IFIC Seminar, Valencia *2nd June 2026*

Rethinking HEP workflows: Leveraging Differentiable Simulations and Open Datasets

César JESÚS-VALLS

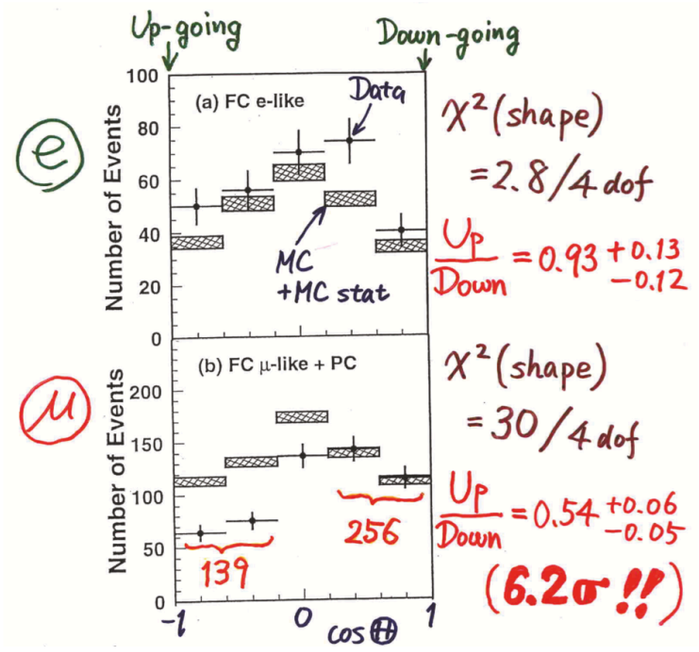
cesar.jesus@cern.ch

The Science: Neutrinos in HEP

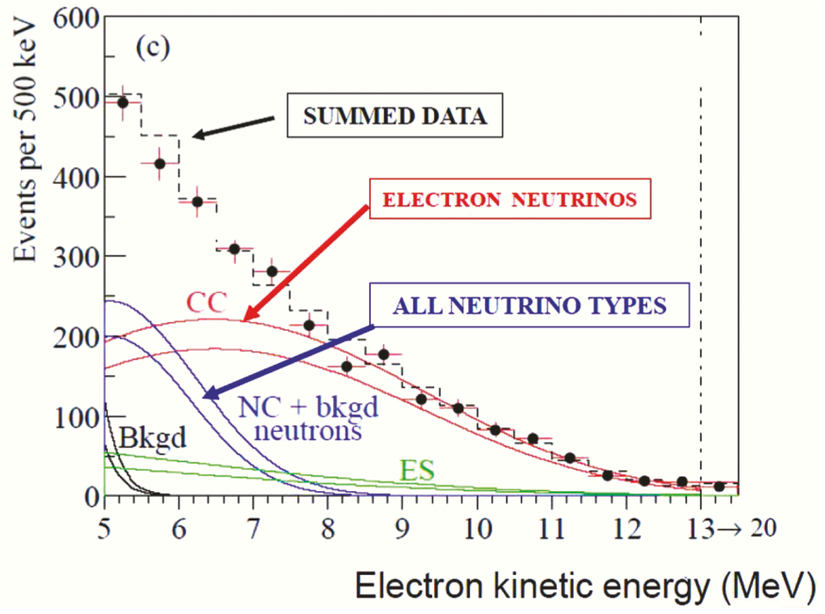
Why do we care about them in the first place?

A FEW NEUTRINO OSC 'HIGHLIGHTS'

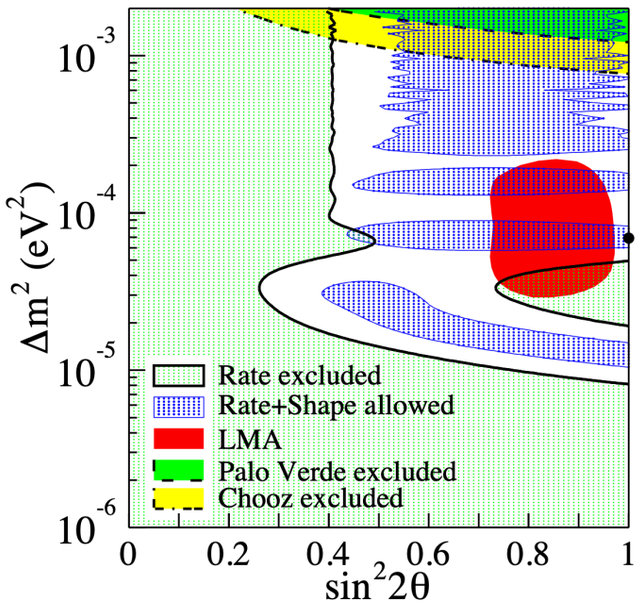
SK DEMONSTRATES ATM OSC



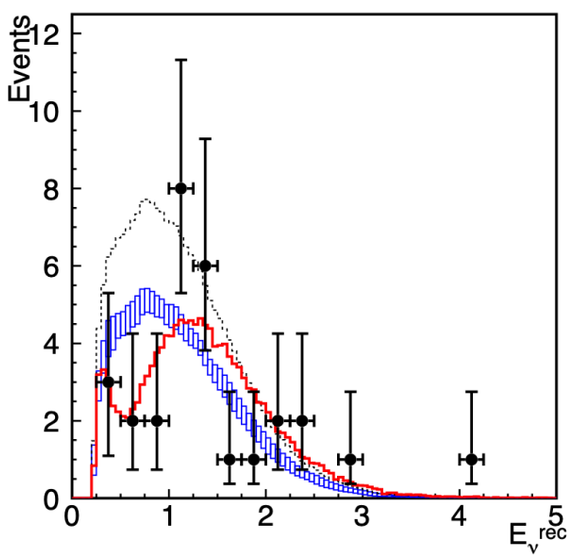
SNO SOLVES SOLAR 'ANOMALY'



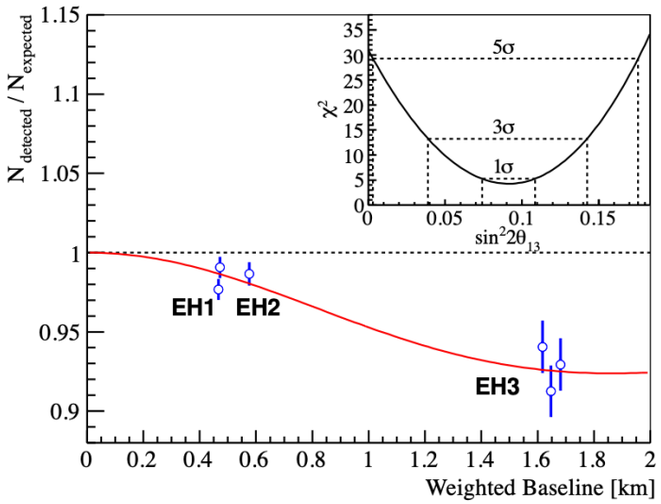
KAMLAND IDENTIFIES 'LMA'



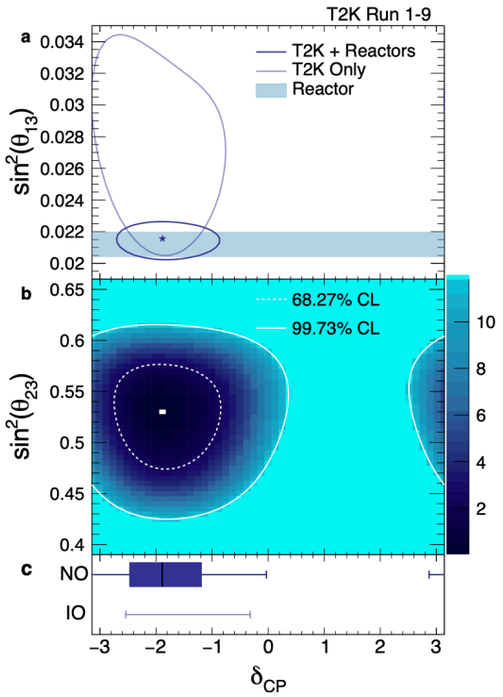
K2K FIRST OBS OF ACC ν OSC



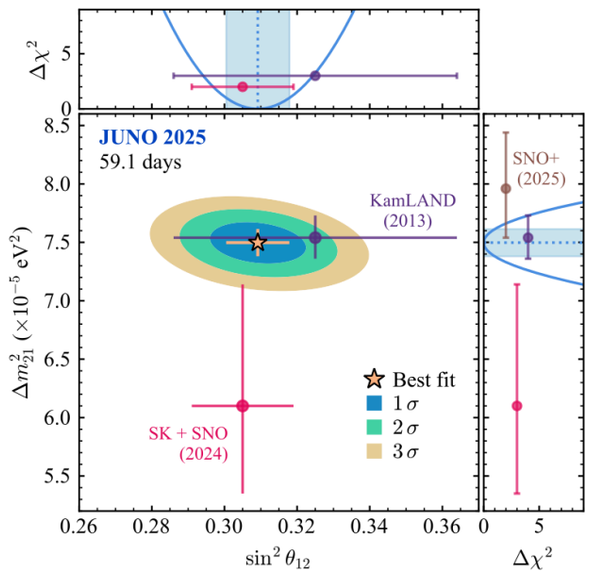
DAYA BAY SHOWS ν_e APPEARANCE



T2K SHOWS STRONG HINTS OF CP VIOLATION

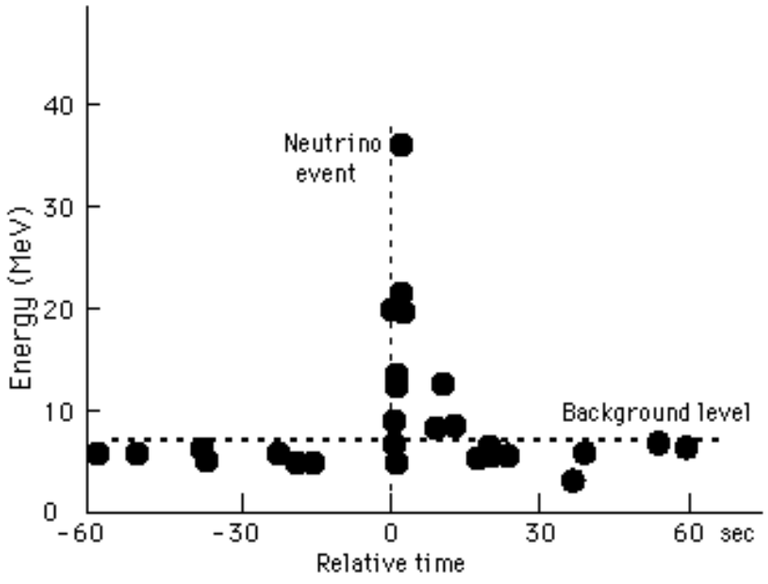


JUNO STARTS PRECISION ERA

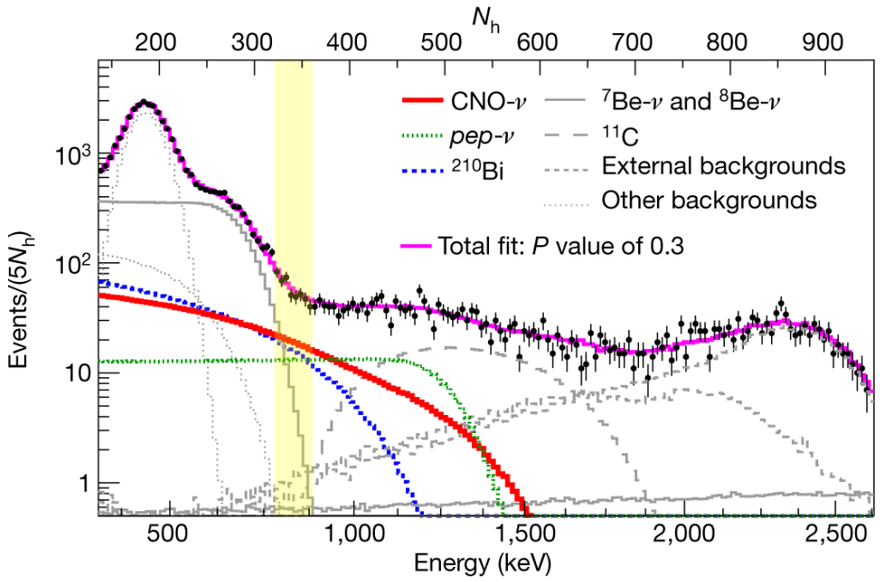


A FEW NEUTRINO ASTRO 'HIGHLIGHTS'

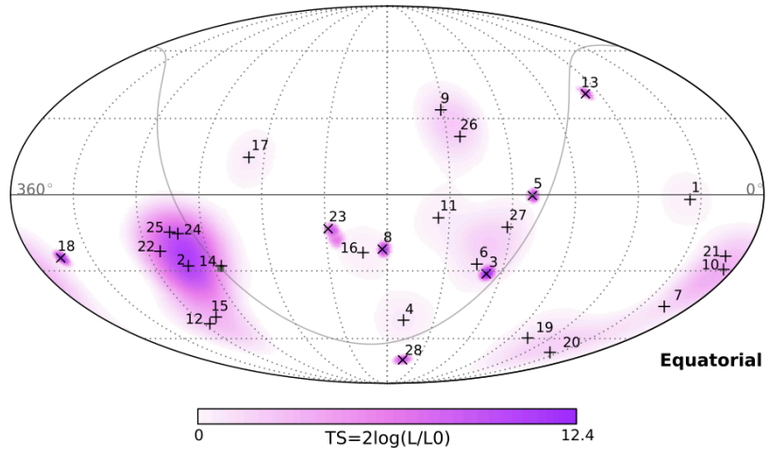
KAMIOKANDE SN 1987A



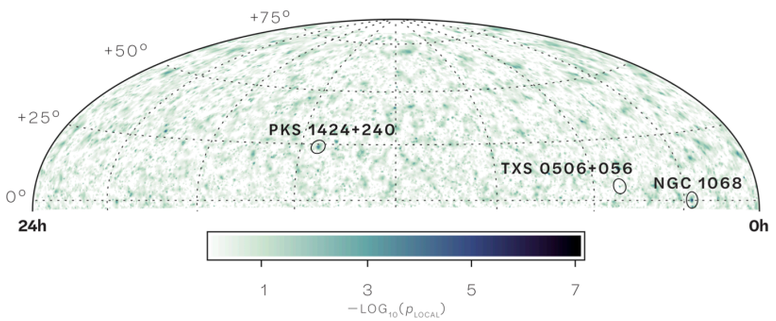
BOREXINO SOLAR SPECTROSCOPY



ICECUBE EXTRATERRESTRIAL ν FLUX

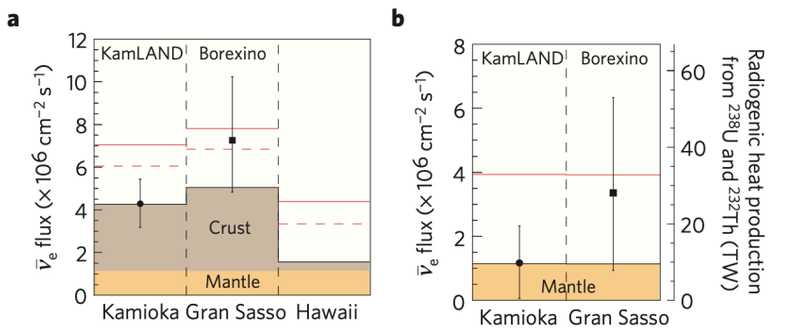


ICECUBE MULTIMESSENGER ASSOCIATATION

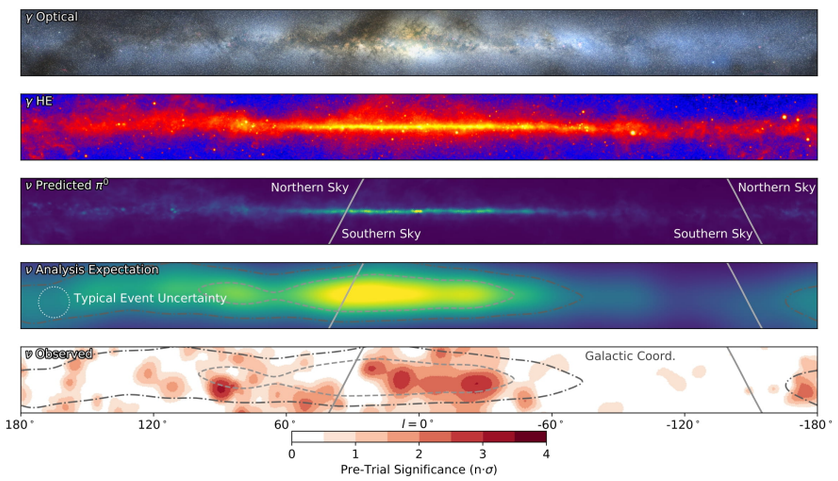


KM3NET DETECTS 220 PeV

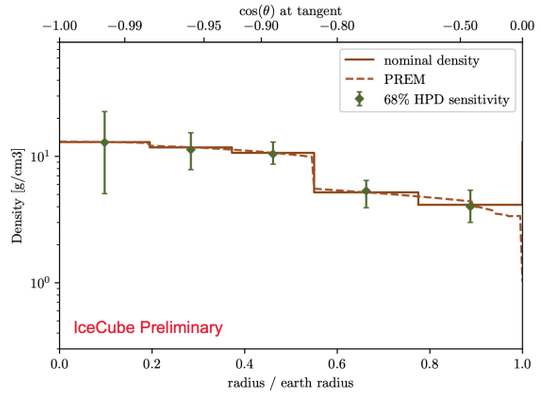
BONUS-I: Geo-neutrinos in KAMLAND/BOREXINO



ICECUBE GALACTIC PLANE ν FLUX



BONUS-II: ICECUBE EARTH TOMOGRAPHY



What do all these highlights have in common?

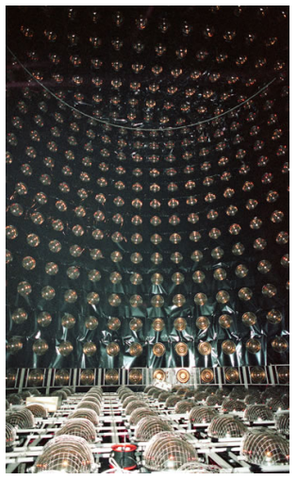
Other than neutrinos...

OPTICAL PARTICLE DETECTORS

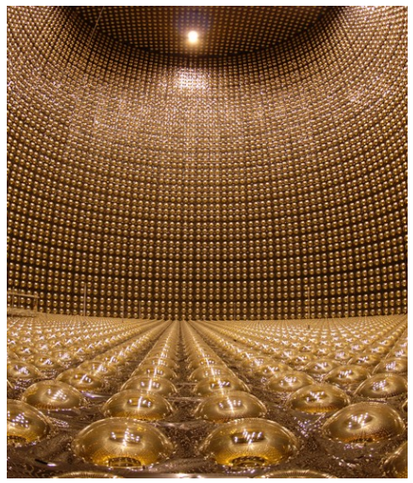
CERN

NEUTRINO PHYSICS HEAVILY RELIES IN THIS TYPE OF DETECTOR TECHNOLOGY

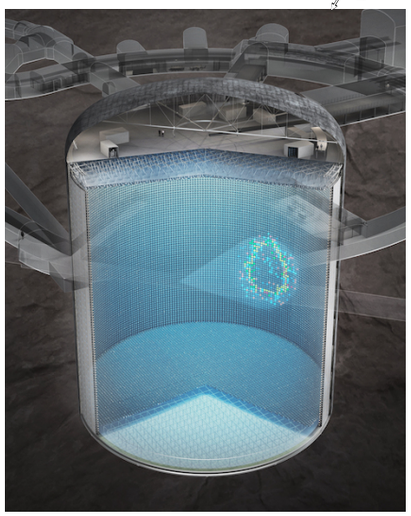
K



SUPER-K

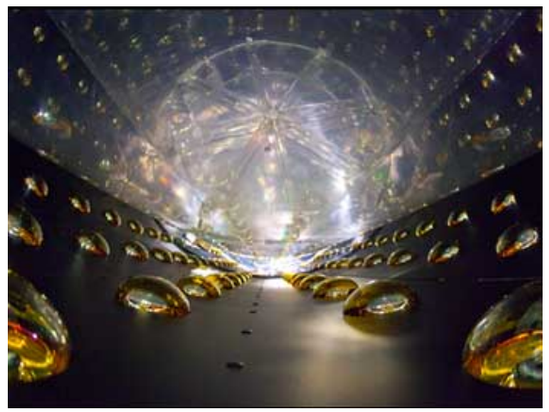


HYPER-K

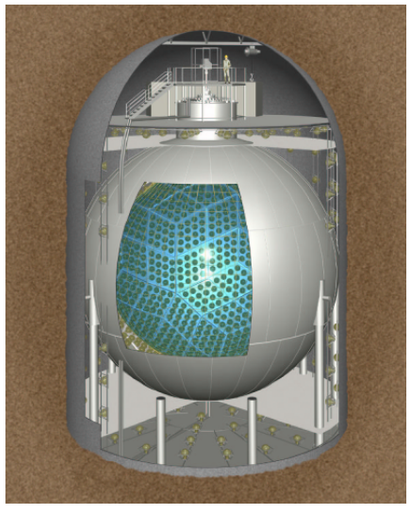


SNO

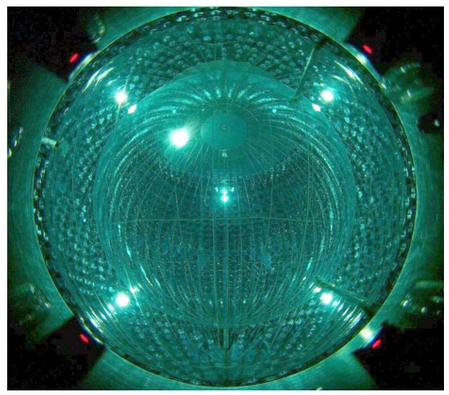
DAYA-BAY



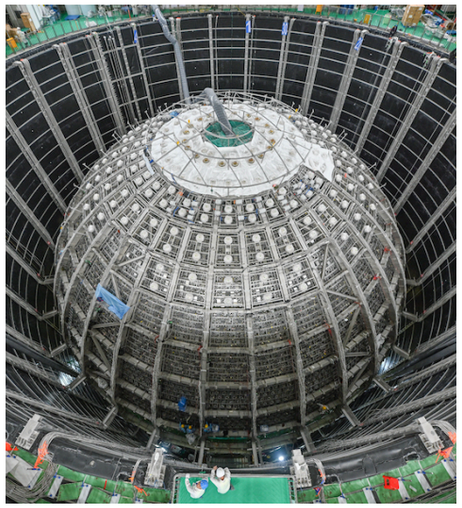
KAMLAND



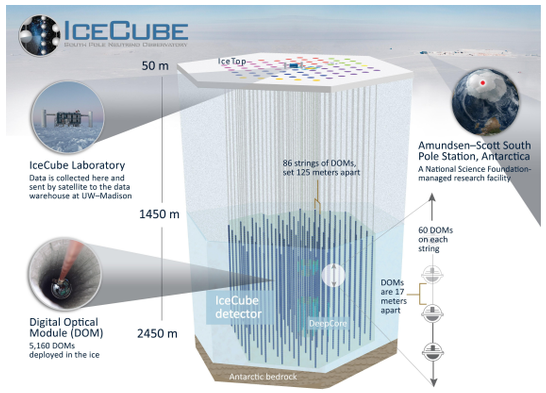
BOREXINO



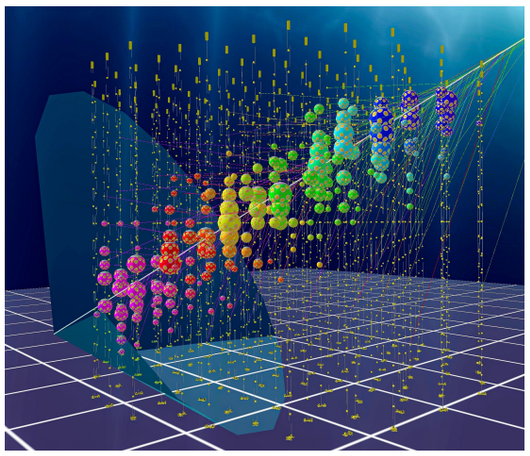
JUNO (~)



ICECUBE



KM3NET



And many others experiments & demos (Double Chooz, LSND, MiniBooNE, ANNIE, WCTE, IWCD, EOS)...

