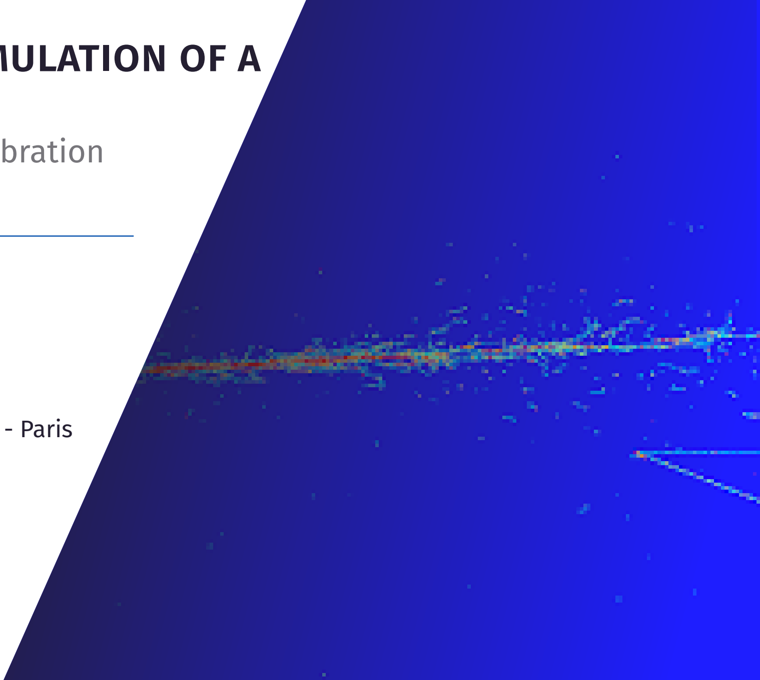
Differentiable Simulation of a Liquid Argon TPC

for multi-dimensional calibration

arXiv:2309.04639

Pierre Granger
granger@apc.in2p3.fr

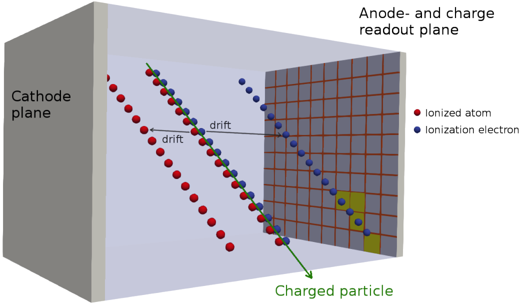
APC (Astroparticule et cosmologie) - Paris

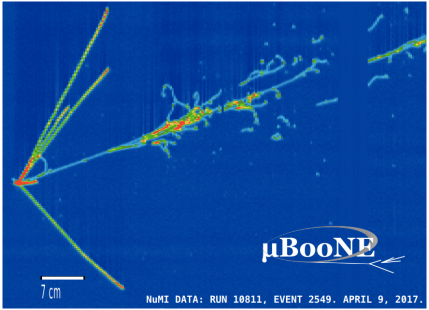


November 28, 2023

Outline

Liquid Argon Time Projection Chambers (LArTPCs)



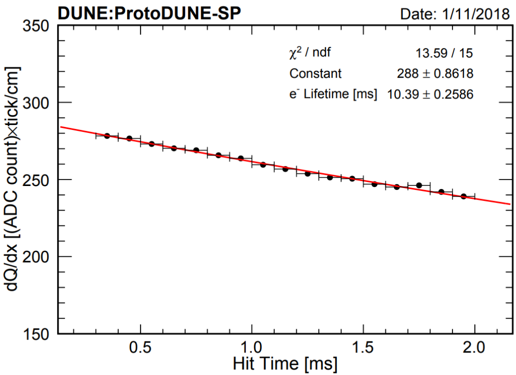


<u>Signal production steps</u>:

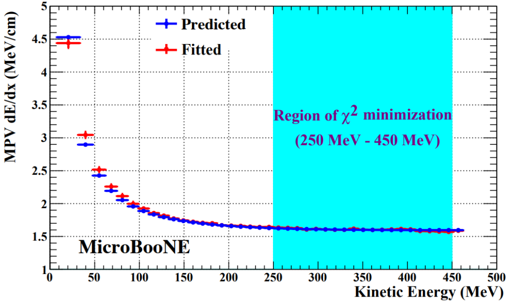
- Argon ionisation
- Ionisation electrons drifted by $\mathbf{E}$ field
- Electrons readout on anode plane

