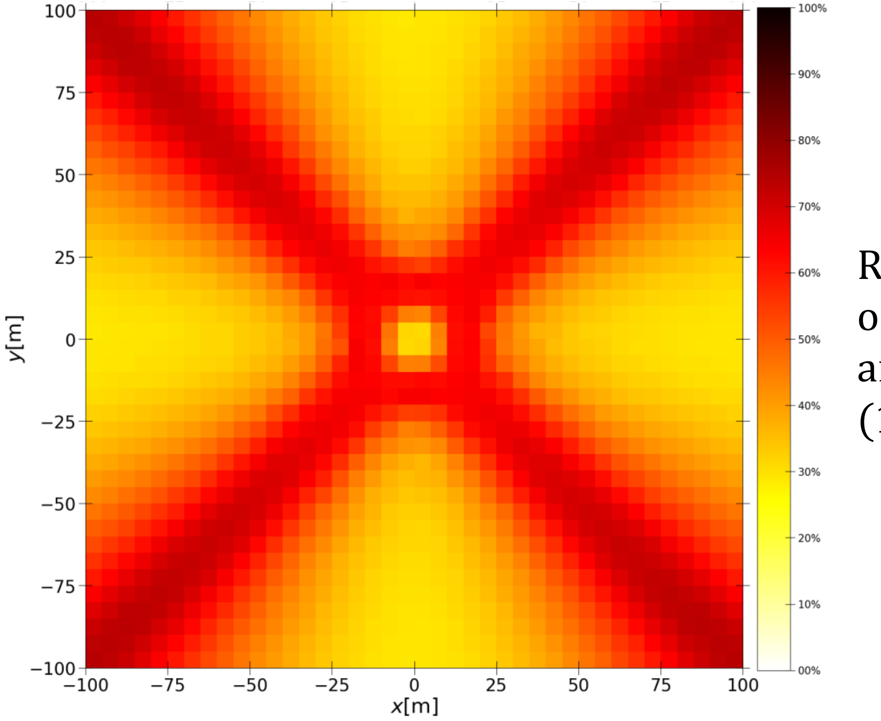
Geometry calculations

- Important to know **how large angular acceptance of detector is**
  - Normally handled by "small angle approximation"
  - Assume detector has small transverse size compared to baseline
  - Assume front face of detector parallel to z axis
  - Not necessarily valid at large viewing angles!