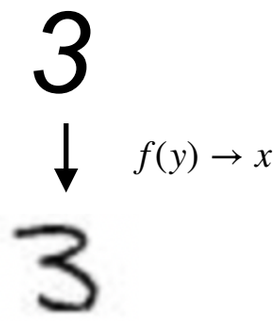
The Tools: Introducing the Challenge

Better Tools → Better Science

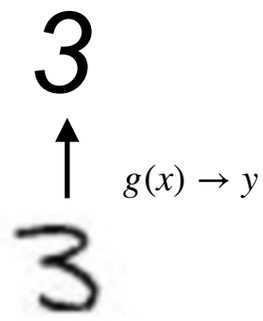
A BASIC CS EXAMPLE

Let's consider MNIST

For a given label "3", we can construct cases asking people to write the label

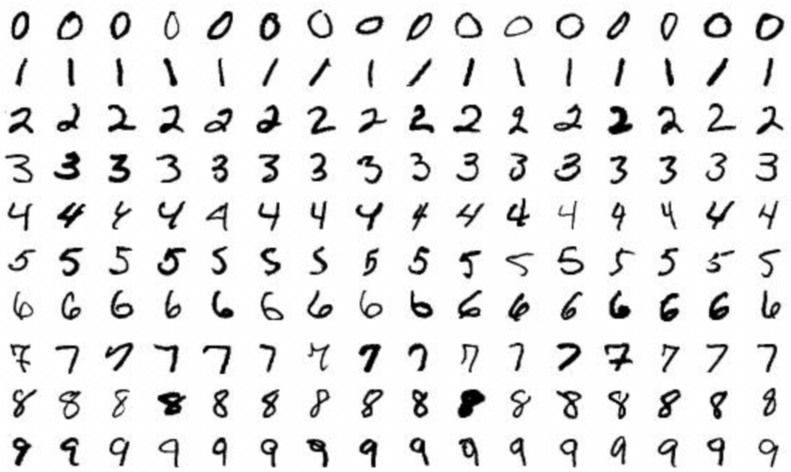


Solving the inverse problem

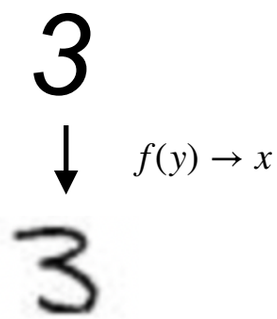


A BASIC CS EXAMPLE

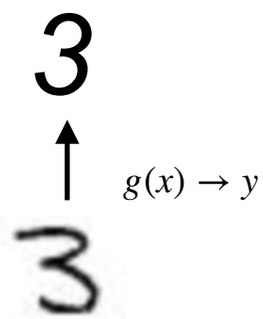
Let's consider MNIST



For a given label "3", we can construct cases asking people to write the label



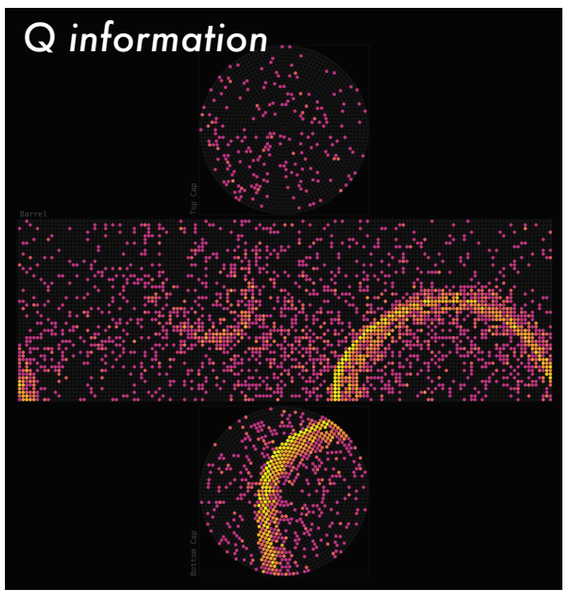
Solving the inverse problem

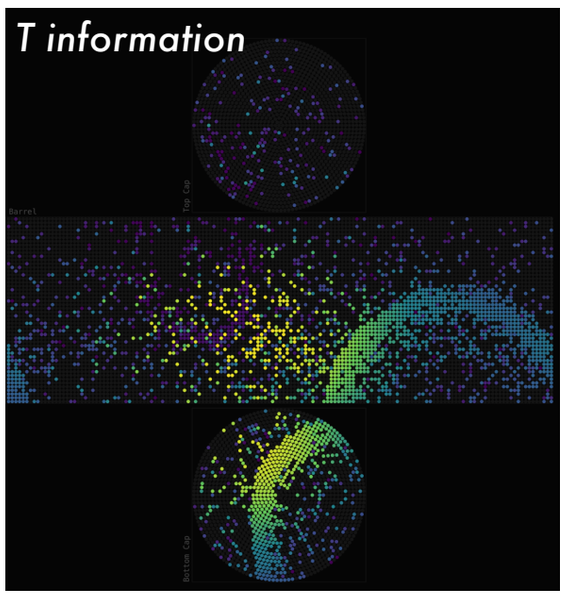


EXPERIMENTAL HIGH-ENERGY PHYSICS (HEP) AS AN INVERSE PROBLEM

Final State

Initial State (y) $\xrightarrow{f_{\vec{\theta}}(y) \to x}$



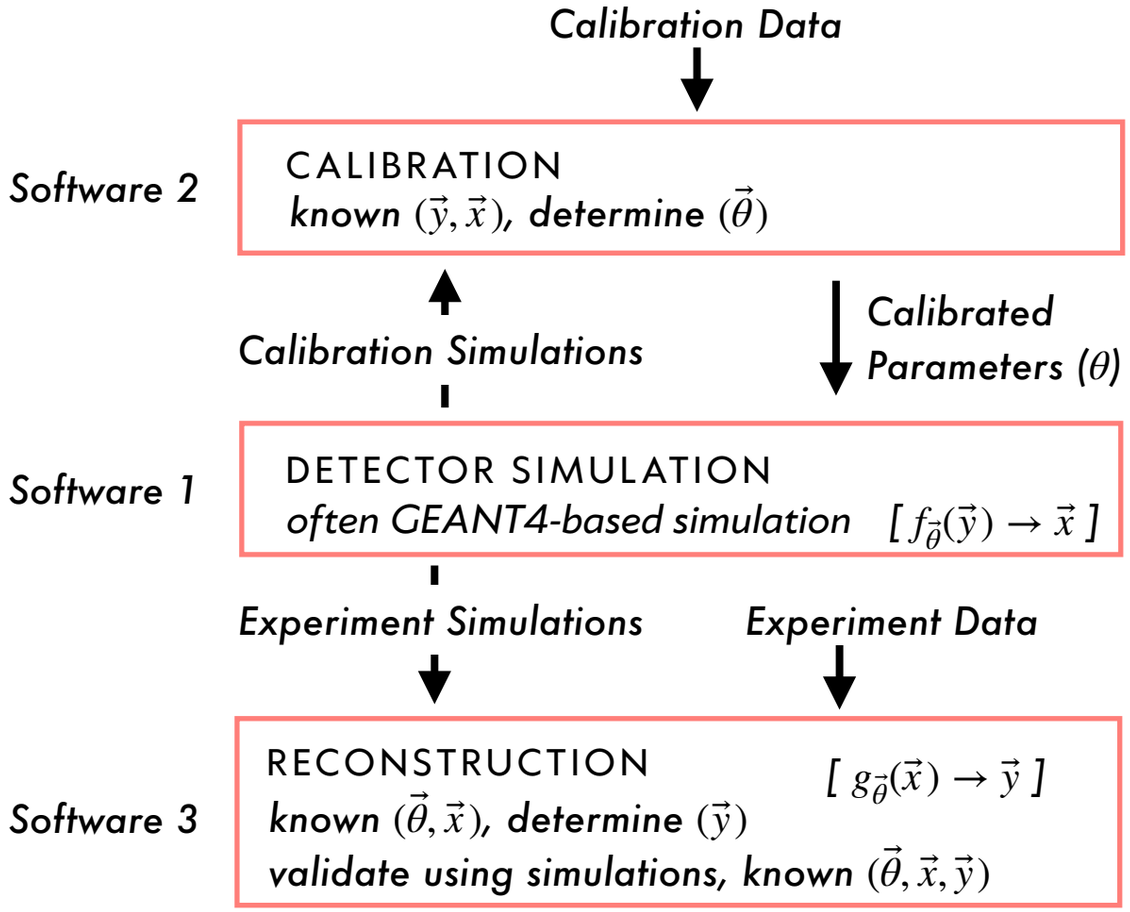


$\xrightarrow{g_{\vec{\theta}}(x) \to y}$ *Inferred Initial State (y')*

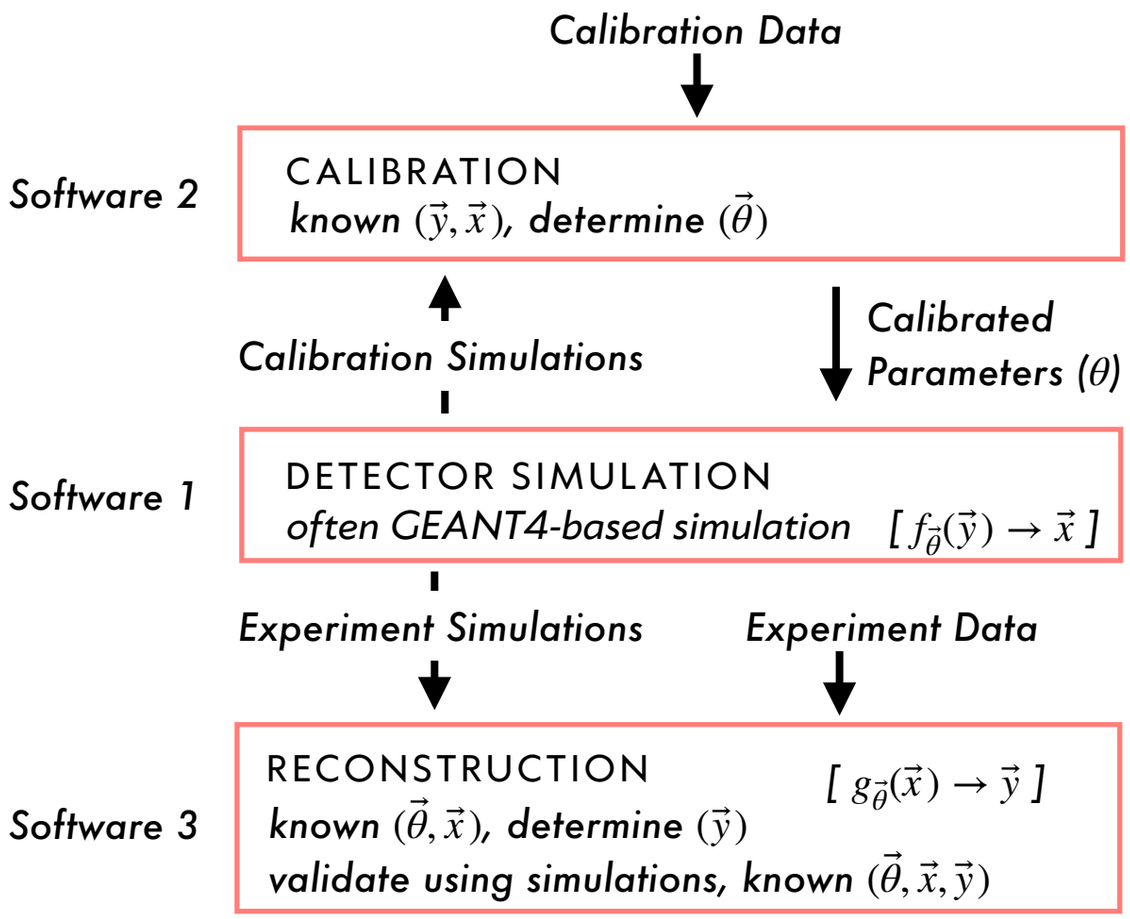
What is $\vec{\theta}$: Properties of a given material (e.g. transparency) or sensor (e.g. detection efficiency) or detector (e.g. electric field), etc.

Black Box reconstruction is NOT enough in HEP! Often we care about intermediate parameters θ!

A TYPICAL HEP WORKFLOW



A TYPICAL HEP WORKFLOW



WHAT WOULD BE IDEAL?

Unified software

IDEAL SOLUTION

Given $f_{\vec{\theta}}(\vec{y}) \to \vec{x}$, automatically defines $g_{\vec{\theta}}(\vec{x}) \to \vec{y}$

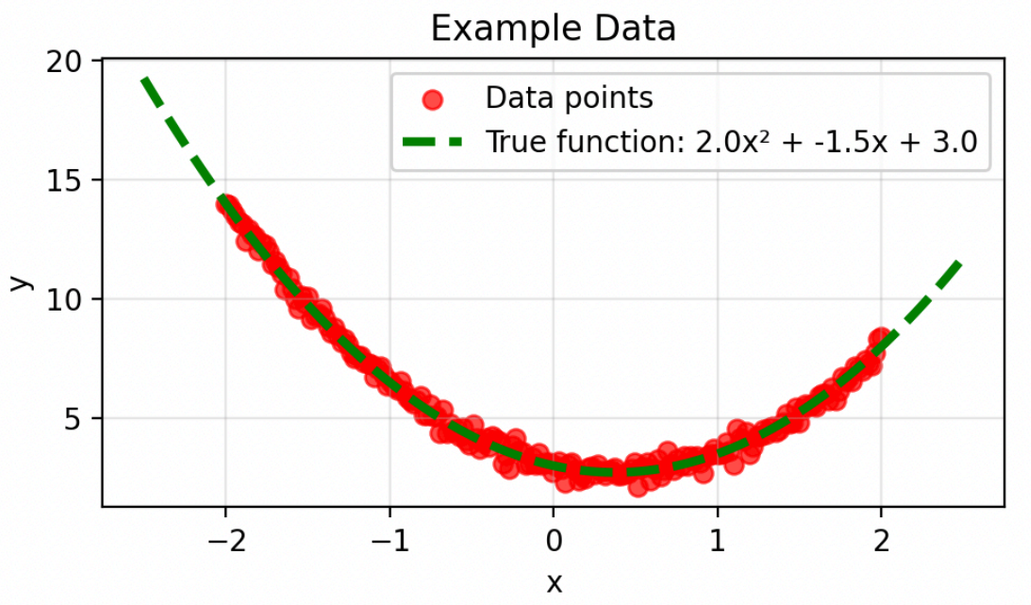
Takes input calibration → Tunes $(\vec{\theta})$

Takes input event $(\vec{x})$ → Predicts event parameters $(\vec{y})$

The goal for this seminar is to show how to construct one such "ideal solution".

Differentiable Simulations

Let's consider a simple example: A 2nd degree polynomial fit.



Consider a set of observations generated by $f(x) = ax^2 + bx + c$.
Find the "**best**" (a, b, c) to explain this data.

First we need to define what "**best**" means → define the **loss** $\mathscr{L}$

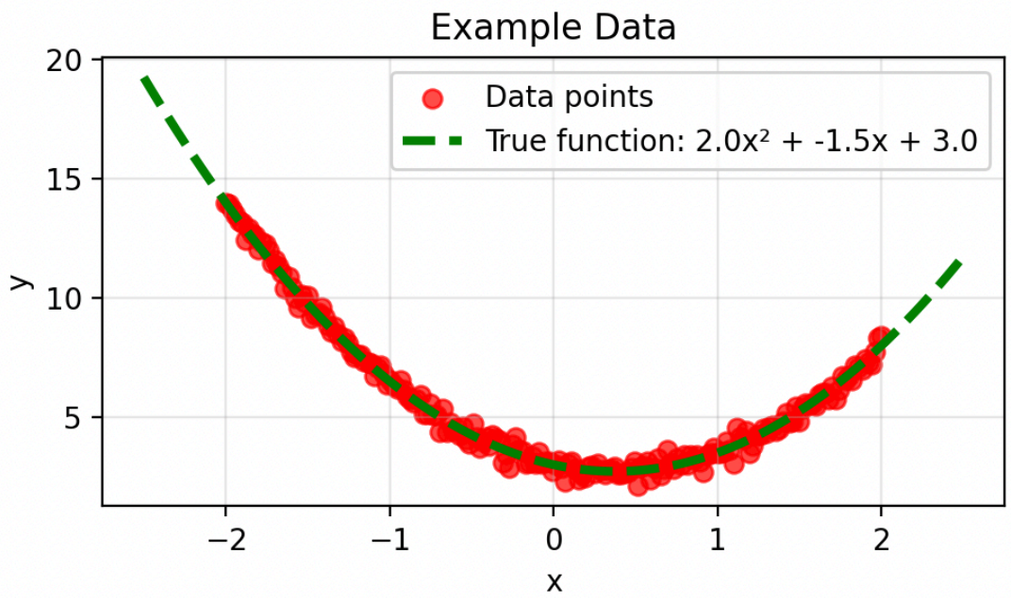
We can try to find a solution in many ways:

A) Stochastic Search Methods.

B) Numerically → $\partial\mathscr{L}/\partial a \approx [\mathscr{L}(a+h, b, c) - \mathscr{L}(a, b, c)]/h$

C) Analytically → Calculating explicitly $\partial\mathscr{L}/\partial a$

Let's consider a simple example: A 2nd degree polynomial fit.



Consider a set of observations generated by $f(x) = ax^2 + bx + c$.
Find the "**best**" (a, b, c) to explain this data.

First we need to define what "**best**" means → define the **loss** $\mathscr{L}$

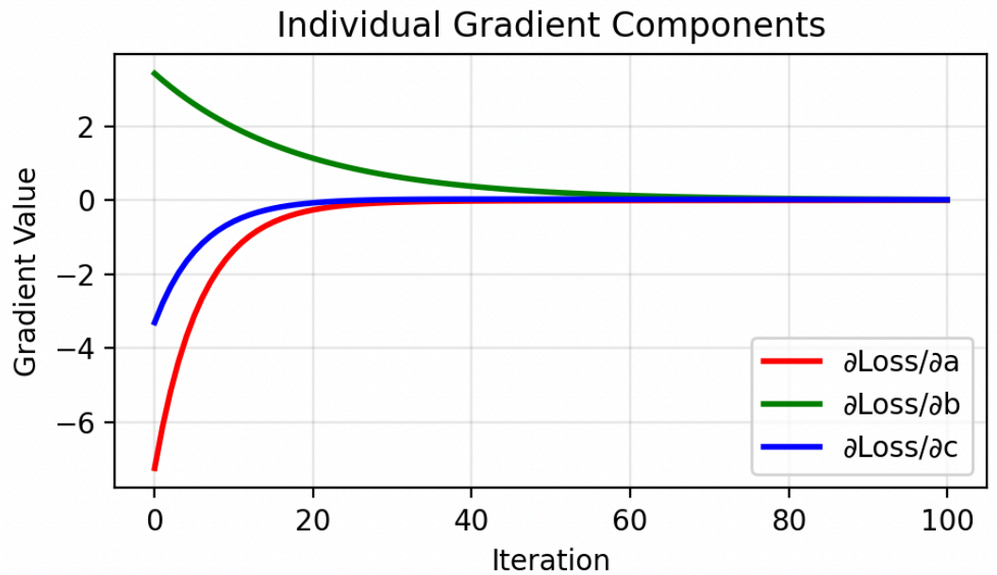
We can try to find a solution in many ways:

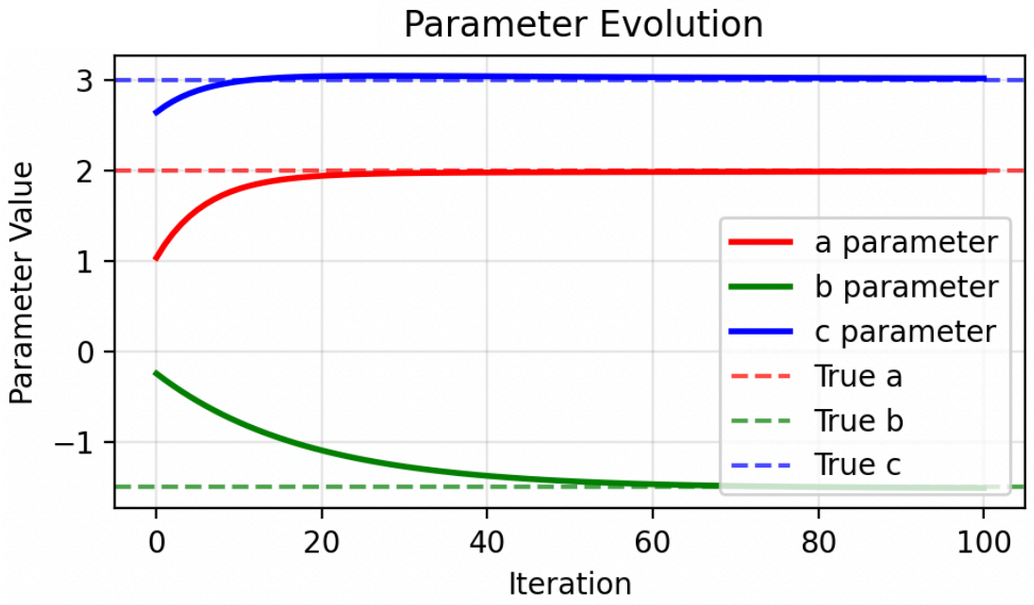
A) Stochastic Search Methods.

B) Numerically → $\partial\mathscr{L}/\partial a \approx [\mathscr{L}(a+h, b, c) - \mathscr{L}(a, b, c)]/h$

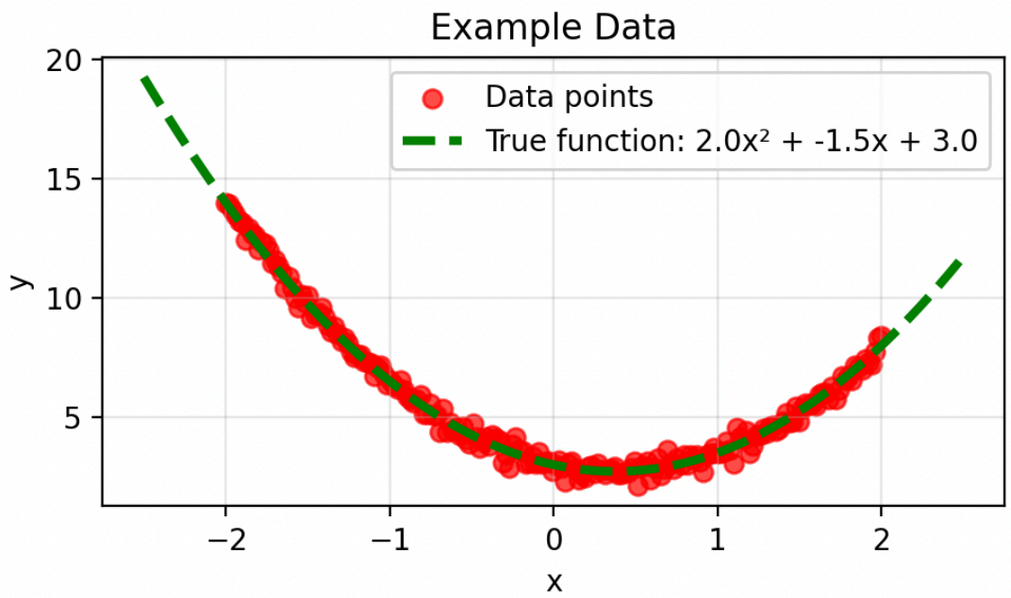
C) Analytically → Calculating explicitly $\partial\mathscr{L}/\partial a$

Look for a solution iteratively $a_{i+1} = a_i + \alpha \cdot \partial L/\partial a$
(where α is a scaling variable —learning rate—).





