Project 1

AD-enabled LArTPC charge simulation

LArTPC for Liquid Argon Time Projection Chamber

Yifan Chen

CIDER-ML workshop
9 April 2025

A LArTPC charge simulation

Track segments ($\frac{dE}{dx}$)

Birks' model (A_B, k_B),
Electric field (ε)

$$\alpha_{\text{recomb}} = \frac{A_B}{1 + \frac{k_B}{\varepsilon \cdot \rho}\frac{dE}{dx}}$$

$$n_{\text{elec}} = \frac{dE}{W_{\text{ion}}}$$

$$n_{\text{elec}}^{\text{recomb}} = \alpha_{\text{recomb}} \cdot n_{\text{elec}}$$

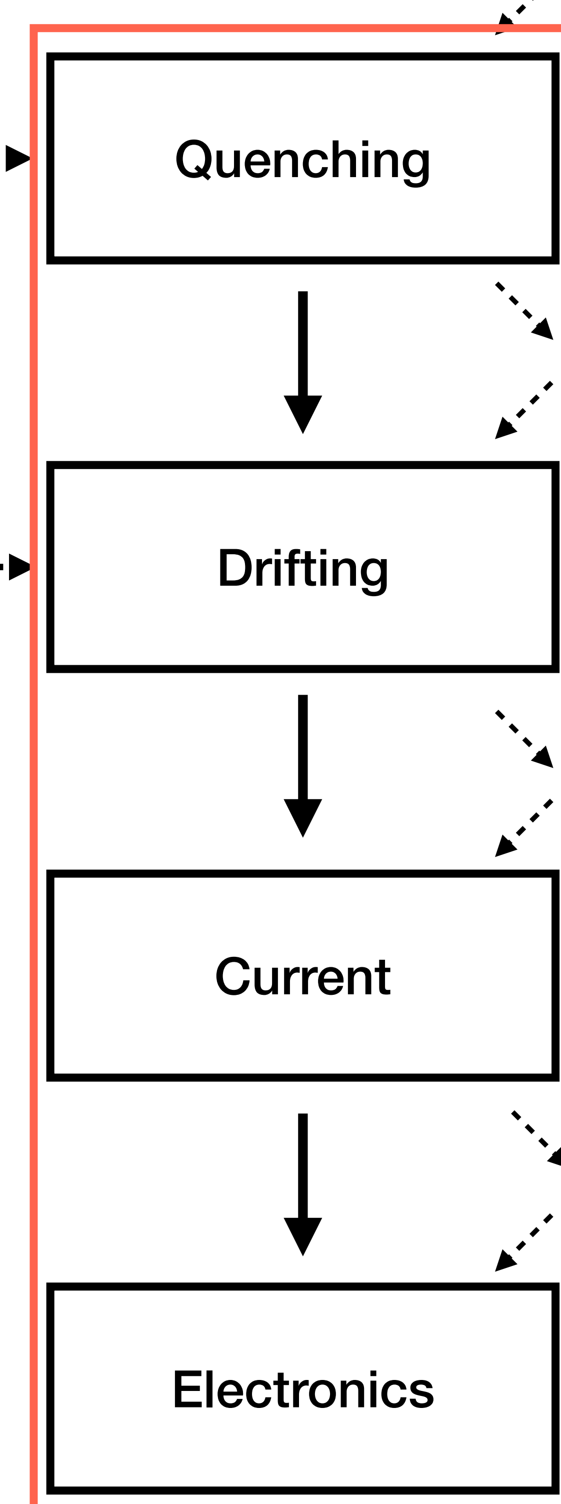
Number of free electrons for drifting

Drift velocity (v_{drift}),
Lifetime (τ),
Transverse/
longitudinal diffusion
(D_T, D_L)

$$v_{\text{drift}} = \mu \cdot \varepsilon$$

$$\sigma_{L,T} = \sqrt{2 \cdot t_{\text{drift}} \cdot D_{L,T}}$$

$$n_{\text{elec}}^{\text{final}} = n_{\text{elec}}^{\text{recomb}} \cdot e^{-t_{\text{drift}}/\tau}$$



Electron distribution at readout

Current induced on each pixel

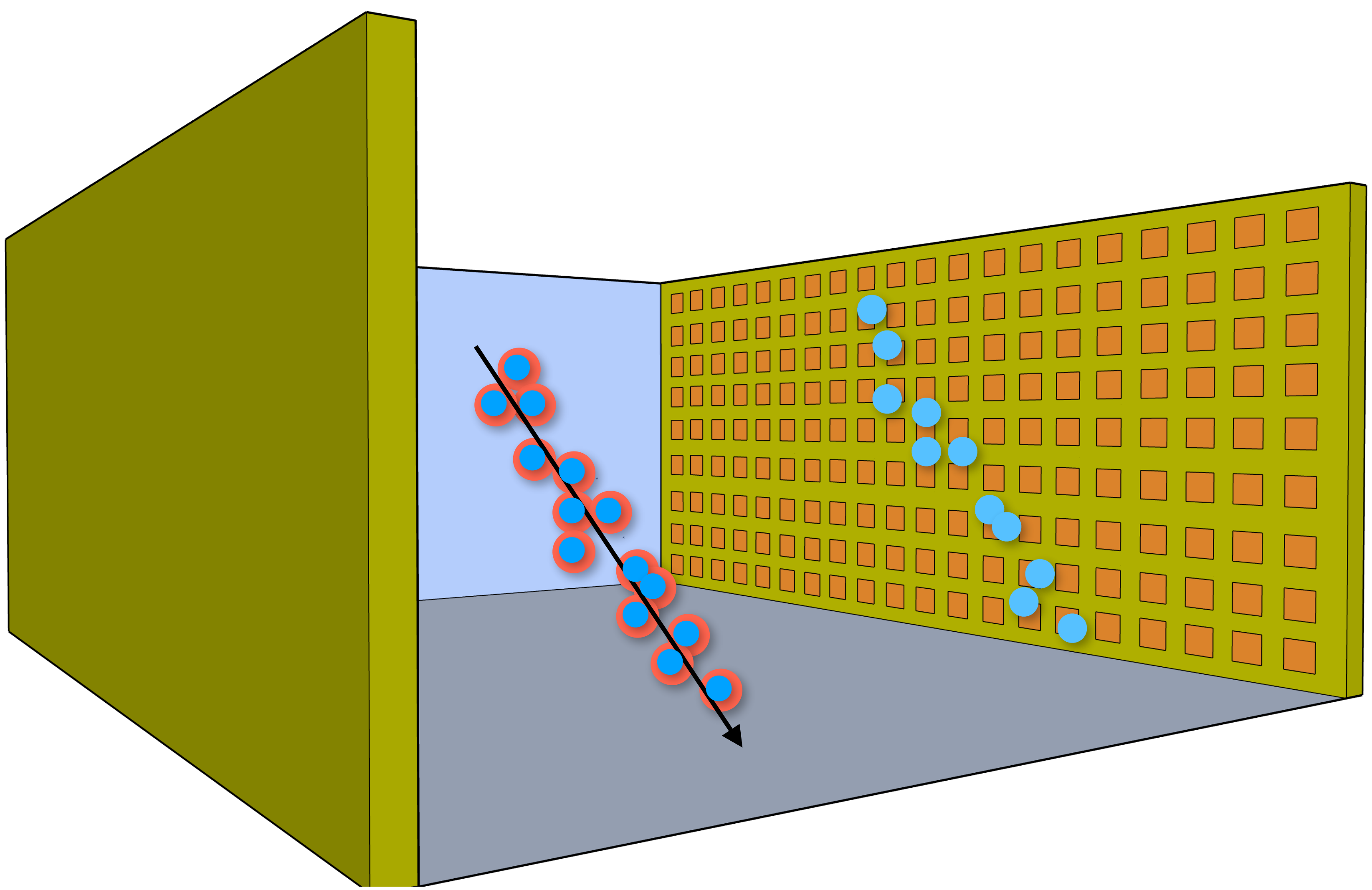
Signal digitized here
dealing with hits, not waveform
in this demonstration