- Allows to get **precise 3D picture** of the interaction
- Relies on **multiple physical processes**
  $\rightarrow$ importance of calibration

TYPICAL LARTPC CALIBRATION



e^- lifetime calibration

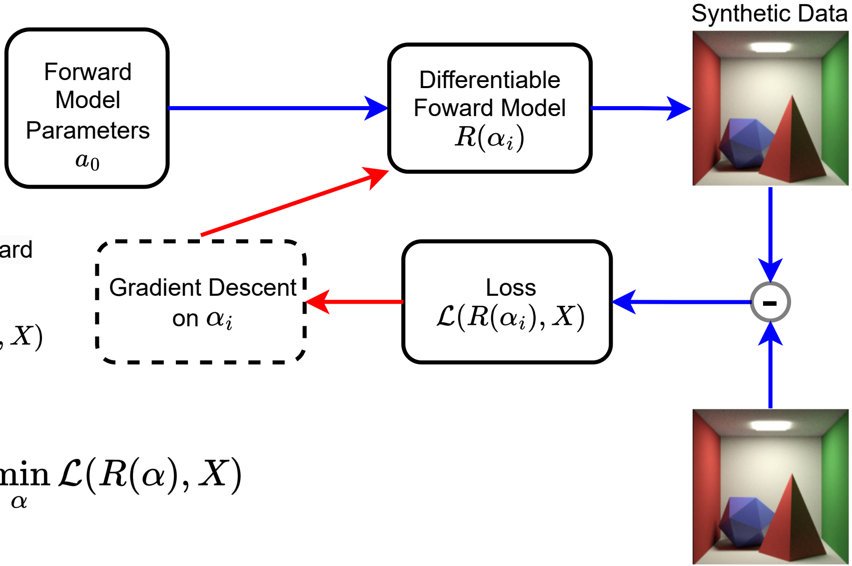


Energy conversion calibration.

Calibration of the different physical parameters are typically done in **different studies.**

→ can be simplified with a differentiable simulator

USING GRADIENT-BASED OPTIMIZATION



Requires forward model to be differentiable: $\nabla_{\alpha_i}\mathcal{L}(R(\alpha_i), X)$

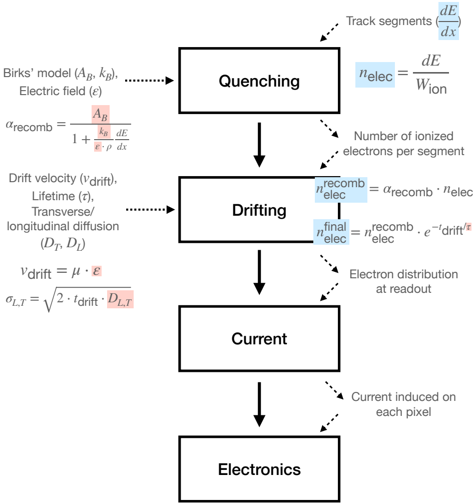
$$\alpha^* = \arg\min_{\alpha} \mathcal{L}(R(\alpha), X)$$

Starting from a non-differentiable LArTPC simulator

Our work: take existing DUNE near-detector simulation (arXiv:2212.09807) and **make it differentiable.**

- Retain physics quality of a tool used collaboration-wide while adding ability to calculate gradient
- Demonstrate the use of this differentiable simulation for **gradient-based calibration**

$\longrightarrow$ How to do it practice?