Let's consider a simple example: A 2nd degree polynomial fit.



Consider a set of observations generated by $f(x) = ax^2 + bx + c$.
Find the "**best**" (a, b, c) to explain this data.

First we need to define what "**best**" means → define the **loss** $\mathscr{L}$

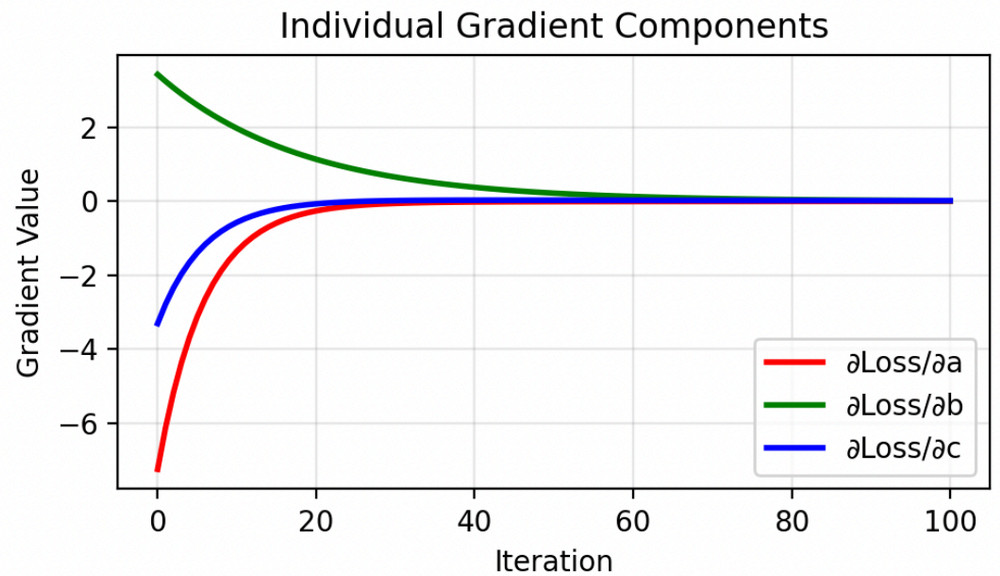
We can try to find a solution in many ways:

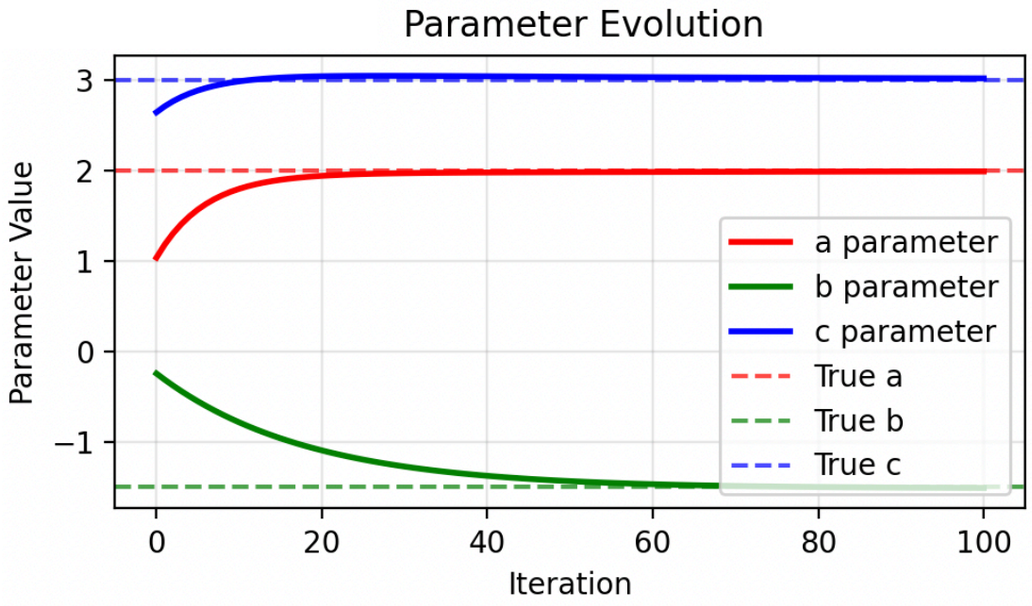
A) Stochastic Search Methods.

B) Numerically → $\partial\mathscr{L}/\partial a \approx [\mathscr{L}(a+h, b, c) - \mathscr{L}(a, b, c)]/h$

C) Analytically → Calculating explicitly $\partial\mathscr{L}/\partial a$

Look for a solution iteratively $a_{i+1} = a_i + \alpha \cdot \partial L/\partial a$
(where α is a scaling variable —learning rate—).

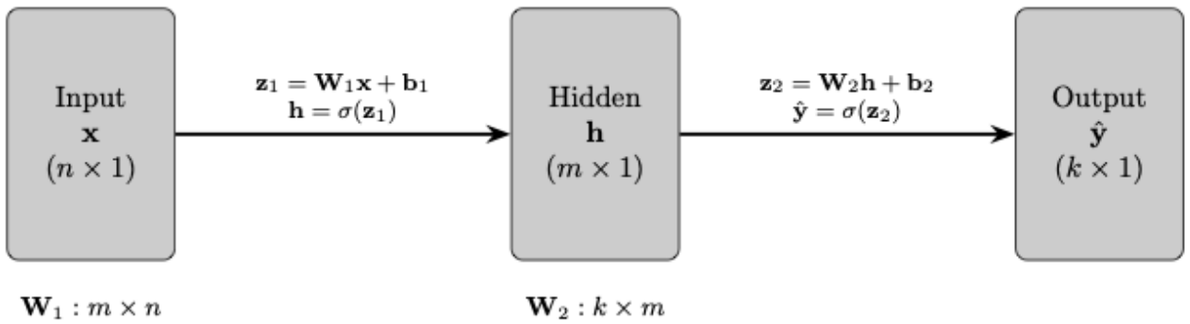




In a differentiable system, given $f_{\vec{\theta}}(\vec{y}) \to \vec{x}$, we can calculate $\partial\mathscr{L}/\partial\vec{y}$, and use it to perform gradient-based optimization, automatically providing $g_{\vec{\theta}}(\vec{x}) \to \vec{y}$

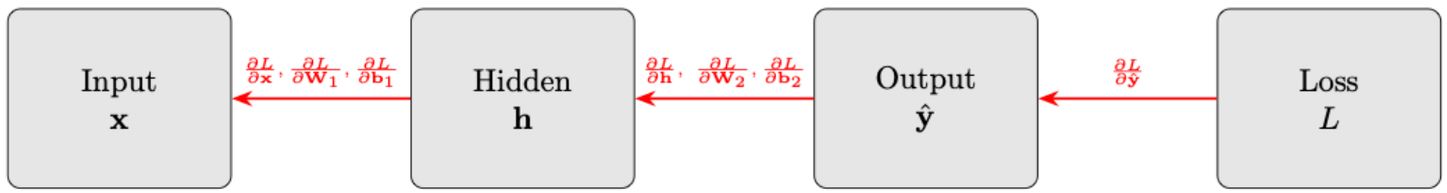
But physics simulators are much more complex than a second degree polynomial, and nobody wants to do all the derivatives by hand!

Automatic gradient calculation is the backbone of neural networks. A simple NN as an example:



To update parameters, we need: $\frac{\partial L}{\partial \mathbf{W}_1}, \frac{\partial L}{\partial \mathbf{b}_1}, \frac{\partial L}{\partial \mathbf{W}_2}, \frac{\partial L}{\partial \mathbf{b}_2}$

Backpropagation



$$L = \frac{1}{2}(\hat{\mathbf{y}} - \mathbf{y})^2$$

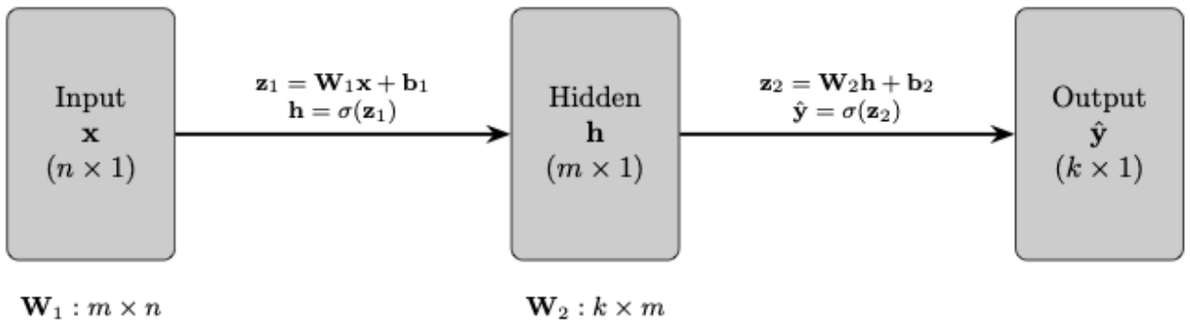
Modern neural networks can be very sophisticated and have an arbitrary number of parameters.

Existing libraries are able to calculate gradients for all their parameters via the chain rule.

GPU-acceleration makes this calculations exceptionally fast.

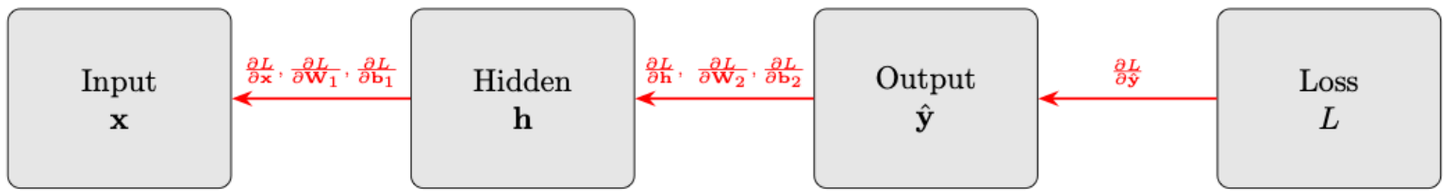
But physics simulators are much more complex than a second degree polynomial, and nobody wants to do all the derivatives by hand!

Automatic gradient calculation is the backbone of neural networks. A simple NN as an example:



To update parameters, we need: $\frac{\partial L}{\partial \mathbf{W}_1}, \frac{\partial L}{\partial \mathbf{b}_1}, \frac{\partial L}{\partial \mathbf{W}_2}, \frac{\partial L}{\partial \mathbf{b}_2}$

Backpropagation



$$L = \frac{1}{2}(\hat{\mathbf{y}} - \mathbf{y})^2$$

Modern neural networks can be very sophisticated and have an arbitrary number of parameters.

Existing libraries are able to calculate gradients for all their parameters via the chain rule.

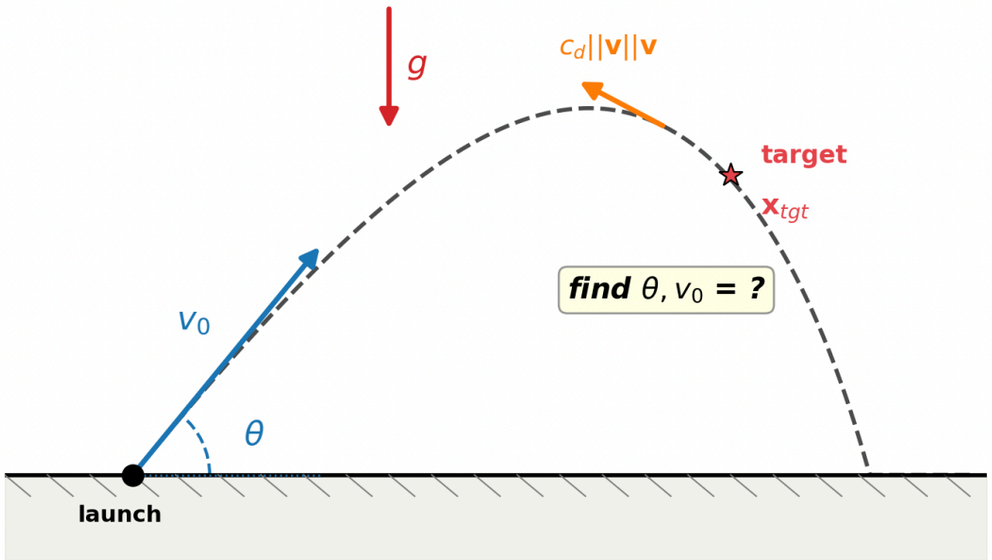
GPU-acceleration makes this calculations exceptionally fast.

Does this mean that we need to use neural networks? No!

We can use differentiable libraries to calculate gradients for any set of composable differentiable functions

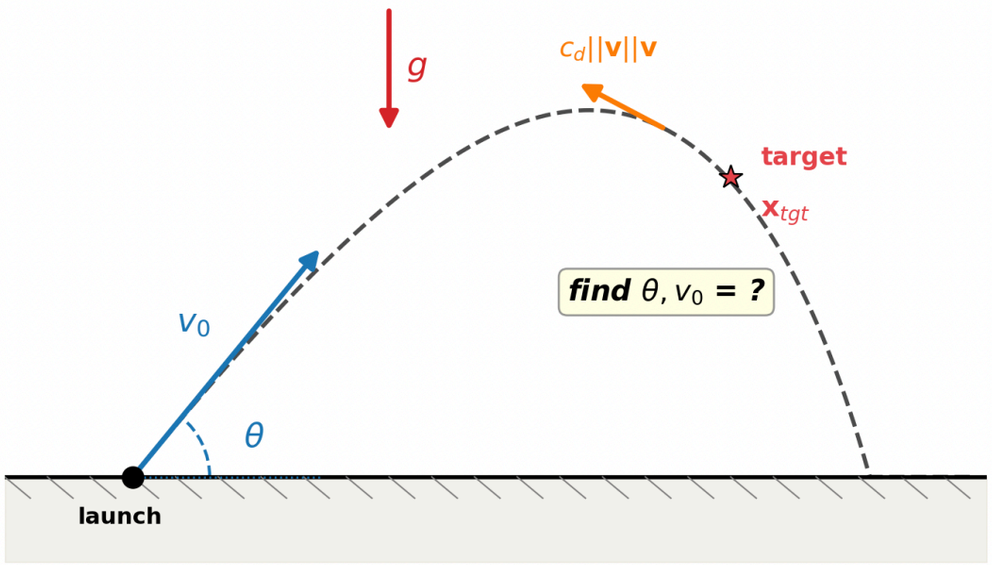
What is a *differentiable physics simulation*?

Inverse problem: optimise launch parameters to hit target



What is a *differentiable physics simulation*?

Inverse problem: optimise launch parameters to hit target



FORWARD

Let's first calculate trajectory taking many small steps forward

$$\mathbf{a}_t = -g\,\hat{y} - c_d\,||\mathbf{v}_t||\,\mathbf{v}_t$$

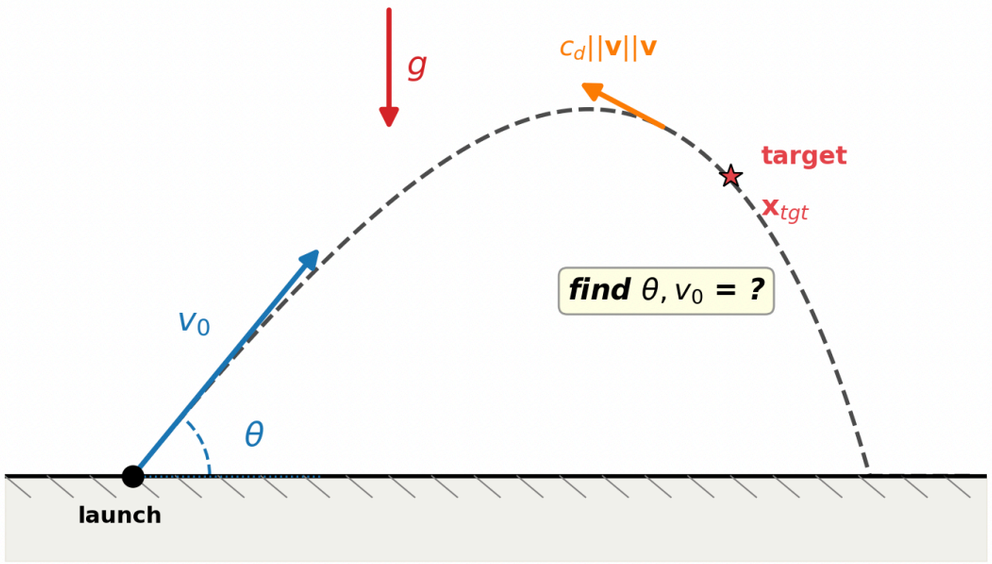
$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{a}_t\,\Delta t$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}\,\Delta t$$

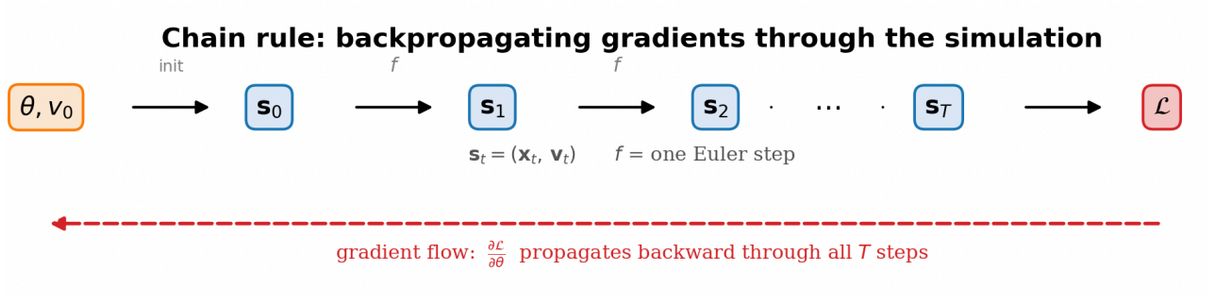
Then lets calculate how far is the closest point in the trajectory to the target, and define the loss $\mathscr{L}$

What is a *differentiable physics simulation*?

Inverse problem: optimise launch parameters to hit target



BACKWARD



Chain rule (reverse mode AD):

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \mathbf{s}_T} \cdot \frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_{T-1}} \cdot \ldots \cdot \frac{\partial \mathbf{s}_1}{\partial \mathbf{s}_0} \cdot \frac{\partial \mathbf{s}_0}{\partial \theta}$$

$$= \frac{\partial \mathcal{L}}{\partial \mathbf{s}_T} \cdot \prod_{t=1}^{T} \frac{\partial f}{\partial \mathbf{s}}\Big|_{\mathbf{s}_{t-1}} \cdot \frac{\partial \mathbf{s}_0}{\partial \theta}$$

FORWARD

Let's first calculate trajectory taking many small steps forward

$$\mathbf{a}_t = -g\,\hat{y} - c_d\,||\mathbf{v}_t||\,\mathbf{v}_t$$

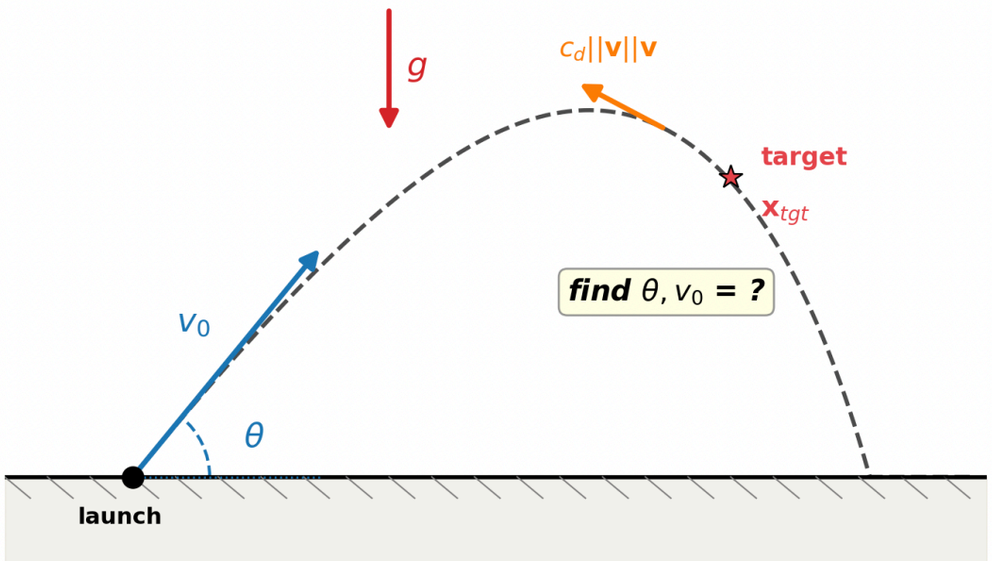
$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{a}_t\,\Delta t$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}\,\Delta t$$

Then lets calculate how far is the closest point in the trajectory to the target, and define the loss $\mathscr{L}$

What is a *differentiable physics simulation*?

Inverse problem: optimise launch parameters to hit target



FORWARD

Let's first calculate trajectory taking many small steps forward

$$\mathbf{a}_t = -g\,\hat{y} - c_d\,||\mathbf{v}_t||\,\mathbf{v}_t$$

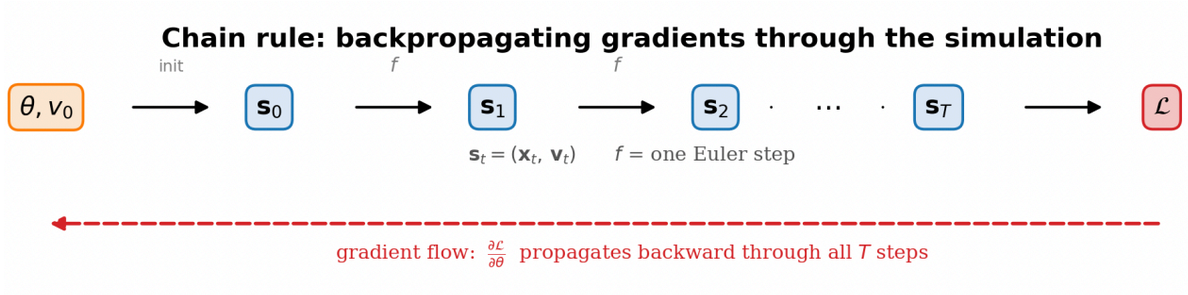
$$\mathbf{v}_{t+1} = \mathbf{v}_t + \mathbf{a}_t\,\Delta t$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}\,\Delta t$$

Then lets calculate how far is the closest point in the trajectory to the target, and define the loss $\mathscr{L}$

BACKWARD

Chain rule: backpropagating gradients through the simulation

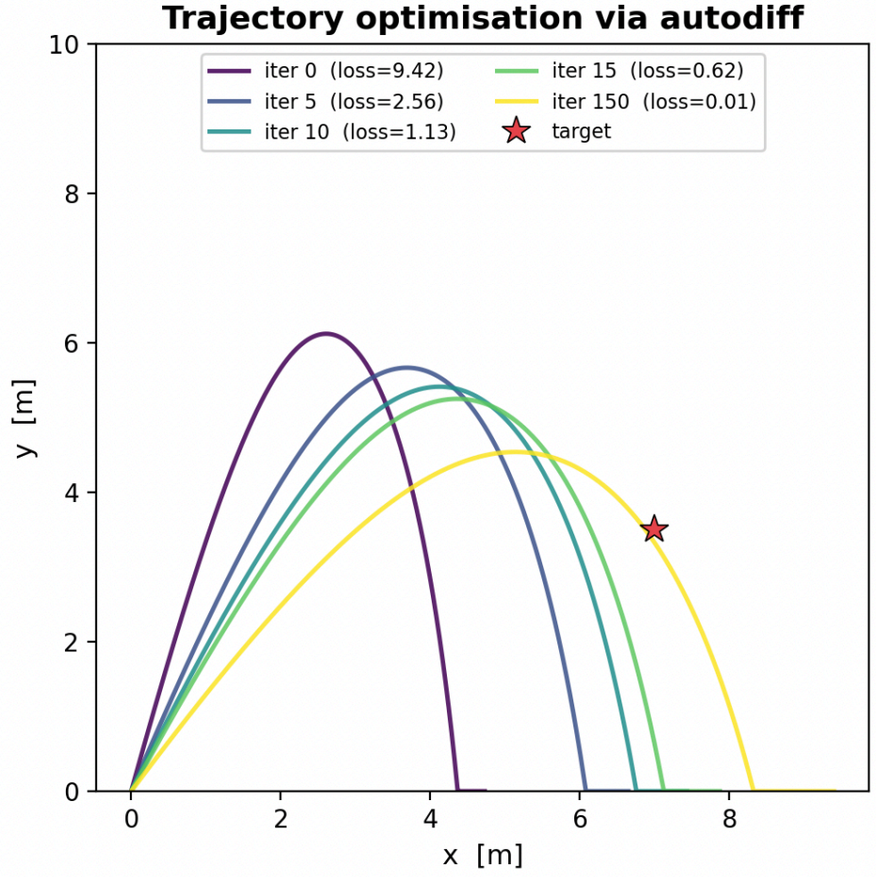


Chain rule (reverse mode AD):

$$\frac{\partial \mathscr{L}}{\partial \theta} = \frac{\partial \mathscr{L}}{\partial \mathbf{s}_T} \cdot \frac{\partial \mathbf{s}_T}{\partial \mathbf{s}_{T-1}} \cdot \ldots \cdot \frac{\partial \mathbf{s}_1}{\partial \mathbf{s}_0} \cdot \frac{\partial \mathbf{s}_0}{\partial \theta}$$

$$= \frac{\partial \mathscr{L}}{\partial \mathbf{s}_T} \cdot \prod_{t=1}^{T} \frac{\partial f}{\partial \mathbf{s}}\Big|_{\mathbf{s}_{t-1}} \cdot \frac{\partial \mathbf{s}_0}{\partial \theta}$$

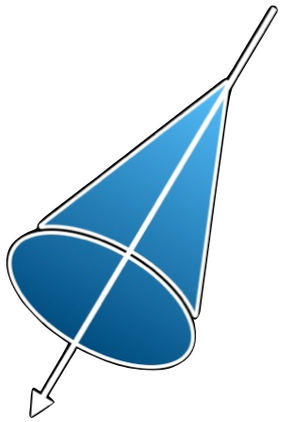
Take home message: We define the forward operation, we tune parameters leveraging AD!