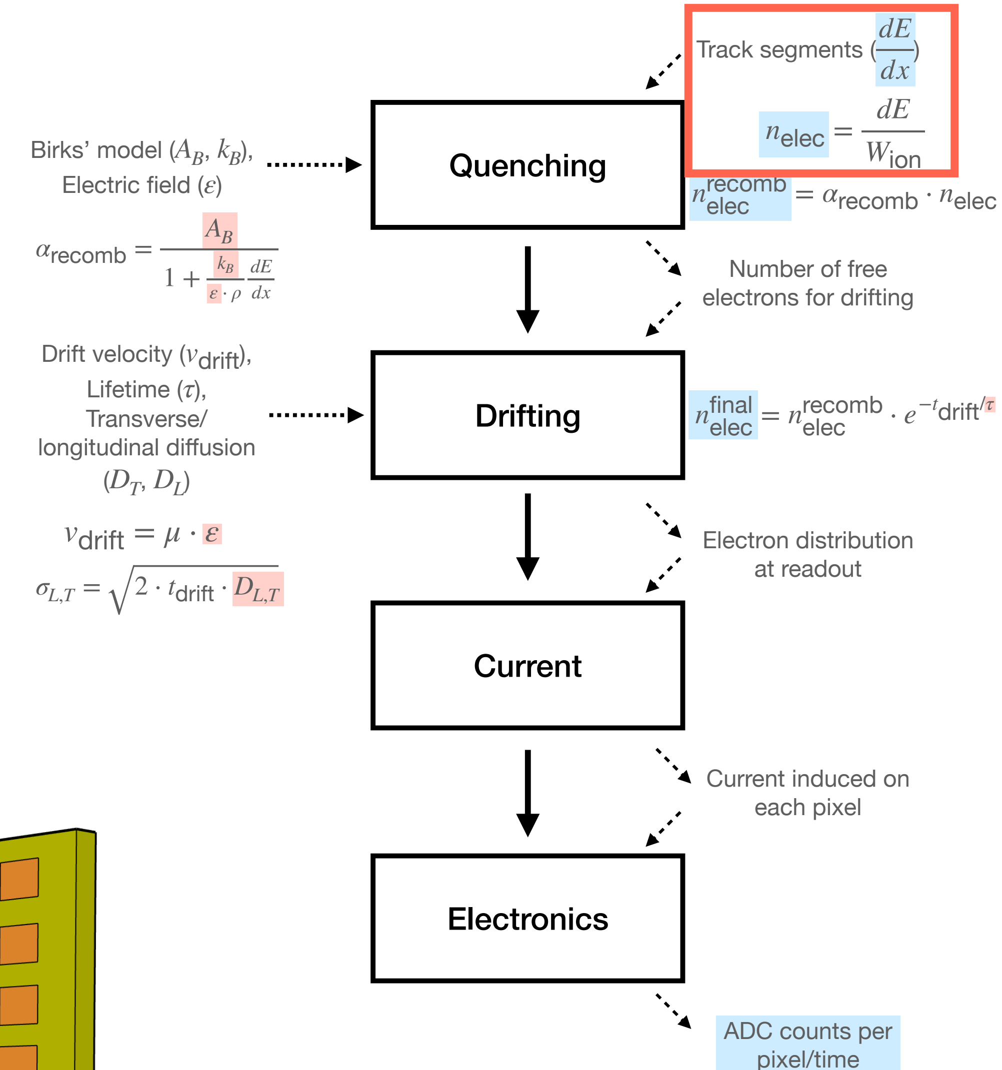
ADC counts per pixel/time

LArTPC: Charge Production

Particle "segments" deposit $\dfrac{dE}{dx}(x, y, z)$

Produce free electrons $n_e = \dfrac{dE}{W_{\text{ion}}}$





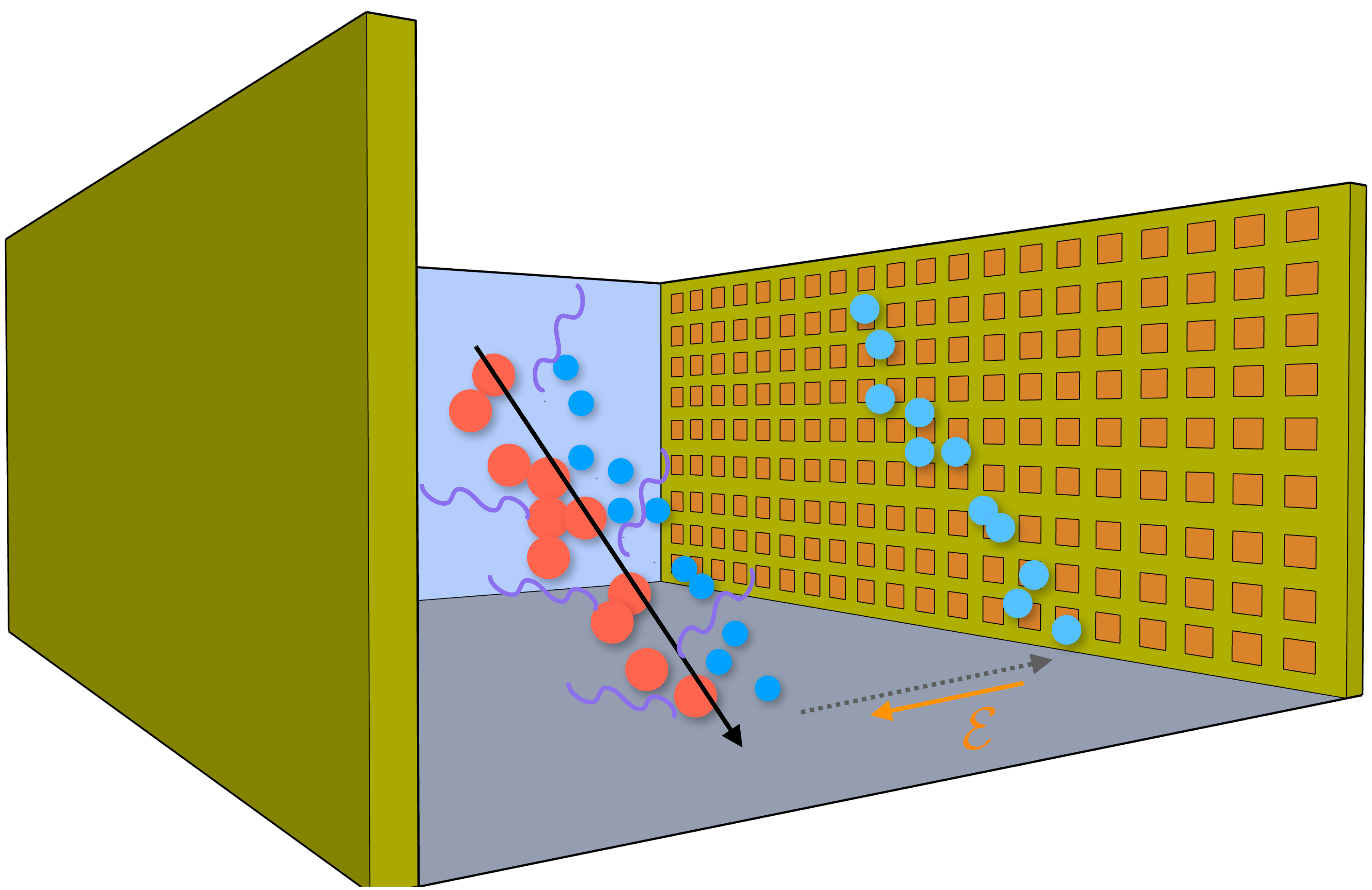
LArTPC: Charge Recombination

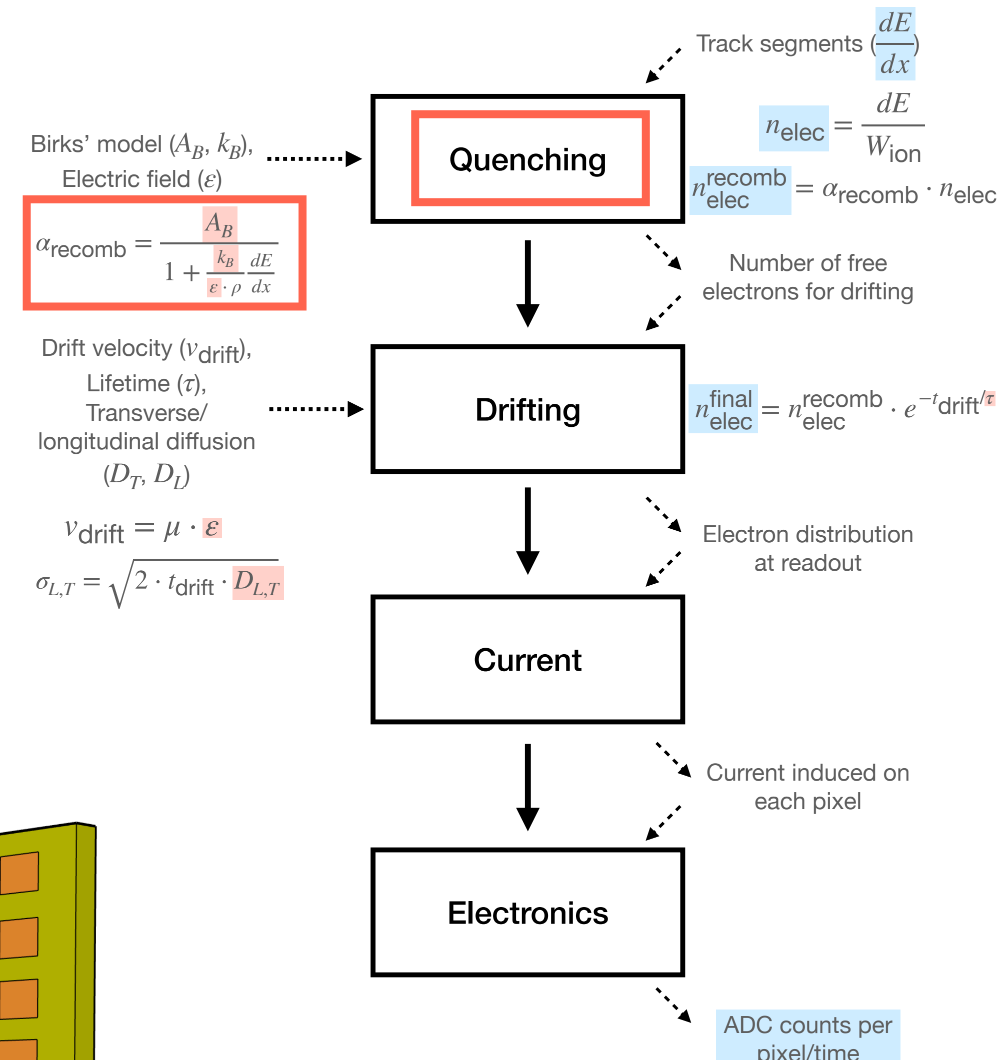
Number of electrons after recombination

$$n_e^{\mathrm{recomb}} = \alpha^{recomb} \cdot n_e$$

Recombination Birks model

$$\alpha^{recomb} = \frac{A_{\mathrm{B}}}{1 + \frac{k_{\mathrm{B}}}{\mathcal{E} \cdot \rho} \frac{dE}{dx}}$$

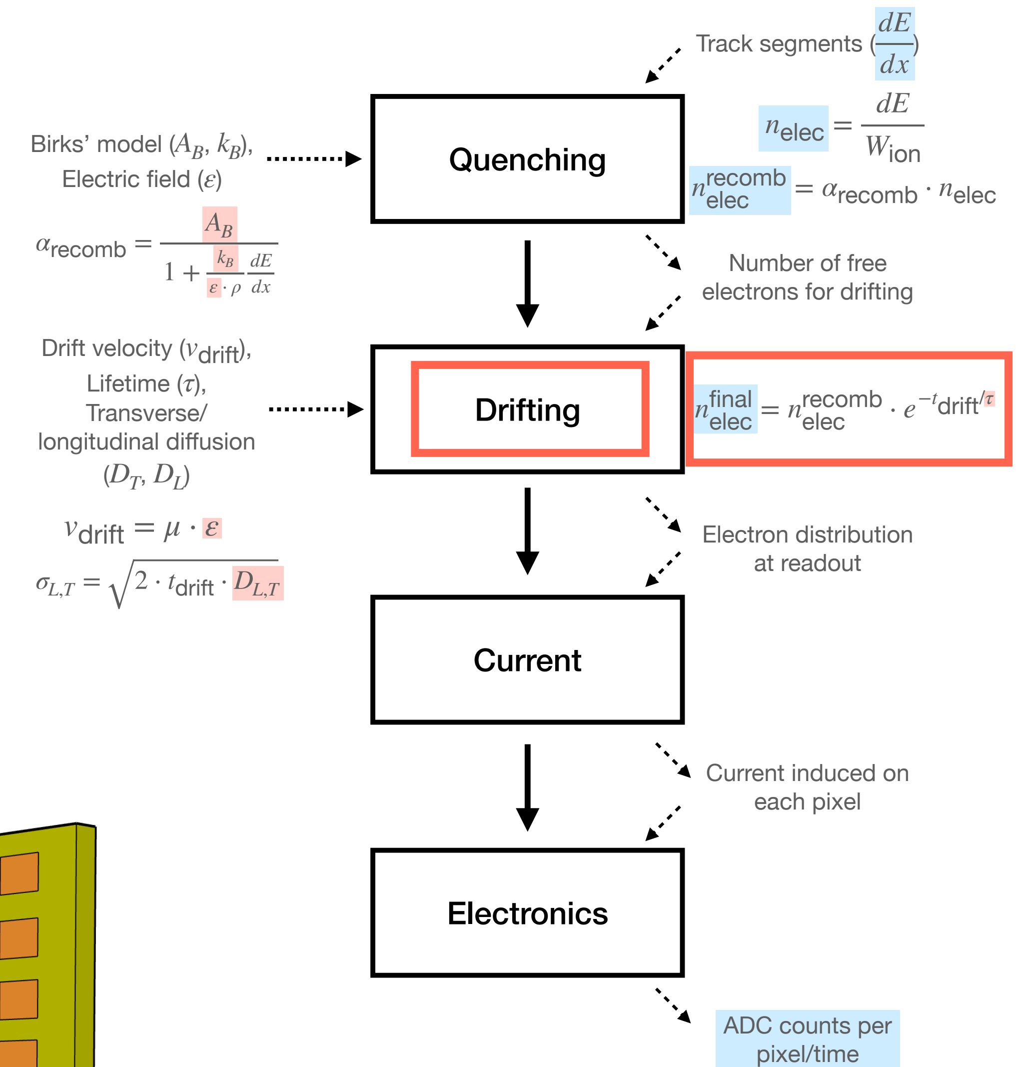


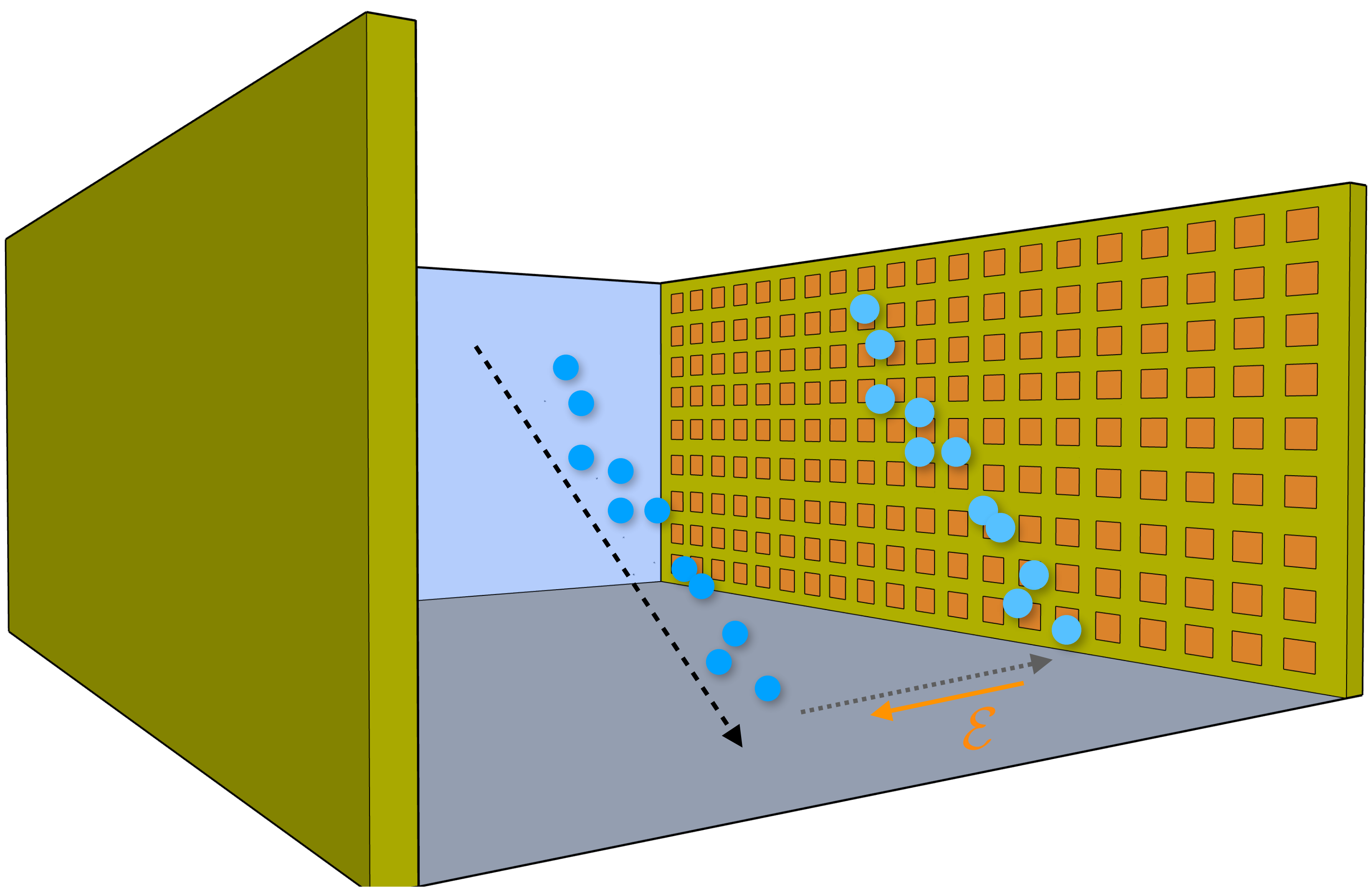


LArTPC: Charge Attenuation

Number of electrons survived charge attenuation

$$n_e^{\text{att}} = n_e^{\text{recomb}} \cdot e^{-\frac{t_{\text{drift}}}{\tau}}$$



