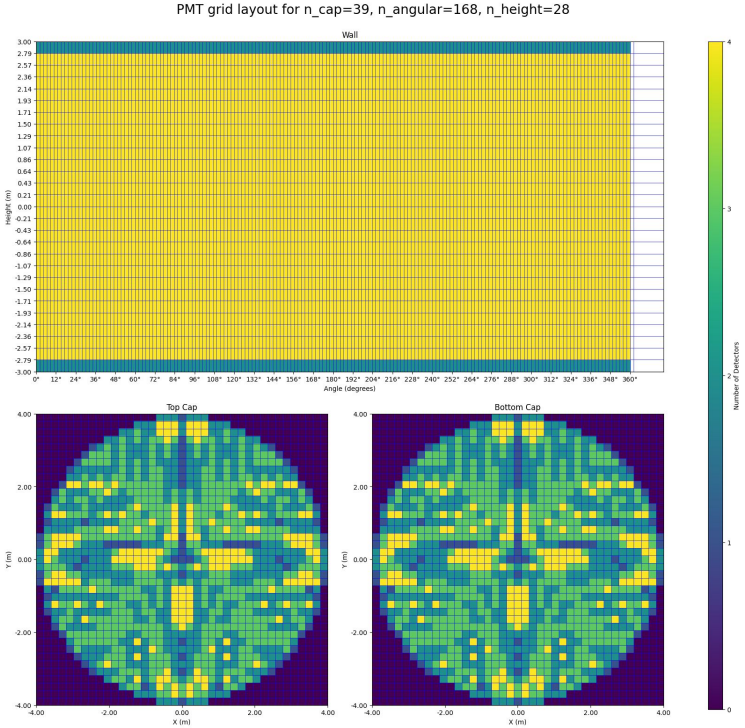
CIDER Weekly Updates

Omar Alterkait

Graduate Student, Tufts University

Grid Aliasing Effect



Performance Enhancement:

- Implemented a grid-based system to partition the detector space
- Primary goal was to reduce computational overhead in ray tracing calculations by limiting the search space

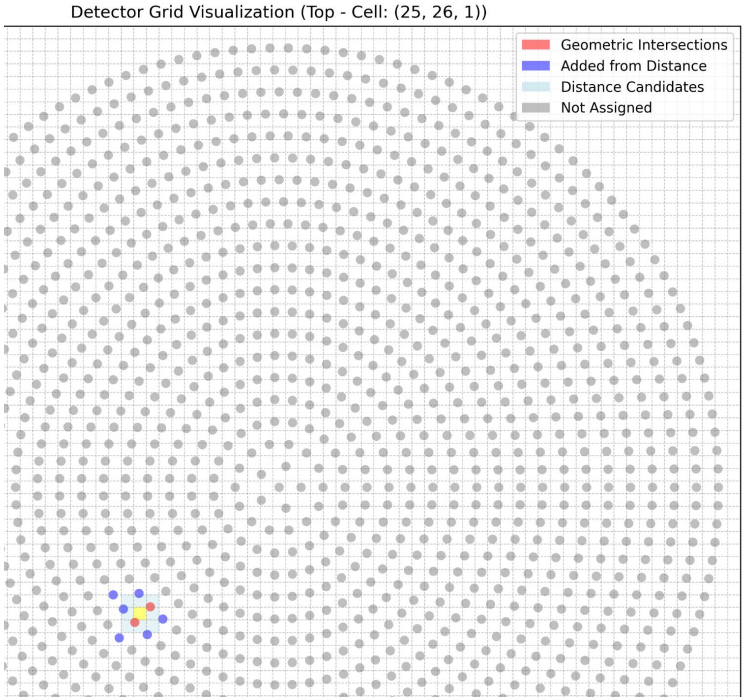
Grid Cell Occupancy Issue:

- Square grid cells didn't align optimally with detector's geometric structure
- Resulted in uneven PMT distribution across cells:
  - Some cells contained no PMTs
  - Other cells had minimal PMT coverage
  - Created "dead zones" in the detection space

Impact on Photon Detection:

- Problem became apparent at higher photon relaxation settings
- When photons could affect more distant PMTs:
  - Photons hitting low-occupancy cells were insufficiently processed
  - Some photons effectively "missed" potential PMT interactions due to grid cell emptiness

Added more PMTs to each grid



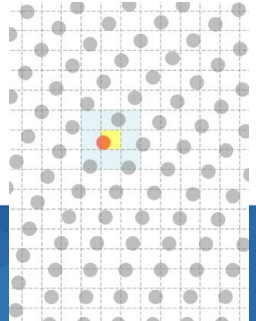
JAX Static Optimization:

- JAX performs best with static-sized arrays
- Previous implementation used fixed-size lists (max_detectors_per_cell) for each grid cell
- Empty cells still maintained allocated space in memory

New Approach - Complete Grid Population:

- Fill every grid cell to maximum capacity using two-step process:
  - First priority: Add PMTs that directly intersect with the cell
  - Second priority: Fill remaining slots with nearest PMTs by distance
- Eliminates empty or partially filled cells, maintaining consistent computation patterns

Before:



Grid Debugging

Previous Grid System:

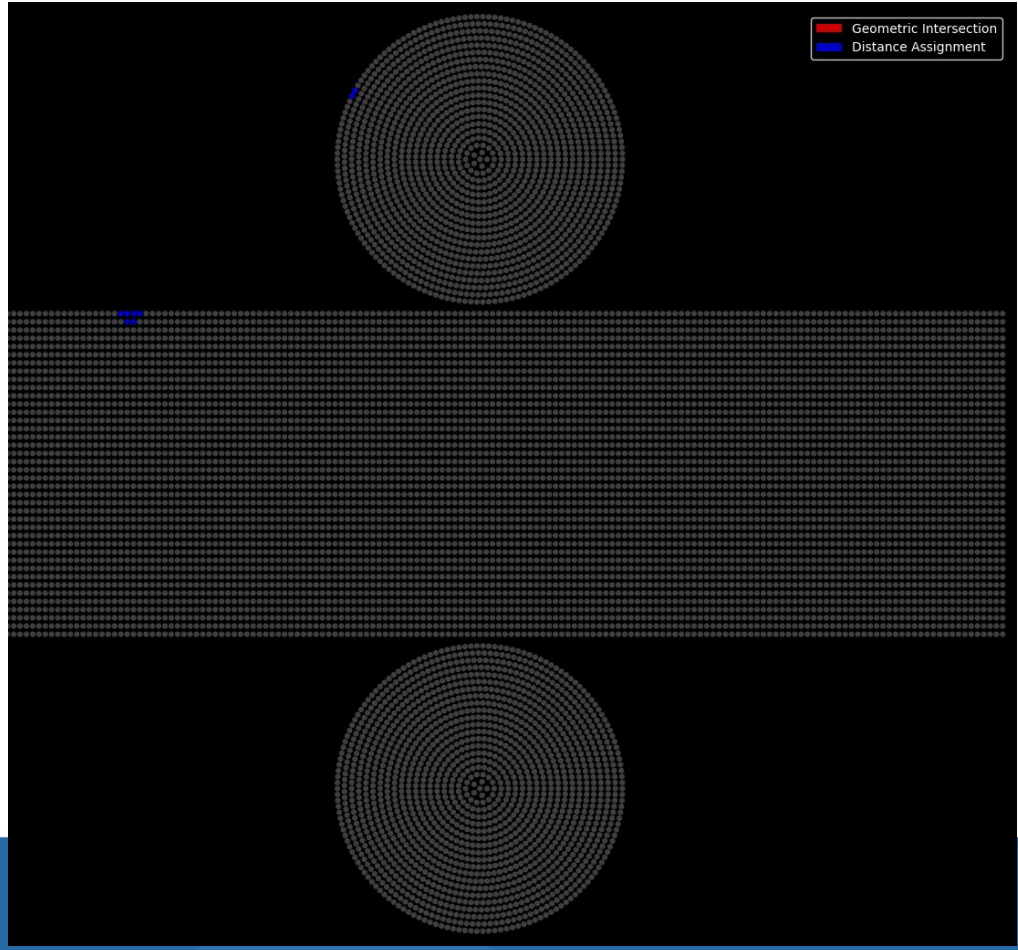
- Sequential filtering process: Check wall/cap hits first
- Each region (walls, caps) had separate PMT lists
- Grid checks were performed only within the hit region

New Complication:

- Grid cells now contain PMTs from multiple regions
- Example: Top cap grids can contain wall PMTs
- Original region-based filtering no longer valid

Resolution:

- Redesigned filtering logic to handle mixed-region grid cells
- Required significant debugging and testing



Photon Relaxation

Photon Relaxation

$$\frac{1}{1+\frac{d^2}{\gamma^2}}$$

Relaxed Ray Tracing:

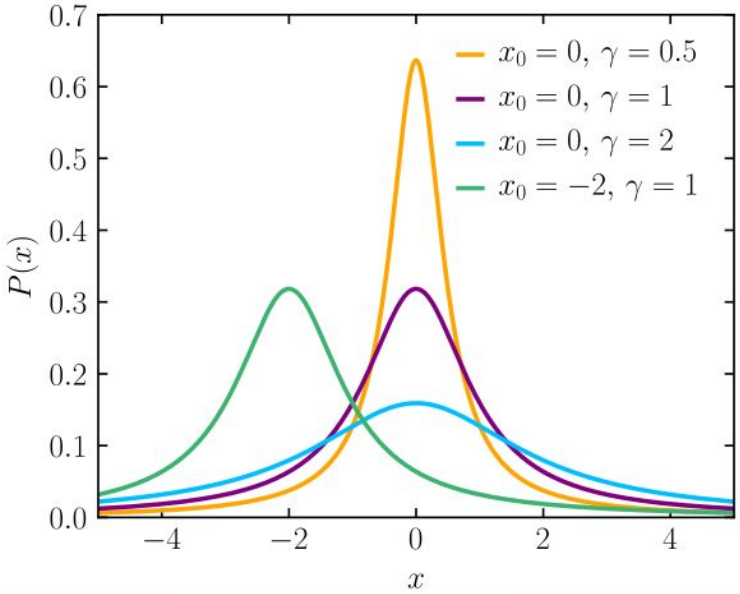
- Moved from binary (hit/miss) to continuous photon contributions
- Single photon now affects multiple PMTs simultaneously
- Results in smoother gradients and reduced degeneracy issues

Contribution Calculation:

- Calculate closest approach between photon path and PMT centers
- Apply Cauchy distribution to determine contribution strength
- Temperature parameter (gamma) controls distribution spread

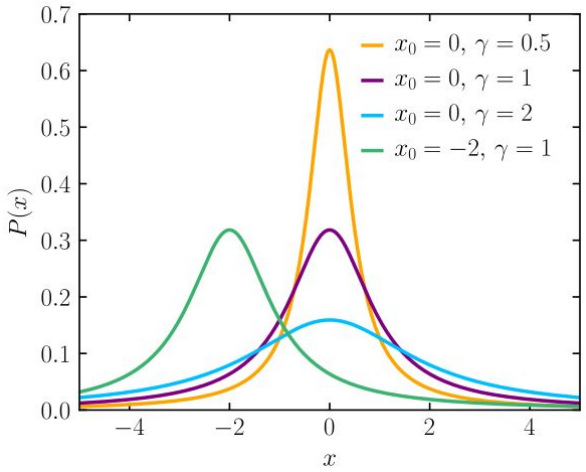
Normalization Process:

- Monitor total contribution of each photon across all PMTs
- When total exceeds 1, normalize by dividing all contributions
- Maintains physical conservation of photon energy



Photon Relaxation

$$\frac{1}{1+\frac{d^2}{\gamma^2}}$$



Current Scaling Approach:

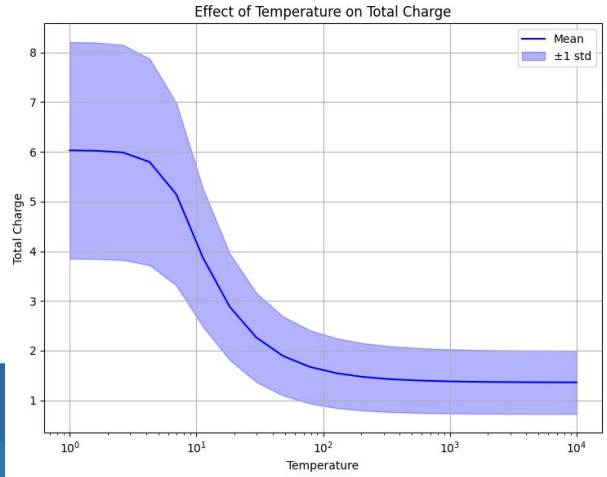
- Normalize photon contributions when sum exceeds 1
- Divide all contributions by total sum
- Simple but creates temperature-dependent variations

Observed Problems:

- Results vary significantly with temperature changes
- Total intensity lacks consistency across settings
- No straightforward normalization method available

Proposed Solution:

- Model the “contribution” to each PMT as the photon overlapping with a circular detector on the wall
- Represent photons as distributions with spatial falloff
- Requires precise calculation of 2D overlap integral between the circle and falloff
- Would provide more physically accurate detection model



Photon Relaxation: Gaussians

Integration Challenge:

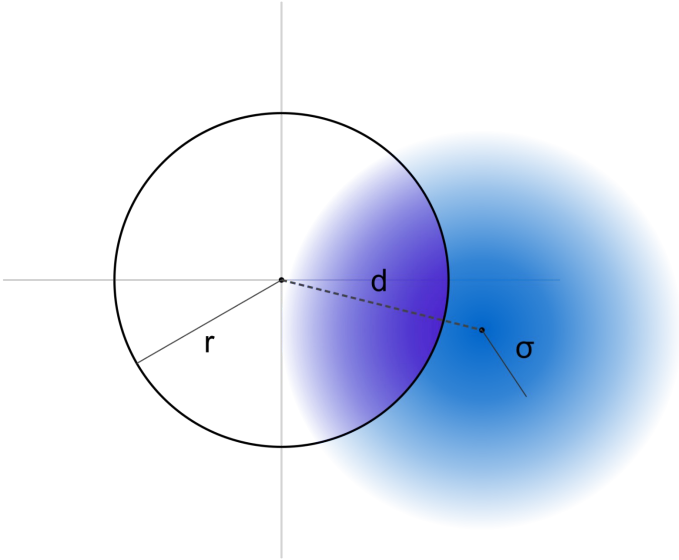
- Full 2D overlap integral needed
- Parameters: PMT radius (r) and distribution width (sigma)
- Computationally intensive: ~4M calculations (1M photons × 4 PMT checks)

Optimization Strategy:

- Leverage problem symmetry
- Pre-calculate overlap as function of distance (d)*
- Distance measured to closest point on trajectory
- Reduces computational overhead significantly

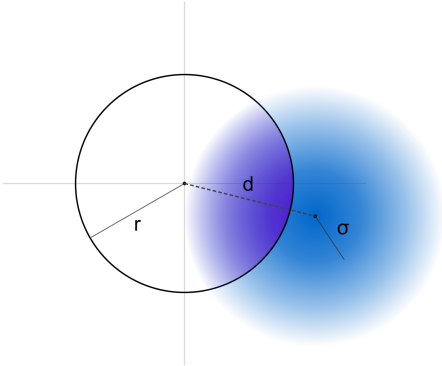
Distribution Options:

- Implemented both Gaussian and Cauchy distributions
- Switched primary focus to Gaussian model
- Code maintains flexibility between both options



* for now d is the closest point along the photon's path and the center of the pmt, not the distance along the wall

Overlap Calculation



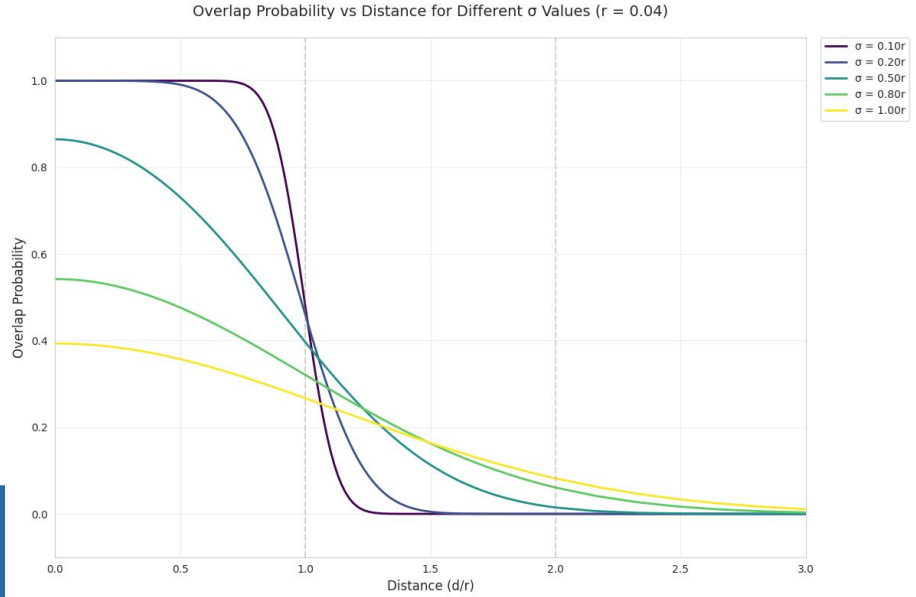
1D differentiable lookup table in JAX.

Save the overlap values and the gradient at that location and interpolates between them

Overlap Probability Values at Selected Distances:

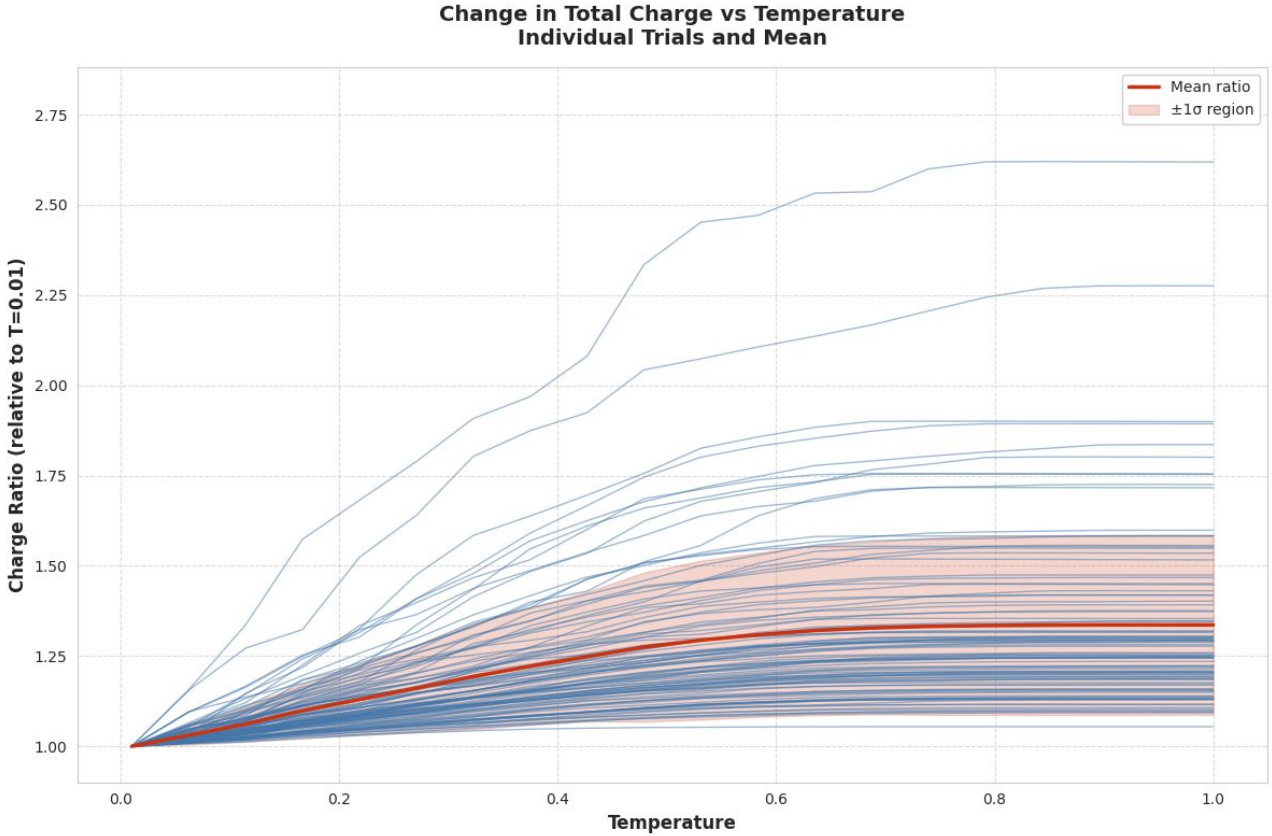
Overlap Probability Values at Selected Distances:

σ	d = 0	d = 2r	d = 3r	d = 4r
0.10r	1	5.37e-24	0	0
0.20r	1	1.99e-07	1.07e-23	0
0.50r	0.865	0.0147	1.87e-05	6.18e-10
0.80r	0.542	0.0609	0.00322	4.28e-05
1.00r	0.393	0.0819	0.0109	0.000604



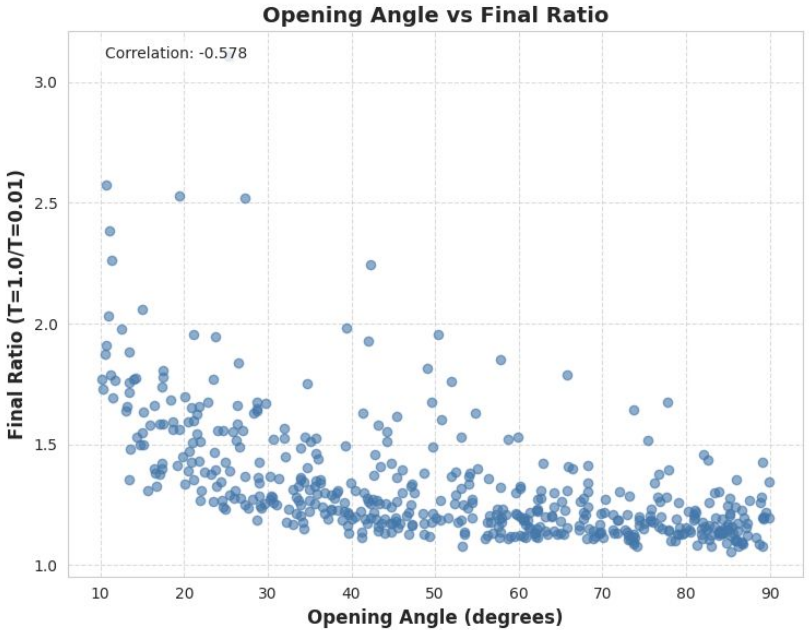
Looking at Intensity Changes

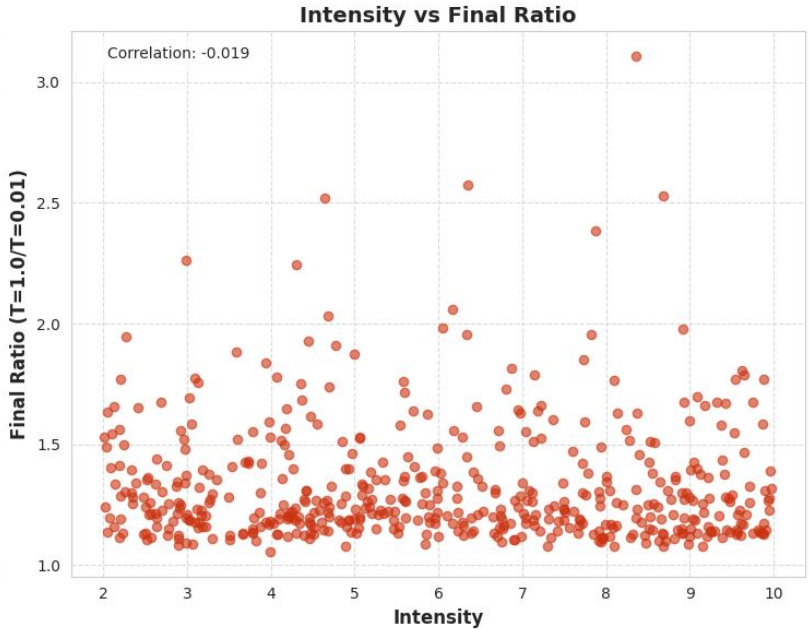
We see that as we increase the temperature, the overall charge increases (we get more contribution to other PMTs from smearing).



Looking at Intensity Changes

For some reason, the biggest factor regarding this is small opening angle cones. We see at a temperature of 1.0, we get an increasing total charge ratio at lower opening angles. These are randomly generated parameters. Needs further investigation. But this ratio is much less than before