Building an end-to-end differentiable optical detector simulation

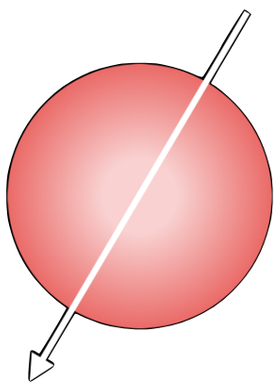
Optical particle detectors rely on the detection of photons generated by energetic particles travelling through a medium.

THERE ARE PRIMARILY TWO MAIN LIGHT-GENERATION MECHANISMS



CHERENKOV LIGHT

Directional. Light pattern is informative of particle properties:
Origin, Direction, Energy and type.
Allows to study multiple final-state particles individually.

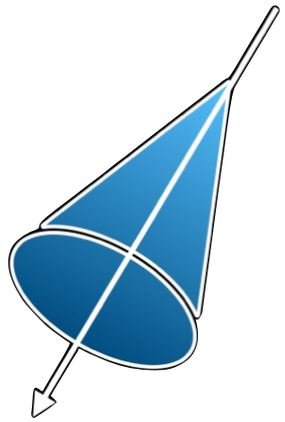


SCINTILLATION LIGHT

Isotropic. Has much higher light yield than Cherenkov emission,
Allows to measure the total energy deposit with excellent resolution.
Relies on energy + timing information to identify signal events.

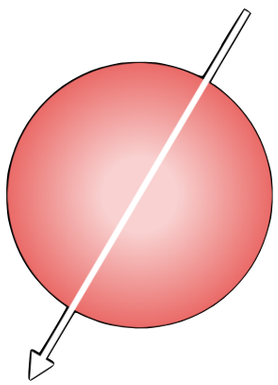
Optical particle detectors rely on the detection of photons generated by energetic particles travelling through a medium.

THERE ARE PRIMARILY TWO MAIN LIGHT-GENERATION MECHANISMS



CHERENKOV LIGHT

Directional. Light pattern is informative of particle properties:
Origin, Direction, Energy and type.
Allows to study multiple final-state particles individually.



SCINTILLATION LIGHT

Isotropic. Has much l… …enkov emission,
Allows t… …posit with excellent resolution.
Relies … …formation to identify signal events.

"Easy" to extend to Scintillation (ongoing)
we focused on Cherenkov-only first

Reconstruction θ_{reco}

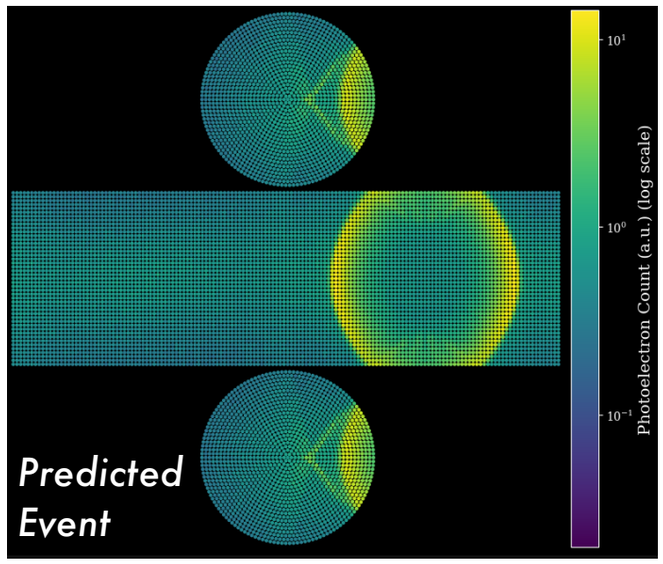
E, x, ω, t_0

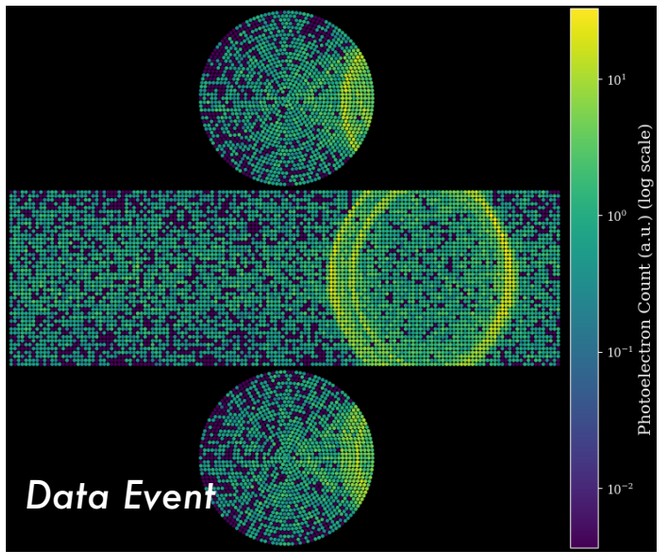
Calibration θ_{calib}

λ_s, λ_a, R

QE per sensor (~10^4 params)

Differentiable simulation





Loss

$\partial \mathcal{L} / \partial \theta_{reco}$

$\partial \mathcal{L} / \partial \theta_{calib}$

Forward pass

Backward pass (autograd)

WHAT DO WE NEED TO DO?

A DIFFERENTIABLE SIMULATION OF AN OPTICAL DETECTOR

RAY GENERATION $f1(\text{Track Parameters}) \rightarrow \{\text{Rays}\}$

RAY PROPAGATION $\sum f2(\text{Ray\_i, Geometry, Detector Parameters}) \rightarrow \{\text{Ray Paths}\}$

SIGNAL PROCESSING $\sum f3(\text{Ray Paths}) \rightarrow \{\text{Hits}\}$

THE PROJECT IDEA

If we are able to build f1, f2 and f3 we should be able to use our differentiable simulation to calculate the track and the detector parameters via gradient descent.

We created a new JAX-based framework to demonstrate the idea

LUCiD: a Light-based Unified Calibration and trackIng Differentiable simulation

LUCiD

https://arxiv.org/abs/2602.24129

https://github.com/CIDeR-ML/LUCiD/

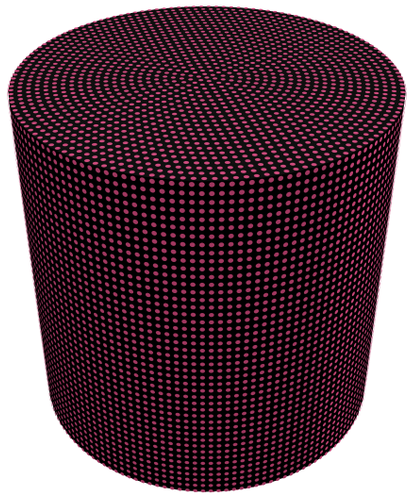
Let's see how to implement f1, f2 and f3 !

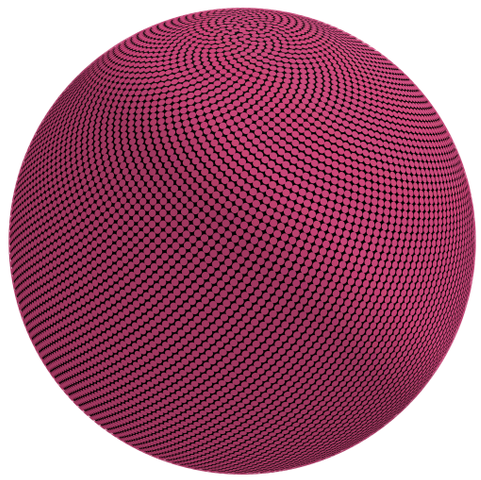
O. Alterkait
(U. Tufts/IAIFI)

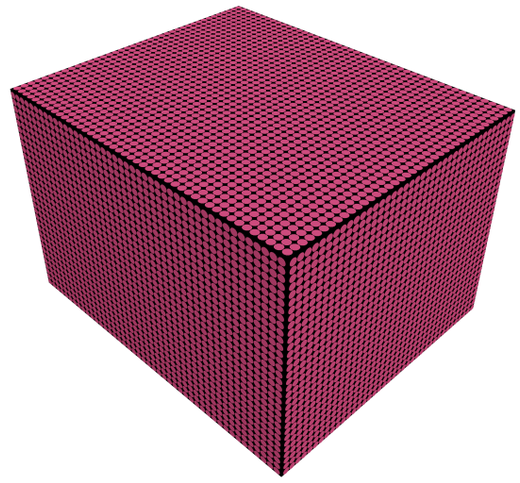
C. Jesús-Valls
(CERN)

CURRENTLY SUPPORTS MOST COMMON DETECTOR GEOMETRIES: CYLINDERS, SPHERES AND BOXES

LUCiD







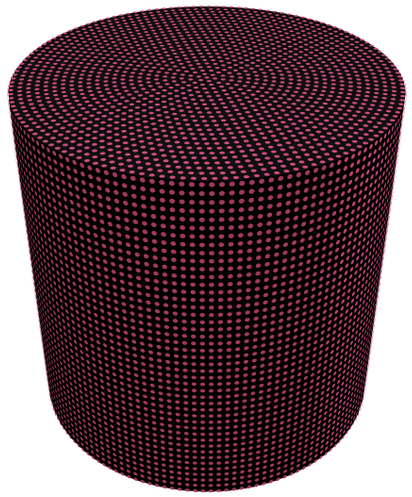
KEY ASPECTS

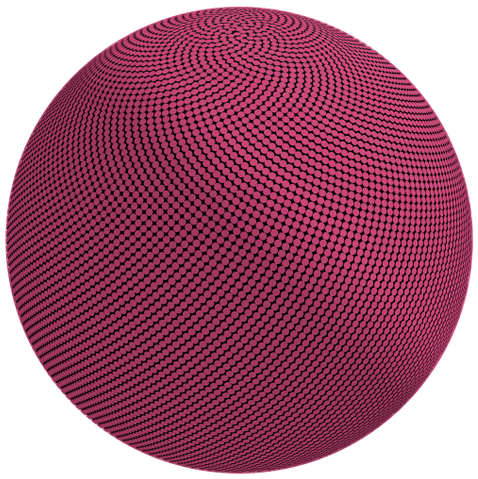
A base Detector class defines abstract methods one needs to fill to create a new detector.

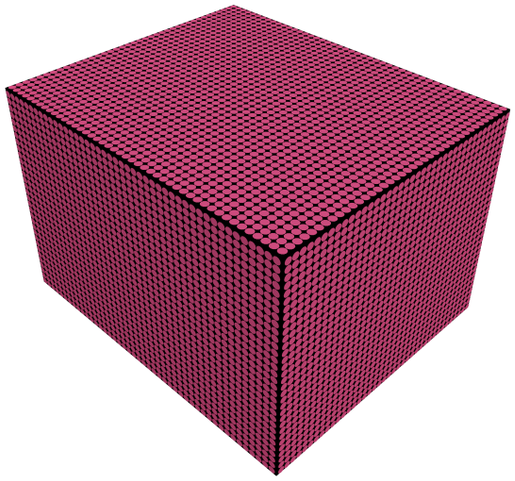
One propagator, shared across geometries, orchestrates the ray-tracing calculation. A detector geometry consists of an array of sensors in 3D space + boundary description + custom propagator primitives.

CURRENTLY SUPPORTS MOST COMMON DETECTOR GEOMETRIES: CYLINDERS, SPHERES AND BOXES

LUCiD







KEY ASPECTS

A base Detector class defines abstract methods one needs to fill to create a new detector.

One propagator, shared across geometries, orchestrates the ray-tracing calculation. A detector geometry consists of an array of sensors in 3D space + boundary description + custom propagator primitives.

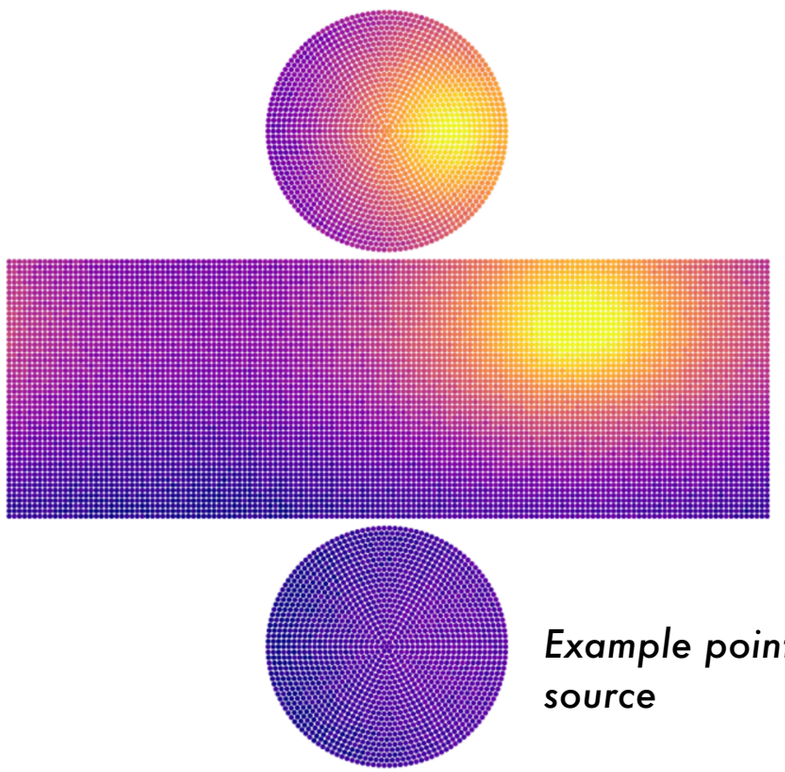
We have templates and examples, it is modular and easy to adapt to individual needs.

Talking to community is critical at this stage, we look for interested partners across ALL experiments.

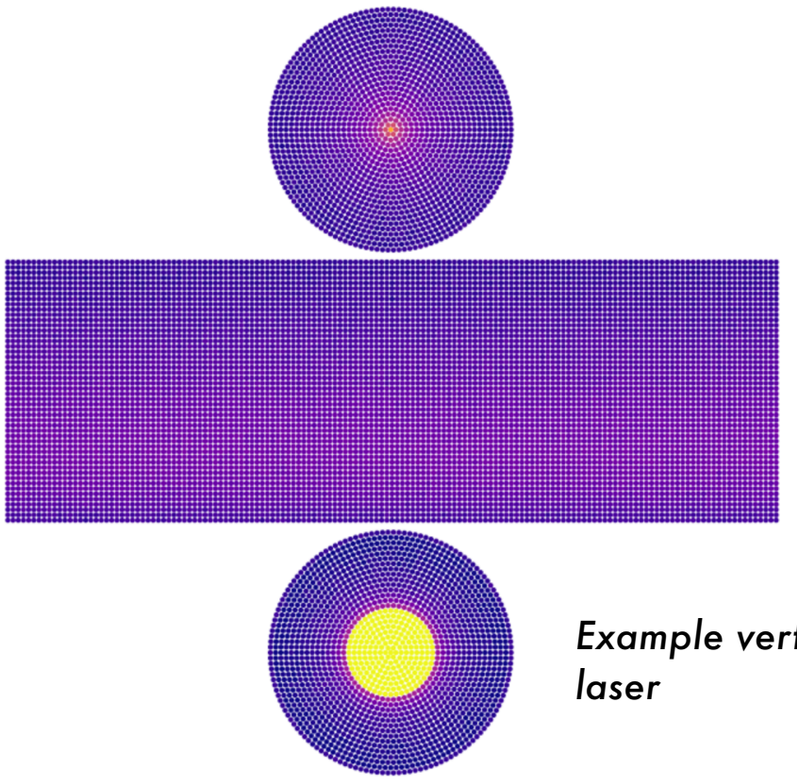
RAY GENERATION (CALIBRATION)

The ray generation step consists in describing the list of the initial ray properties (photon initial positions, times, directions, etc).

Calibration sources are generally straightforward, e.g. a a point source has all rays with common position, and random isotropic direction. A laser has also common position and the direction distribution depends on assumptions.



Example point source

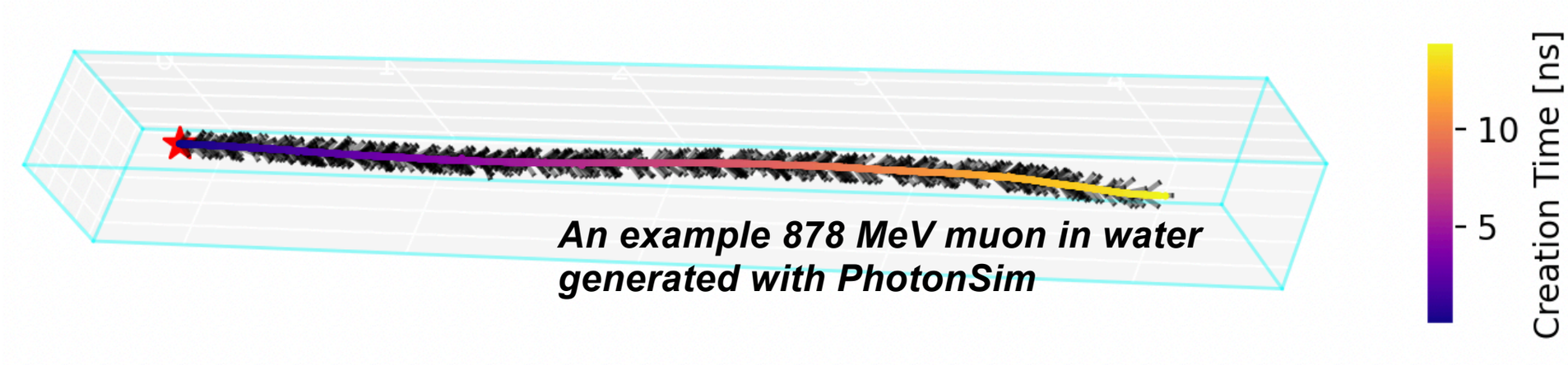


Example vertical laser

Modelling ray generation in tracks is more challenging: We need to model a distribution that incorporates all the microphysics, stochasticity, etc. → Goal was to mimic FitQun, SK reconstruction algorithm, so we parametrised the 'average' ray emission for a given track.

LET'S USE GEANT4

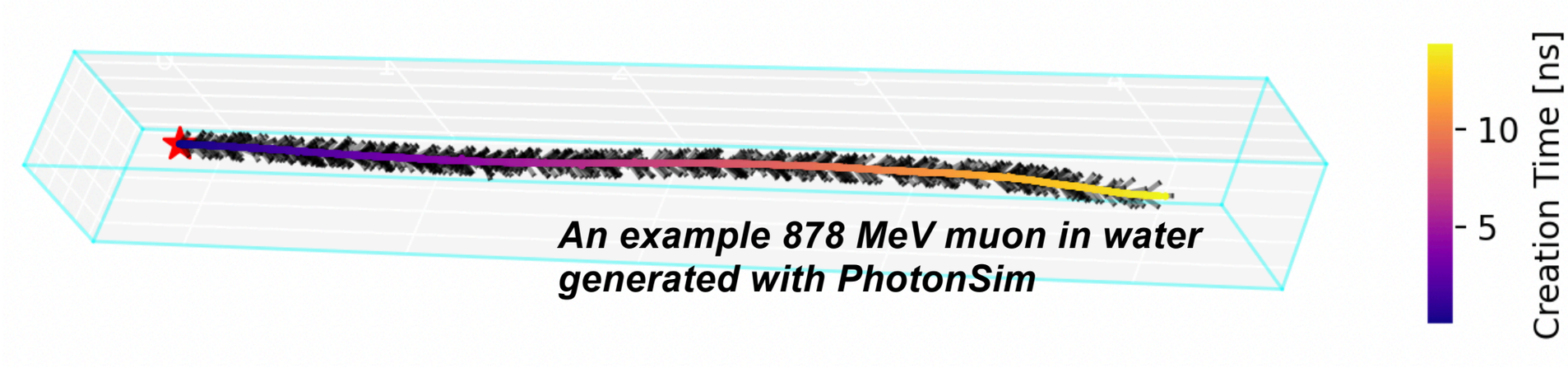
GEANT4 is not differentiable, but can be used to generate inputs to create a differentiable surrogate.
For this purpose, in the context of this project, I created PhotonSim.



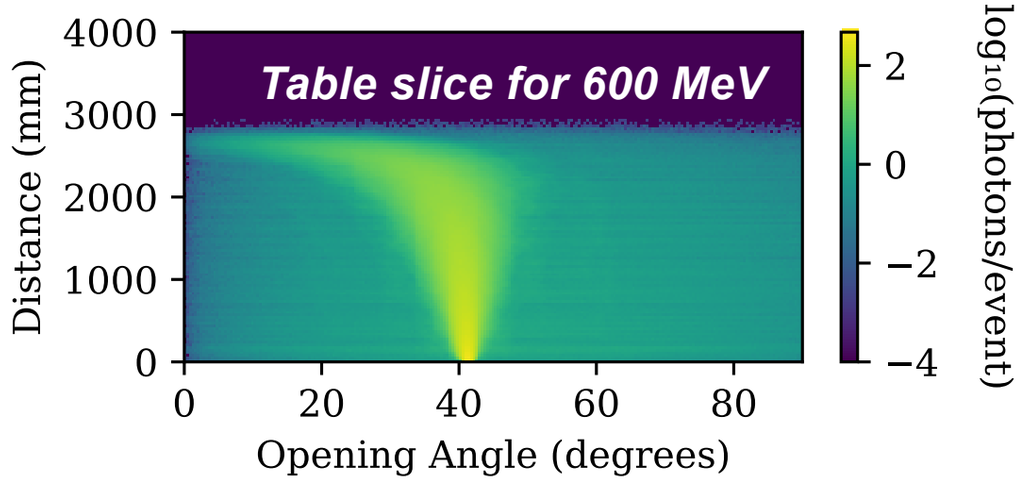
Modelling ray generation in tracks is more challenging: We need to model a distribution that incorporates all the microphysics, stochasticity, etc. → Goal was to mimic FitQun, SK reconstruction algorithm, so we parametrised the 'average' ray emission for a given track.

LET'S USE GEANT4

GEANT4 is not differentiable, but can be used to generate inputs to create a differentiable surrogate.
For this purpose, in the context of this project, I created PhotonSim.