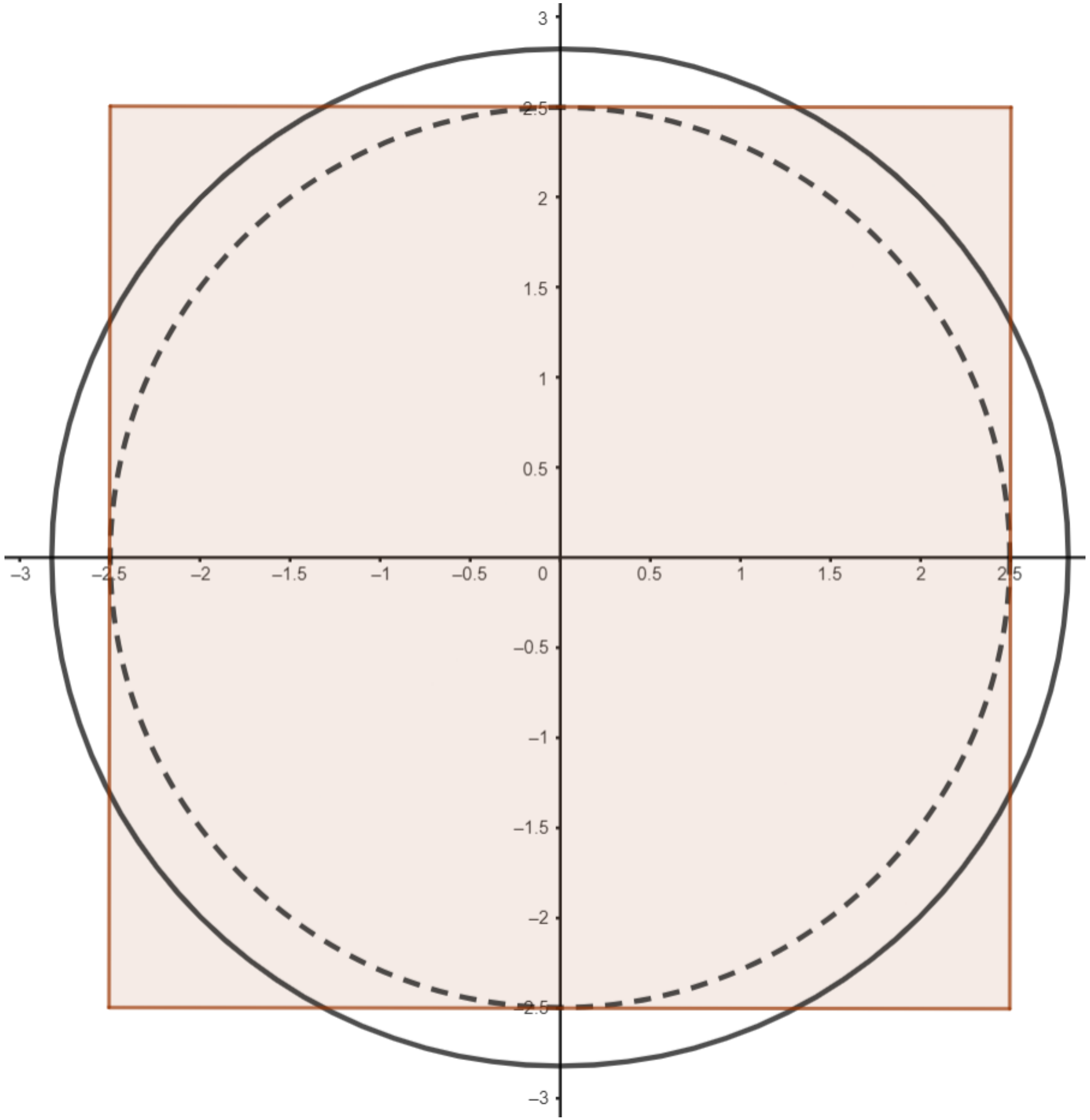




Angular acceptance / 4π ($10 \leq d\ [m] \leq 20$)



Relative difference of small-angle from analytical calc ($10 \leq d\ [m] \leq 20$)