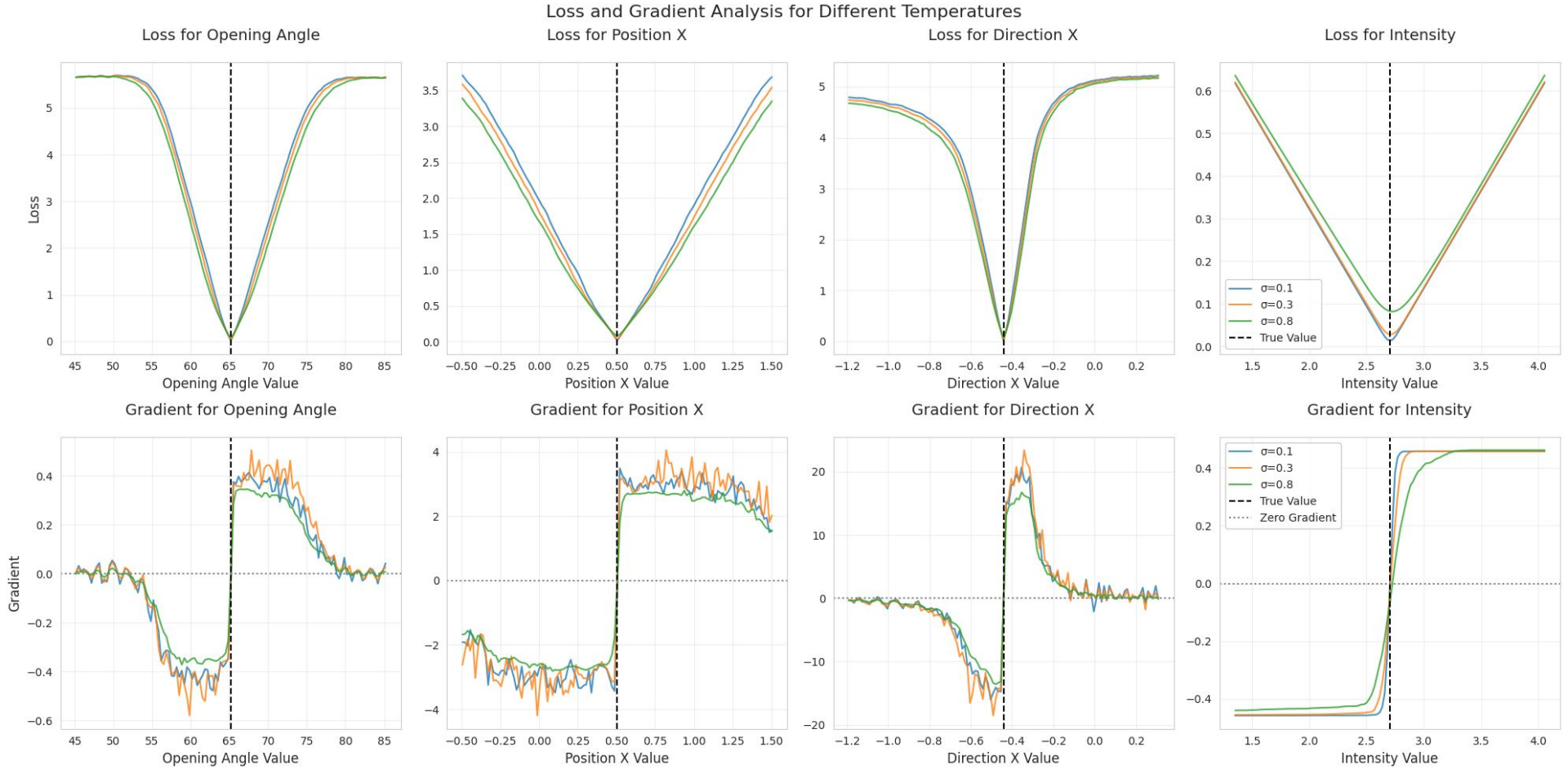
Loss and Gradients

PMT Comparison Loss Function

- Core Concept:
  - We compare the relationships between all PMTs in both simulated and true data, using two key hyperparameters (λ, τ)
- Distance Calculation:
  - Compute pairwise distances between all PMT combinations.
  - Scale these distances using parameter τ to establish a meaningful distance metric
  - Apply softmin operation between simulated and true data points to create weight matrices in both directions. Ignore the non active PMTs
  - This is a smoothing of finding the single closest sim point to each true point.
- Time Alignment:
  - Remove mean values from both simulated and true time measurements
  - This adjustment compensates for unknown t0 timing offsets
- Loss Components:
  - Calculate weighted differences between all PMT pairs using the weight matrices
  - Combine charge and time losses: Total Loss = L_charge + λ L_time