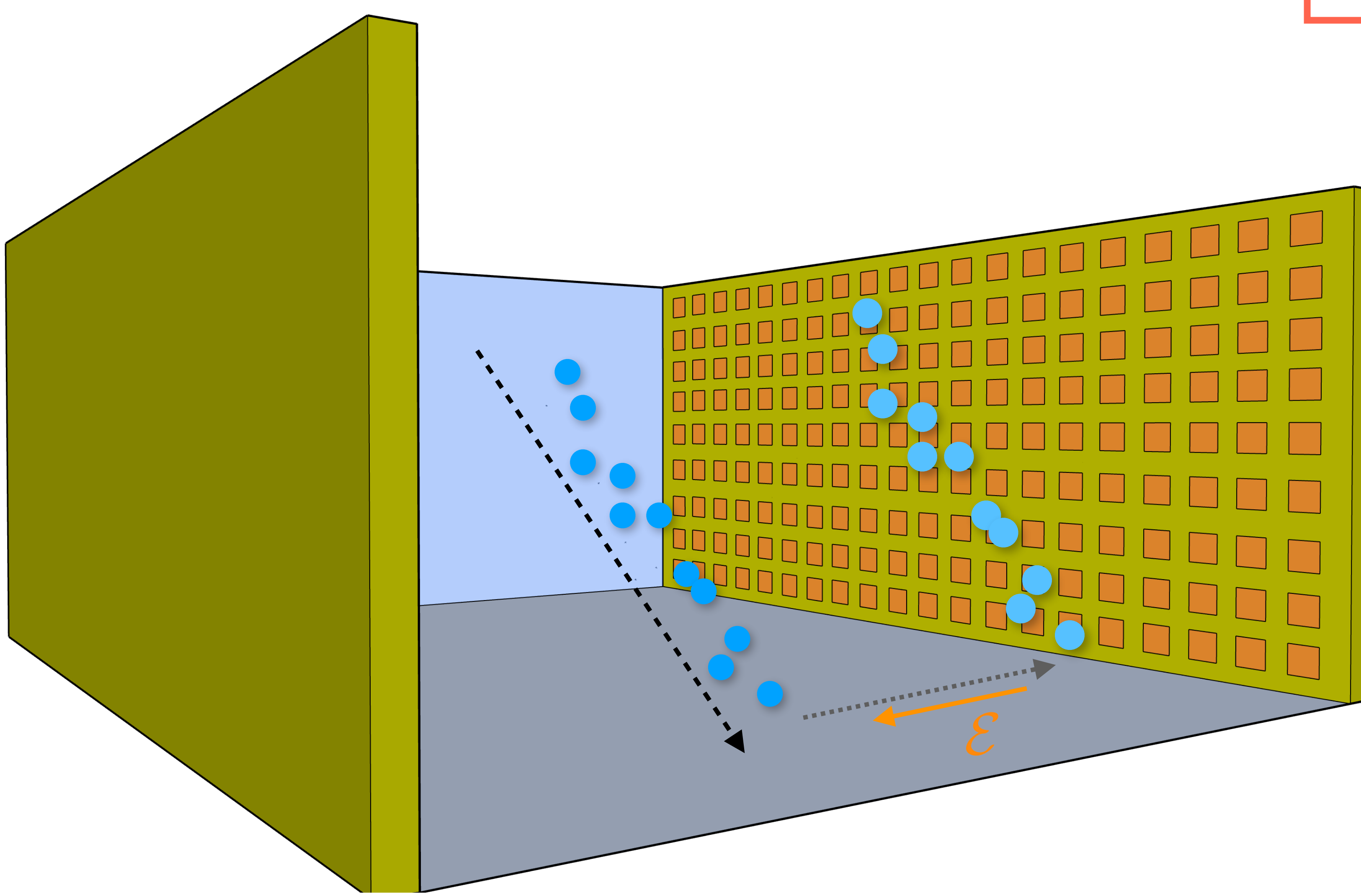


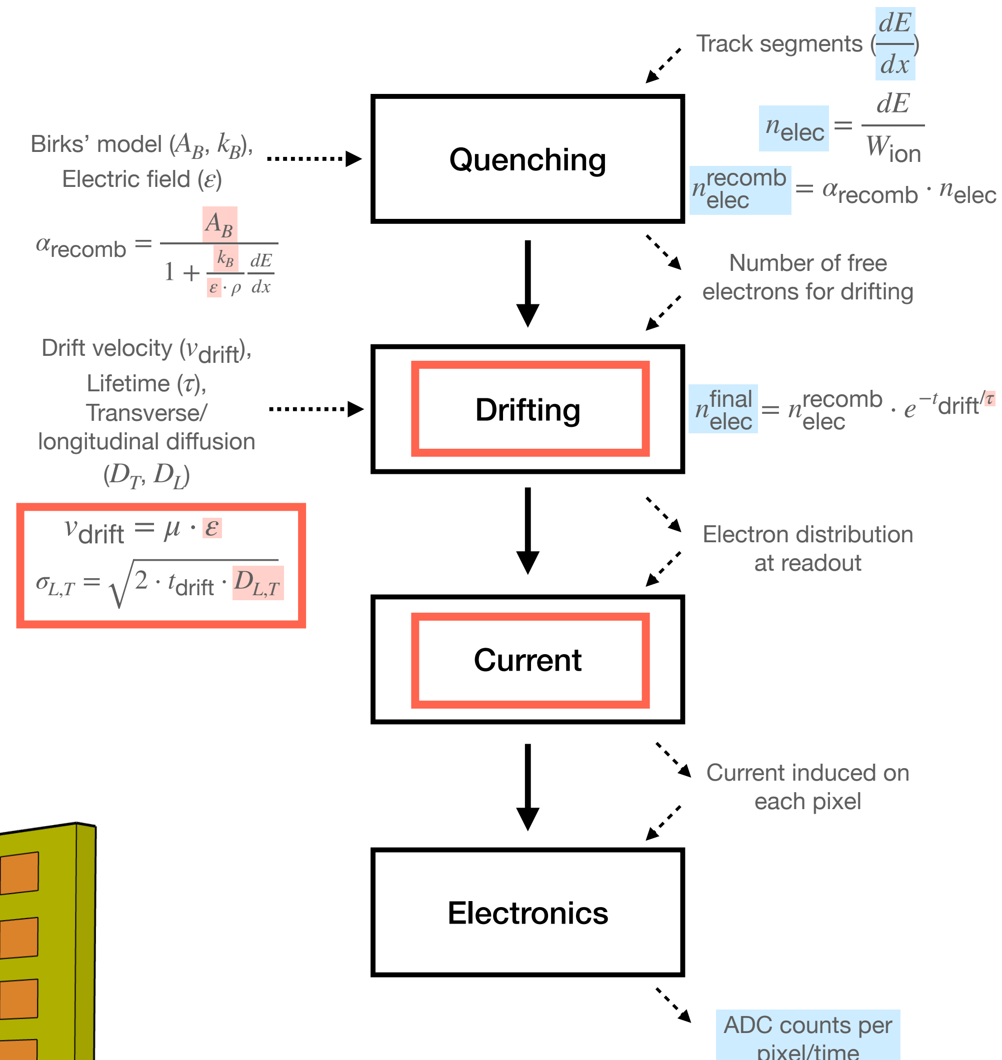
LArTPC: Charge Drift and Diffusion

$$v_{\text{drift}} = \mu \cdot \mathcal{E}$$

$$\text{drift position} = v_{\text{drift}} \cdot t_{\text{drift}}$$

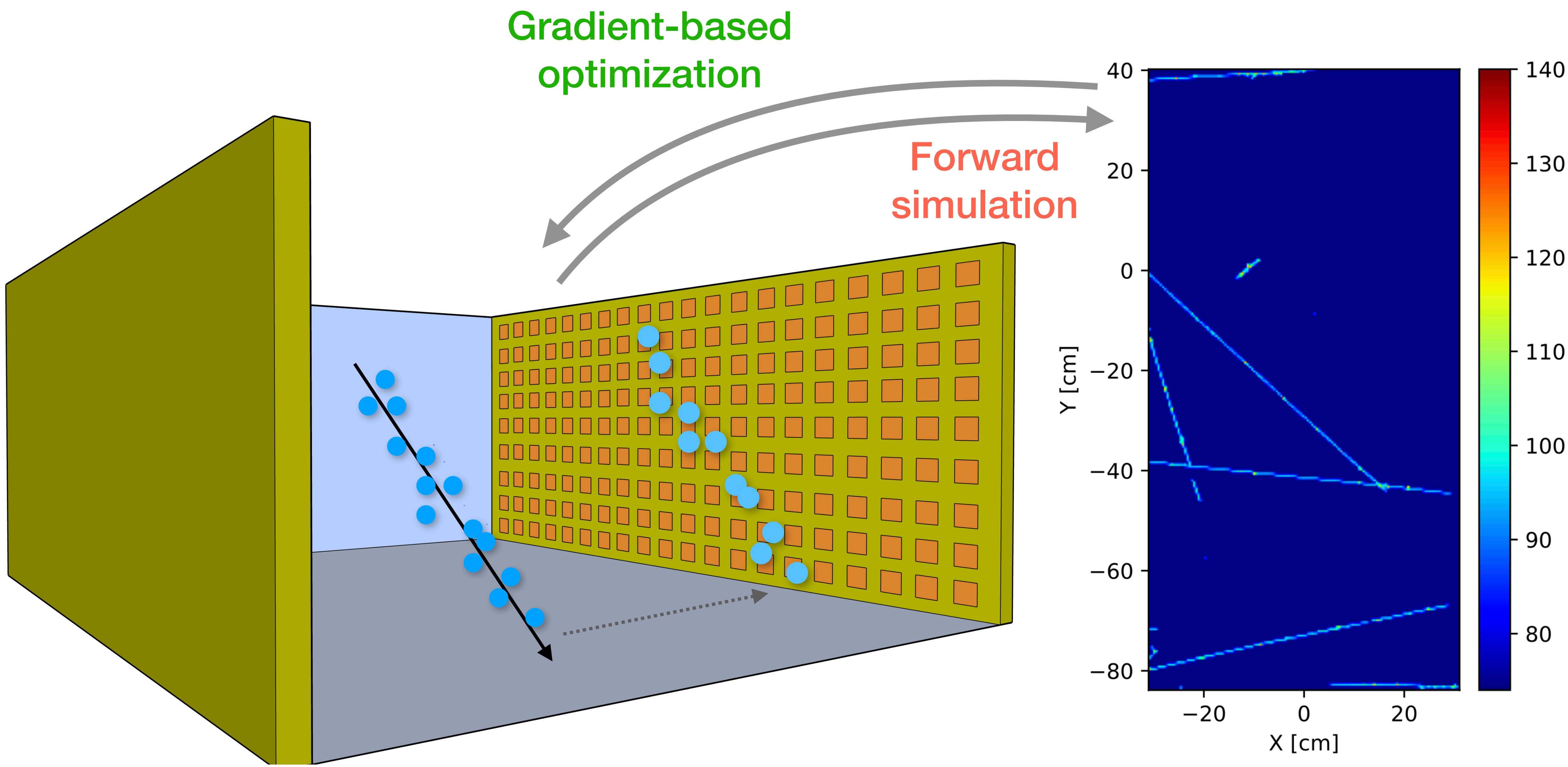
$$\sigma_{L,T} = \sqrt{2 \cdot t_{\text{drift}} \cdot D_{L,T}}$$





Simultaneous High-dimensional Calibration

- Challenging for conventional calibration methods
- Development of a differentiable simulation for high-dimensional calibration
  - Simultaneous optimization for multiple model parameters
  - Straightforward application of the calibration
  - Improve simulation fidelity



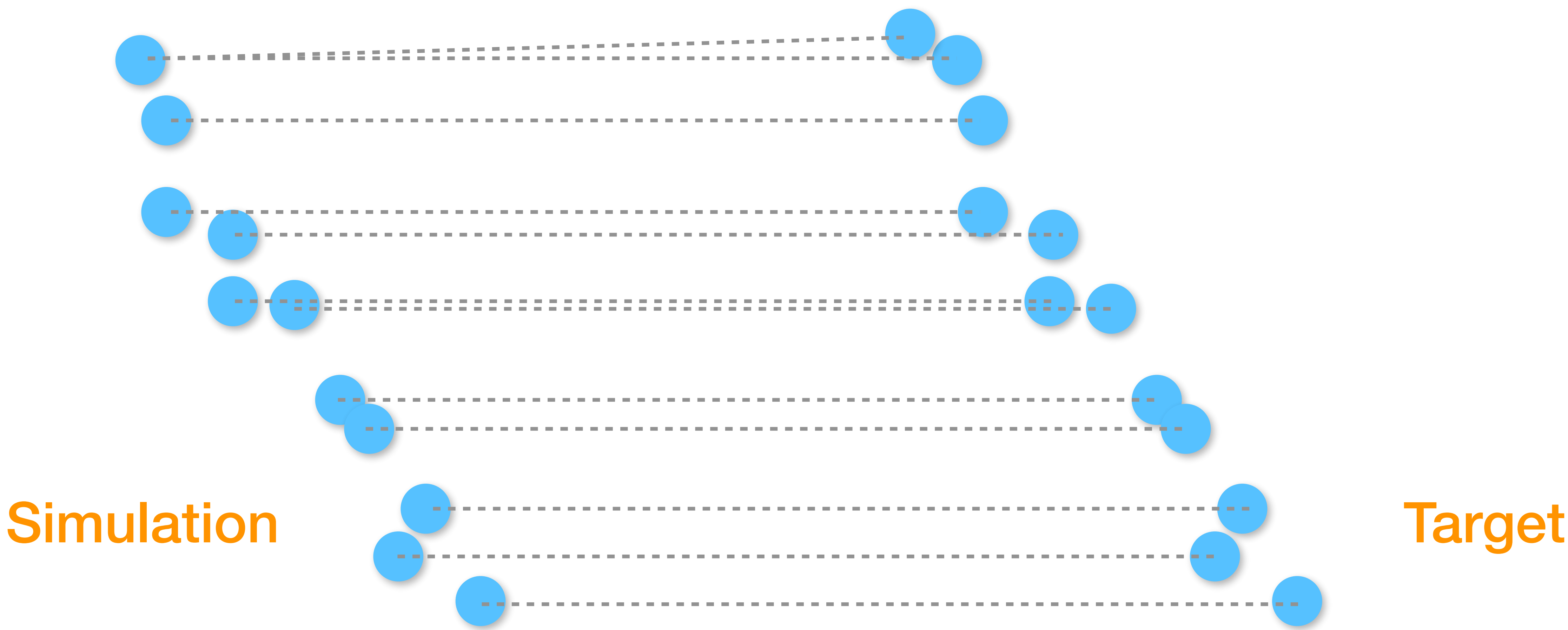
From Simulation Output to Loss

Calculate gradients for the parameters $\nabla_\theta \mathcal{L}(f(\chi, \theta_0), F_{\text{target}})$

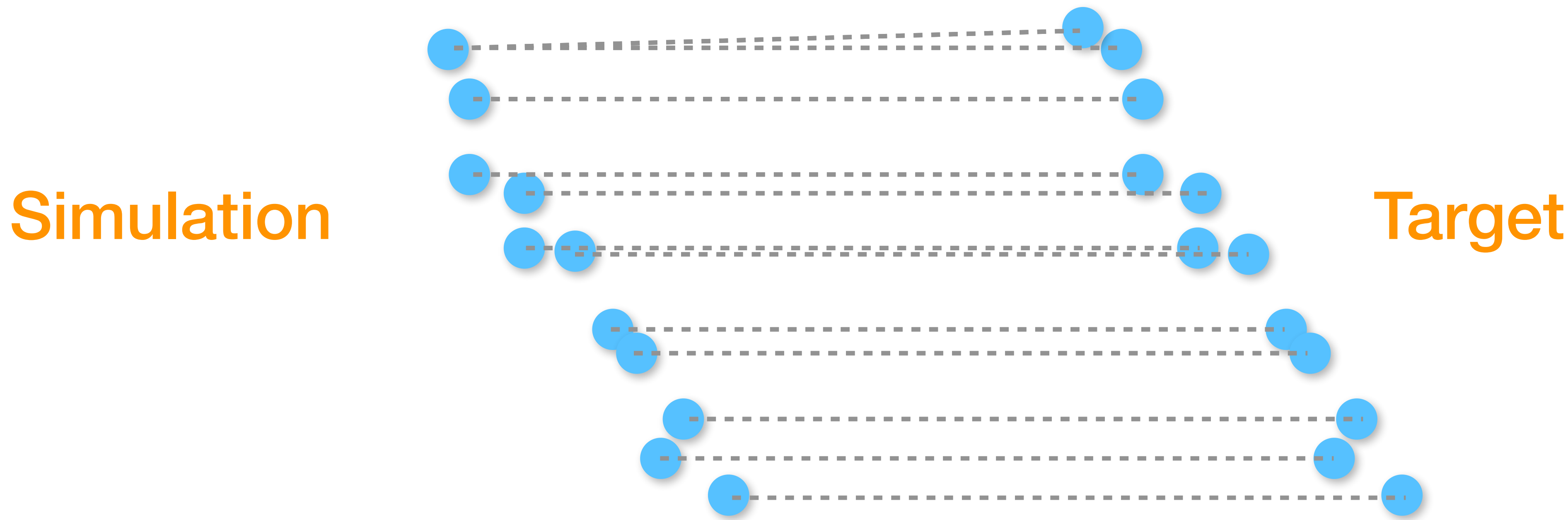
Gradient-based parameter optimization $\theta_0 \to \theta_i$

Challenges

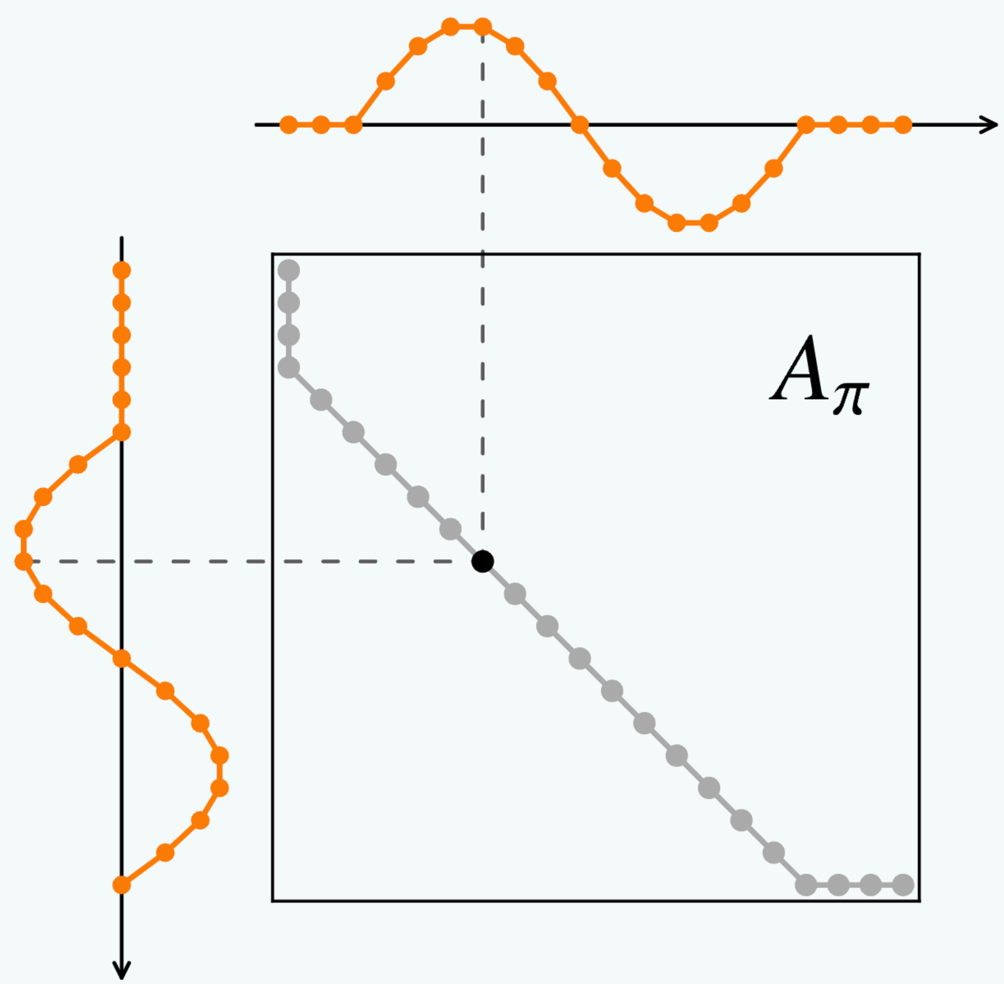
- Sparse data
- Potentially different length of the simulation output and the target data
- Obscure correspondence between hits from the two sets

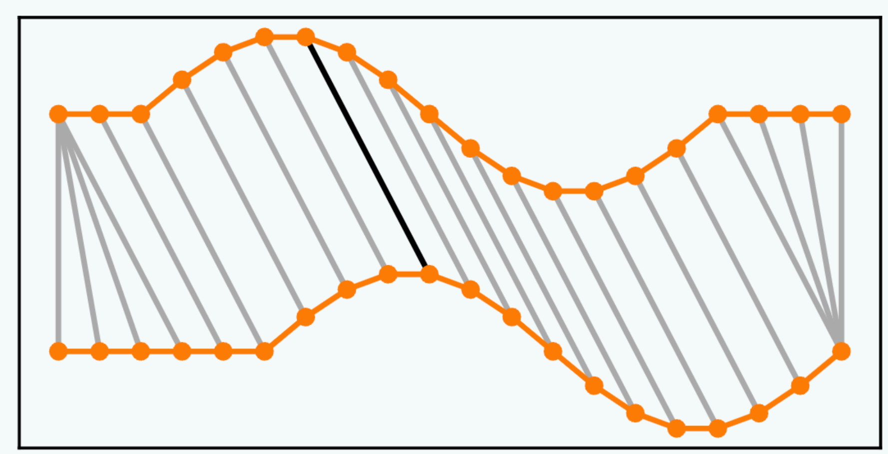


Tested Losses: Chamfer, Soft DTW, MSE



Soft Dynamic Time Warping (DTW)