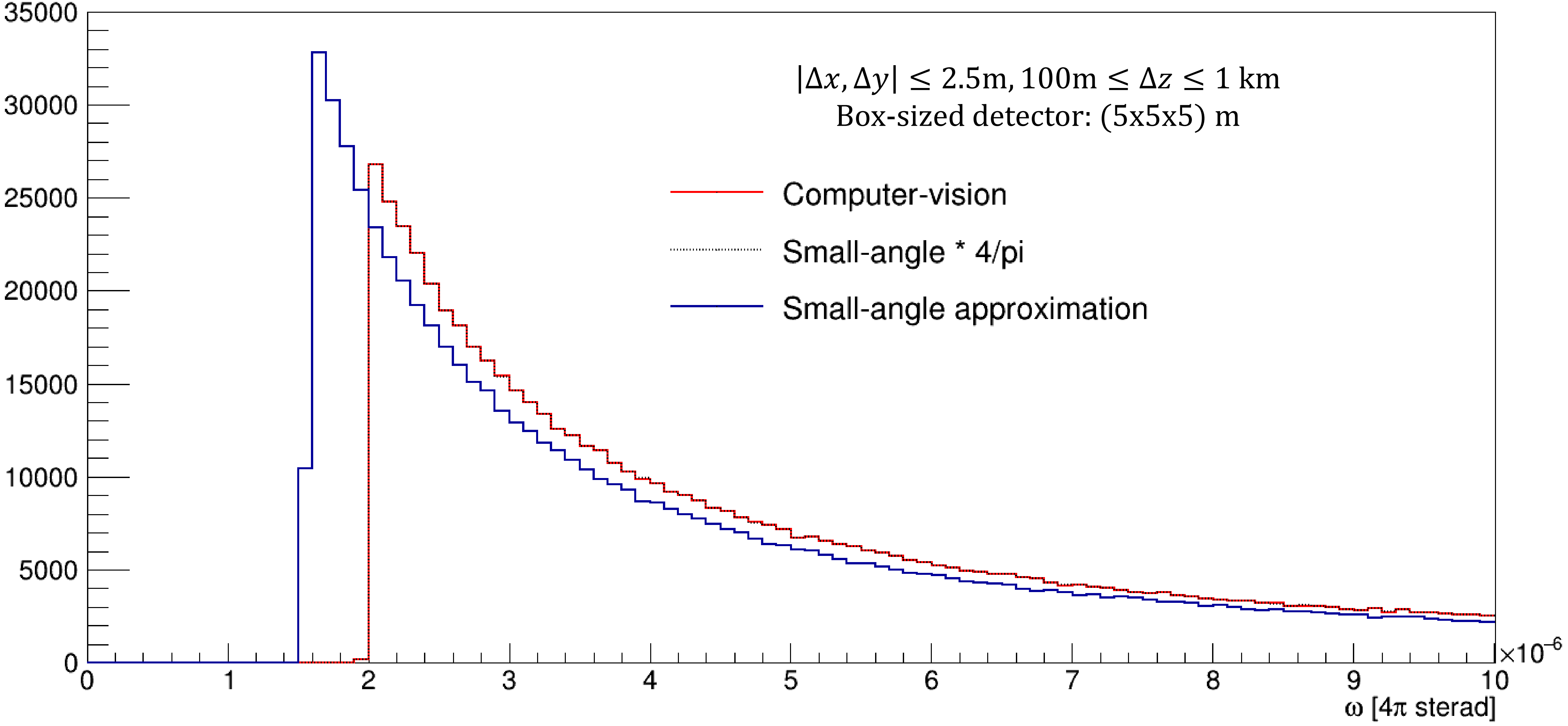


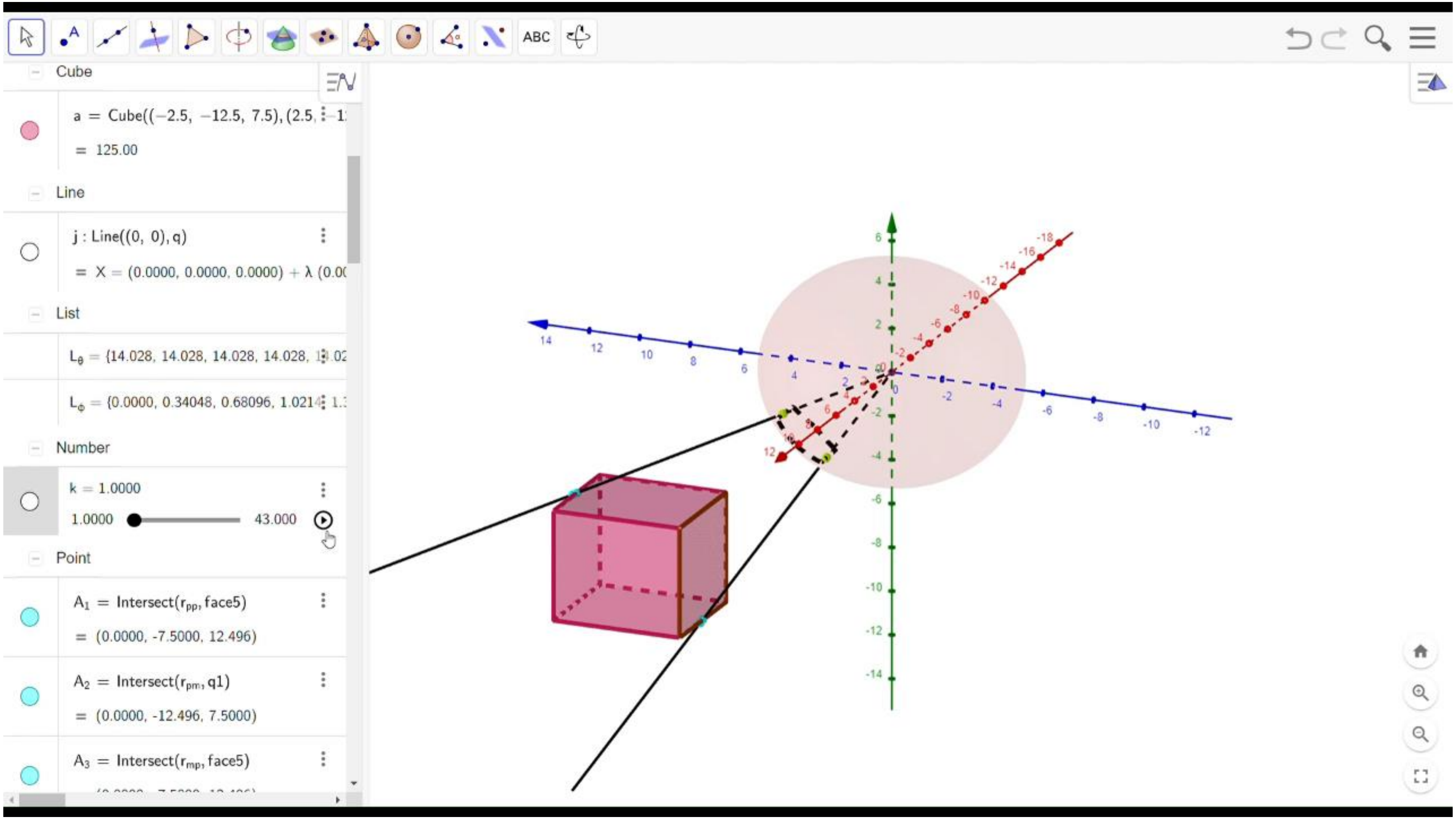
Why does this happen?

In prior calculations (BeamHNL) flux was calculated based on illuminated area $= 1\text{m}^2$ with **face normal to z**