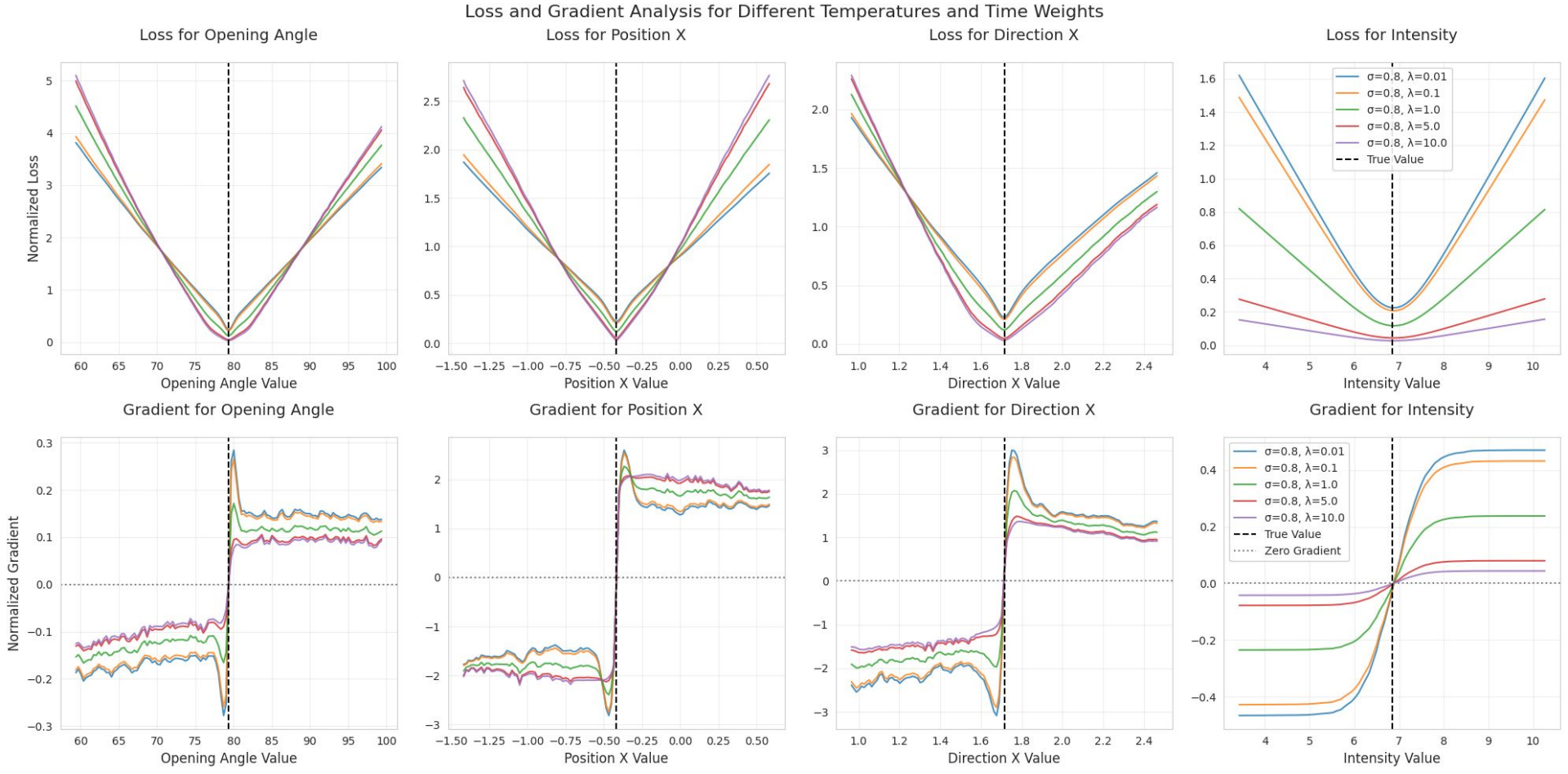
Gradient Validation

Loss and gradients are more robust to hyperparameters

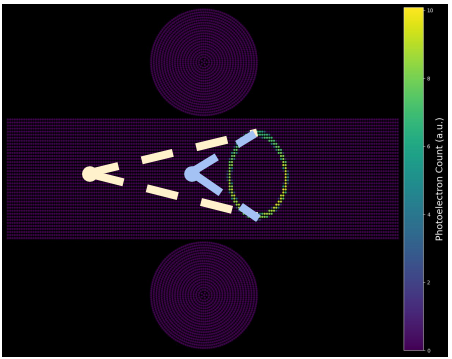


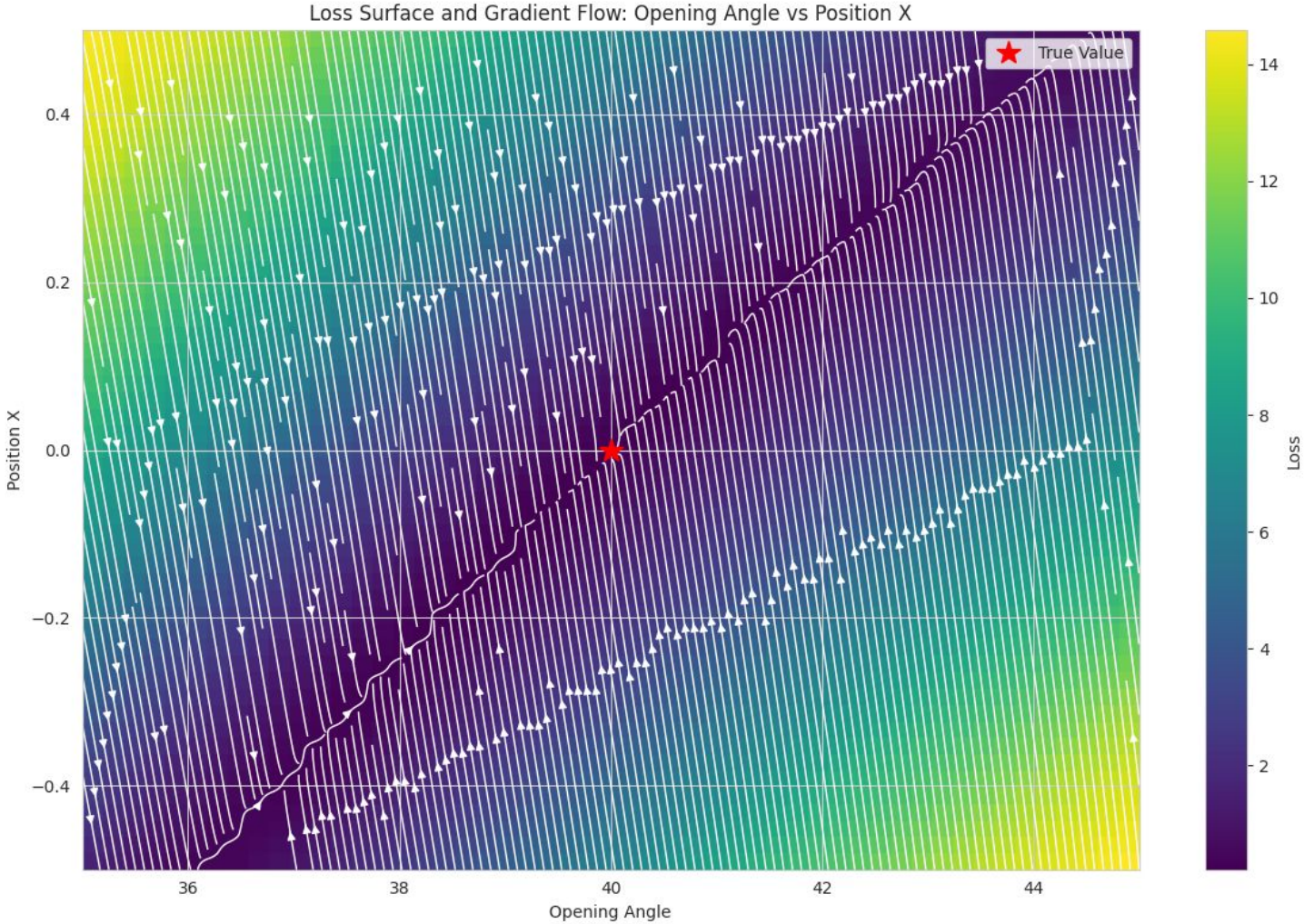
Gradient Validation

Loss and gradients are more robust to hyperparameters



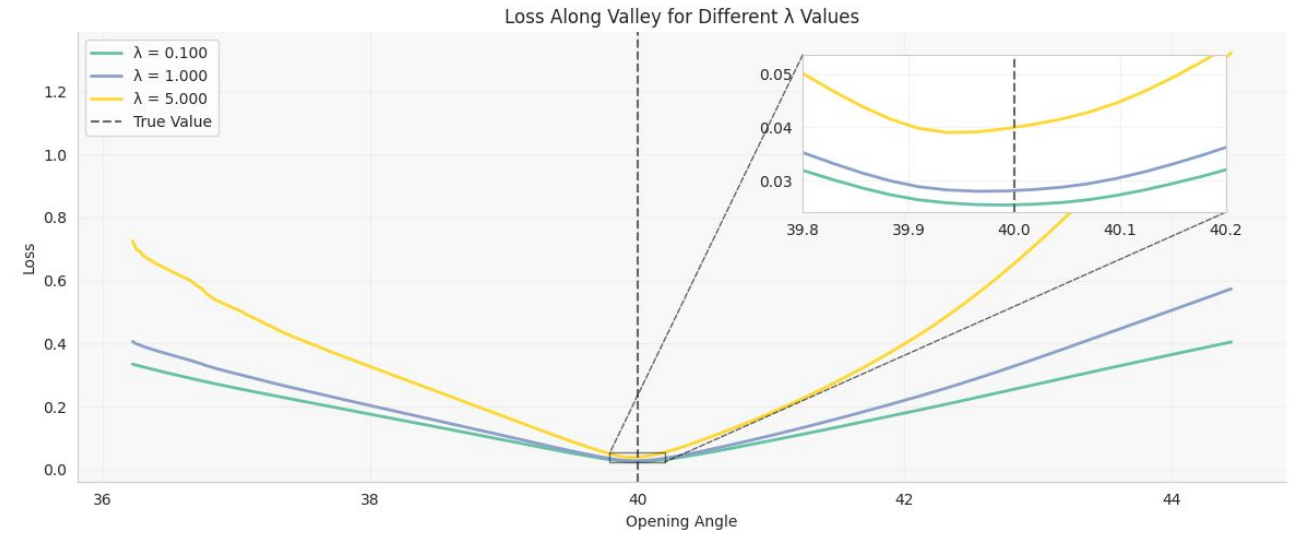
Degeneracy

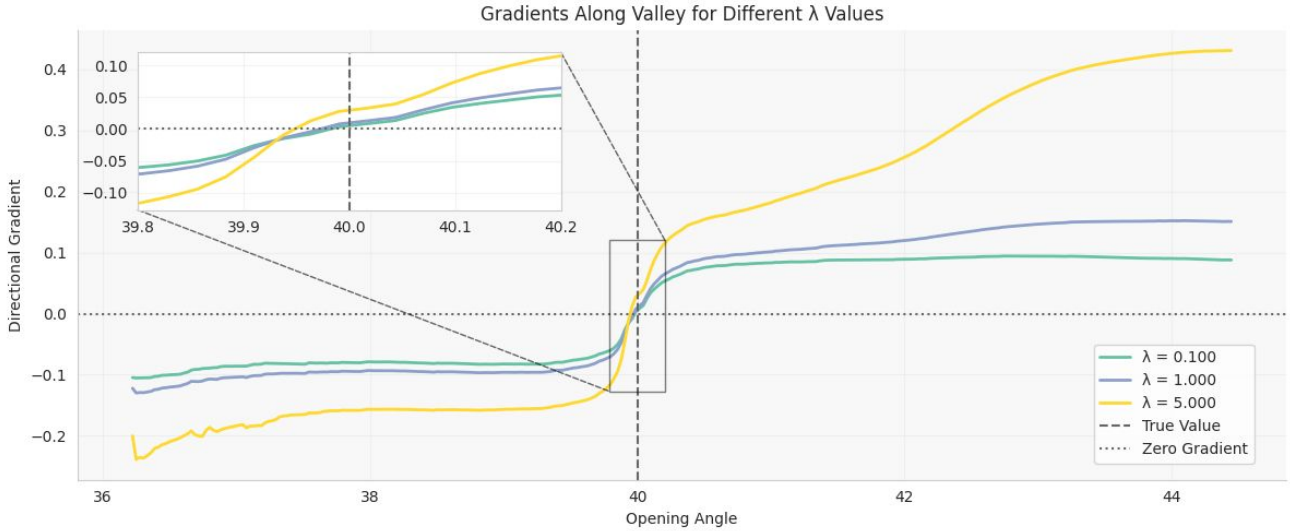

Photoelectron Count (a.u.)


Loss Surface and Gradient Flow: Opening Angle vs Position X
True Value
Position X
0.4
0.2
0.0
−0.2
−0.4
Opening Angle
36
38
40
42
44
Loss
14
12
10
8
6
4
2

Degeneracy Valley

This is the gradient along the valley in the direction of the valley. This is with sigma = 0.1





Training

Gradient Descent: Initial Guess

Created a simple initial parameter guess estimator

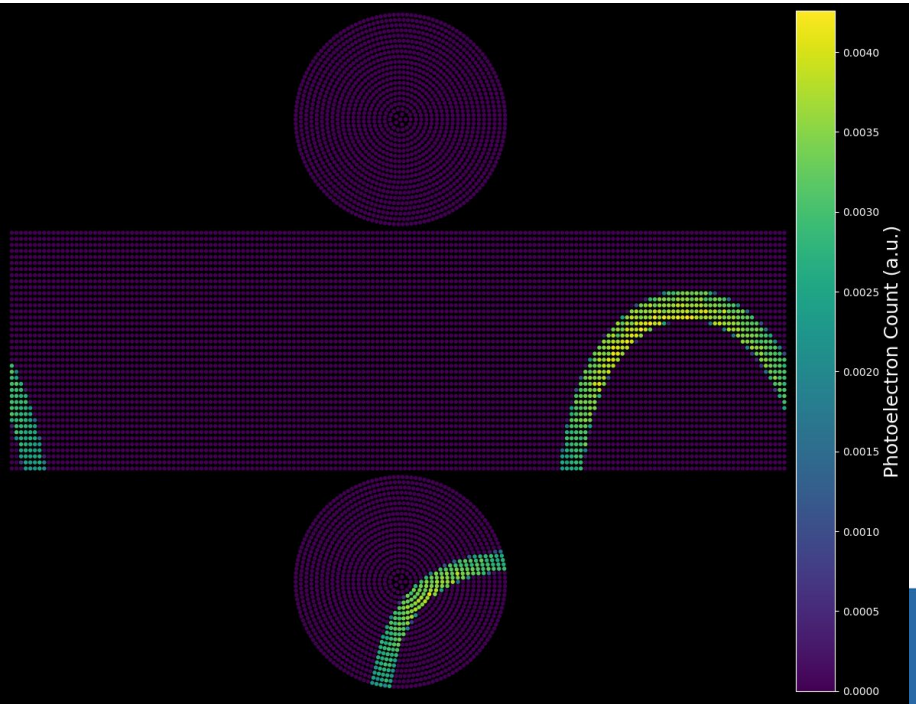
- Starting Position:
  - Begin with a defined reference point (typically choosing the coordinate origin)
  - This provides an anchor for subsequent calculations
- Direction Calculation:
  - Compute the primary direction using charge-weighted contributions from all PMTs
  - Direction vector emerges from the starting position
- Angular Analysis:
  - Calculate the opening angle formed by PMT positions relative to the trajectory
  - This angle helps characterize the event geometry
- Intensity Estimation:
  - Scale the total observed charge by a calibration factor
  - Final intensity = Total Charge × Intensity Scale Factor

Examples

Event Parameters:

Opening Angle: 45.32 degrees
Initial Position: (0.59, 0.00, 0.66)
Initial Direction: (0.53, 0.60, -0.63)
Initial Intensity: 8.33