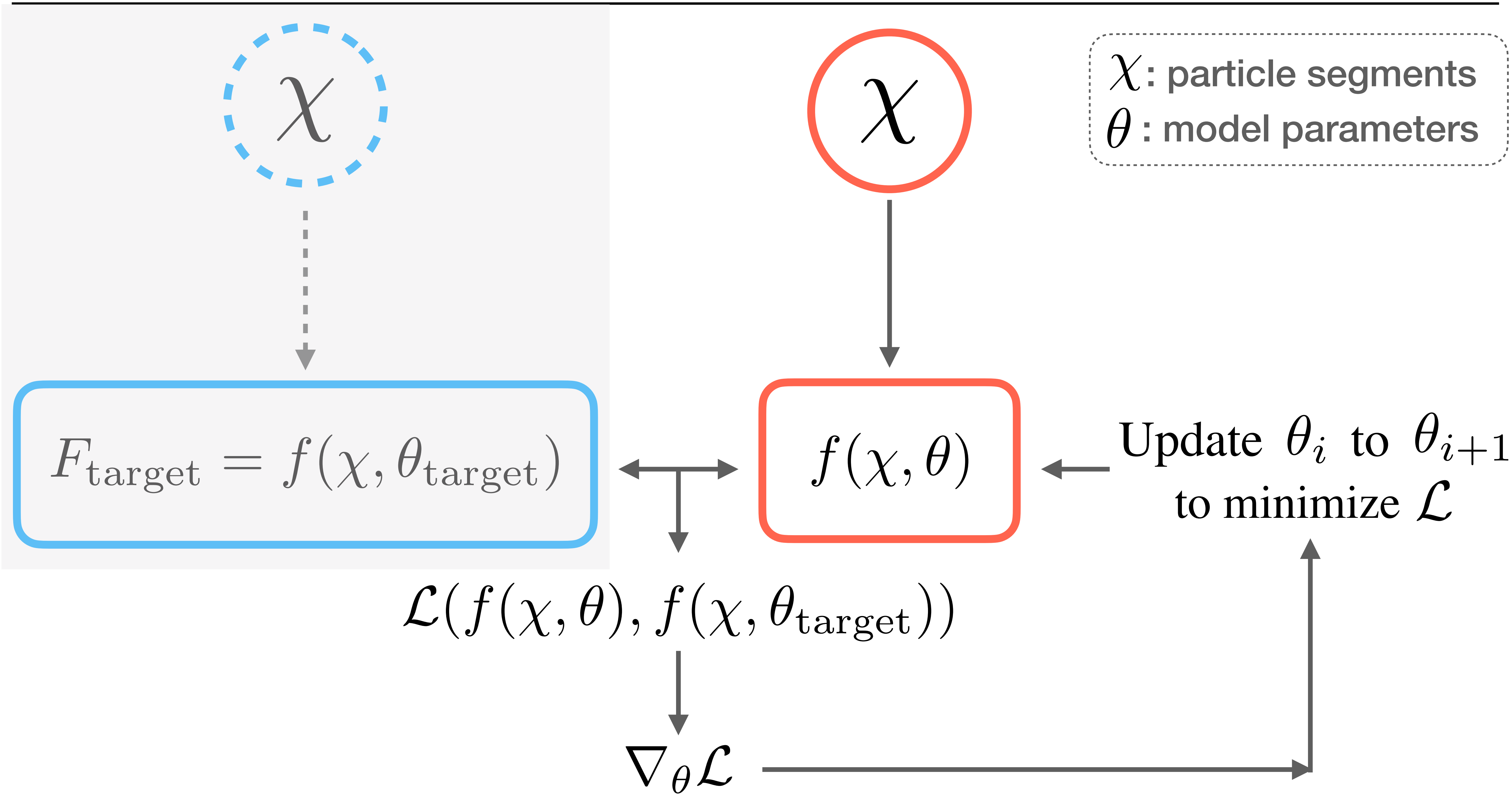


Romain Tavenard

Post: DTW

Post: Soft DTW

Developing and Evaluating the Calibration Fit



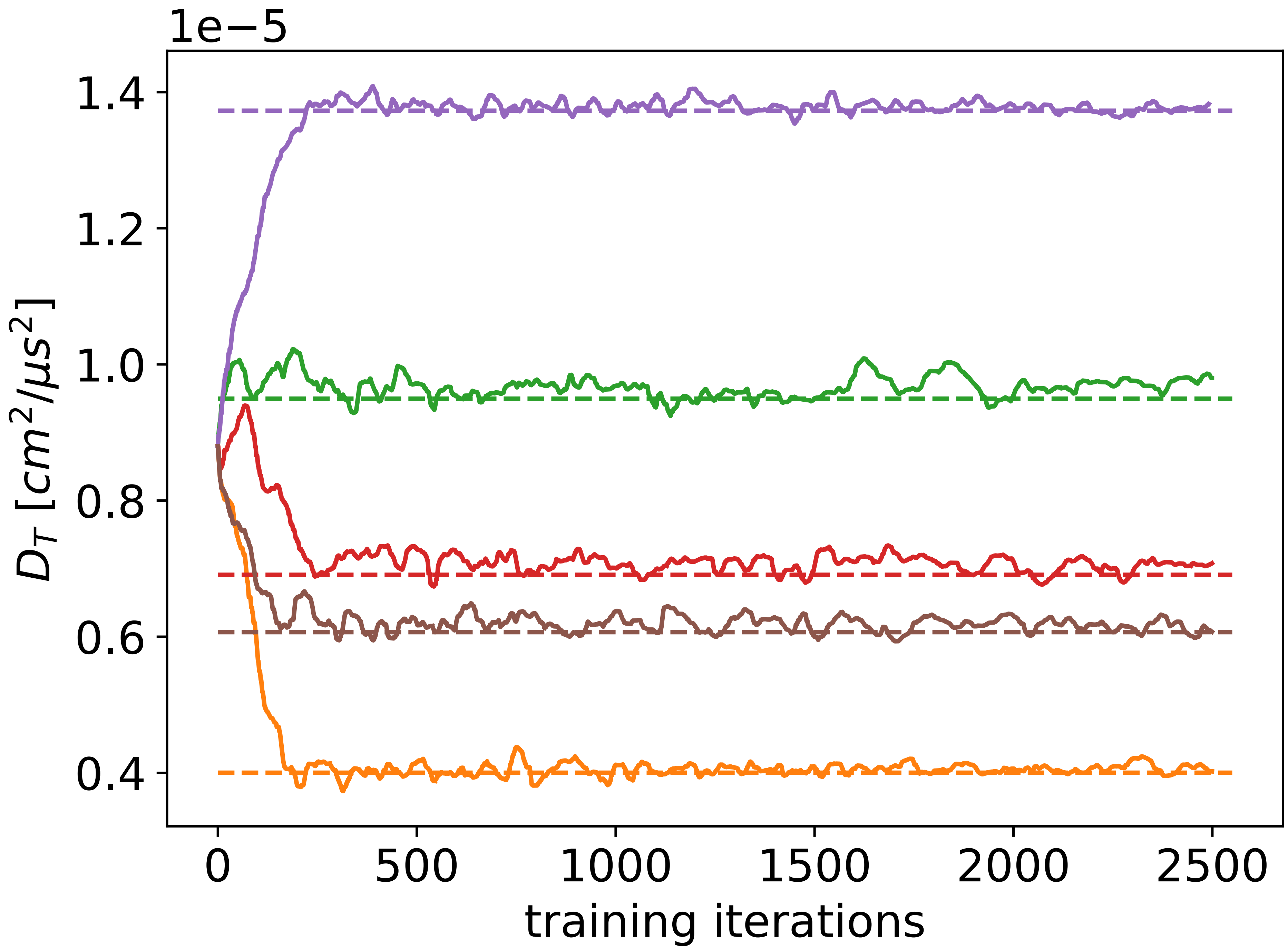
Evaluate if θ converge to θ_{target}

- Start with different initial values and check the parameter convergence to the same target
- Start with the same initial values and check the parameter convergence to different targets

Fit Result (with Electronic Noise)

Soft DTW

5 fits with different targets in 6D phase space.
The fits use 100 cm batch.

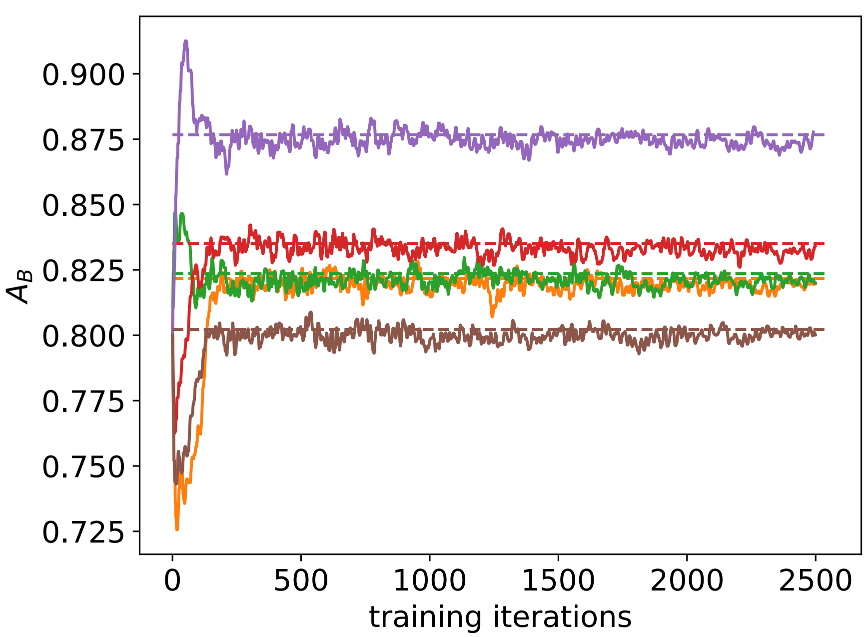


A Full Picture of the Fit Result (with Electronic Noise)

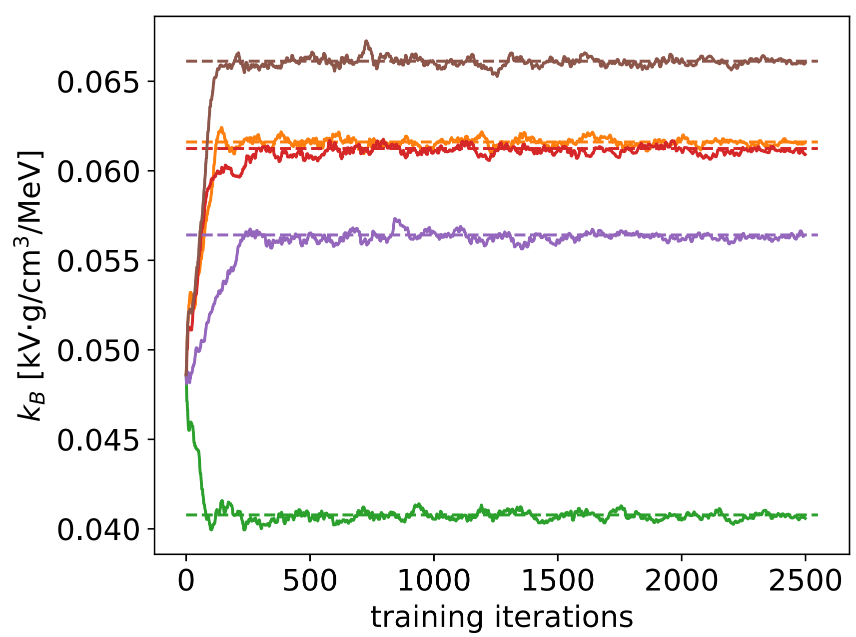
Soft DTW

5 fits with different targets in 6D phase space.
The fits use 100 cm batch.