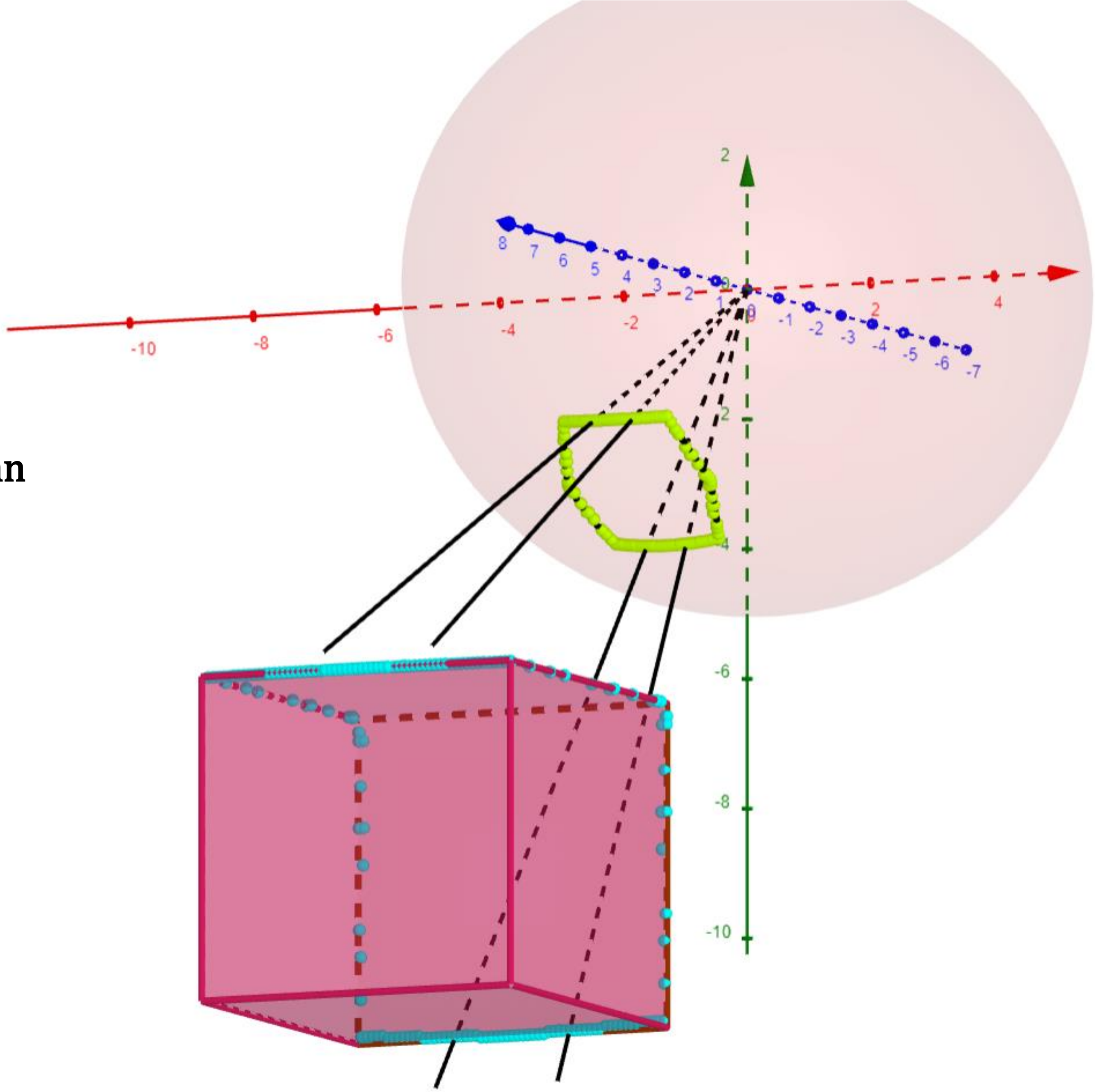
- Instead of scaling to detector element size:
- Use detector volume bounding box (can be your entire detector, or subdetector…) to estimate angle
- “Small-angle” uses the inscribed circle: **needs to be corrected** by factor $4/\pi$

Is that all there is to it?


$|\Delta x, \Delta y| \leq 2.5\text{m}, 100\text{m} \leq \Delta z \leq 1\text{ km}$
Box-sized detector: (5x5x5) m
Computer-vision
Small-angle * 4/pi
Small-angle approximation
35000
30000
25000
20000
15000
10000
5000
0
0
1
2
3
4
5
6
7
8
9
10
$\times 10^{-6}$
ω [4π sterad]

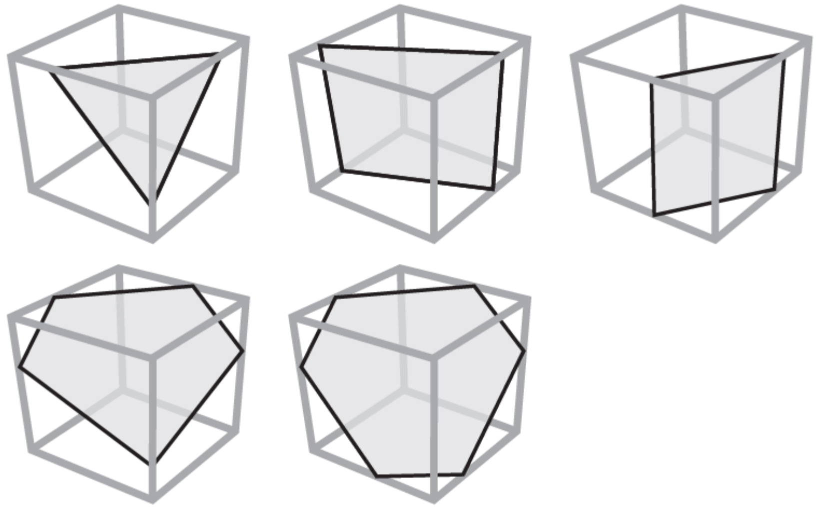

ABC
Cube
a = Cube((−2.5, −12.5, 7.5), (2.5,
= 125.00
Line
j : Line((0, 0), q)
= X = (0.0000, 0.0000, 0.0000) + λ (0.0
List
Lθ = {14.028, 14.028, 14.028, 14.028,
Lφ = {0.0000, 0.34048, 0.68096, 1.021
Number
k = 1.0000
1.0000
43.000
Point
A1 = Intersect(rpp, face5)
= (0.0000, -7.5000, 12.496)
A2 = Intersect(rpm, q1)
= (0.0000, -12.496, 7.5000)
A3 = Intersect(rmp, face5)