REWRITING THE SIMULATOR

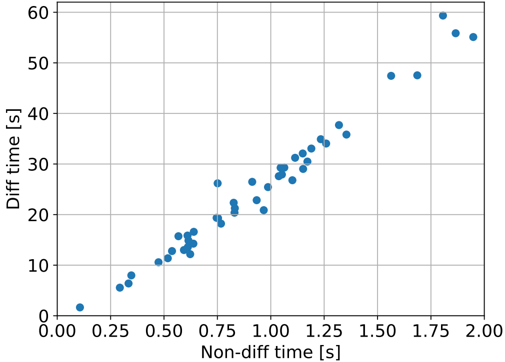
Numba code using **CUDA JIT compiled kernels** → Framework change for diff version:

- Differentiable version rewritten using EagerPy(backend agnostic)/PyTorch, which are based around tensor operations.
- New version rewritten in a **vectorized** way to fit within these frameworks

Performance drawbacks:

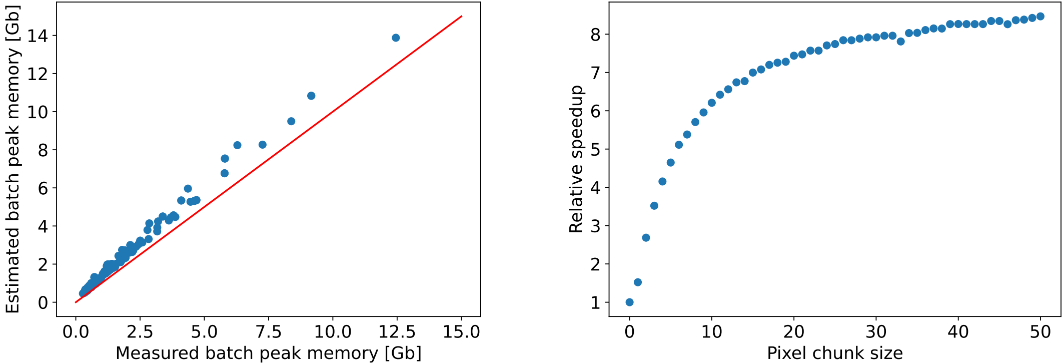
- Use of dense tensors to represent a sparse problem
- Moving from **CUDA JIT compiled dedicated kernel** to a **long chain of generic kernels** (vectorized operations).

→ also impacting memory usage



Memory challenge

Because of the use of dense tensors, **memory** $\propto \Delta_z \times \cot\theta$. (length in drift direction and angle) $\rightarrow$ introduced **automatic memory estimation** for each batch to estimate best pixel chunk size.

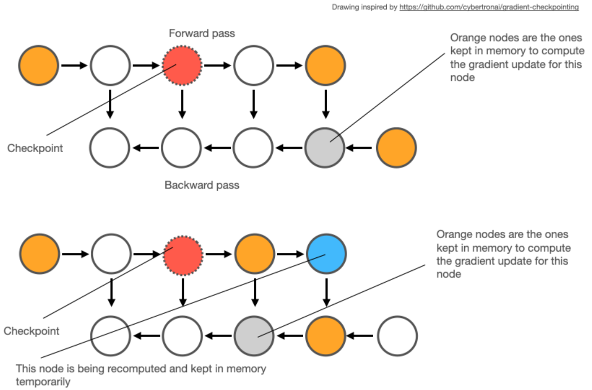


$\rightarrow$ gradient accumulation required by backward pass also saturate the memory

Memory challenge: checkpointing

Reducing the memory used through PyTorch **checkpointing**:

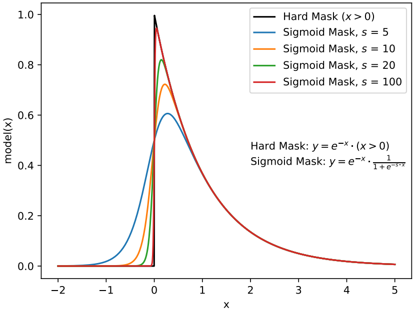
- Gradient accumulation memory intensive due to stored intermediate results
- Trades memory for computation time by recomputing intermediates

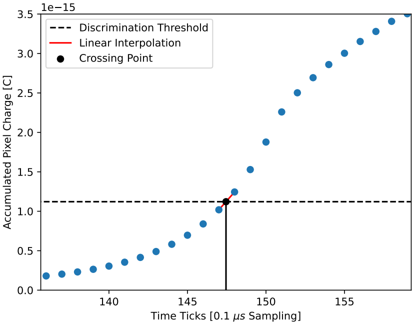


source

Differentiable relaxations

The base simulation contains **discrete operations** → non-differentiable.
Requires differentiable relaxations to be able to get usable gradients.