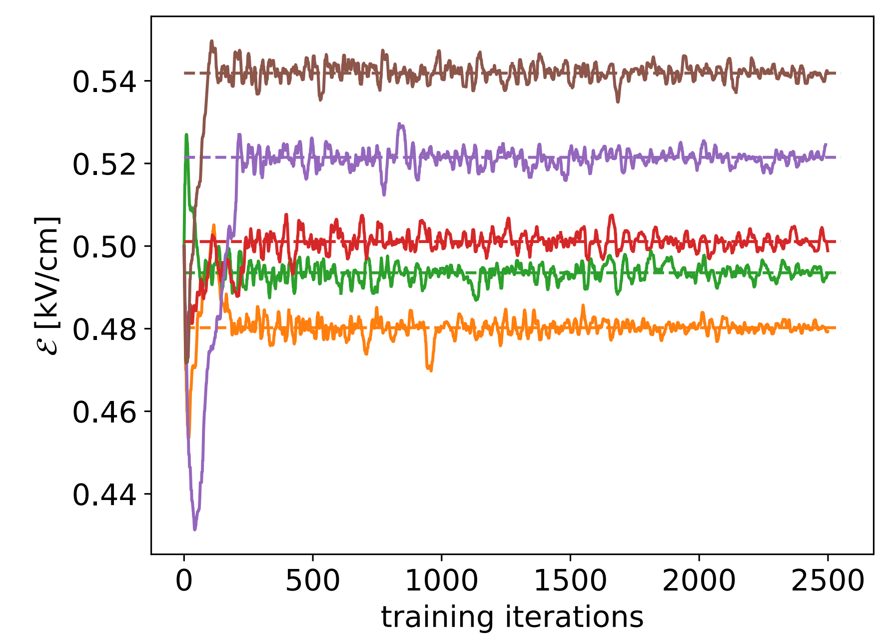
Recombination model A_B



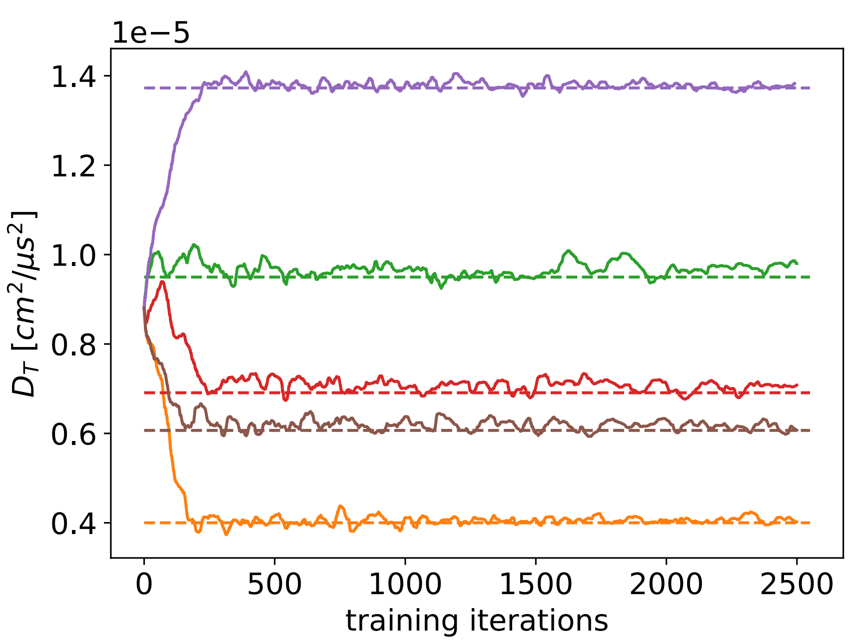
Recombination model k_B



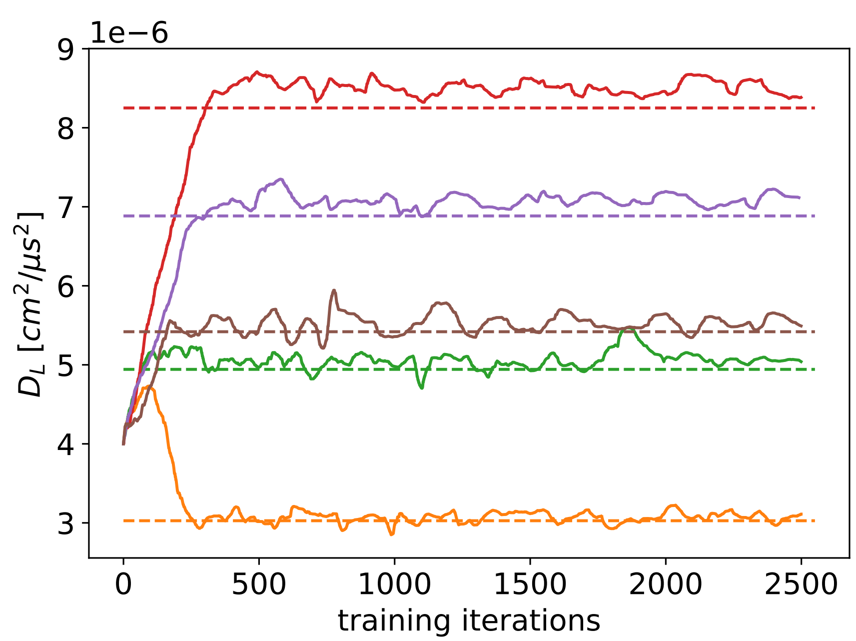
Electric field



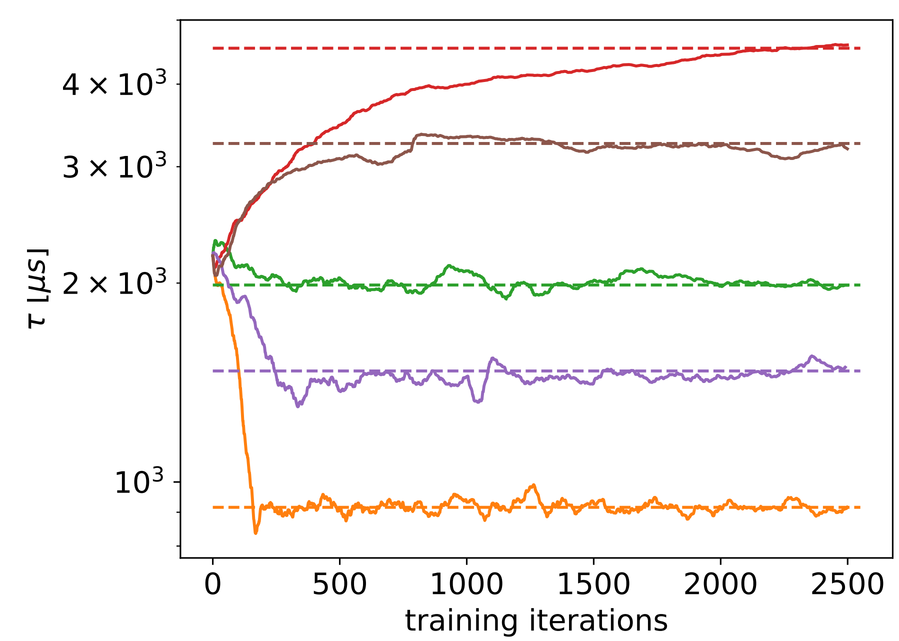
Transverse diffusion coefficient



Longitudinal diffusion coefficient

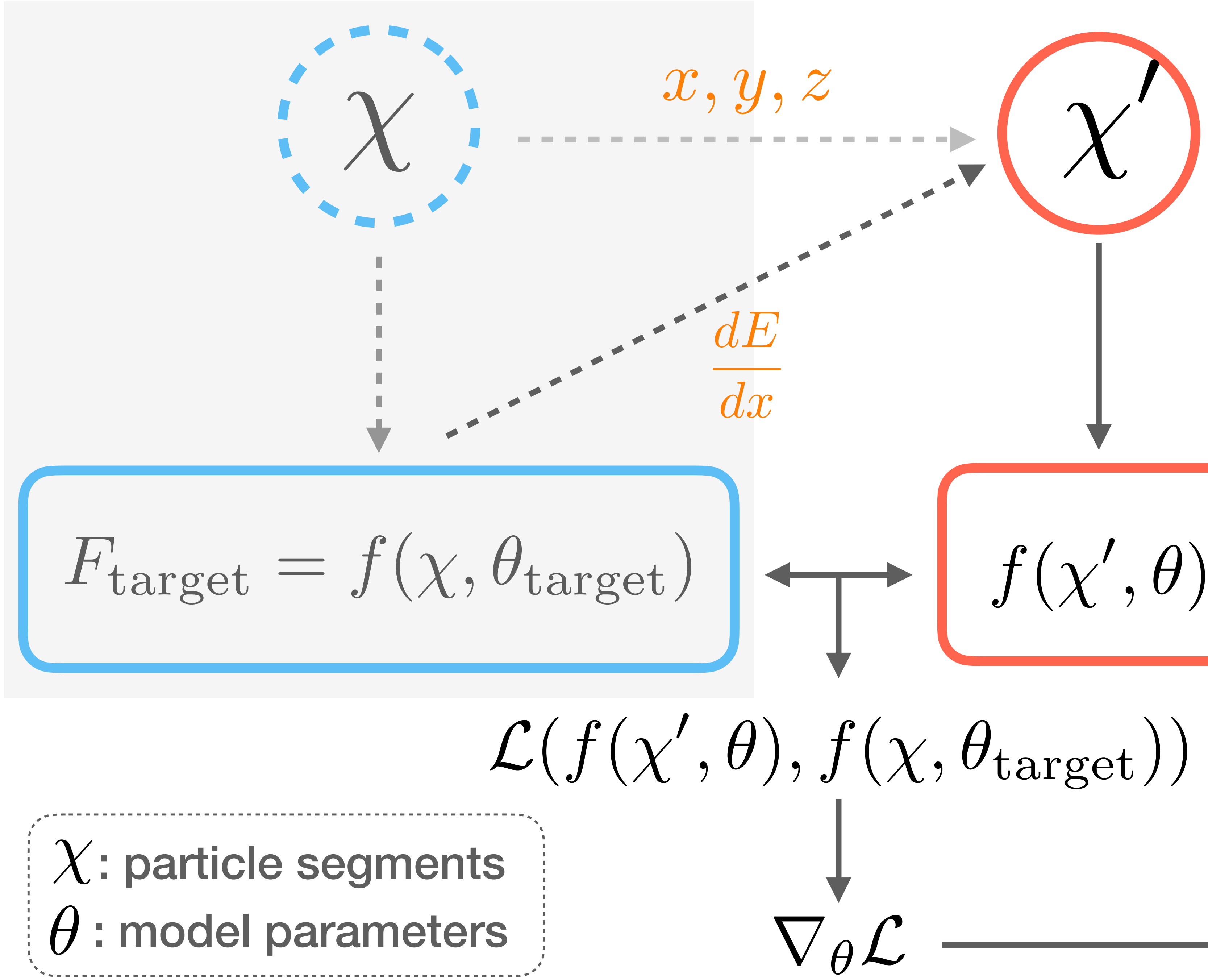


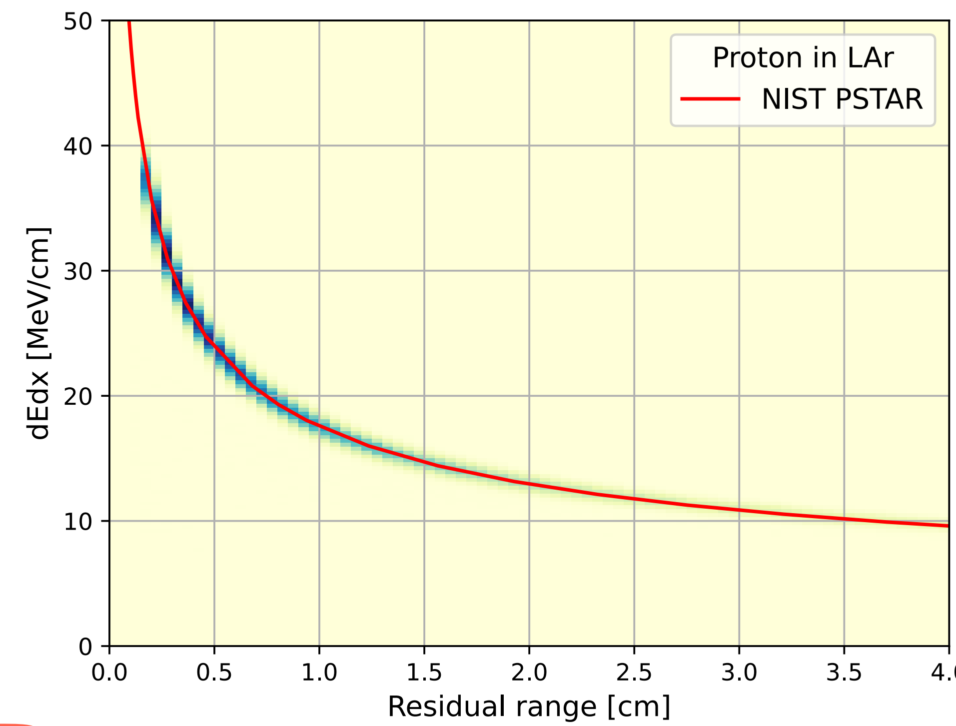
Electron lifetime



Data Application: From the Readout to the Simulation Input

Updated closure test





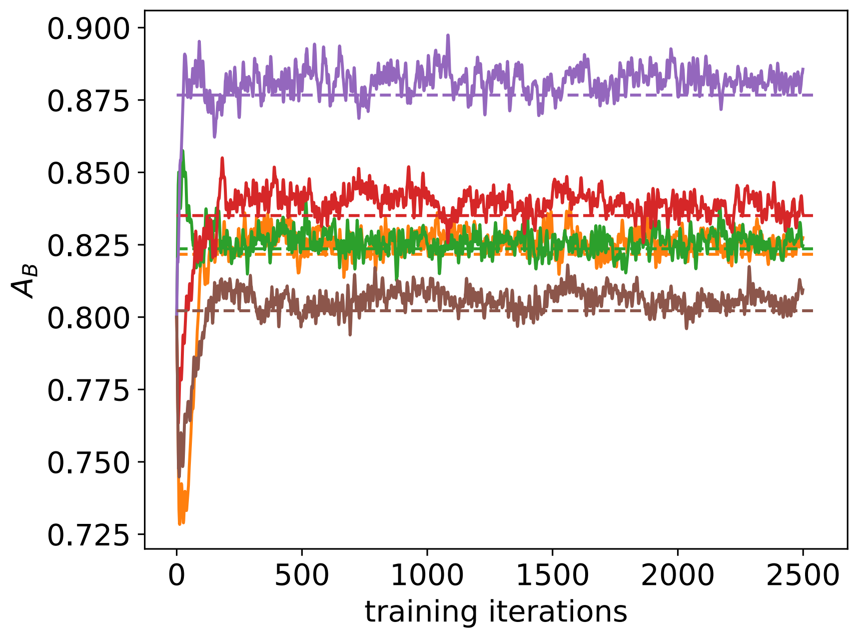
Evaluate if θ converge to θ_{target}

Fit Result (with Reconstructed dE/dx)

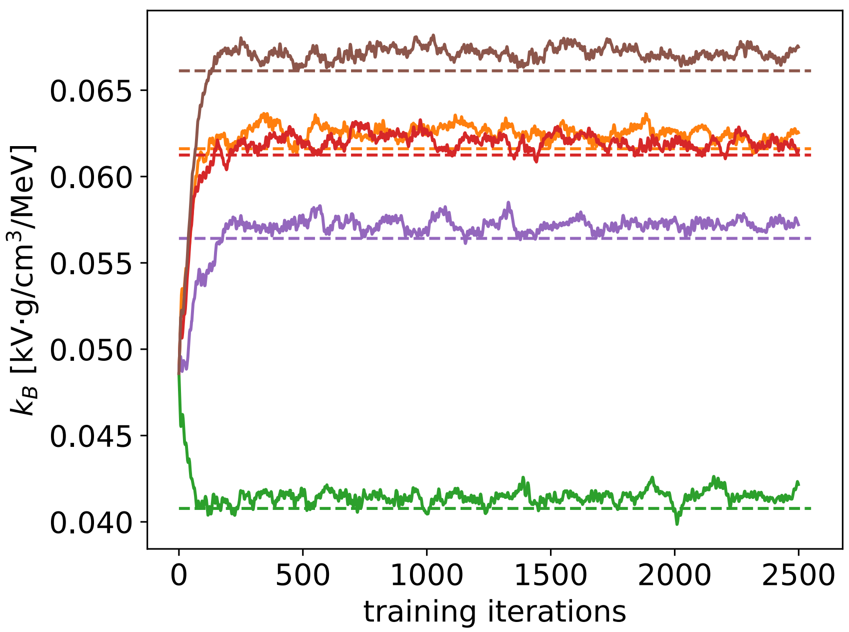
Soft DTW

5 fits with different targets in 6D phase space.
The fits use 100 cm batch.