We model the distribution of opening angles vs the distance to the track origin as a function of the track energy averaging 10k mono-energetic initial particles. We store the outcome in a 3D table:



We model the 3D look-up table using a neural network.
In particular we use SIREN (SInusoidal REpresentation Network). https://arxiv.org/pdf/2006.09661

HOW DOES IT WORK?

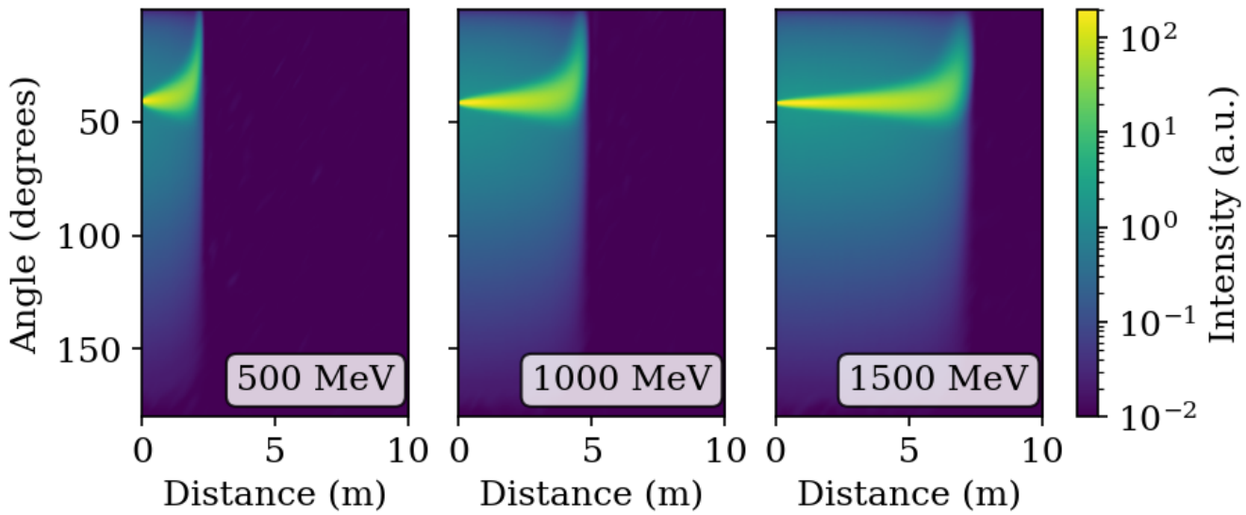
SIREN is essentially an MLP that uses periodic (sinusoidal) activation functions. SIRENs are great for representing continuous signals with fine detail and smooth derivatives, and are therefore well suited as implicit neural representations of images, audio and similar high-frequency targets. It is also a clean fit for tasks where the gradient of the network is also part of the pipeline.

We model the 3D look-up table using a neural network.
In particular we use SIREN (SInusoidal REpresentation Network). https://arxiv.org/pdf/2006.09661

HOW DOES IT WORK?

SIREN is essentially an MLP that uses periodic (sinusoidal) activation functions. SIRENs are great for representing continuous signals with fine detail and smooth derivatives, and are therefore well suited as implicit neural representations of images, audio and similar high-frequency targets. It is also a clean fit for tasks where the gradient of the network is also part of the pipeline.

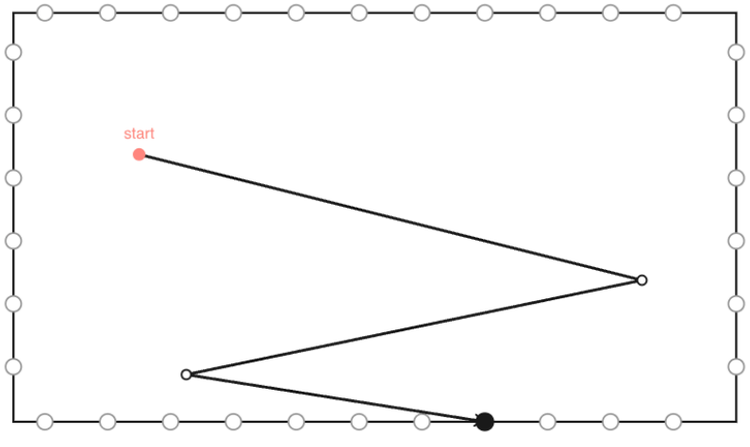
Learned Intensity distributions with SIREN



TWO PROPAGATION MODES

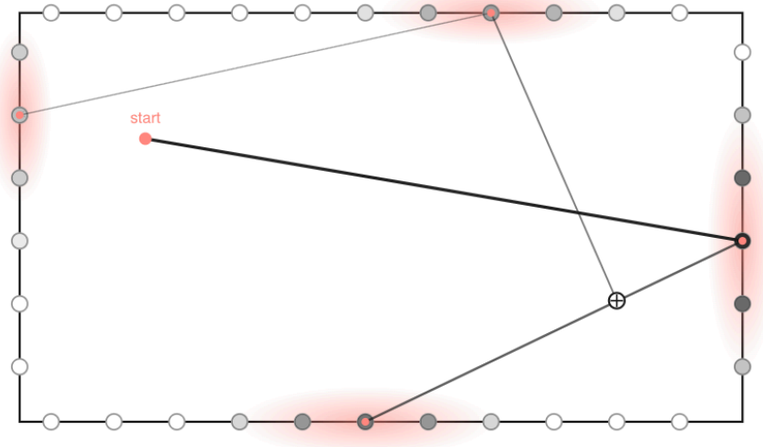
MC MODE

Discrete sampling



EXPECTED-VALUE MODE

Continuous & differentiable



Each iteration, a ray traverses some distance to its next interaction: either a scatter point in the medium or a wall/sensor surface, and then either scatters, reflects, or contributes to detection at a sensor. Absorption acts over the path length traversed in that iteration.

repeat K times

① Medium interactions

$P_{reach} = \exp(-d_{surf} / \lambda_s)$ $P_{det} = P_{reach} \cdot (1 - R)$ $A(d) = \exp(-d_{surf} / \lambda_a)$

implicit capture

② Deposit at sensors (always — ray is never killed)

$n_{dep} = I_i \cdot \alpha_i \cdot P_{det} \cdot A(d) \cdot w_{geo}$ *gaussian relaxation* $t_{hit} = t + d_{surf} / v_g$

straight through estimator

③ Path selection (scatter vs. reflect — detection excluded)

$P_{ref} = P_{reach} \cdot R$
$P_{scat} = 1 - P_{reach}$
$P = [\, P_{scat}, P_{ref} \,] / (P_{scat} + P_{ref})$

$h \sim \text{Categorical}(P)$
$w = h - sg(P) + P$
$x_{new} = w_0\, x_{scat} + w_1\, x_{surf}$

④ State update

$\alpha_i \leftarrow \alpha_i \cdot \exp(-d_{step} / \lambda_a)$ $t \leftarrow t + d_{step} / v_g$

Legend

λ_s scattering length
d_{surf} ray–surface distance
w_{geo} geometric overlap wei
h one-hot path sample

λ_a absorption length
I_i generation intensity
v_g group velocity in medium
P path probability vector

R surface reflectivity
α_i survival factor
d_{step} path length this step
$sg(\cdot)$ stop-gradient operator

HOW DO WE CREATE HITS FROM RAYS?

When working in 'data mode':

CANDIDATE RAYS

From the propagator, each candidate carries:

- a weight
- which sensor it hits
- arrival time

M = K × max_sensors × n_rays entries in total

QE DETECTION

Each candidate is kept with a per-sensor probability:

```
p = QE × QE_corr[sensor]
keep ~ Bernoulli(p)
```

Hard 0/1 decision — what a real sensor does.

AGGREGATE PER SENSOR

Group surviving rays by target sensor and reduce:

```
charge[sensor] = Σ weights
time[sensor] = min(times)
```

Total charge and first-arrival time per sensor.

DETECTOR SMEARING

Apply SK-like resolution to mimic real electronics:

- Gaussian jitter on times (TTS)
- Gain smearing on charges

Matches the resolution of the real detector.

HOW DO WE CREATE HITS FROM RAYS?

When working in 'data mode':

CANDIDATE RAYS

From the propagator, each candidate carries:

- a weight
- which sensor it hits
- arrival time

M = K × max_sensors × n_rays entries in total

→

QE DETECTION

Each candidate is kept with a per-sensor probability:

```
p = QE × QE_corr[sensor]
keep ~ Bernoulli(p)
```

Hard 0/1 decision — what a real sensor does.

→

AGGREGATE PER SENSOR

Group surviving rays by target sensor and reduce:

```
charge[sensor] = Σ weights
time[sensor] = min(times)
```

Total charge and first-arrival time per sensor.

→

DETECTOR SMEARING

Apply SK-like resolution to mimic real electronics:

- Gaussian jitter on times (TTS)
- Gain smearing on charges

Matches the resolution of the real detector.

When working in 'prediction mode':

CANDIDATE RAYS

From the propagator, each candidate carries:

- a weight
- which sensor it hits
- arrival time

M = K × max_sensors × n_rays entries in total

→

QE WEIGHTING

Same per-sensor probability as data mode — applied as a continuous weight:

```
p = QE × QE_corr[sensor]
weight ← weight × p
```

Continuous — keeps the pipeline differentiable.

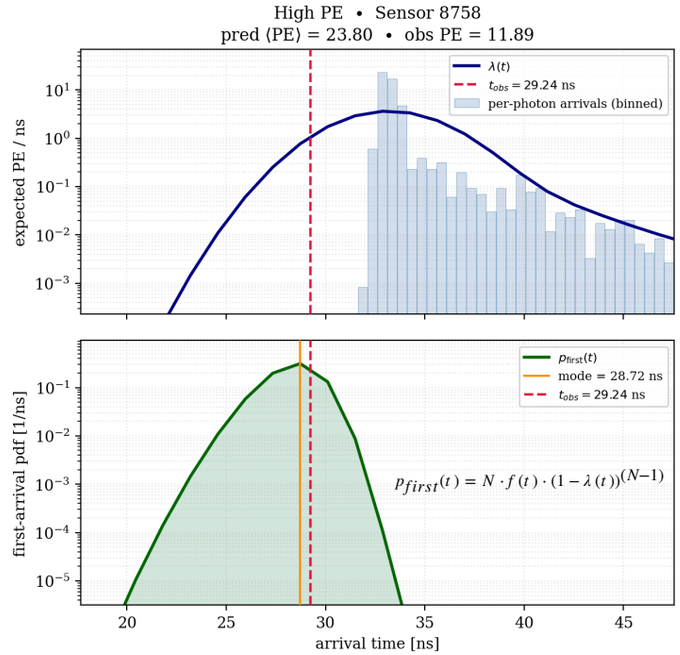
→

AGGREGATE PER SENSOR

Reduce only the charge — per-ray list stays intact:

```
charge[sensor] = Σ weights
(times, sensor_id → kept)
```

Likelihood consumes the full per-ray list directly.



time likelihood example for one sensor

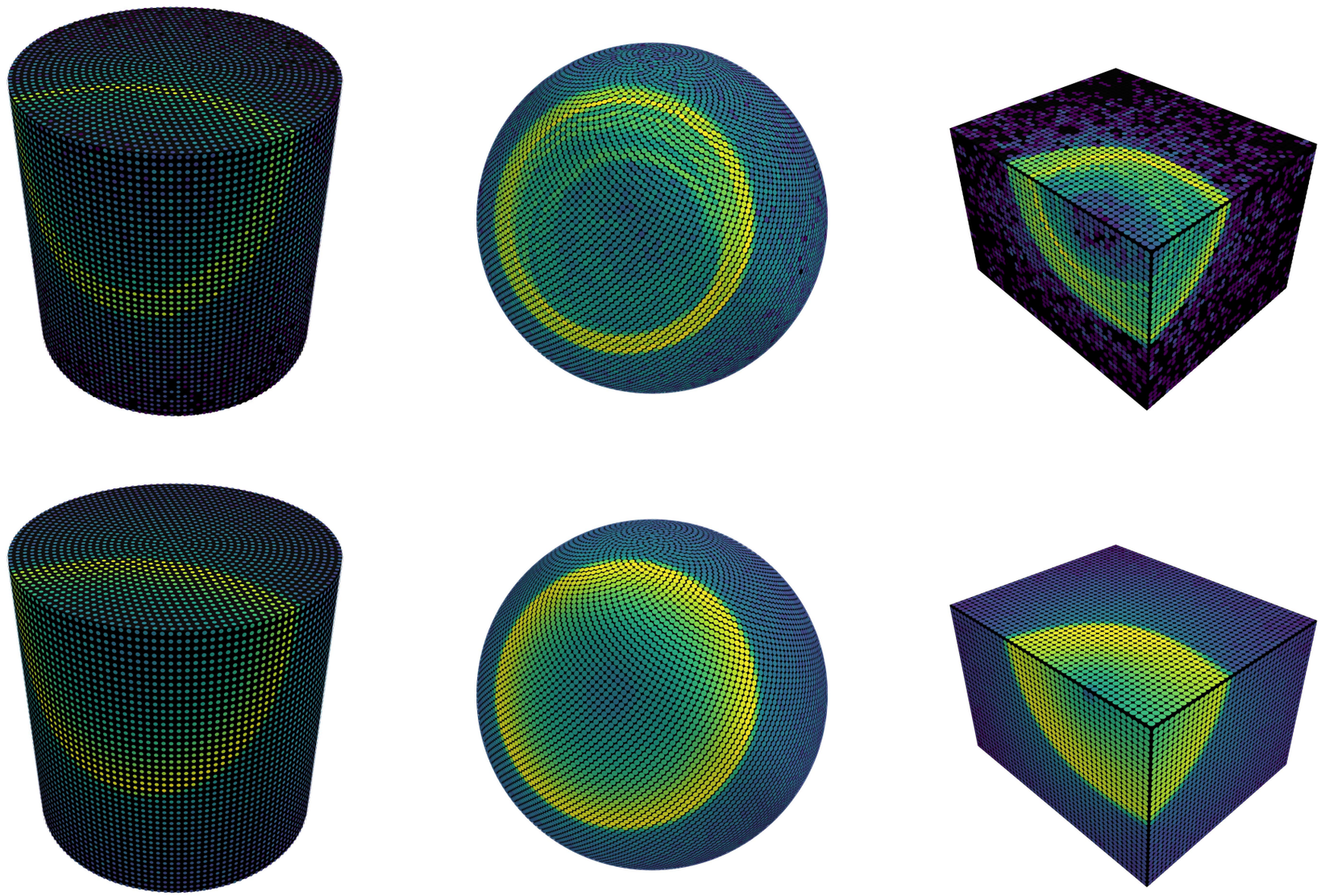
Summed charged weights correspond directly to charge expectation per sensor.
Full list of arrival times is used to calculate first time of arrival likelihood in the loss.

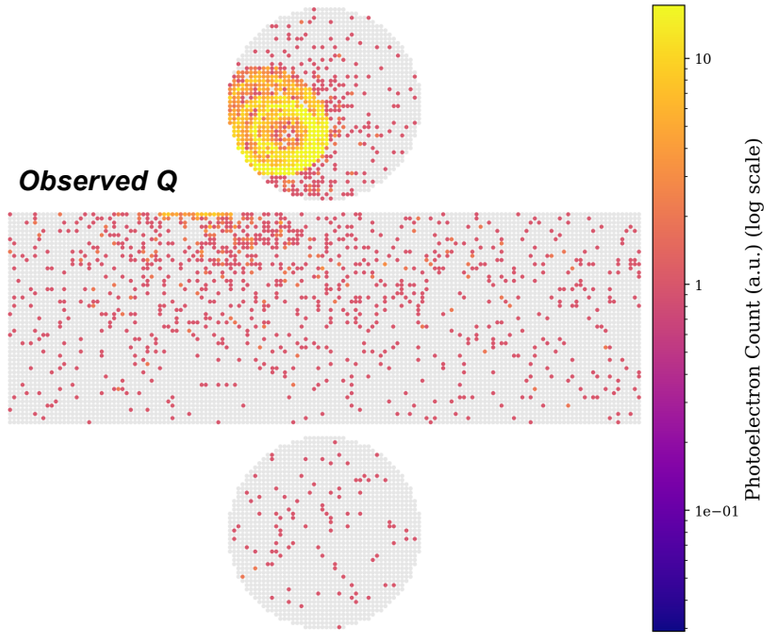
- Generation: We store individual photons using PhotonSim and then we inject them in our ray-tracer using ‘MC mode’ propagation.

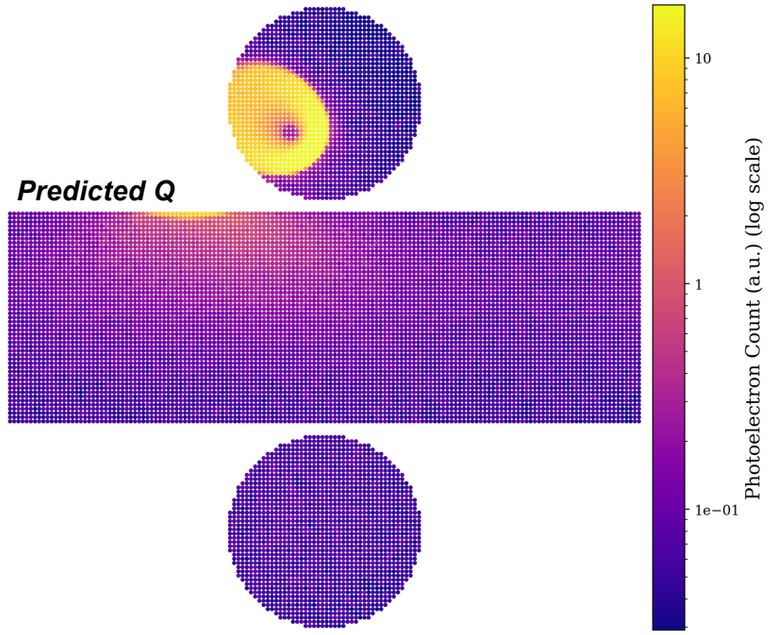
DATA LIKE EVENTS

- Generation: We store individual photons using PhotonSim and then we inject them in our ray-tracer using 'MC mode' propagation.

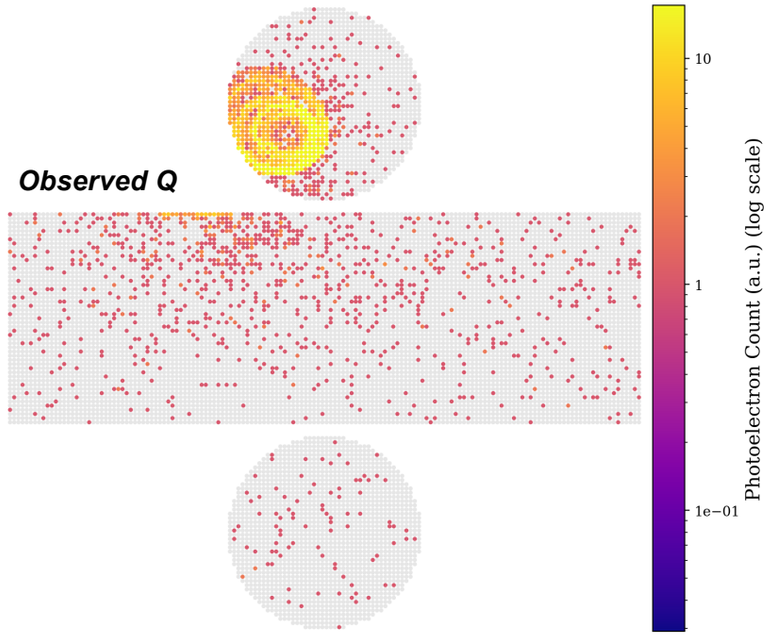
EXAMPLE OF DATA-LIKE EVENTS (TOP) AND ASSOCIATED PREDICTIONS (BOTTOM)

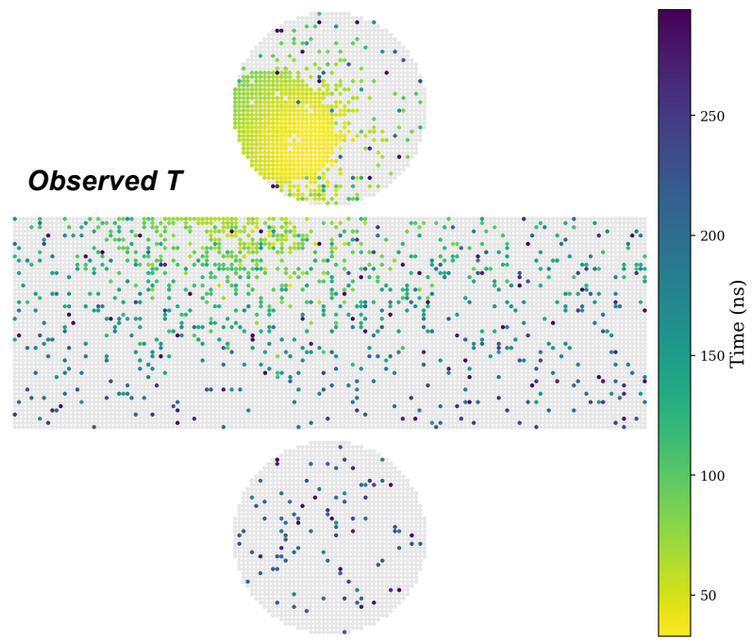






$$\mathscr{L}_{charge} = \sum_i \left[n_i^{pred} - n_i^{obs} \log n_i^{pred} + \log(n_i^{obs}!) \right]$$



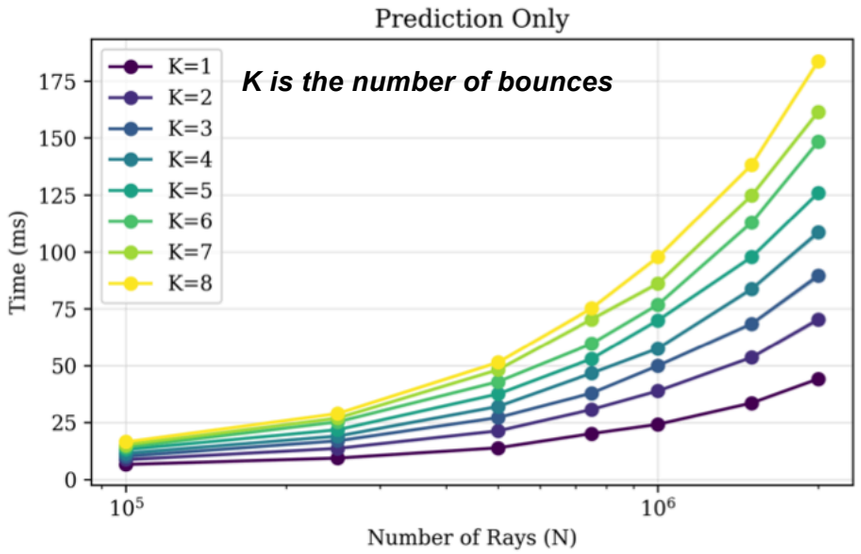


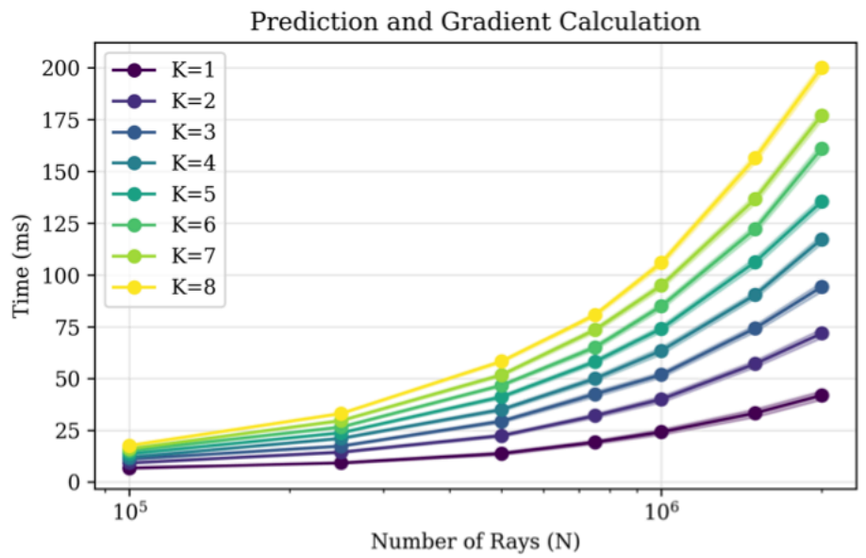
$$\mathscr{L}_{charge} = \sum_i \left[n_i^{pred} - n_i^{obs} \log n_i^{pred} + \log(n_i^{obs}!) \right]$$

$$\mathscr{L}time = \sum i \in H - \log p_{first}^{(i)}(t_i^{obs}) \qquad H = i : n_i^{obs} > 0$$

- The combination of JAX, JIT compiled kernels and modern GPUs allows generating accurate predictions in O(ms).
- Gradient calculation adds a small overhead.
- We play some tricks to minimise JIT overhead → Rays are padded in fixed-sized buckets, we ignore padded rays.