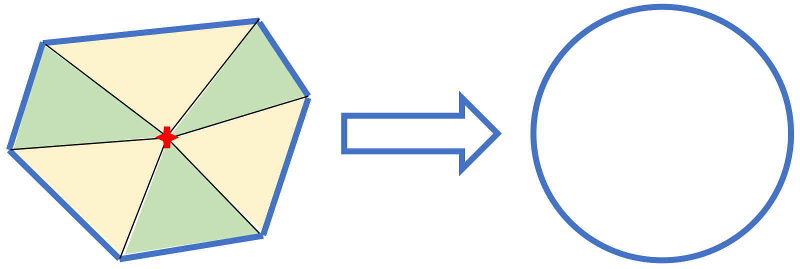
Large viewing angles **can distort shapes**



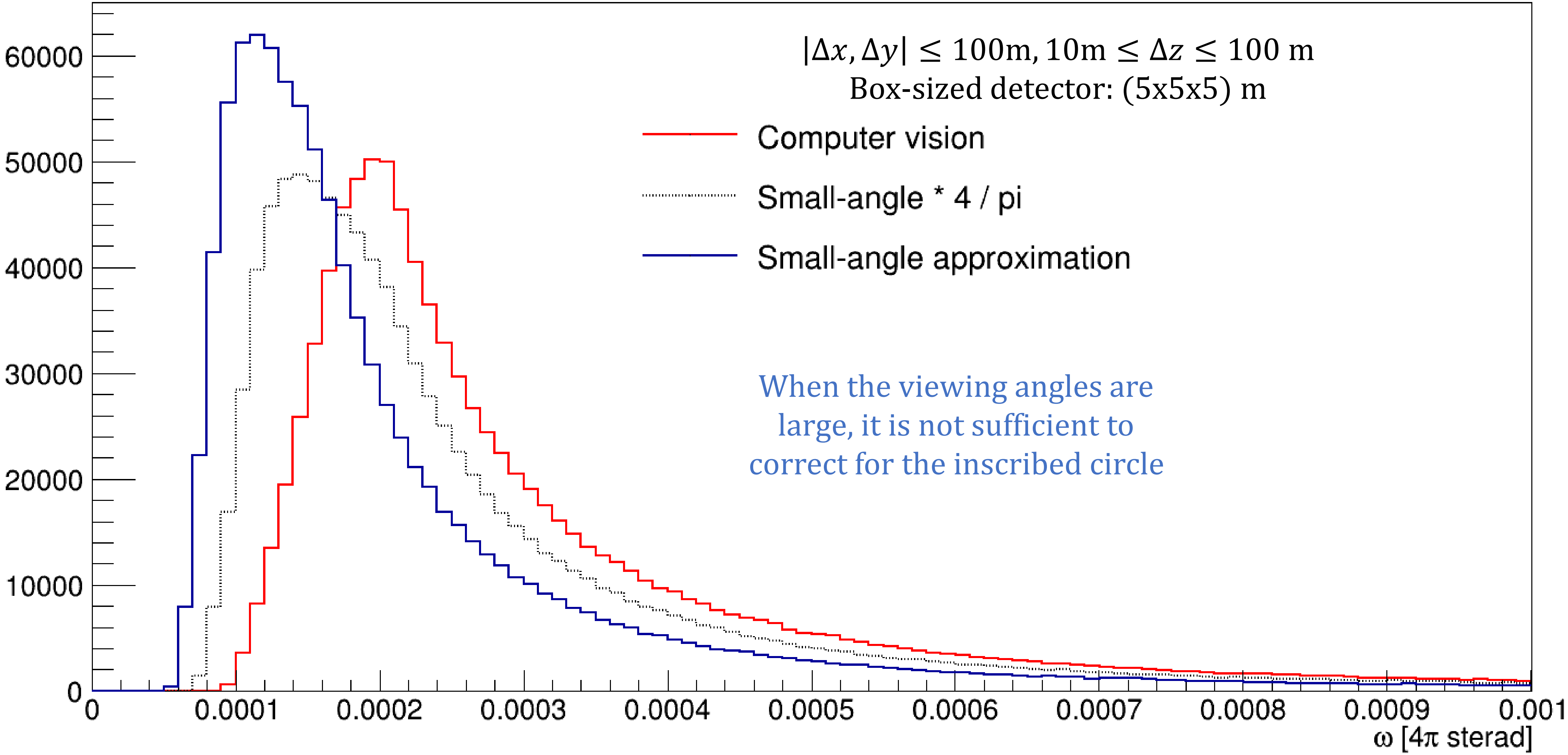
Geometry improvement

- Algorithm to calculate generic intersections of viewing plane with a bounding box is known (Rezk Salama and Kolb, 2005)
  - Anywhere between 3 and 6 vertices
- Implemented this "computer-vision" algorithm into GENIE
  - "Small-angle-like" approach: calculate area of polygon and replace with circle of equivalent area
- About O(3) slowdown with respect to small-angle approximation
- Option available to user to switch between the computer-vision algorithm and small-angle approximation