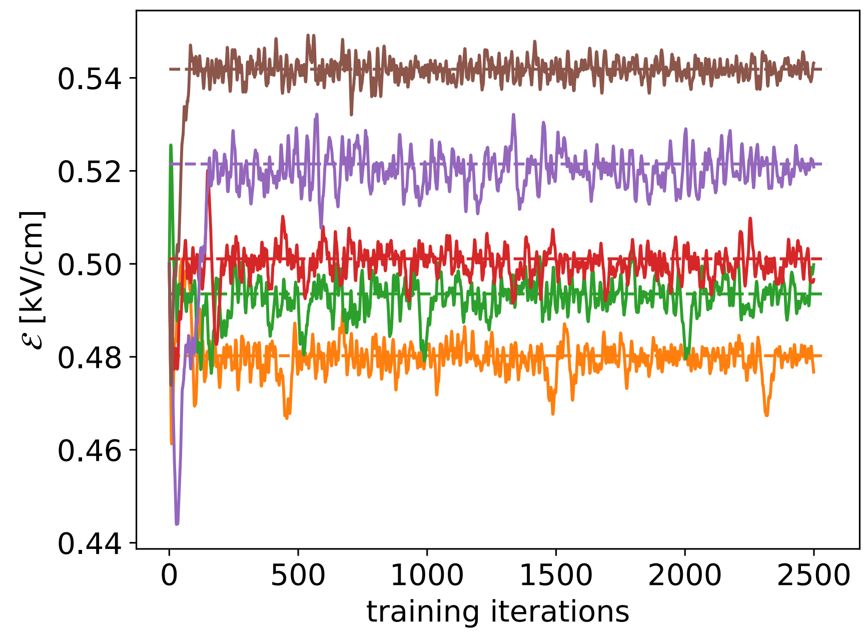
Recombination model A_B



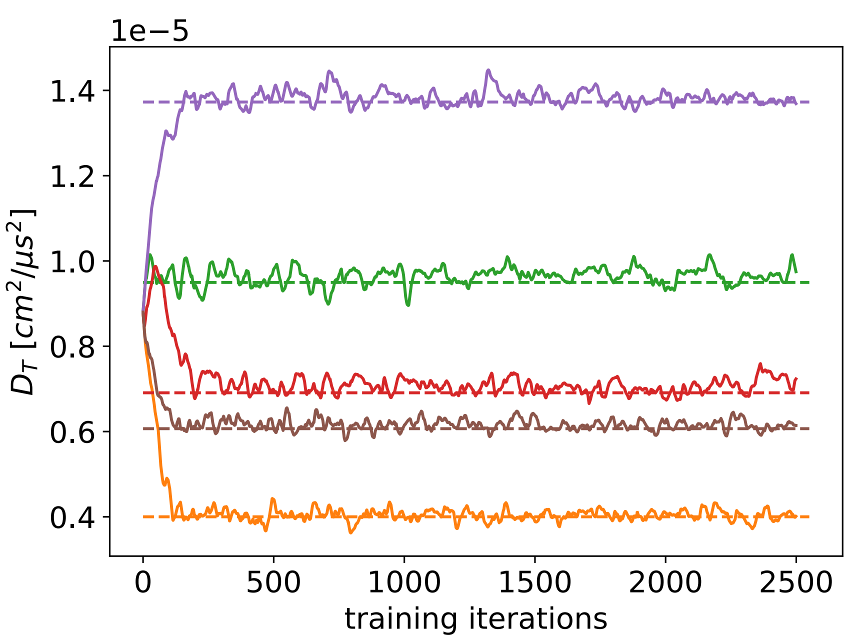
Recombination model k_B



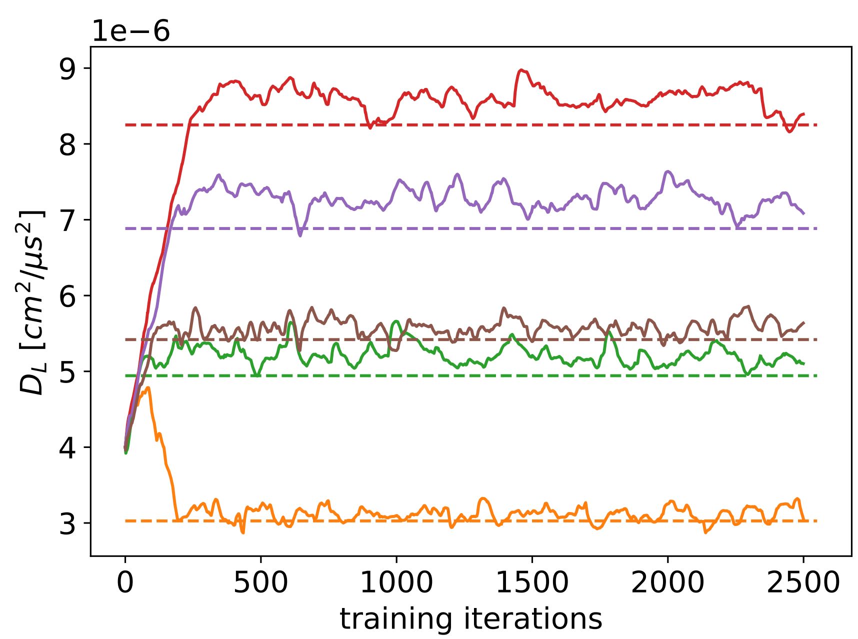
Electric field



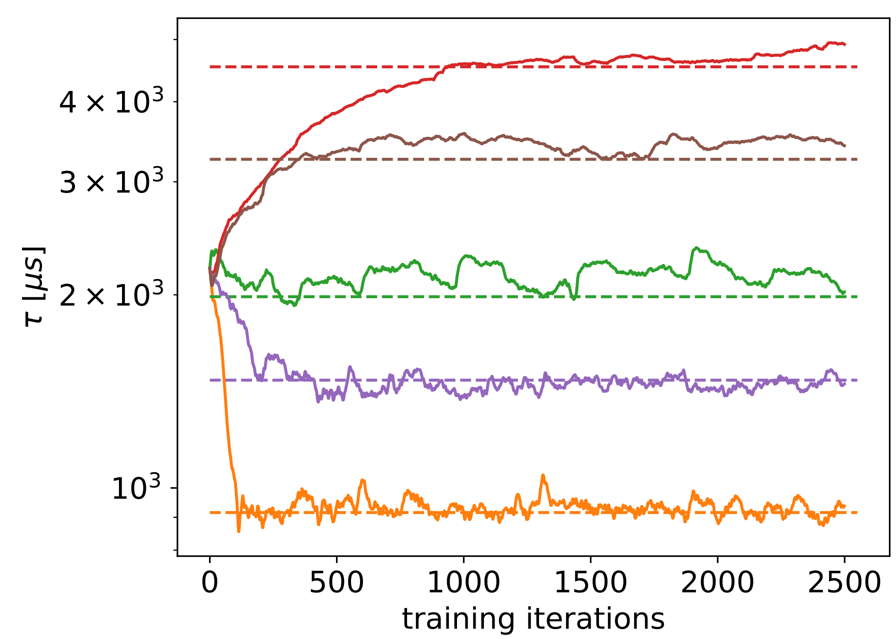
Transverse diffusion coefficient



Longitudinal diffusion coefficient



Electron lifetime

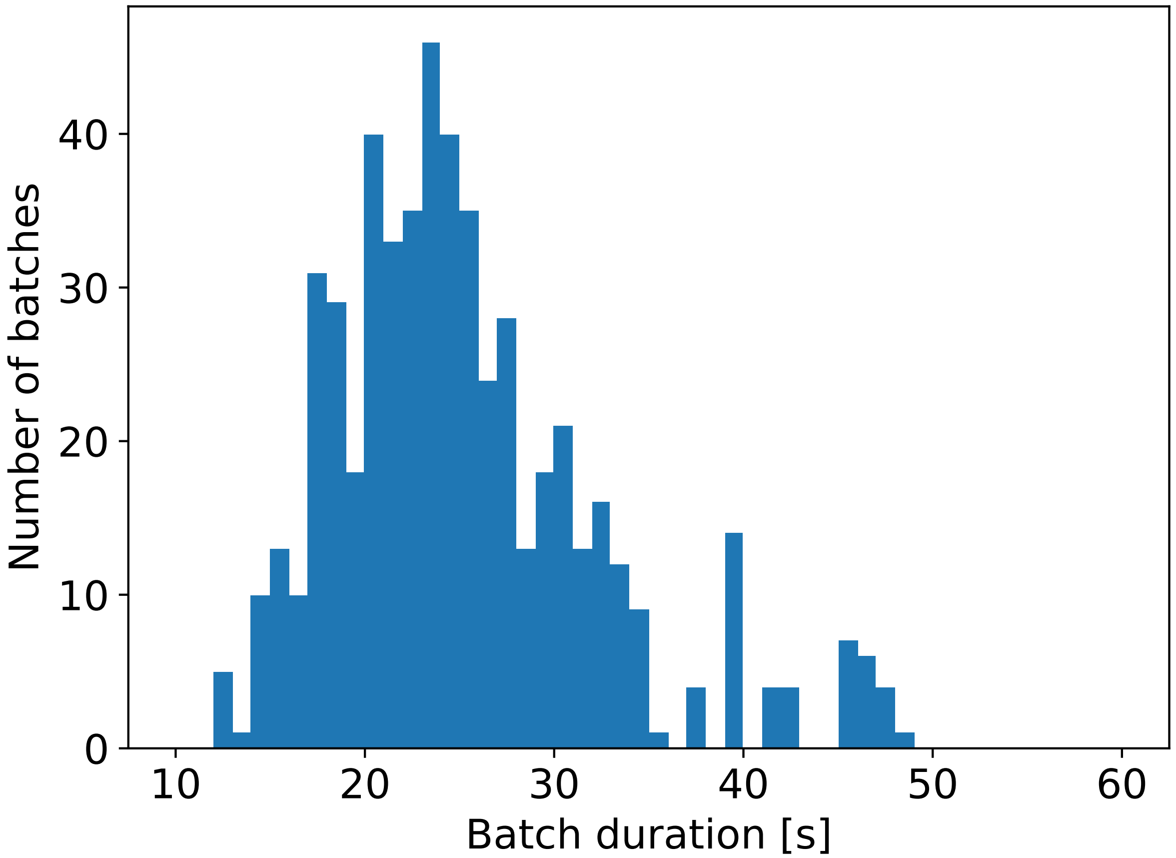


Differentiable *larnd-sim* implemented in PyTorch and JAX

PyTorch implementation
https://github.com/ynashed/larnd-sim

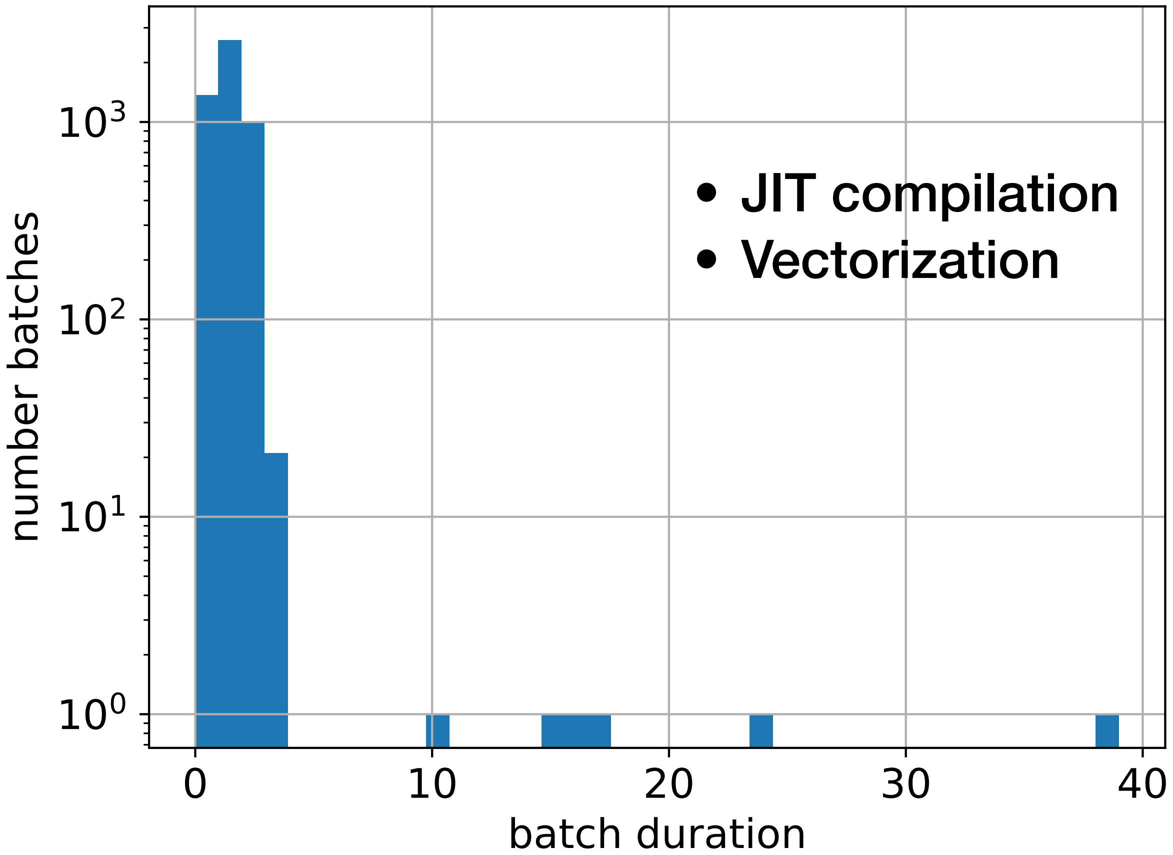
JAX implementation
https://github.com/pgranger23/larnd-sim-jax/tree/main

PyTorch implementation



Batch size = 100 cm
Batch run time ~ 25 s

JAX implementation



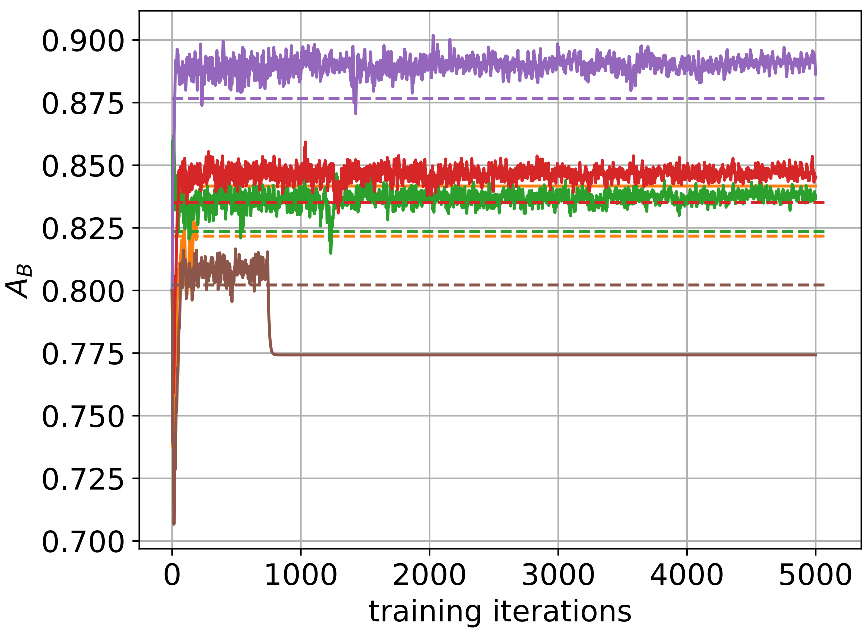
Batch size = 2000 cm
Batch run time < 2s
Extrapolate from PyTorch version would be 500s

JAX Fit Result (without Electronic Noise) 🥵

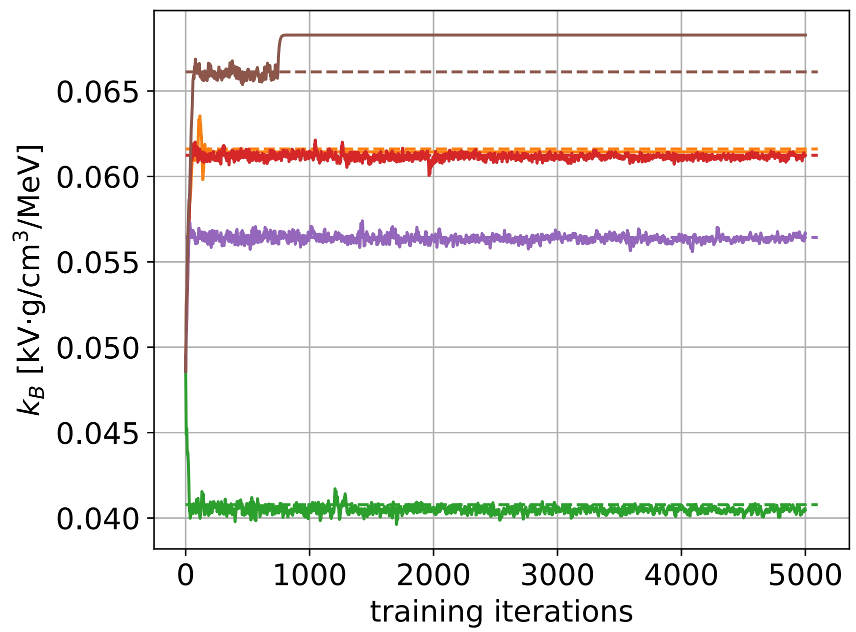
Chamfer

5 fits with different targets in 6D phase space.
The fits use 2000 cm batch.