CALCULATIONS USING NVIDIA A100





Results

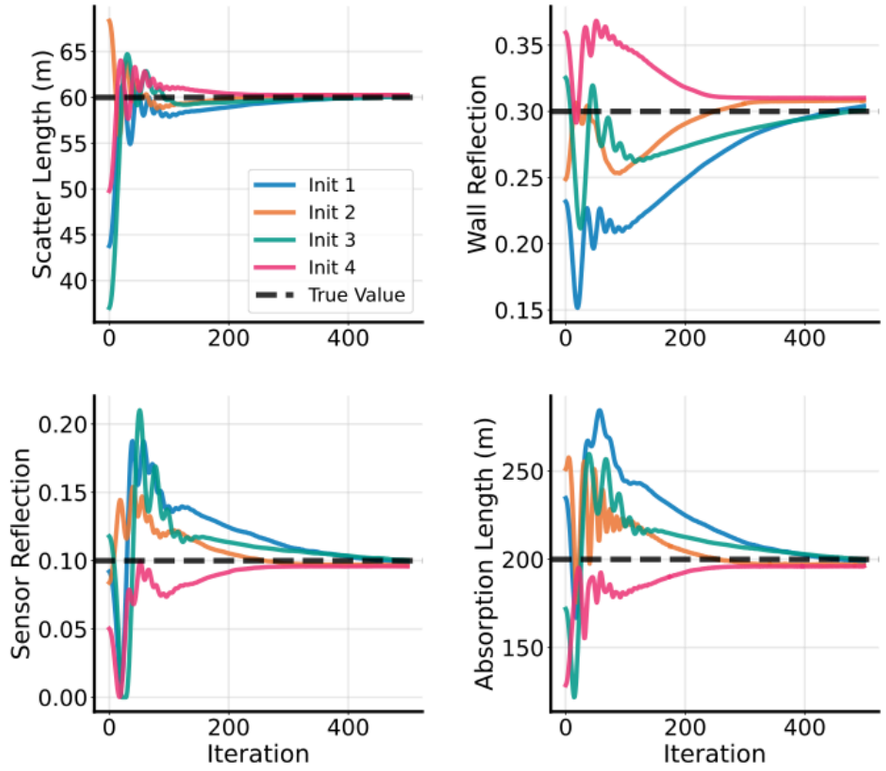
Calibration objective

minimise $\mathcal{L}_{\text{counts}}$ via Adam · gradient from differentiable simulation

Campaign 1 · optical properties

Parameters: λ_s, λ_a, R_w, R_s

Source: central laser, vertically downward



Calibration objective

minimise $\mathcal{L}_{\text{counts}}$ via Adam · gradient from differentiable simulation

Campaign 1 · optical properties

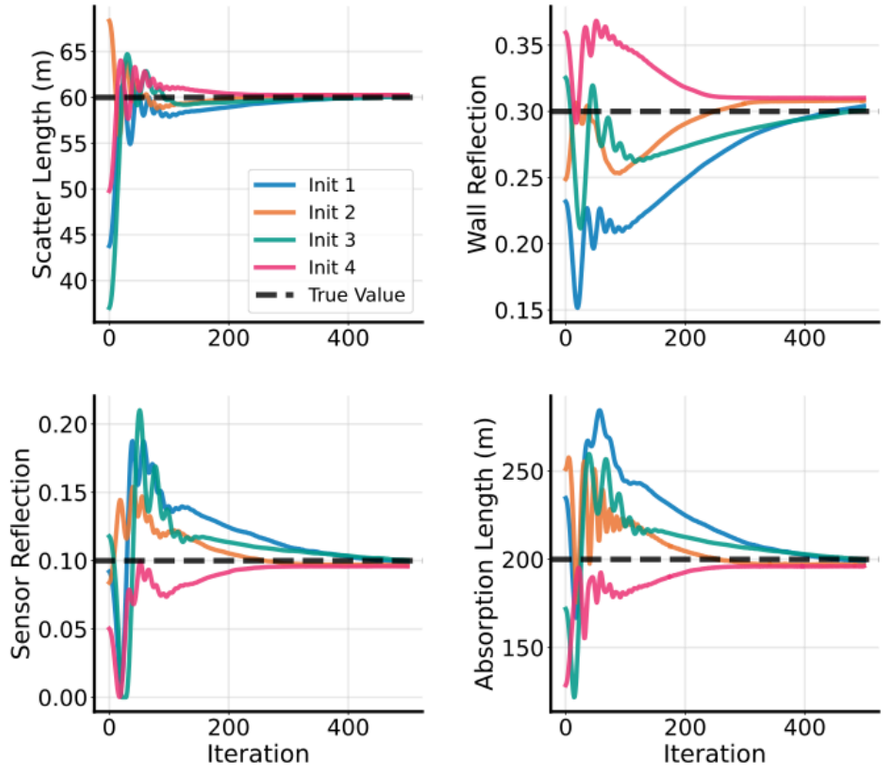
Parameters: λ_s, λ_a, R_w, R_s

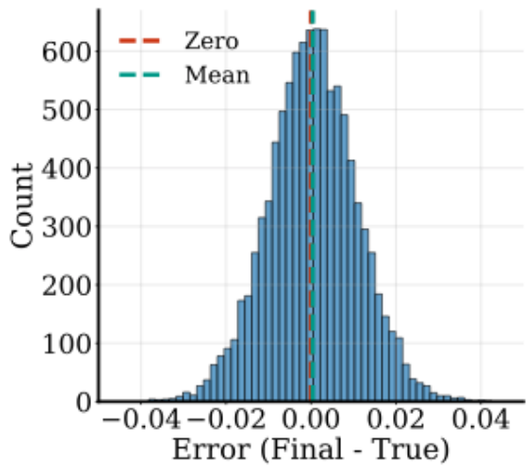
Source: central laser, vertically downward

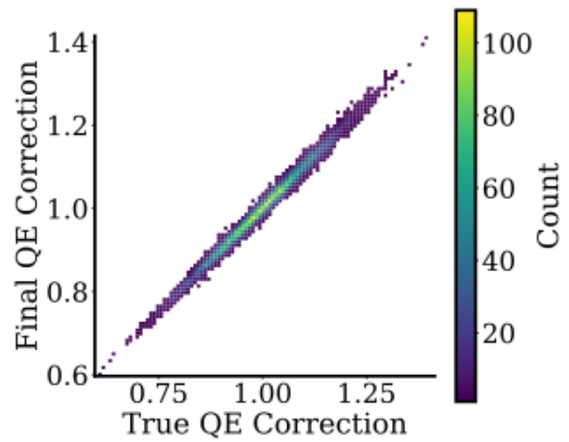
Campaign 2 · sensor efficiency

Parameters: QE_j per sensor (~10^4)

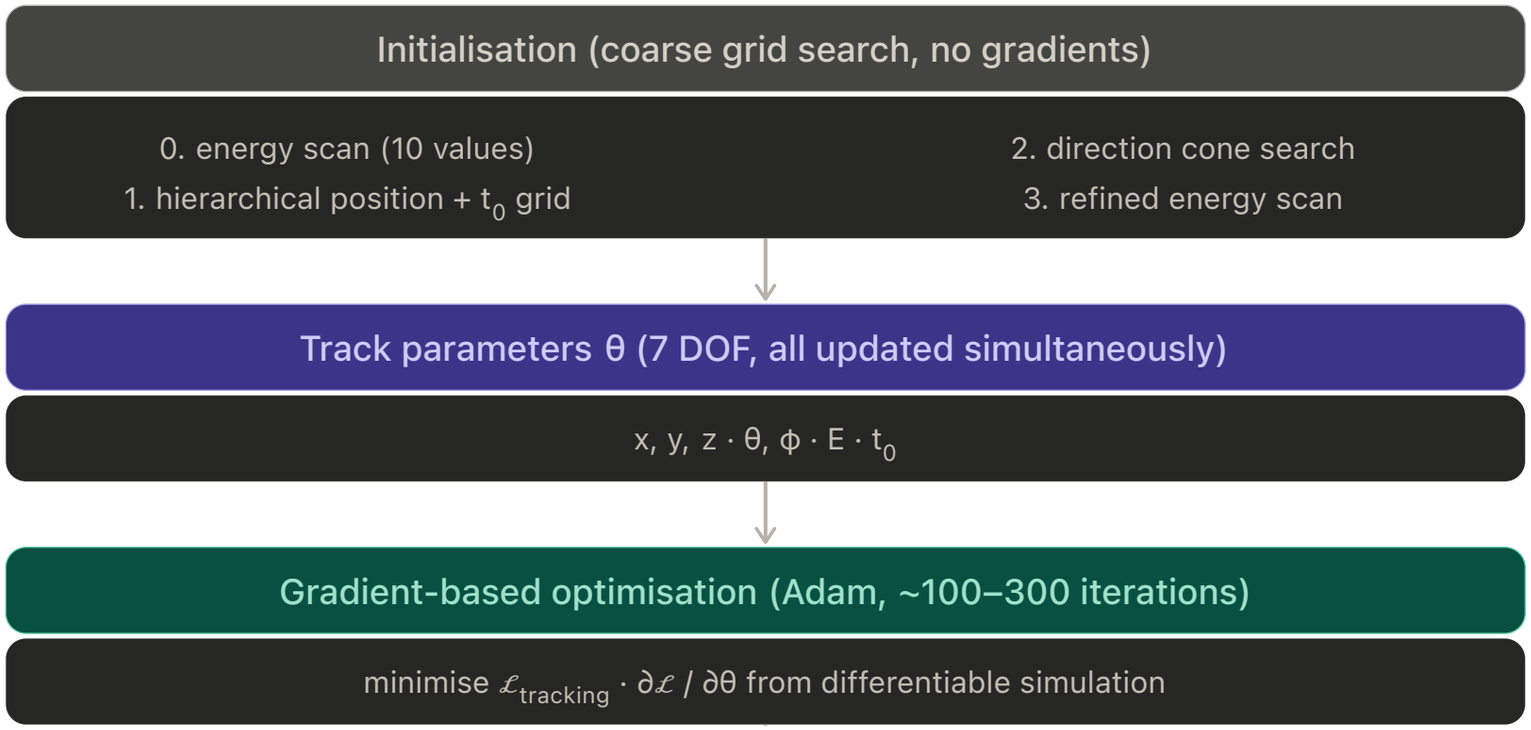
Sources: 15 isotropic (1 central + 14 grid)







HOW DOES IT WORK?



Takes ~1/3 of total optimisation time

Iterations depend on number of rays, among others

HOW DOES IT WORK?

Initialisation (coarse grid search, no gradients)

0. energy scan (10 values)
1. hierarchical position + t_0 grid
2. direction cone search
3. refined energy scan

Takes ~1/3 of total optimisation time

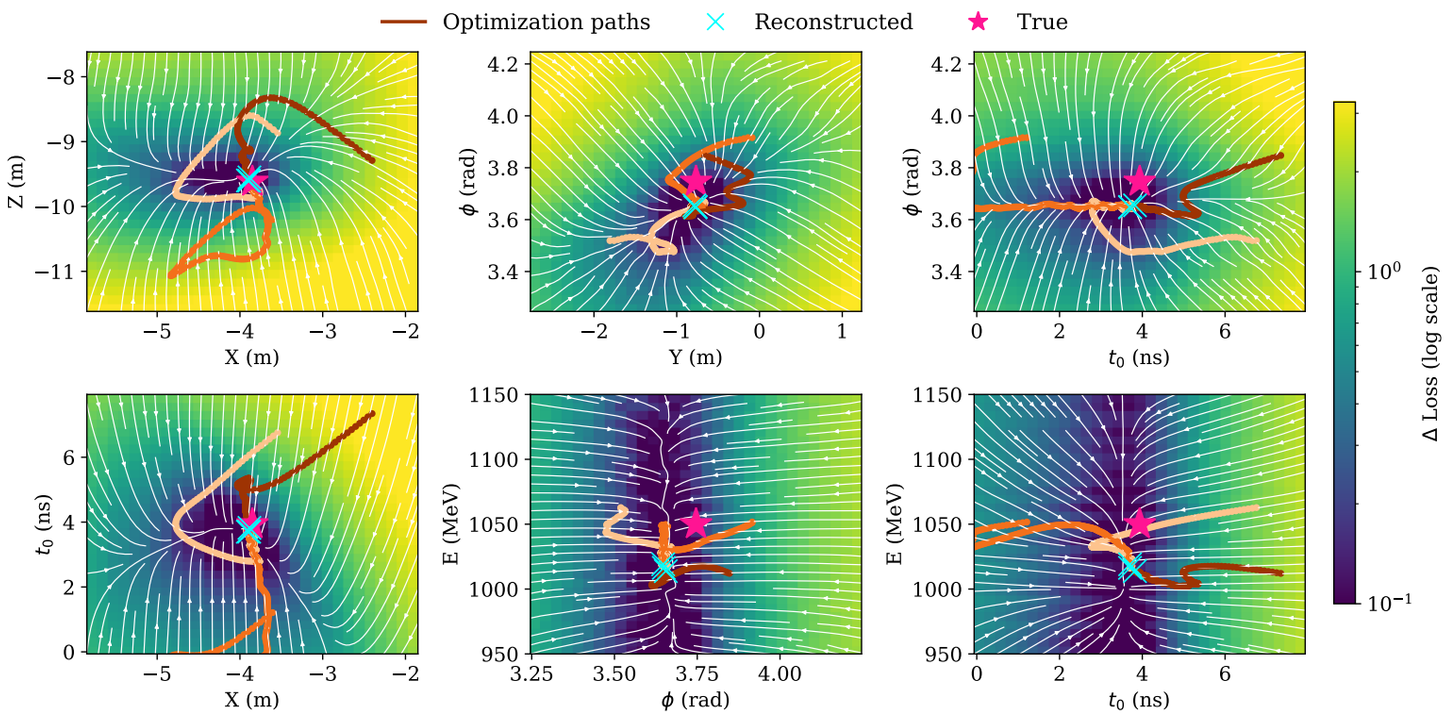
Track parameters θ (7 DOF, all updated simultaneously)

x, y, z · θ, ϕ · E · t_0

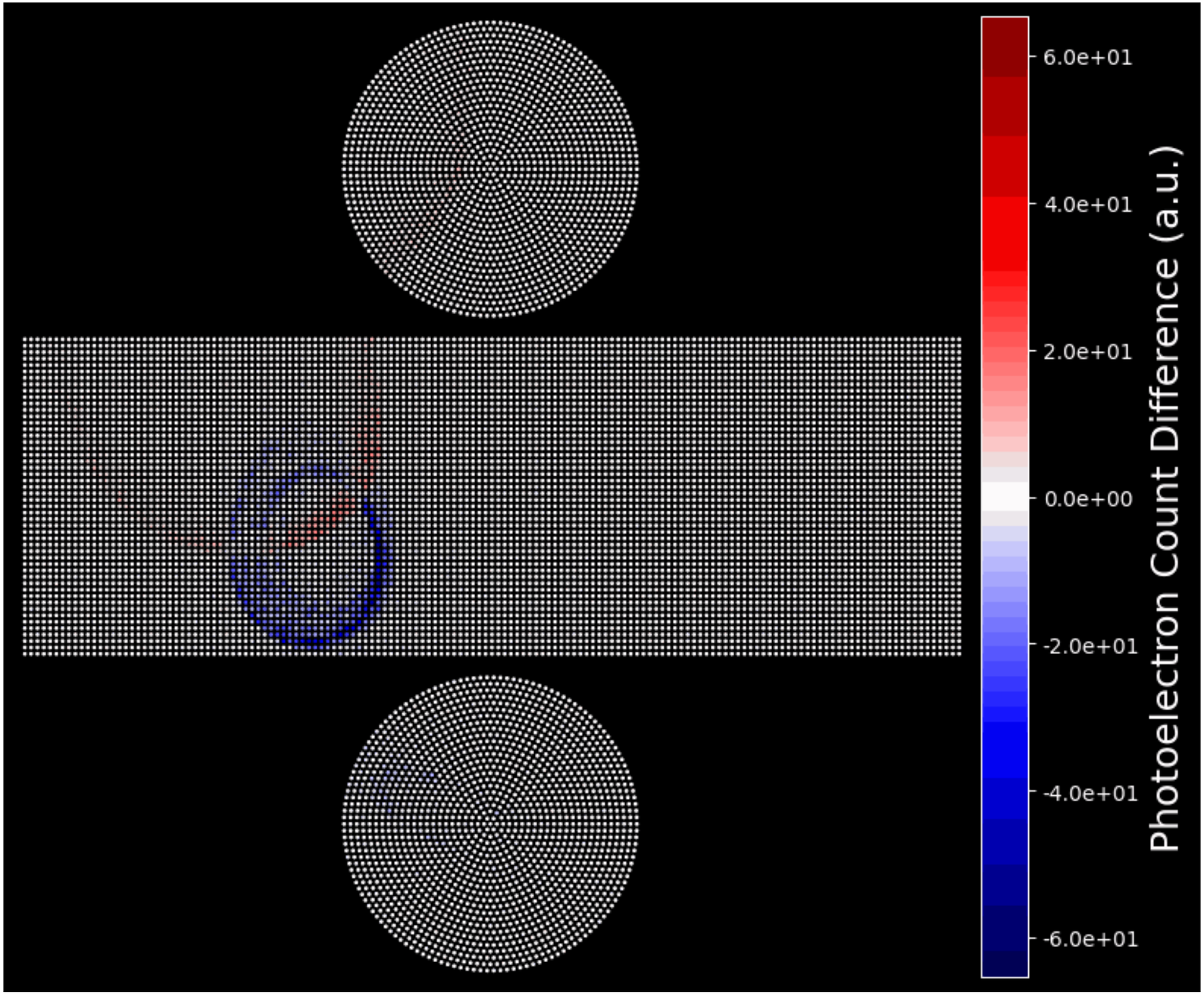
Gradient-based optimisation (Adam, ~100–300 iterations)

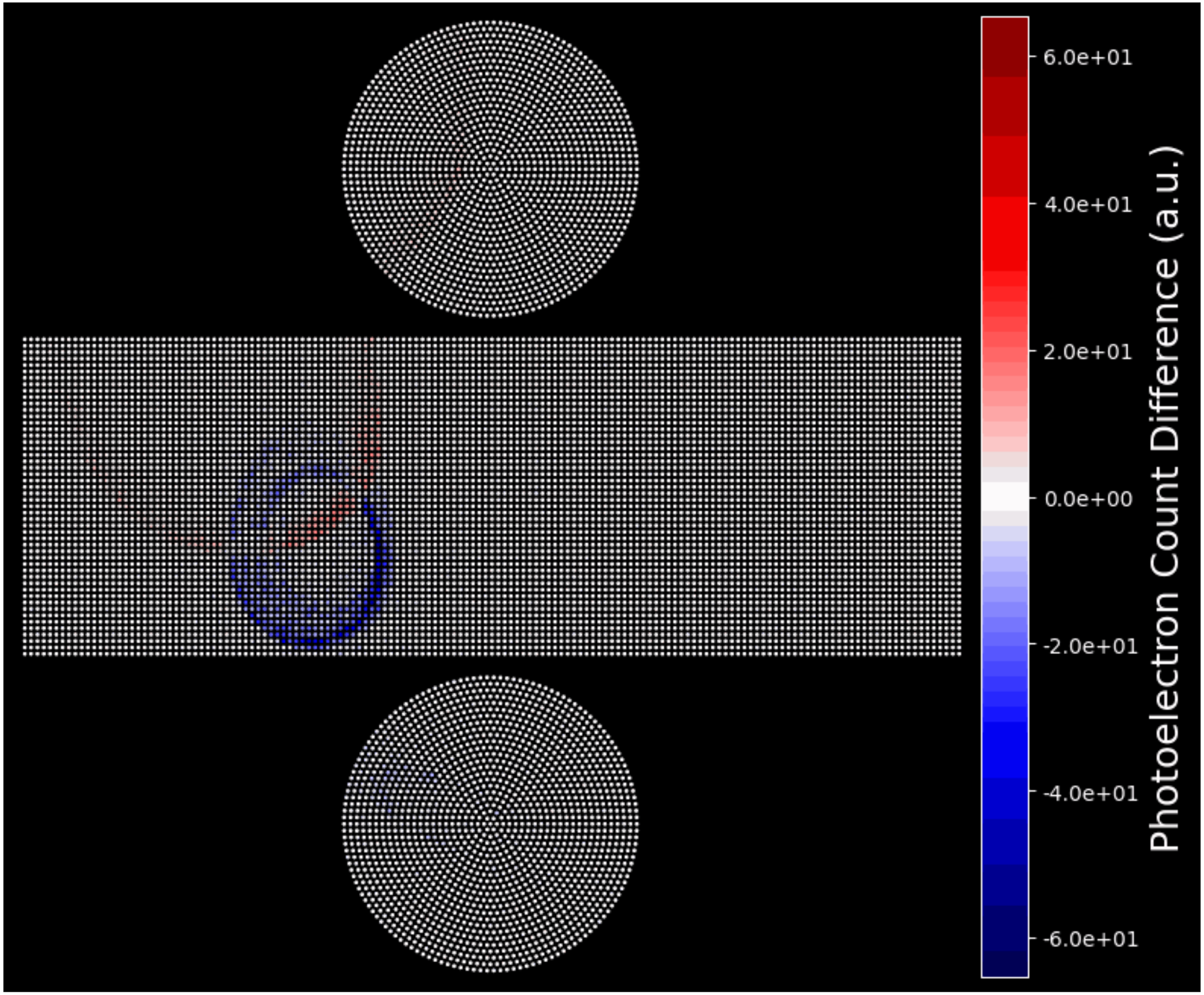
minimise $\mathcal{L}_{\text{tracking}}$ · ∂$\mathcal{L}$ / ∂θ from differentiable simulation