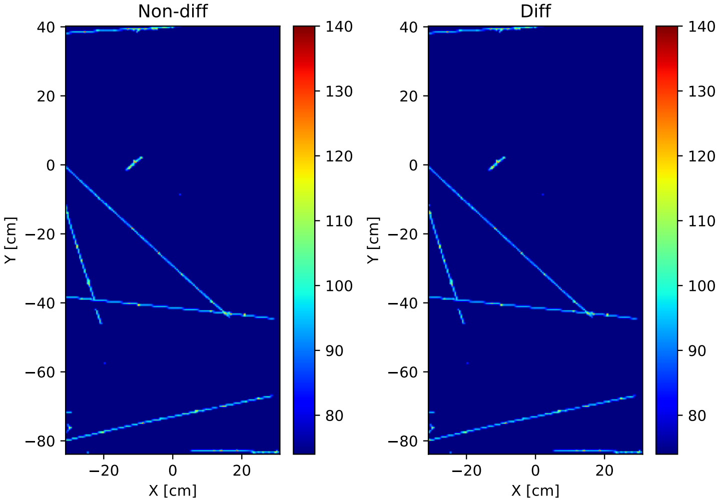


- Cuts (e.g. x > 0) → smooth sigmoid threshold
- Integer operations (e.g. floor division) → floating point (e.g. regular division)
- Discrete sampling → interpolation

Checking the result

Checking that the **relaxations don't modify the simulator output.**

Average deviation of 0.04 ADC/pixel $\rightarrow$ well below the typical noise level of few ADCs.

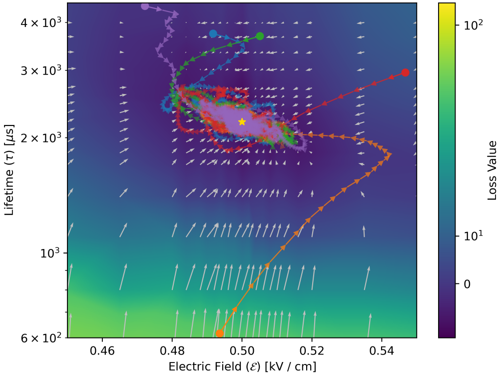


Optimization of the Model Parameters

- Input particle segments (position and energy deposition): χ
- Model parameters: θ
- Differentiable simulation: $f(\chi, \theta)$
- Target data: F_{target}

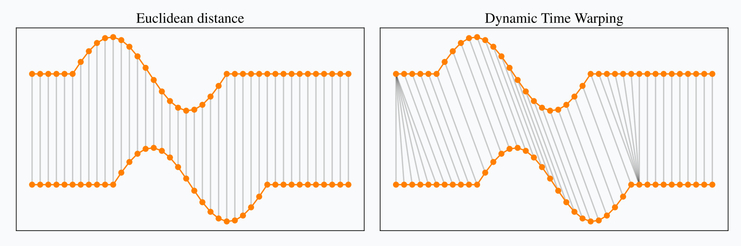
1. Choose the initial parameter values θ_0
2. Run the forward simulation $f(\chi, \theta_0)$
3. Compare the simulation output and the target data with a loss function $\mathcal{L}(f(\chi, \theta_0), F_{\text{target}})$
4. Calculate gradients for the parameters $\nabla_\theta \mathcal{L}(f(\chi, \theta_0), F_{\text{target}})$
5. Update parameter values $\theta_0 \rightarrow \theta_i$ to minimize the loss

   Iterate step 2. to 5.



Optimization choices: Loss function

Loss function choice is crucial for minimization quality



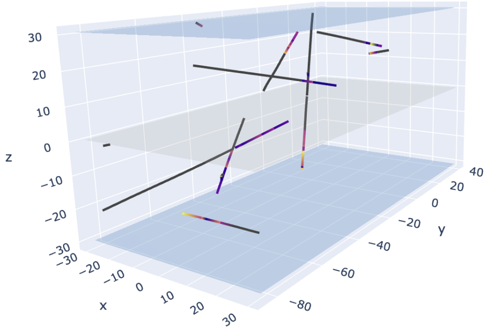
Source

Two main ways of computing the loss:

- Comparison of 3D voxel grids of charges (x, y, t $\rightarrow$ z, q).
  - Difficulty of taking gradients through discrete pixelization.
  - Risk of flat loss if not enough overlap in distributions.
- **Considering the waveforms for each pixel (time sequence) and using Dynamic Time Warping**
  - Using a relaxed SoftDTW version that is differentiable.