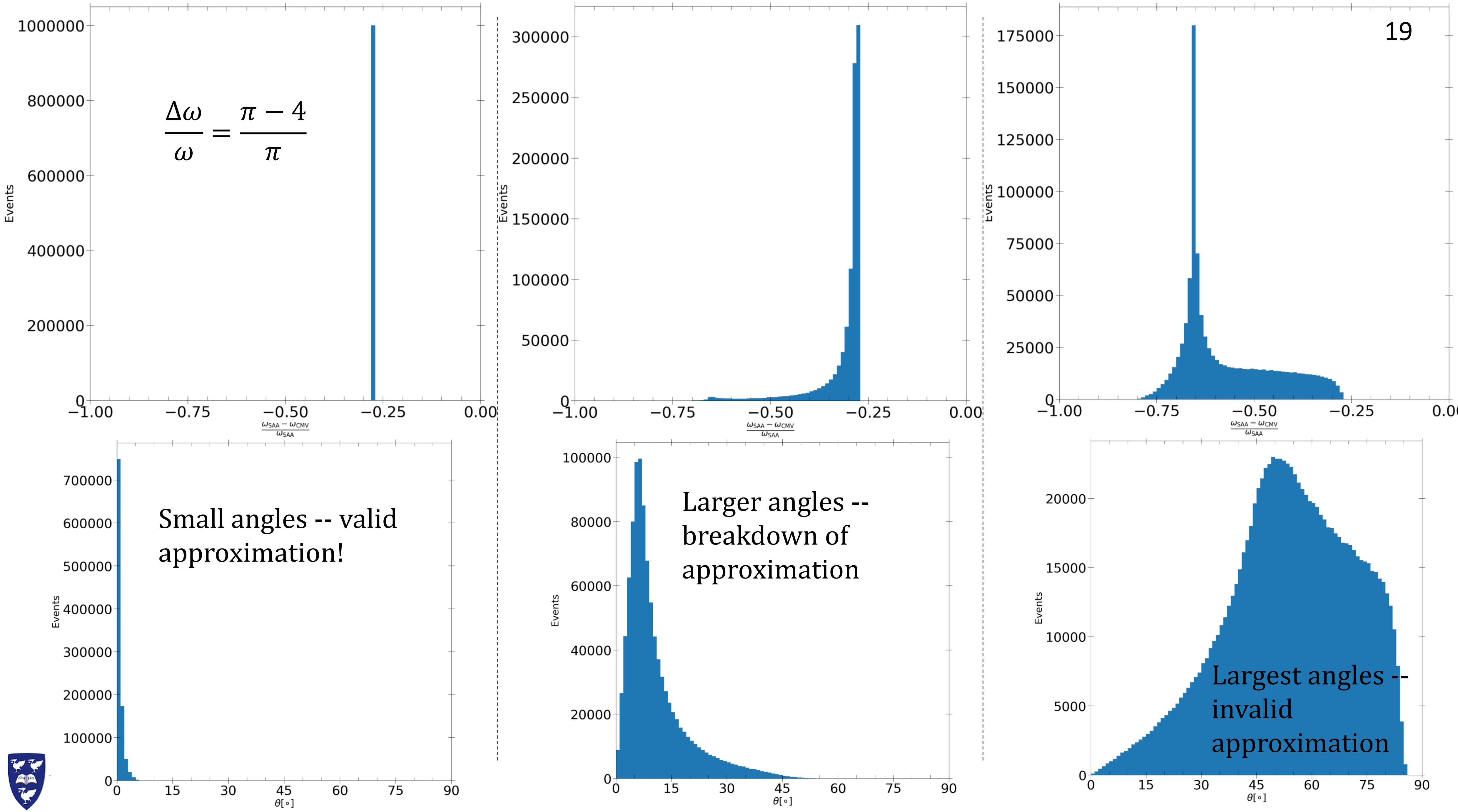




$$A = \Sigma_i A_i \mapsto \theta = \tan^{-1}\left(\frac{A}{\pi}\right) \mapsto \frac{\Omega}{4\pi} = \frac{1}{2}(1 - \cos\theta)$$



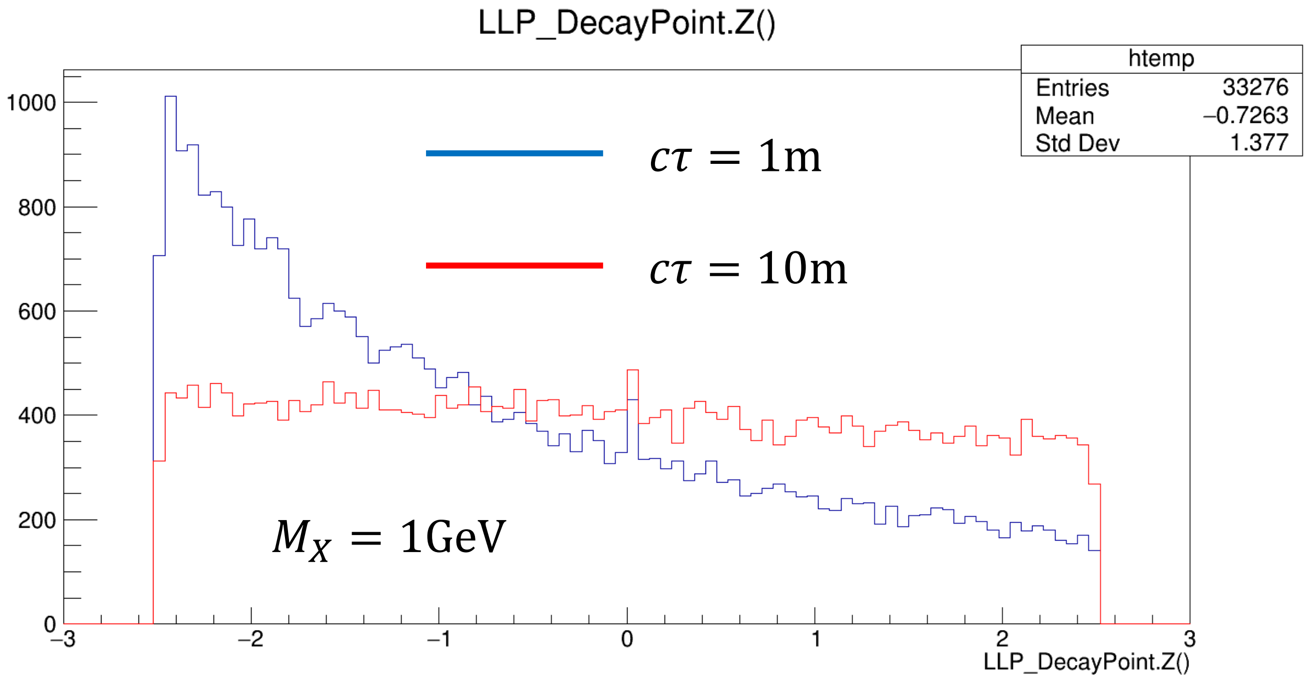
When the viewing angles are large, it is not sufficient to correct for the inscribed circle