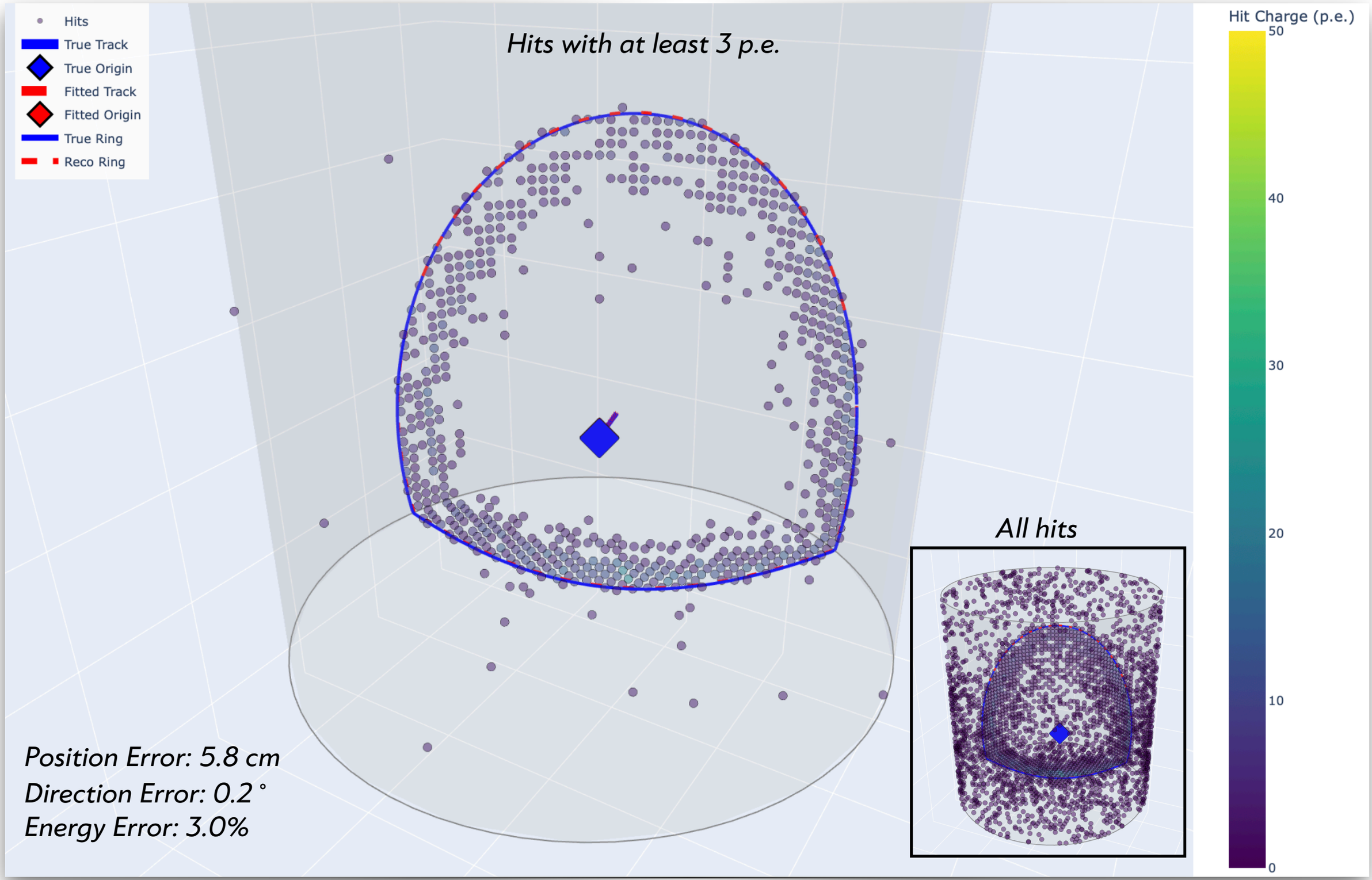
Iterations depend on number of rays, among others

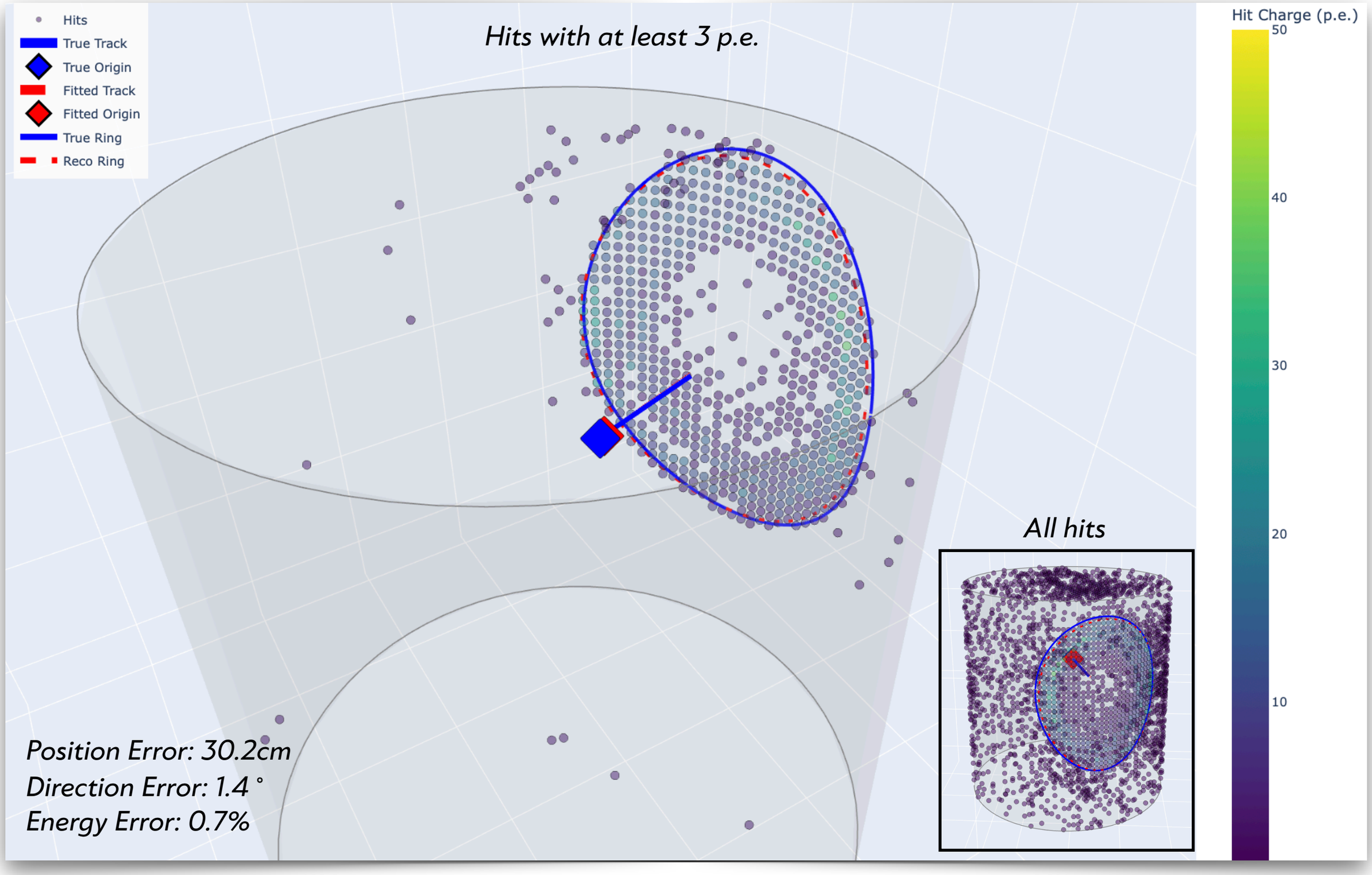


Example optimisation for the same event from three different initial guess points


6.0e+01
4.0e+01
2.0e+01
0.0e+00
-2.0e+01
-4.0e+01
-6.0e+01
Photoelectron Count Difference (a.u.)


6.0e+01
4.0e+01
2.0e+01
0.0e+00
-2.0e+01
-4.0e+01
-6.0e+01
Photoelectron Count Difference (a.u.)


Hits
True Track
True Origin
Fitted Track
Fitted Origin
True Ring
Reco Ring
Hits with at least 3 p.e.
Hit Charge (p.e.)
50
40
30
20
10
0
All hits
Position Error: 5.8 cm
Direction Error: 0.2 °
Energy Error: 3.0%


Hits
True Track
True Origin
Fitted Track
Fitted Origin
True Ring
Reco Ring
Hits with at least 3 p.e.
Hit Charge (p.e.)
50
40
30
20
10
All hits
Position Error: 30.2cm
Direction Error: 1.4 °
Energy Error: 0.7%

How does this compare to SK reconstruction software?

Performance (1 GeV muon, SK-like)

	Vertex	Direction	Momentum	Time / event
LUCiD	15.6 cm	1.1°	1.5%	~10 s
fiTQun	15.8 cm	1.0°	2.3%	~30 s

Approximate comparison using **very similar dataset*

What is next?


O. Alterkait
(U. Tufts/IAIFI)


R. Shah
(Kavli IPMU)


N. Brito
(E. Polytechnique)


R. Tsuchii
(Tokyo Tech)

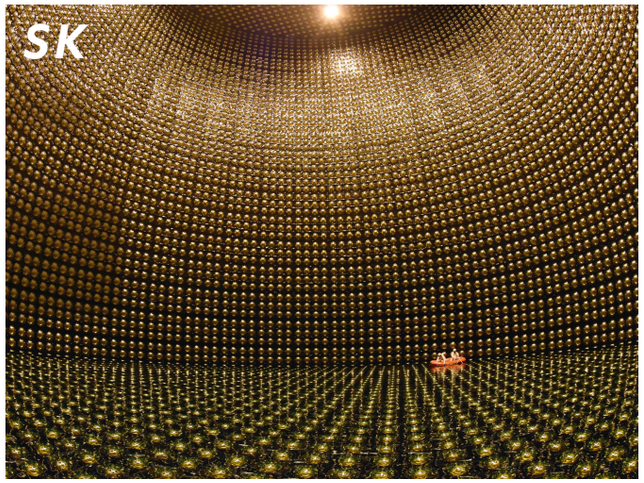

Z. Xie
(U. Minnesota)

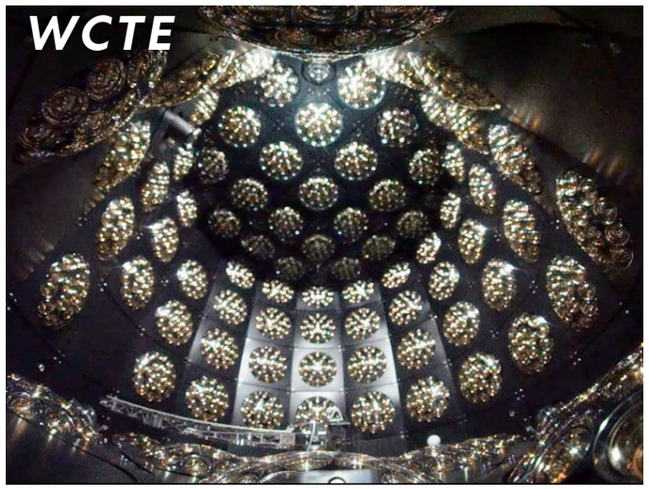

F. Morais
(Amherst College)


T. Gressier
(Uppsala U.)

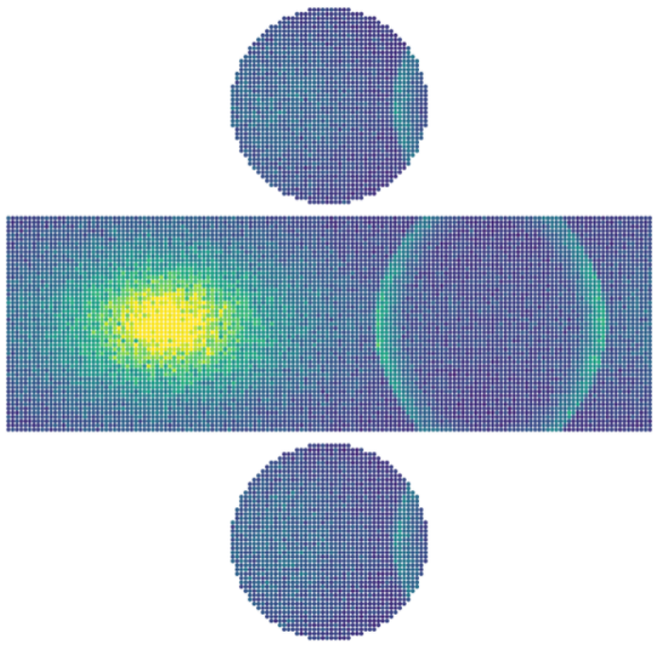

C. Jesús-Valls
(CERN)

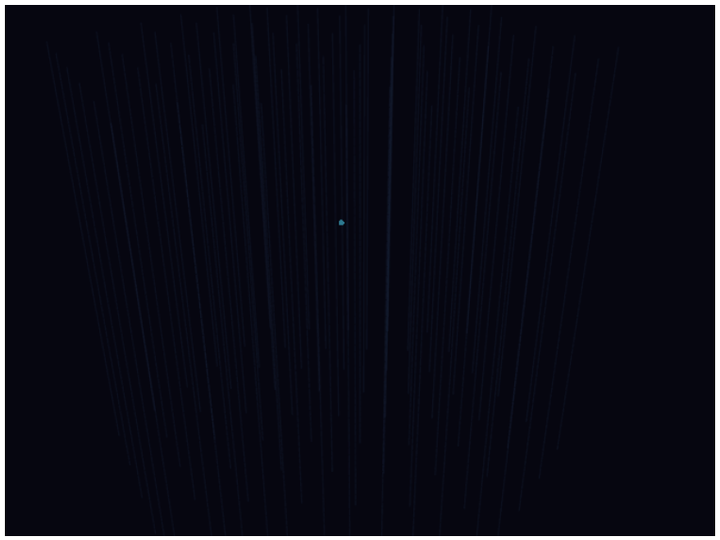
Highest priority is to demonstrate the technique using data





But we are also expanding into other technologies: Now we support scintillation and Telescopes






O. Alterkait
(U. Tufts/IAIFI)


R. Shah
(Kavli IPMU)


N. Brito
(E. Polytechnique)


R. Tsuchii
(Tokyo Tech)

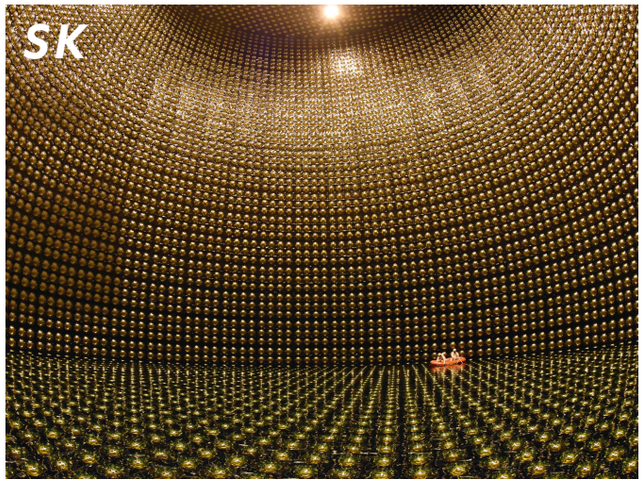

Z. Xie
(U. Minnesota)

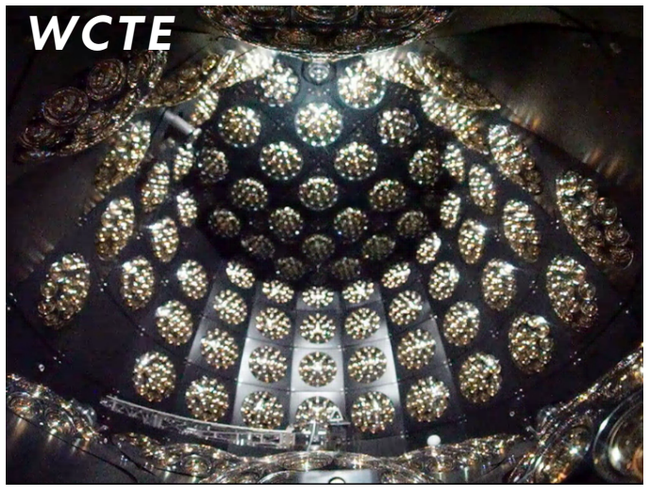

F. Morais
(Amherst College)


T. Gressier
(Uppsala U.)

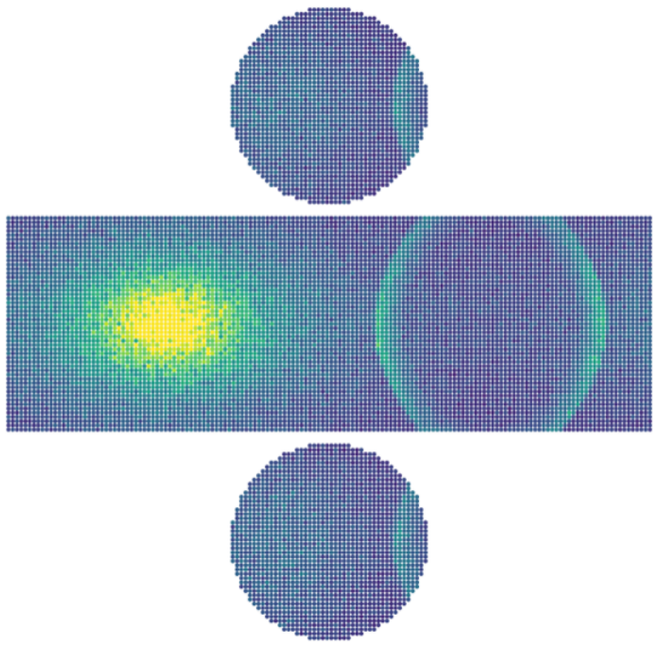

C. Jesús-Valls
(CERN)

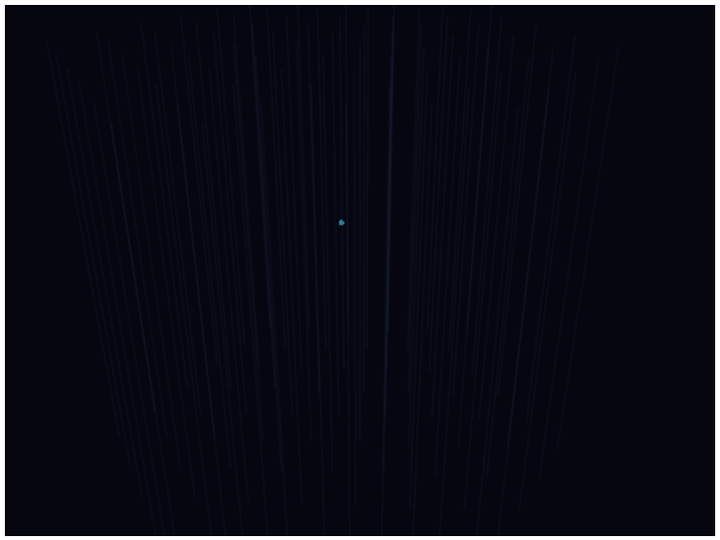
Highest priority is to demonstrate the technique using data





But we are also expanding into other technologies: Now we support scintillation and Telescopes





O. Alterkait
(U. Tufts/IAIFI)

C. Jesús-Valls
(CERN)

LUCiD implemented in JAX: https://arxiv.org/abs/2602.24129
https://github.com/CIDeR-ML/LUCiD/

P. Granger
(CERN)

Y. Chen
(SLAC)

LArNDsim - Now in JAX: https://github.com/pgranger23/larnd-sim-jax
(Pixel readout)

Paper coming in 2026

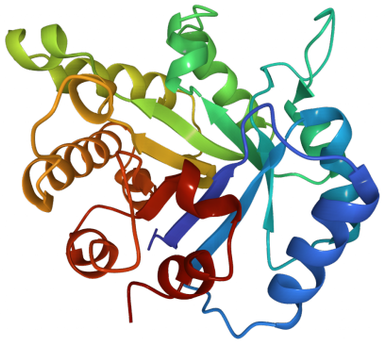
O. Alterkait
(U. Tufts/IAIFI)

Another LArTPC framework in JAX: https://github.com/OmarAlterkait/JAXTPC
(Wire readout)

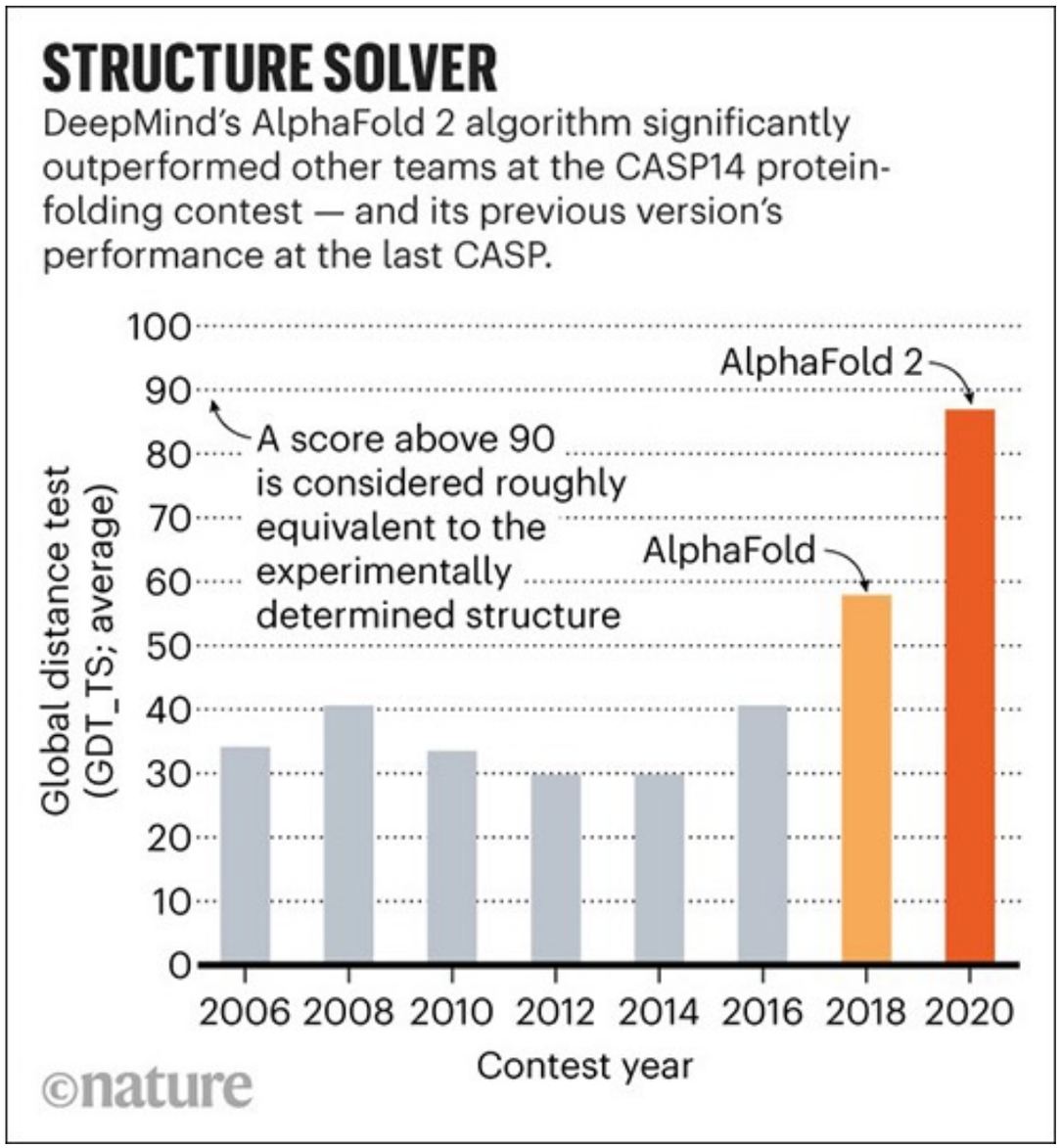
- ***Discussions ongoing to merge LArTPC work in a single framework.***
- ***Most technical challenges to fully model LArTPC (Charge + Light) are solved... but one would need a dedicated effort to merge.***

The power of scientific datasets

Learning from a different field



A CHANGE OF PARADIGM: DATA > DOMAIN KNOWLEDGE



- *In other domains, AI-tools have shown that, if good datasets exist, one can enable transformative performance boosts.*
- *HEP is uniquely poised to generate massive datasets of exceptional quality.*
- *HEP communities are often closed: Software and data are private → Surprising absence of good datasets/ benchmarks*

What can we do about it?

DORAEMON | Open Dataset Challenge

OpenDC Data Hub Documentation Software Updates

DORAEMON Open Dataset Challenge

Public simulated datasets and shared evaluation tasks for AI methods in neutrino physics.