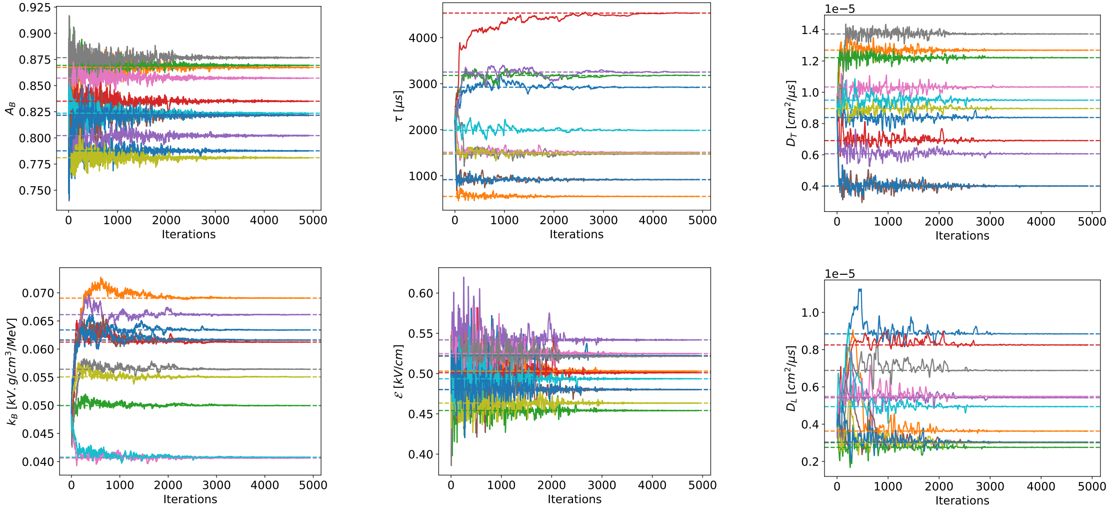
INPUT SAMPLE AND SIMULATED DETECTOR

- Input sample consisting of 1 GeV simulated muon tracks
- Second sample of muons, pions and protons (1 GeV to 3 GeV)
- Geometry of a DUNE ND module: 60 cm $\times$ 60 cm $\times$ 120 cm
- Noise model available in simulator but not used.



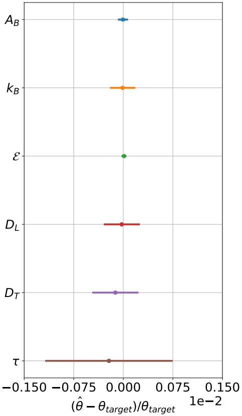
Doing a "closure test" based on simulated data, $F_{\text{target}} = f(\chi, \theta_{\text{target}})$:
$\rightarrow$ Fit of 6 physical parameters **simulteanously** on simulated data for multiple targets and initial values.

Results

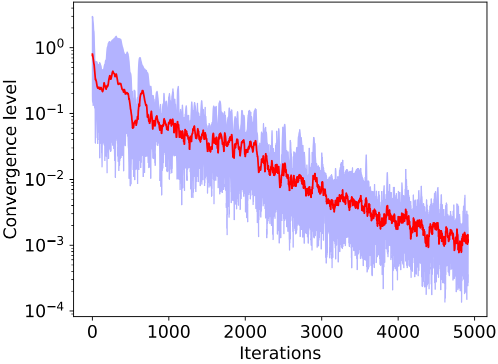


We have **convergence** of the fits for all the parameters.