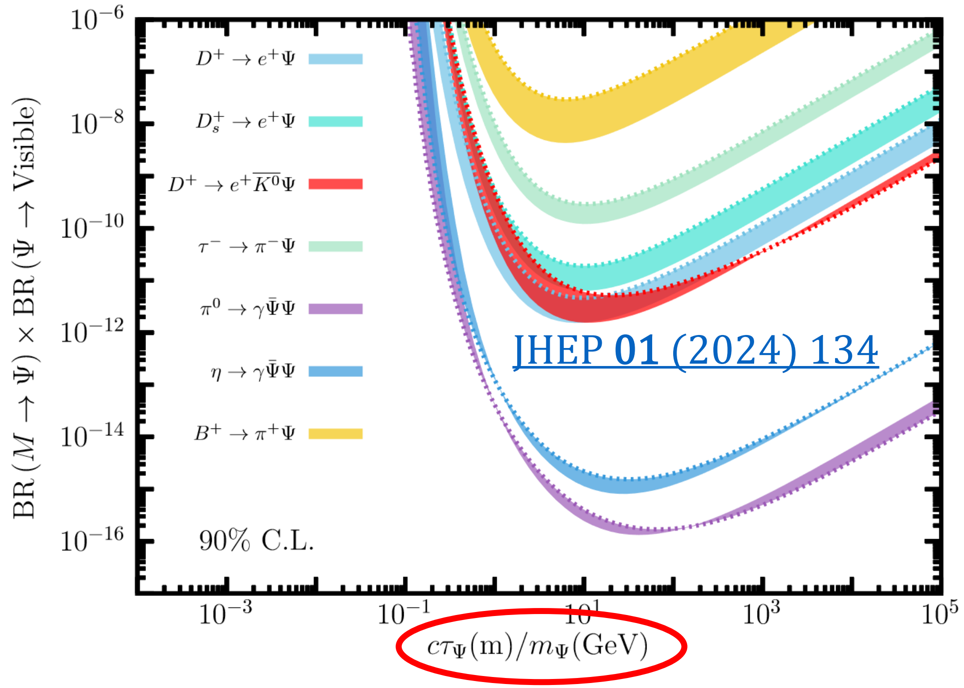

$\frac{\Delta\omega}{\omega} = \frac{\pi - 4}{\pi}$
Events
$\frac{\omega_{SAA} - \omega_{CMV}}{\omega_{SAA}}$
Small angles -- valid approximation!
Larger angles -- breakdown of approximation
Largest angles -- invalid approximation
$\theta[\circ]$

Vertex positioning

- Done in similar way to BeamHNL: force evaluation along a point in the detector top volume (passed as argument), check if acceptable, calculate entry and exit points using ROOT if yes