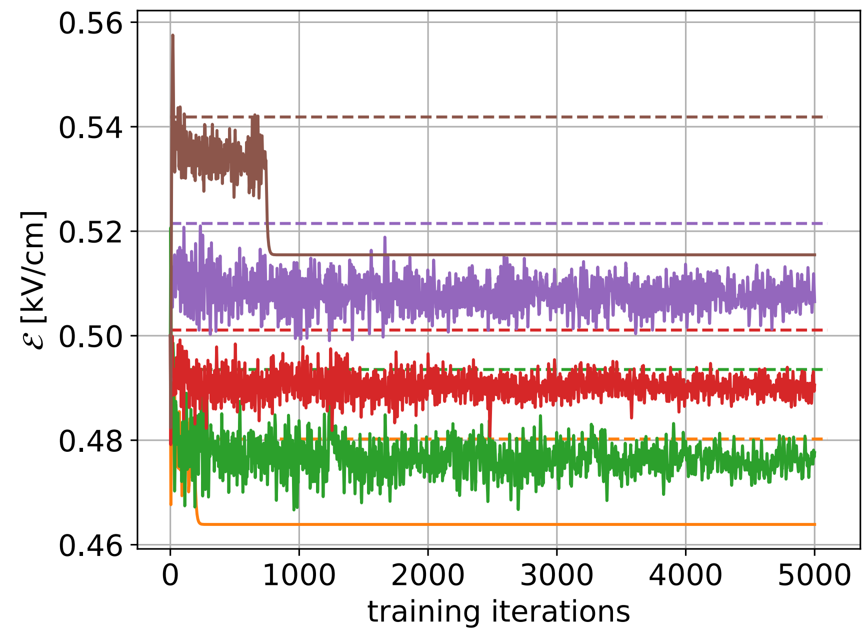
Recombination model A_B



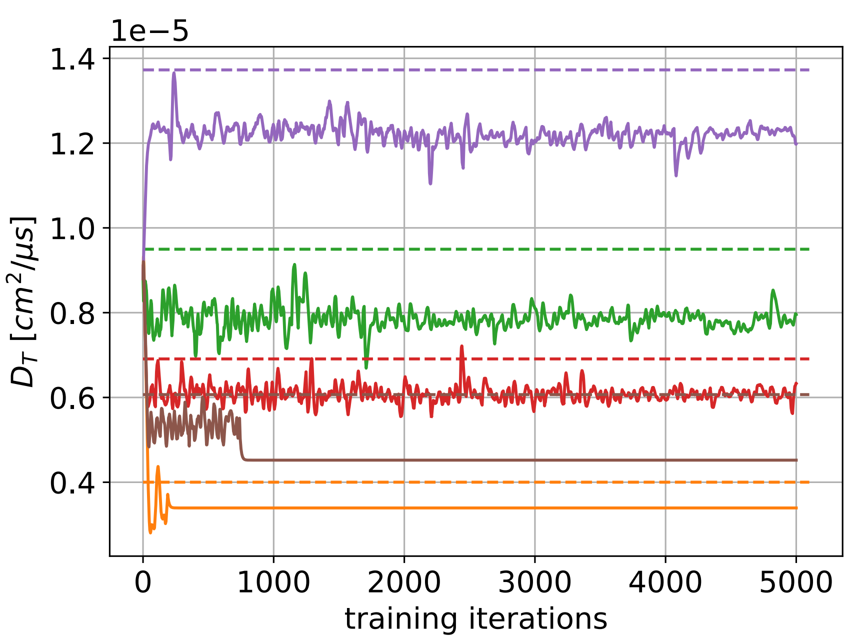
Recombination model k_B



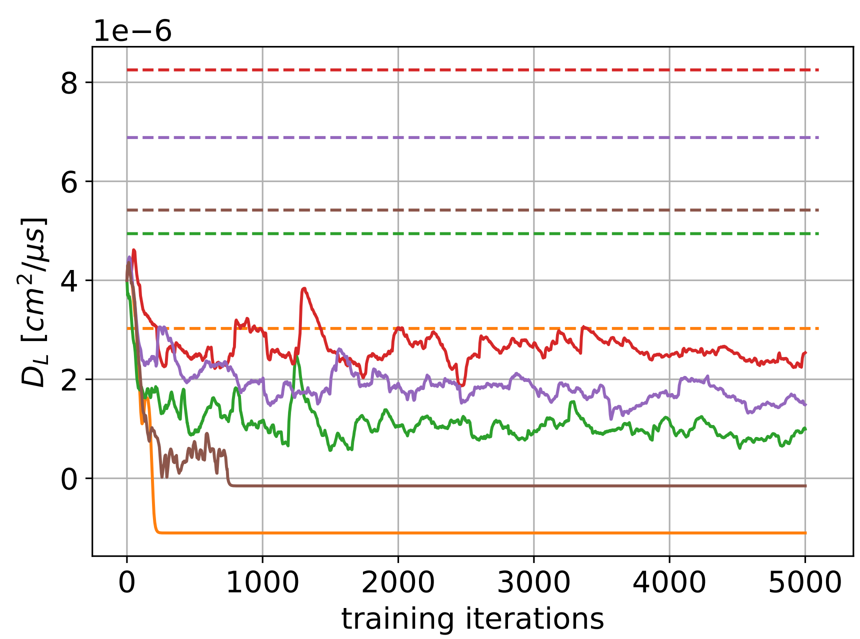
Electric field



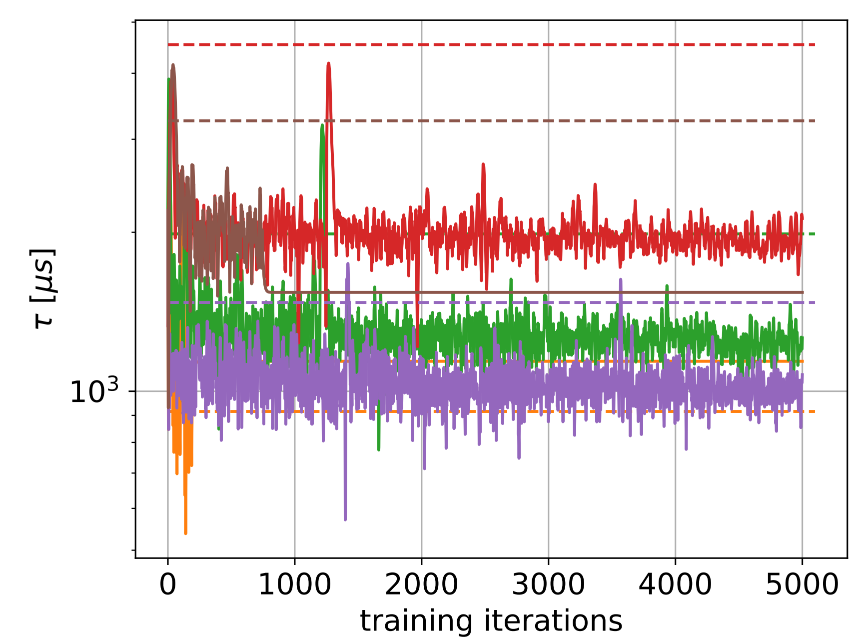
Transverse diffusion coefficient



Longitudinal diffusion coefficient



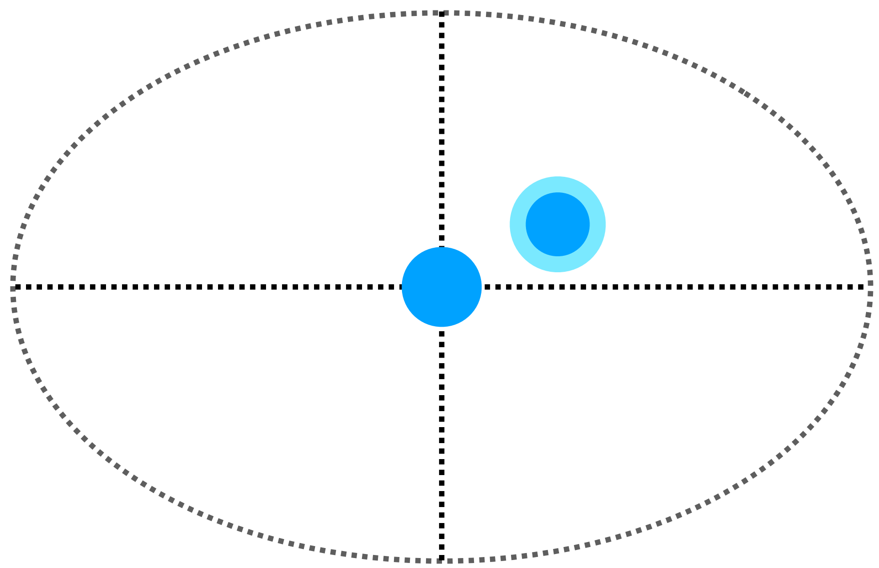
Electron lifetime



Electron Sampling according to the Diffusion

$$\sigma_T = \sqrt{2 \cdot t_{\mathrm{drift}} \cdot D_T}$$

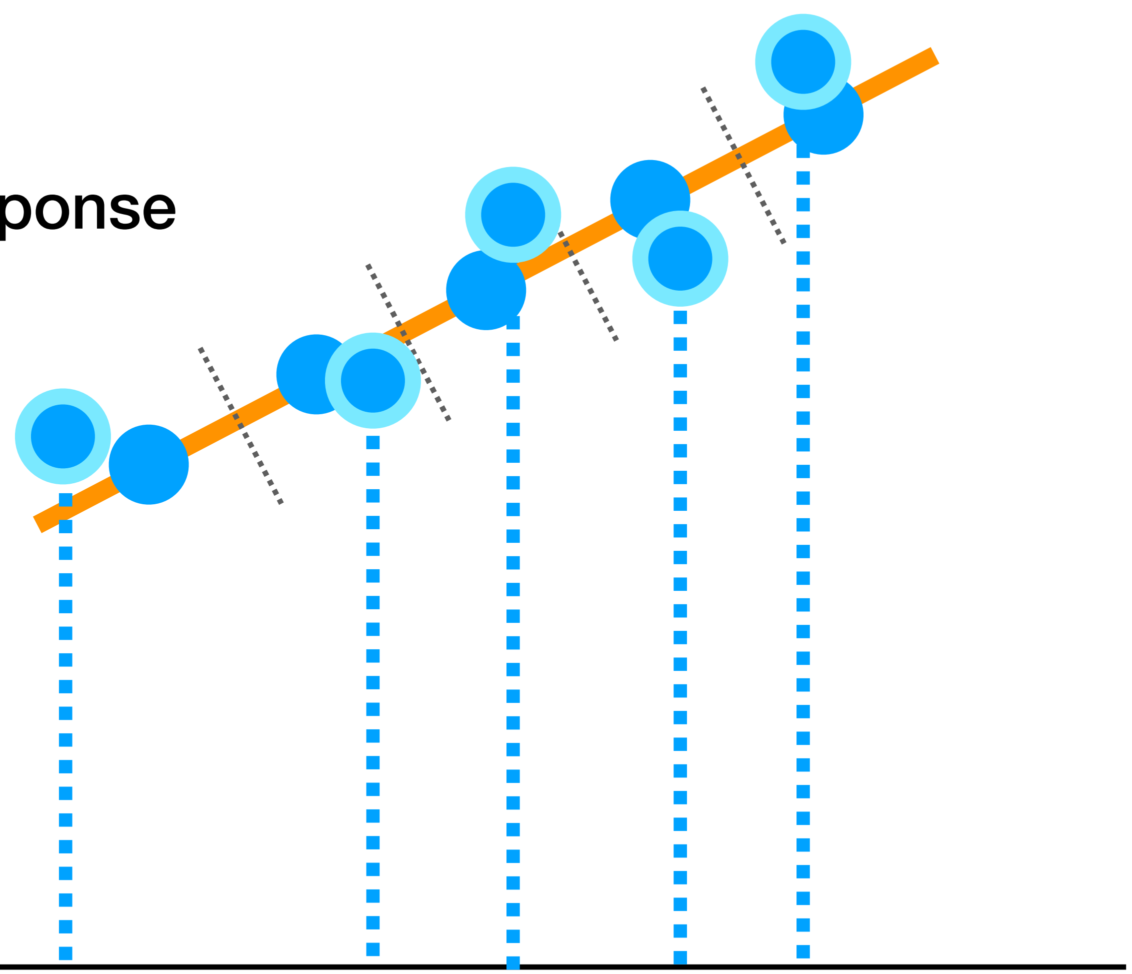
Random Gaussian sampling



$$\sigma_L = \sqrt{2 \cdot t_{\mathrm{drift}} \cdot D_L}$$

The sampled charge for detector response

The initial charge



Subject to sampling noise

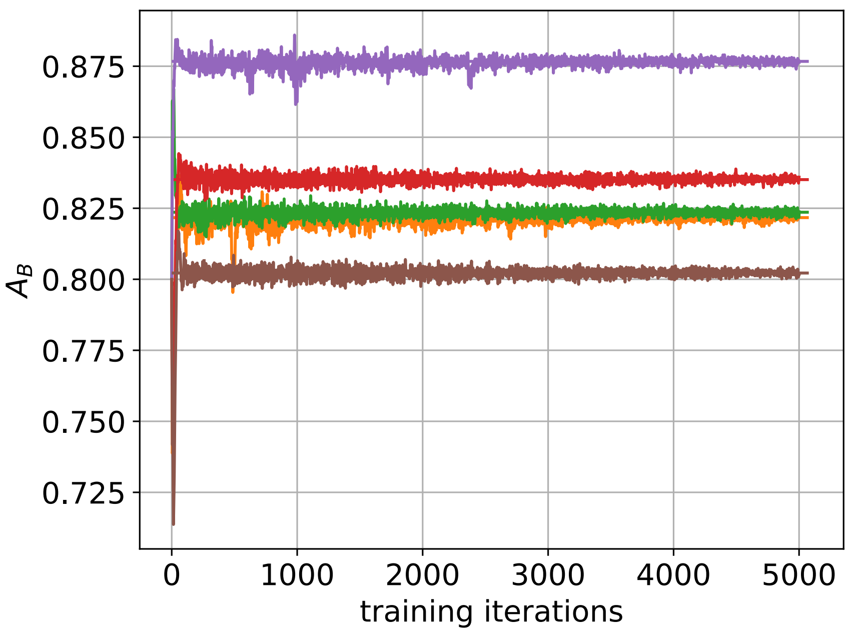
Anode

JAX Fit Result (without Electronic Noise)

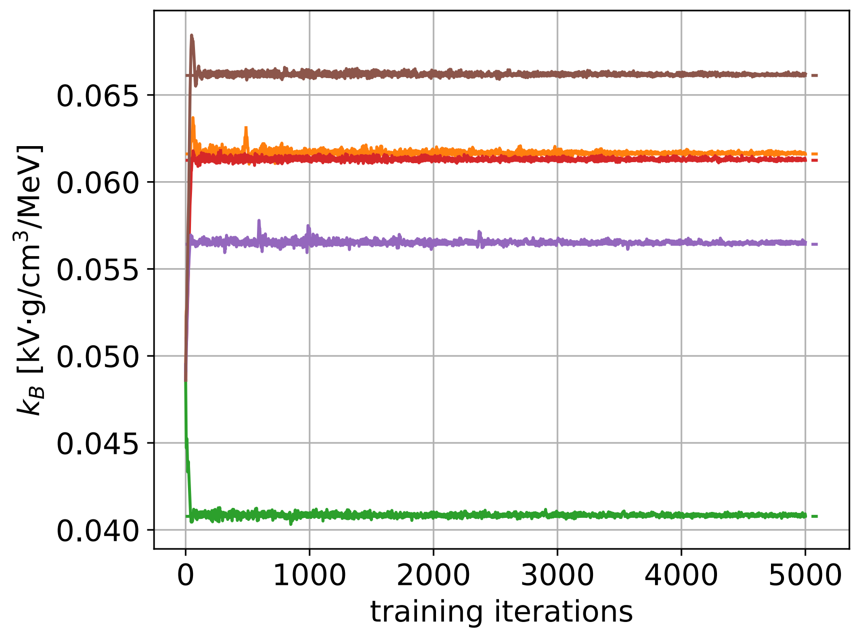
Chamfer

5 fits with different targets in 6D phase space.
The fits use 2000 cm batch.
Consistent random electron sampling seed