Explore OpenDC | Read documentation

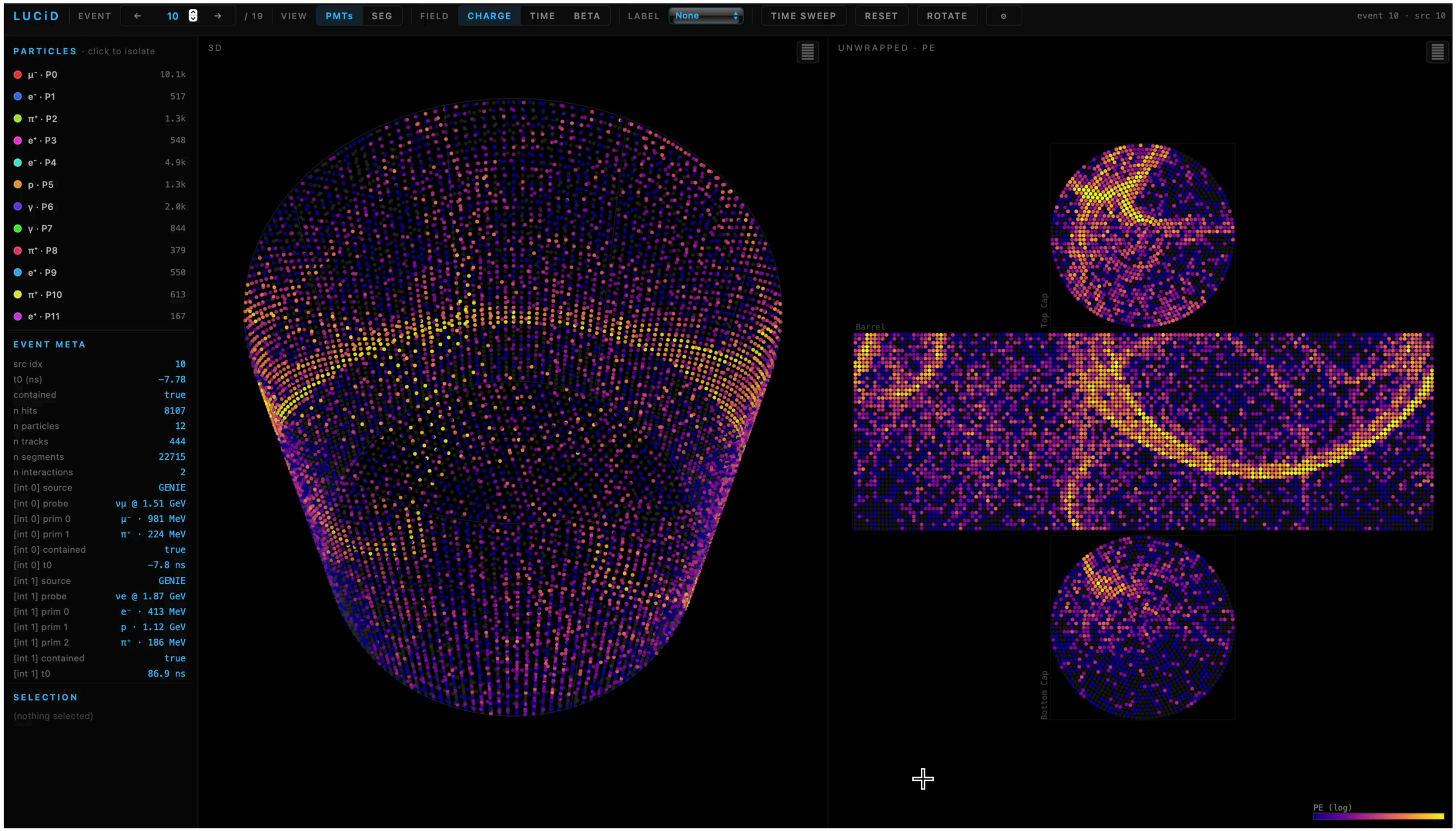
https://www.deeplearnphysics.org/doraemon_site/

Site under construction!

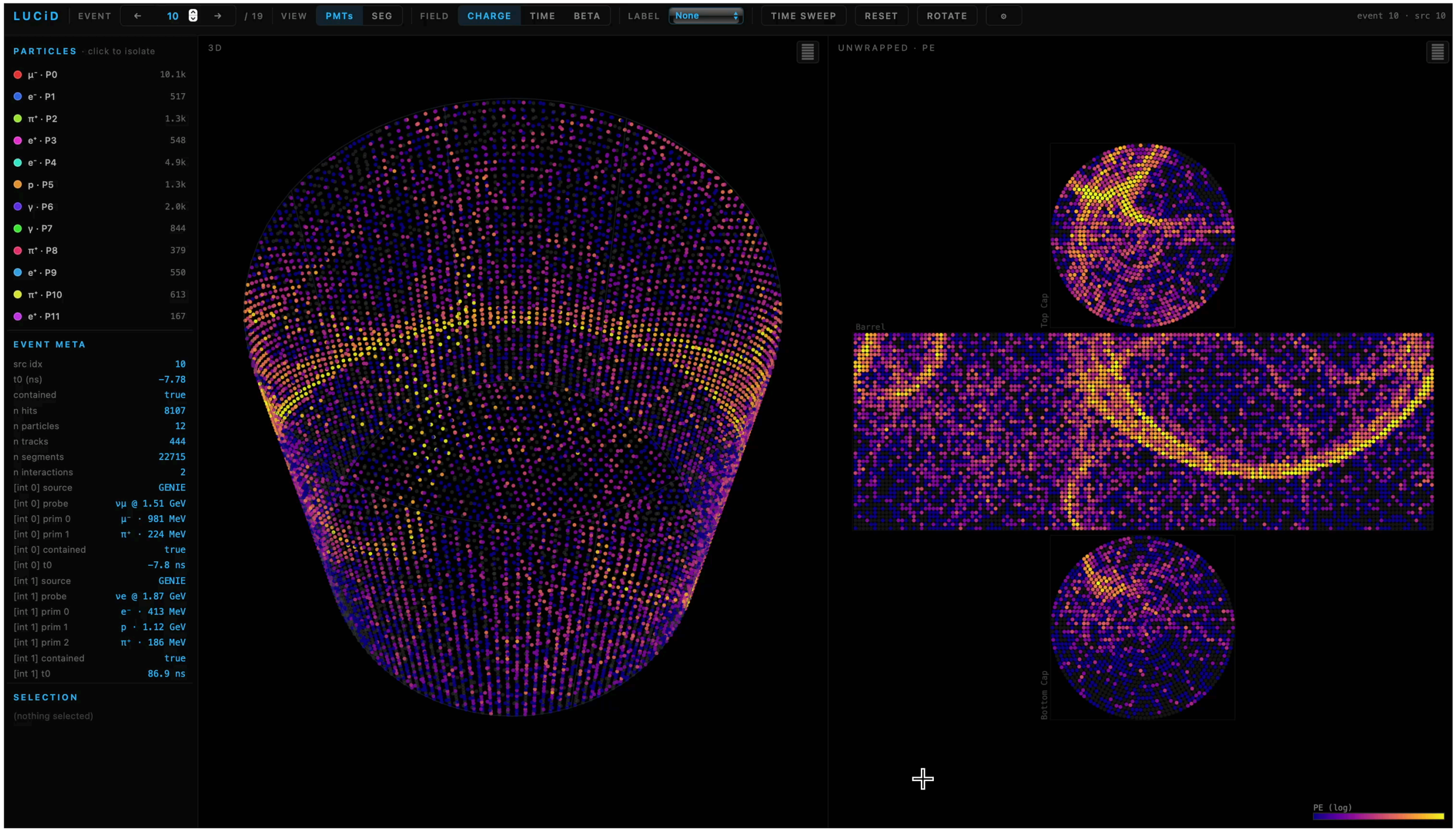
CHANGING THE FIELD APPROACH

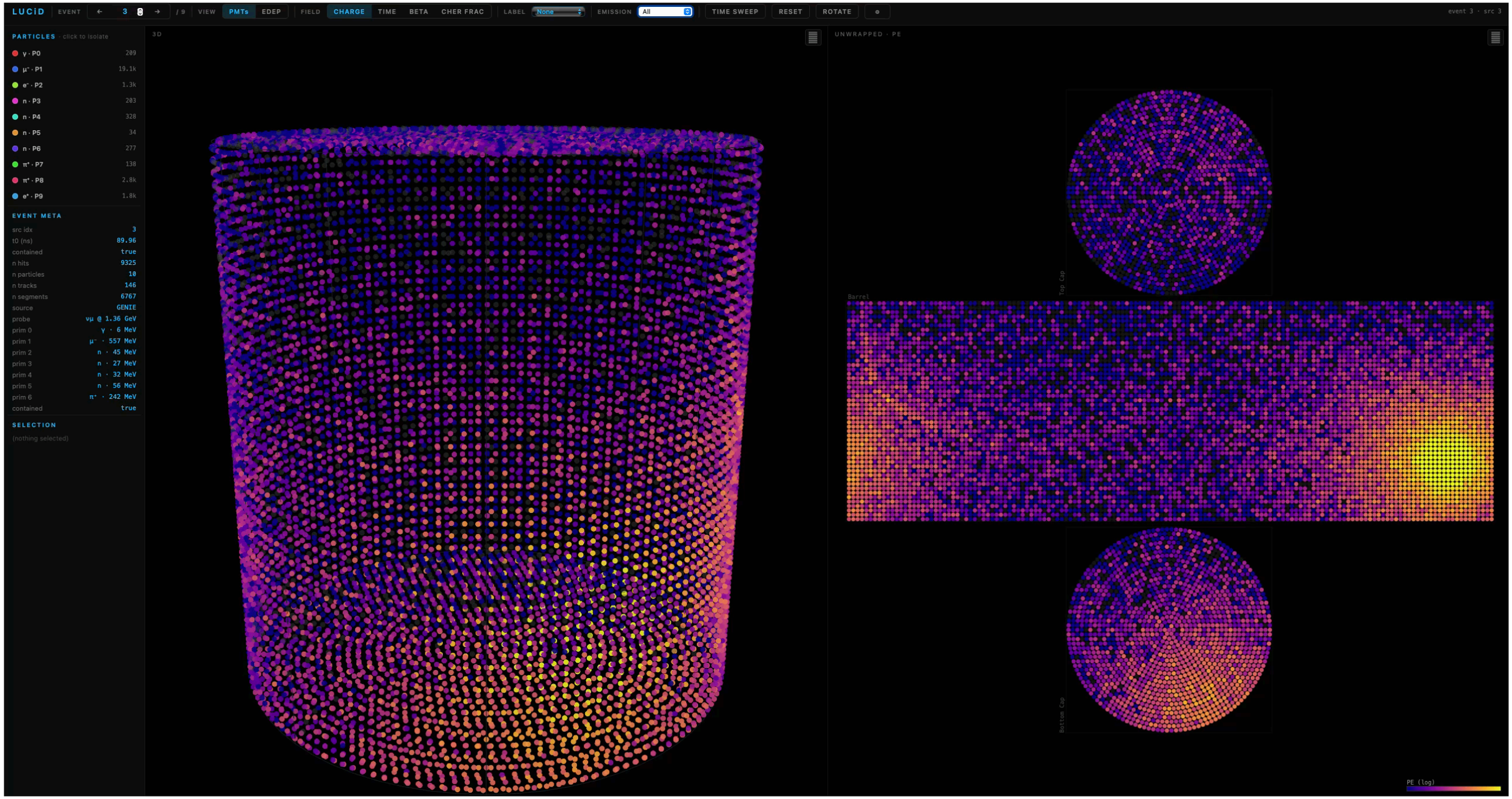
- *We are building a site to coordinate distribution of datasets, publication of challenges and evaluation of benchmarks in neutrino physics.*
- *Wide approach: From calibration/reconstruction challenges to cross-detector transfer, interaction modelling…*
- *Labels as downstream views: One can redefine the semantic labels without re-generating the datasets.*

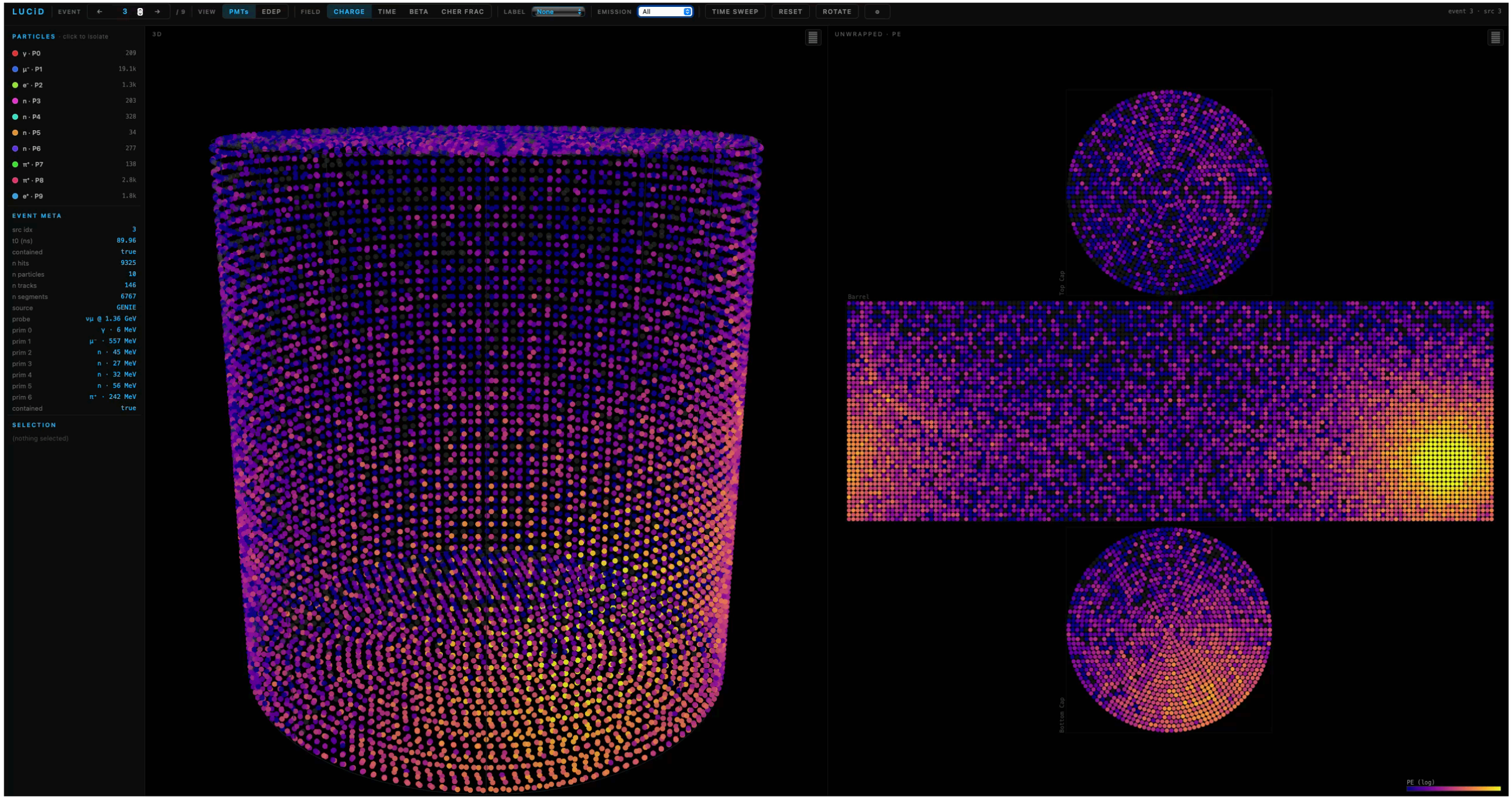
A SNEAK PEEK


LUCiD
EVENT
10
/ 19
VIEW
PMTs
SEG
FIELD
CHARGE
TIME
BETA
LABEL
None
TIME SWEEP
RESET
ROTATE
event 10 · src 10
PARTICLES · click to isolate
μ⁻ · P0 10.1k
e⁻ · P1 517
π⁺ · P2 1.3k
e⁺ · P3 548
e⁻ · P4 4.9k
p · P5 1.3k
γ · P6 2.0k
γ · P7 844
π⁺ · P8 379
e⁺ · P9 550
π⁺ · P10 613
e⁺ · P11 167
EVENT META
src idx 10
t0 (ns) −7.78
contained true
n hits 8107
n particles 12
n tracks 444
n segments 22715
n interactions 2
[int 0] source GENIE
[int 0] probe νμ @ 1.51 GeV
[int 0] prim 0 μ⁻ · 981 MeV
[int 0] prim 1 π⁺ · 224 MeV
[int 0] contained true
[int 0] t0 −7.8 ns
[int 1] source GENIE
[int 1] probe νe @ 1.87 GeV
[int 1] prim 0 e⁻ · 413 MeV
[int 1] prim 1 p · 1.12 GeV
[int 1] prim 2 π⁺ · 186 MeV
[int 1] contained true
[int 1] t0 86.9 ns
SELECTION
(nothing selected)
3D
UNWRAPPED · PE
Top Cap
Barrel
Bottom Cap
PE (log)

A SNEAK PEEK

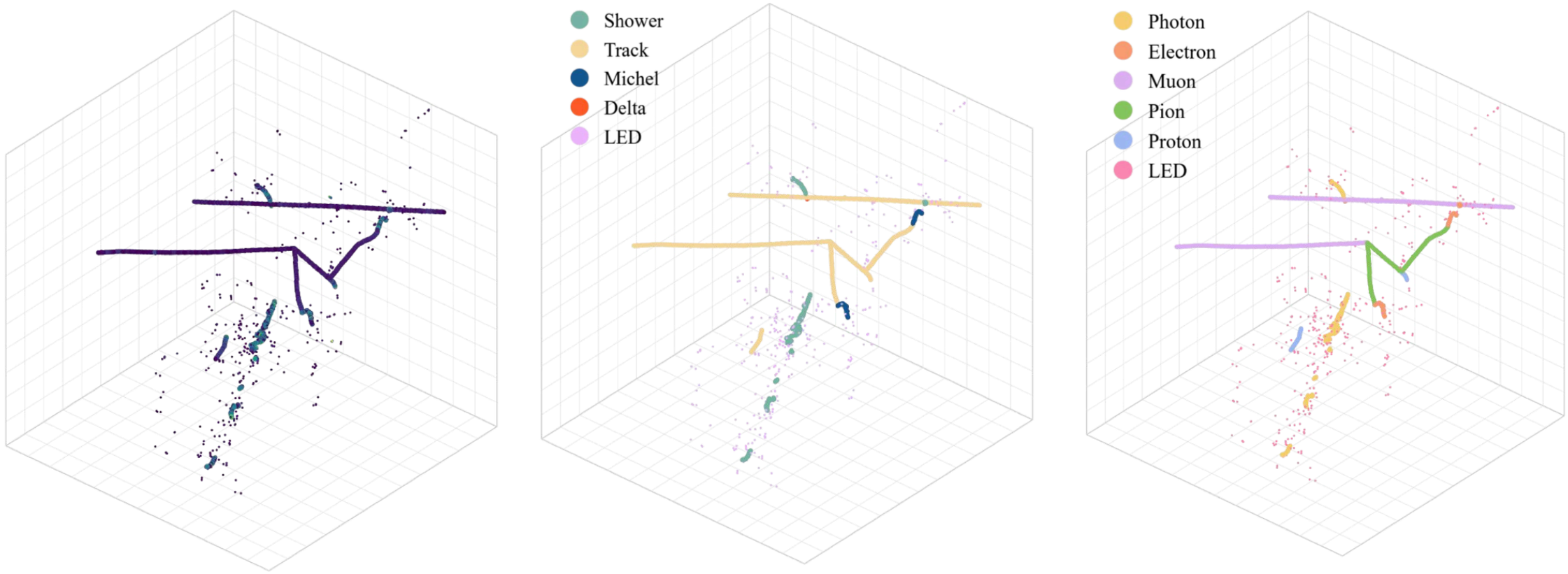

LUCiD
EVENT
10
/ 19
VIEW
PMTs
SEG
FIELD
CHARGE
TIME
BETA
LABEL
None
TIME SWEEP
RESET
ROTATE
event 10 · src 10
PARTICLES · click to isolate
μ⁻ · P0 10.1k
e⁻ · P1 517
π⁺ · P2 1.3k
e⁺ · P3 548
e⁻ · P4 4.9k
p · P5 1.3k
γ · P6 2.0k
γ · P7 844
π⁺ · P8 379
e⁺ · P9 550
π⁺ · P10 613
e⁺ · P11 167
EVENT META
src idx 10
t0 (ns) −7.78
contained true
n hits 8107
n particles 12
n tracks 444
n segments 22715
n interactions 2
[int 0] source GENIE
[int 0] probe νμ @ 1.51 GeV
[int 0] prim 0 μ⁻ · 981 MeV
[int 0] prim 1 π⁺ · 224 MeV
[int 0] contained true
[int 0] t0 −7.8 ns
[int 1] source GENIE
[int 1] probe νe @ 1.87 GeV
[int 1] prim 0 e⁻ · 413 MeV
[int 1] prim 1 p · 1.12 GeV
[int 1] prim 2 π⁺ · 186 MeV
[int 1] contained true
[int 1] t0 86.9 ns
SELECTION
(nothing selected)
3D
UNWRAPPED · PE
Top Cap
Barrel
Bottom Cap
PE (log)


LUCiD
EVENT
3
/ 9
VIEW
PMTs
EDEP
FIELD
CHARGE
TIME
BETA
CHER FRAC
LABEL
None
EMISSION
All
TIME SWEEP
RESET
ROTATE
event 3 · src 3
PARTICLES · click to isolate
γ · P0 209
μ⁺ · P1 19.1k
e⁻ · P2 1.3k
n · P3 203
n · P4 328
n · P5 34
n · P6 277
π⁺ · P7 138
π⁺ · P8 2.8k
e⁺ · P9 1.8k
EVENT META
src idx 3
t0 (ns) 89.96
contained true
n hits 9325
n particles 10
n tracks 146
n segments 6767
source GENIE
probe νμ @ 1.36 GeV
prim 0 γ · 6 MeV
prim 1 μ⁻ · 557 MeV
prim 2 n · 45 MeV
prim 3 n · 27 MeV
prim 4 n · 32 MeV
prim 5 n · 56 MeV
prim 6 π⁺ · 242 MeV
contained true
SELECTION
(nothing selected)
3D
UNWRAPPED · PE
Top Cap
Barrel
Bottom Cap
PE (log)


LUCiD
EVENT
3
/ 9
VIEW
PMTs
EDEP
FIELD
CHARGE
TIME
BETA
CHER FRAC
LABEL
None
EMISSION
All
TIME SWEEP
RESET
ROTATE
event 3 · src 3
PARTICLES · click to isolate
γ · P0 209
μ⁺ · P1 19.1k
e⁻ · P2 1.3k
n · P3 203
n · P4 328
n · P5 34
n · P6 277
π⁺ · P7 138
π⁺ · P8 2.8k
e⁺ · P9 1.8k
EVENT META
src idx 3
t0 (ns) 89.96
contained true
n hits 9325
n particles 10
n tracks 146
n segments 6767
source GENIE
probe νμ @ 1.36 GeV
prim 0 γ · 6 MeV
prim 1 μ⁻ · 557 MeV
prim 2 n · 45 MeV
prim 3 n · 27 MeV
prim 4 n · 32 MeV
prim 5 n · 56 MeV
prim 6 π⁺ · 242 MeV
contained true
SELECTION
(nothing selected)
3D
UNWRAPPED · PE
Top Cap
Barrel
Bottom Cap
PE (log)

PILARNET https://arxiv.org/abs/2006.01993

- *LArTPC datasets already exist and have been in a number of AI/ML papers.*



- *LArTPC datasets already exist and have been in a number of AI/ML papers and are helping to push SSL.*

https://arxiv.org/abs/2502.02558

https://arxiv.org/abs/2604.07037

https://arxiv.org/abs/2512.01324

Conclusions

We are living a new era, the era of AI/ML. Experiments need to re-think how they do science in this rapidly changing environment.

- **Redesigning scientific workflows with AI/ML at the core.** LUCiD is the first end-to-end demonstrator showing that differentiable simulators are ready to become a paradigm-shifting tool in HEP. Momentum is building: several groups are now working to integrate LUCiD into WCTE, SK and Hyper-K.

- **DORAEMON: Large-scale public HEP neutrino datasets for AI.** The generation pipeline is in place; a first release of ~10M fully labeled events will be public soon (mirrored structure in several detector technologies), aiming to transform how new tools are developed and benchmarked in HEP. Valuable datasets for the broader ML community.

Towards the future:

- **We aim to expand to as many experiments as possible, and ignite a community effort. If interested in joining please get in contact!**

cesar.jesus@cern.ch