- **Difference**: lifetime is now explicitly passed as config-level argument





UNIVERSITY OF LIVERPOOL

Output handling

- Too many handles to the conditional probability to pack into a single object
  - Need to know points of entry / exit for survival / decay probabilities…
  - Full kinematics of LLP production channel for polarisations…
  - Collimation-effect weight for the acceptance correction…
  - Branching ratio to production and decay channels…
- Solution: Written as a new class, FluxContainer, which keeps all this information
  - Technically present in BeamHNL but not written in meaningful way to output
  - FluxContainer is written as flat branches to a rootracker tree

```
                 -1, -1, -1
LLP_Mass          = 1
LLP_Lifetime      = 10
LLP_Flux_evtno    = 2
LLP_Parent_pdg    = 411
LLP_Parent_v4_NEAR = (TLorentzVector*)0x5611f30ae010
LLP_Parent_v4     = (TLorentzVector*)0x5611f3101f80
LLP_Parent_p4_NEAR = (TLorentzVector*)0x5611f2da1f50
LLP_Parent_p4     = (TLorentzVector*)0x5611f30eeee0
LLP_p4_NEAR       = (TLorentzVector*)0x5611f30da670
LLP_p4            = (TLorentzVector*)0x5611f30d8f80
LLP_EntryPoint_NEAR = (TLorentzVector*)0x5611f30e73d0
LLP_EntryPoint    = (TLorentzVector*)0x5611f3107a30
LLP_ExitPoint_NEAR = (TLorentzVector*)0x5611f3107ab0
LLP_ExitPoint     = (TLorentzVector*)0x5611f30f2c30
LLP_DecayPoint_NEAR = (TLorentzVector*)0x5611f30fdb70
LLP_DecayPoint    = (TLorentzVector*)0x5611f3113e60
LLP_Wgt_geom      = 3.21994e-06
LLP_Boost_factor = 6.57443
LLP_Wgt_collimation = 1.40078
LLP_Wgt_survival = 0.00013532
LLP_Wgt_detdecay = 0.0620888
LLP_Vertex_rng    = 0.605103
LLP_production_pdgs = (vector<int>*)0x5611f30f3270
LLP_production_p4xs = (vector<double>*)0x5611f30fd160
LLP_production_p4ys = (vector<double>*)0x5611f2dcbce0
LLP_production_p4zs = (vector<double>*)0x5611f30bf200
LLP_production_p4Es = (vector<double>*)0x5611f2db5920
```

Looking forward

- Finalising the architecture but some things need doing still:
  - Double ($n-$ple?) LLP production channels? ← In progress!
  - Any more branches that need to be added to FluxContainer output?
  - Unit tests / validation for computer vision algorithm ← In progress!
  - Implement "polarisation weight" perhaps? Would it be useful to have a way for user to input a formula to modify said weight based on kinematics?

- Code is public! (lives in my personal Generator fork (feature/ExoticLLP) for now, feel free to try it out but **caveat emptor: this is not a validated official release!**)
  - But you'll hear from us when it is official ☺

- **Comments and ideas are welcome!**

Thank you!

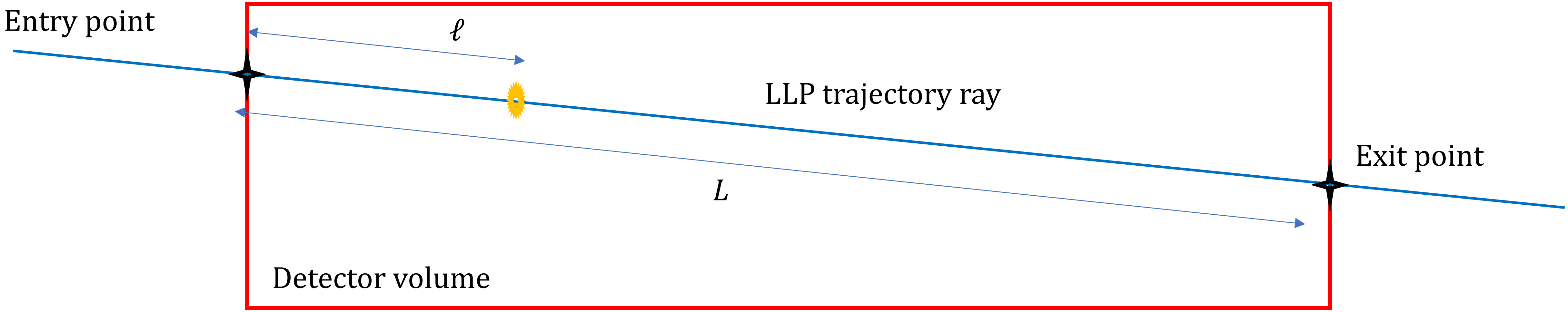
Backup

- Function to assign uniform number u to elapsed length ℓ:

$$\ell(u;\beta_X,\tau_X) = c\tau_X\beta_X\gamma_X \cdot \log_{10}\left[\left(1 - \exp\frac{-L}{c\tau_X\beta_X\gamma_X}\right)^{-1}\right],$$

where L is the maximum distance in the detector

(= distance between entry and exit points of LLP ray in the detector)



Branch	Description
LLP_Mass	Mass [MeV]
LLP_Lifetime	$c\tau$ [m]
LLP_Flux_evtno	Index of parent entry in input flux tree
LLP_Parent_pdg	PDG code of parent
LLP_Parent_v4 (v4_NEAR)	Parent decay vtx in local detector or flux (NEAR) coords
LLP_Parent_p4 (p4_NEAR)	Parent momentum in local detector or flux (NEAR) coords
LLP_p4 (p4_NEAR)	LLP momentum in local or flux coords
LLP_EntryPoint (EntryPoint_NEAR)	Entry point of LLP trajectory into detector volume
LLP_ExitPoint (ExitPoint_NEAR)	Exit point of LLP trajectory into detector volume
LLP_DecayPoint (DecayPoint_NEAR)	LLP decay vertex
LLP_Wgt_geom	Angular size of detector volume / 4pi
LLP_Boost_factor	Lab-frame E_X / parent-rest-frame E_X^*
LLP_Wgt_collimation	Modification of angular region due to collimation effect
LLP_Wgt_survival (Wgt_detdecay)	Probability of LLP survival to detector (decay inside vol)
LLP_Vertex_rng	Random number used to generate LLP vertex
LLP_production_(pdg, p4(x,y,z,E))s	PDG codes and 4-momenta (parent rest frame) of all particles in LLP production channel

Collimation effect: reminder