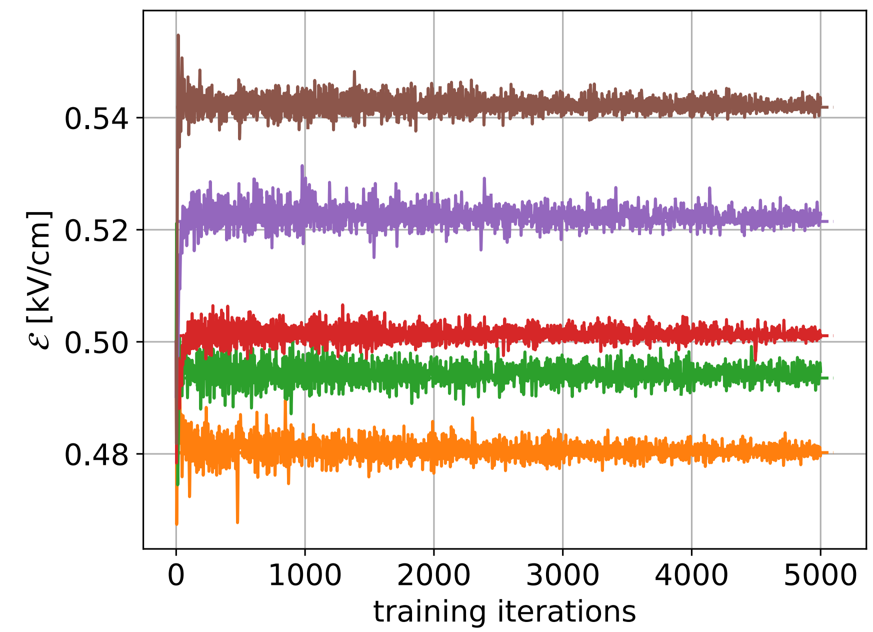
Recombination model A_B



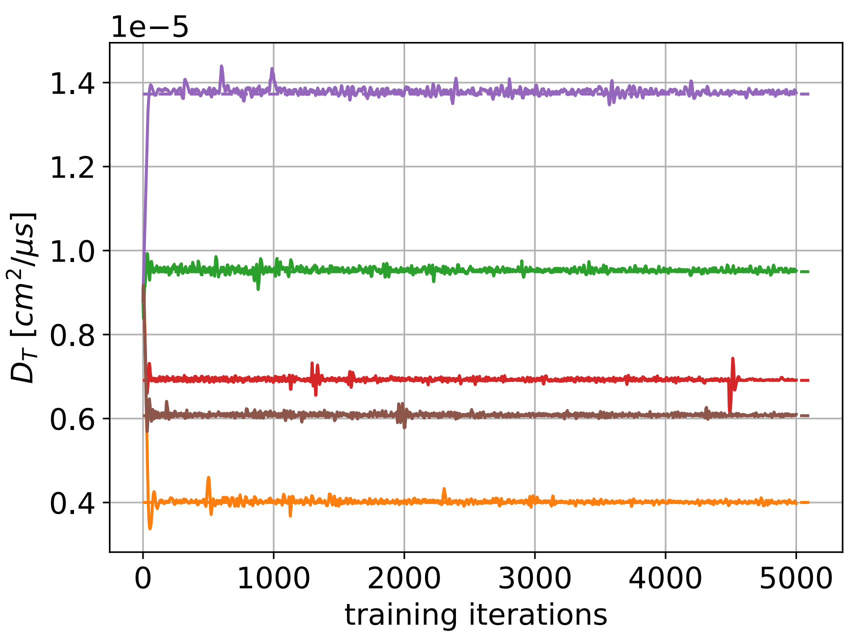
Recombination model k_B



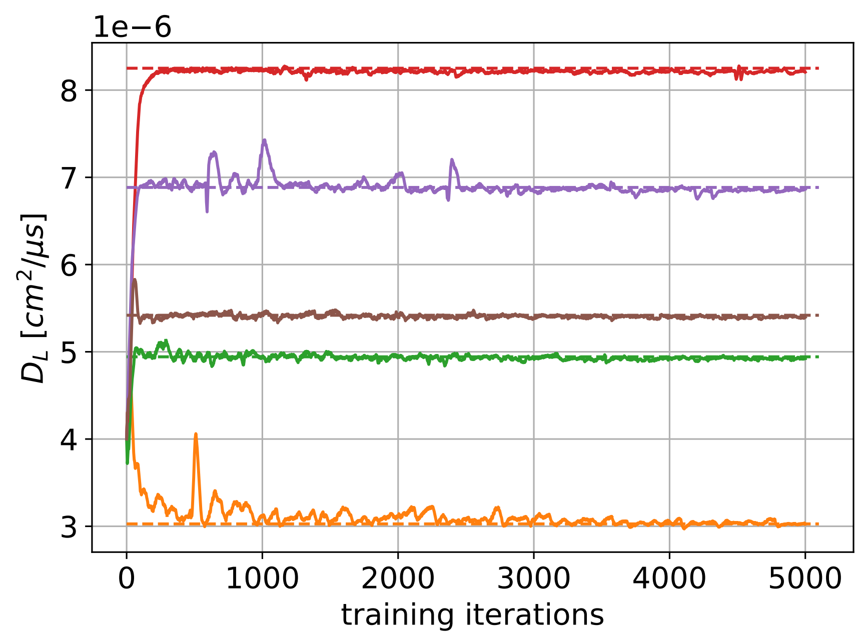
Electric field



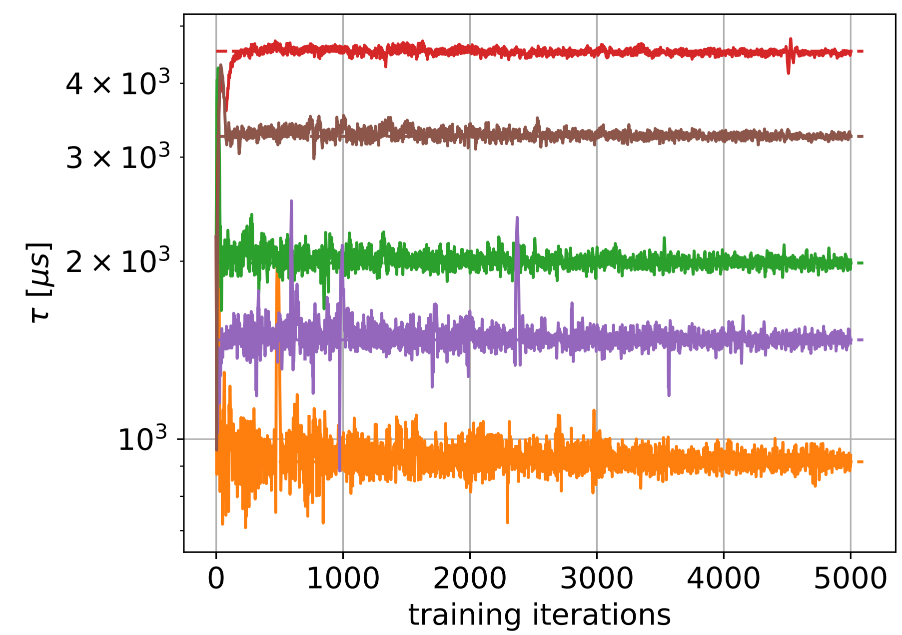
Transverse diffusion coefficient



Longitudinal diffusion coefficient



Electron lifetime



Milestones Towards Data Application

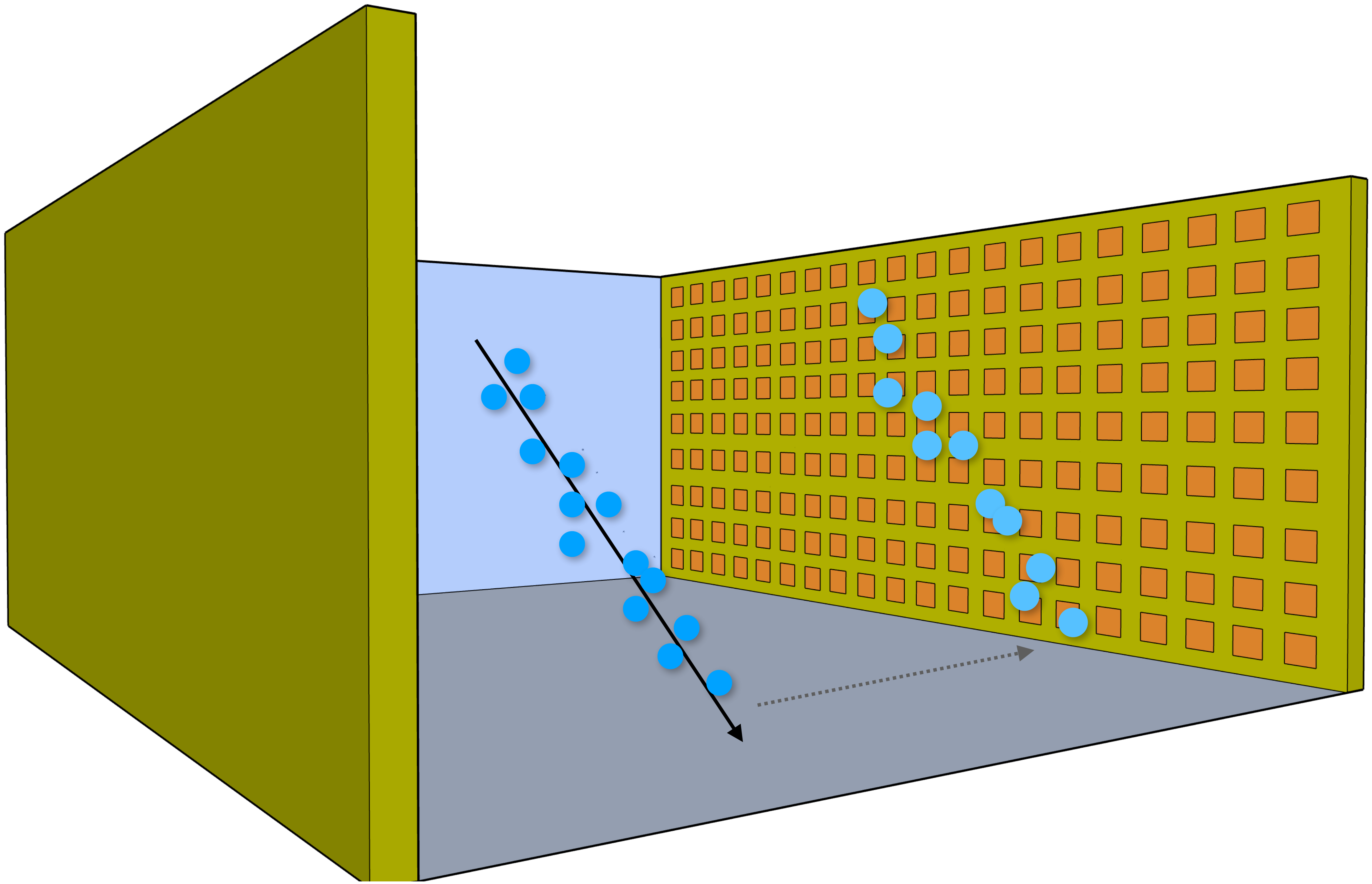
Overarching goal: simultaneous data calibration with 2x2 data

- Fit fake data with the current electron sampling
  - Perhaps the sampling noise is to be avoided with other integration methods
- Fit fake data with electronic noise
- Fit fake data with reconstructed tracks
  - Only reconstructed dE/dx
  - Reconstructed dE/dx and position
- Understand the impact of the selection purity and reconstruction bias
- Data sample selection
  - Proton sample
  - Estimate the sample statistics
  - Data – (forward) simulation comparison
  - Understand the differences and mitigate the non-calibration related differences
    - May need an implicit model to absorb the differences?
- Data calibration with single module data
  - Potentially introducing more parameters for the calibration

Workshop Goal and To-do's

- Fit fake data with the current electron sampling
  - Check the simulation output distribution with respect to chopping steps and batch size (4/10-11)
  - Test chopping steps and batch size within the tolerance of memory (4/10-11)
  - If necessary explore other integration methods (second week)
- Fit fake data with electronic noise (possibly 4/14-16)
- Fit fake data with reconstructed dE/dx (possibly 4/16-18)

LArTPC: Liquid Argon Time Projection Chamber



About the Fit

Parameter [Units]	**Nominal Value**	**Range**
A_B	0.8	[0.78, 0.88]
k_B $[kV.g/cm^3/MeV]$	0.0486	[0.04, 0.07]
$\mathcal{E}$ $[kV/cm]$	0.5	[0.45, 0.55]
τ $[\mu s]$	2200	[500, 5000]
D_L $[cm^2/\mu s]$	4×10^{-6}	$[2 \times 10^{-6}, 9 \times 10^{-6}]$
D_T $[cm^2/\mu s]$	8.8×10^{-6}	$[4 \times 10^{-6}, 14 \times 10^{-6}]$

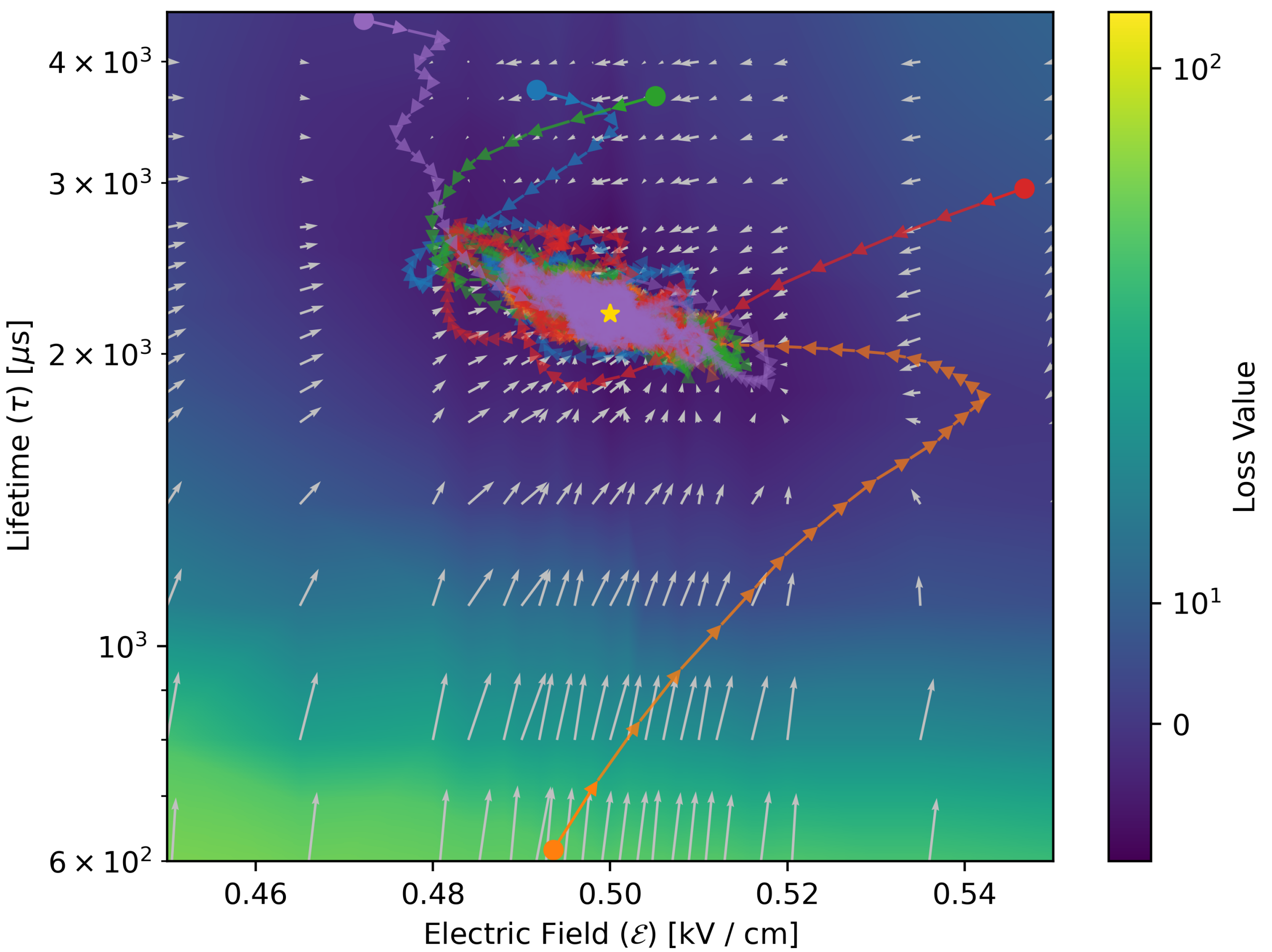
- Normalize the parameters with their nominal values for gradient calculation
- Gradient clips on normalized gradient
- Exponential learning rate decay
- Use Chamfer / MSE / Soft Dynamic Time Warping (Soft DTW) for the loss
- Recover the parameter values for the forward simulation

Calibration: Fit 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 $\theta_0 \rightarrow \theta_i$ based on the gradients

Iterate step 2. to 5.



For gradient descent, the parameter update takes form of

$$\theta_{i+1} = \theta_i - \eta \cdot \nabla_\theta \mathcal{L}(f(\chi, \theta_i), F_{\text{target}})$$

We use Adam for the optimiser