RESULTS



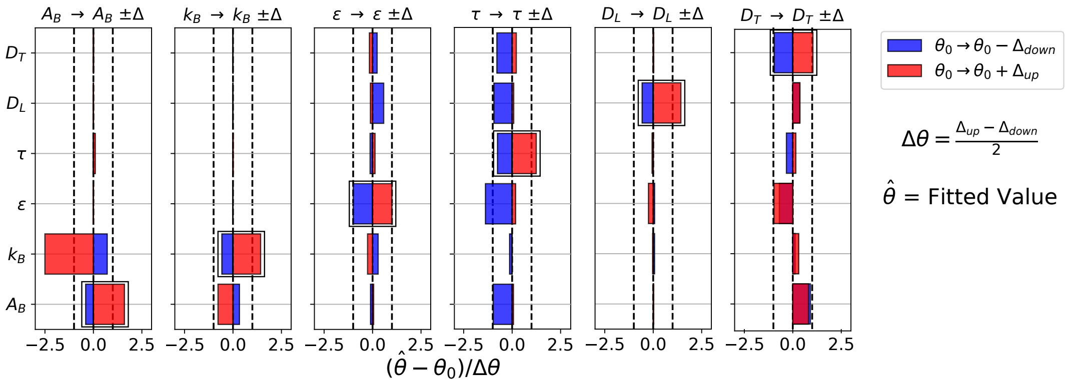
Parameters convergence



6D simulteanous fit converging under L_∞

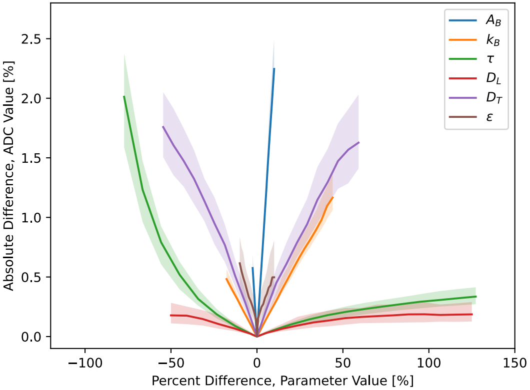
Demonstration of gradient-based calibration on simulation data through a "closure test" ($\theta \rightarrow \theta_{\text{target}}$).

Demonstration of multidimensional fit usefulness

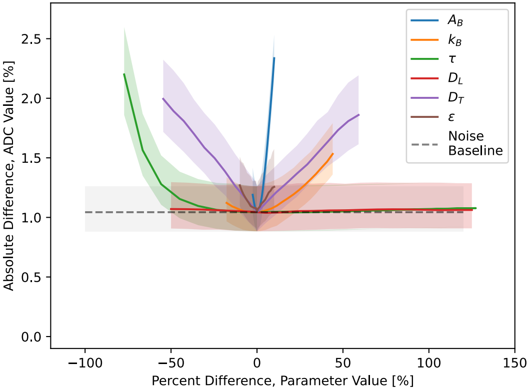


The various physical parameters **are correlated.** Fitting them independently leads to some inaccuracies and biases.

Fit sensitivity

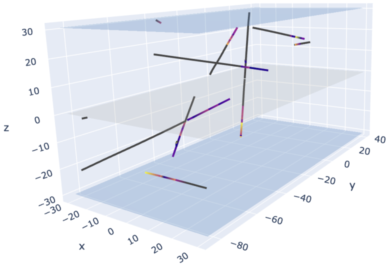


Different sensitivities to the various physical parameters (w.o. noise).



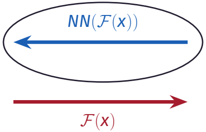
Decrease in sensitivity when considering noise.

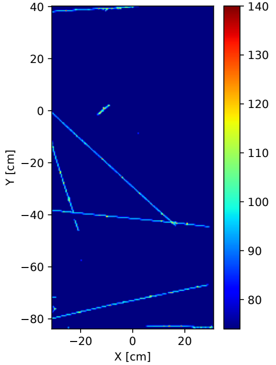
Going further



Energy deposits $\mathrm{d}E/\mathrm{d}x$
(inaccessible in data)

Inverse mapping step to developp





Detector readout

Combining our differentiable simulator with an inverse mapping would allow for direct model constraining, fully data driven: $\mathcal{L}_{CC} = (\mathcal{F}(NN(y_{\text{data}})) - y_{\text{data}})^2$

Conclusions

Proof of concept for the calibration of a LArTPC using a differentiable simulator.

Multidimensional fit converging correctly on simulated data with the differentiable simulator.

Upcoming challenges:

- Applying this framework to real data (DUNE 2x2 ND data)
- Improving the performances (not limiting at the moment)
- Fitting more physical parameters (such as Efield map)

Going further:

- Extend the framework to inverse problem solving.

Pierre Granger

November 28, 2023

granger@apc.in2p3.fr