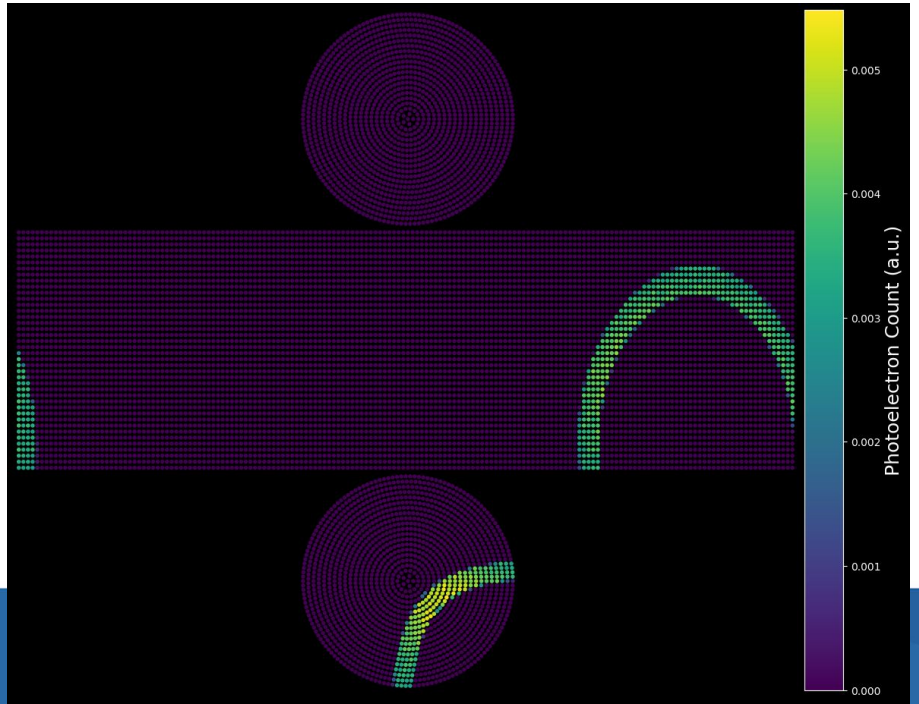
True



Event Parameters:

Opening Angle: 42.30 degrees
Initial Position: (-0.04, -0.49, -0.07)
Initial Direction: (0.64, 0.68, -0.35)
Initial Intensity: 9.46

Guess

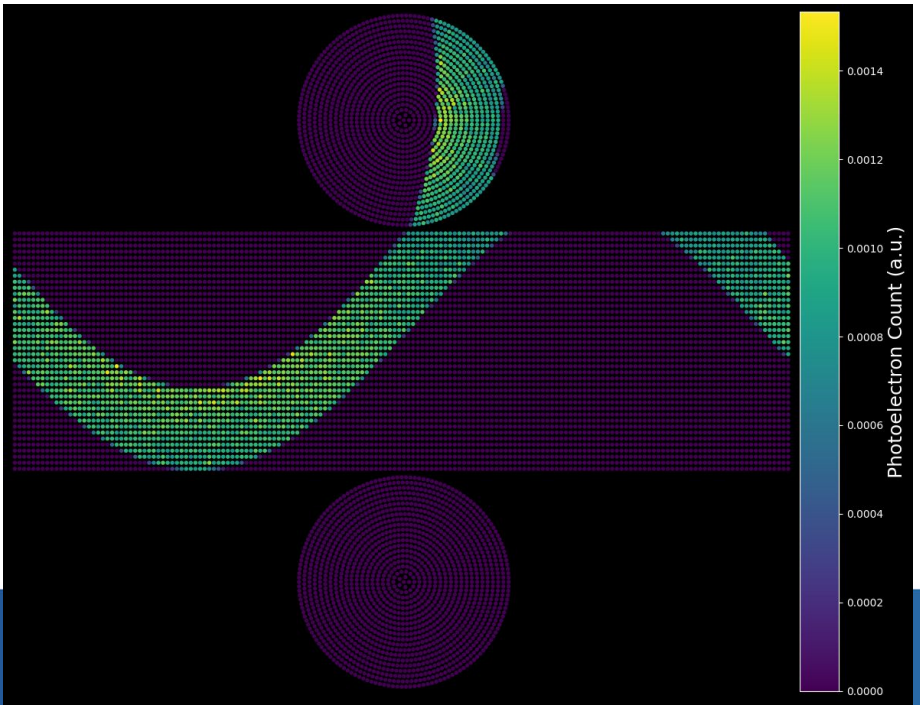


Examples

Event Parameters:

Opening Angle: 78.54 degrees
Initial Position: (-0.40, -0.05, -0.13)
Initial Direction: (-1.10, 0.13, 1.44)
Initial Intensity: 9.28

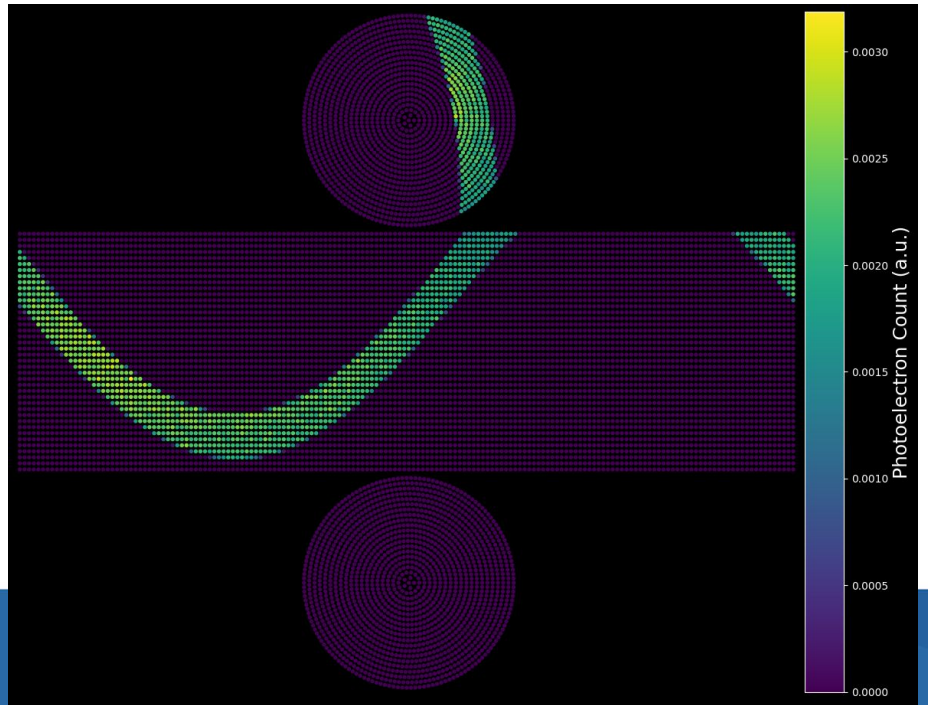
True



Event Parameters:

Opening Angle: 78.48 degrees
Initial Position: (-0.37, 0.27, -0.16)
Initial Direction: (-0.61, -0.12, 0.78)
Initial Intensity: 11.25

Guess

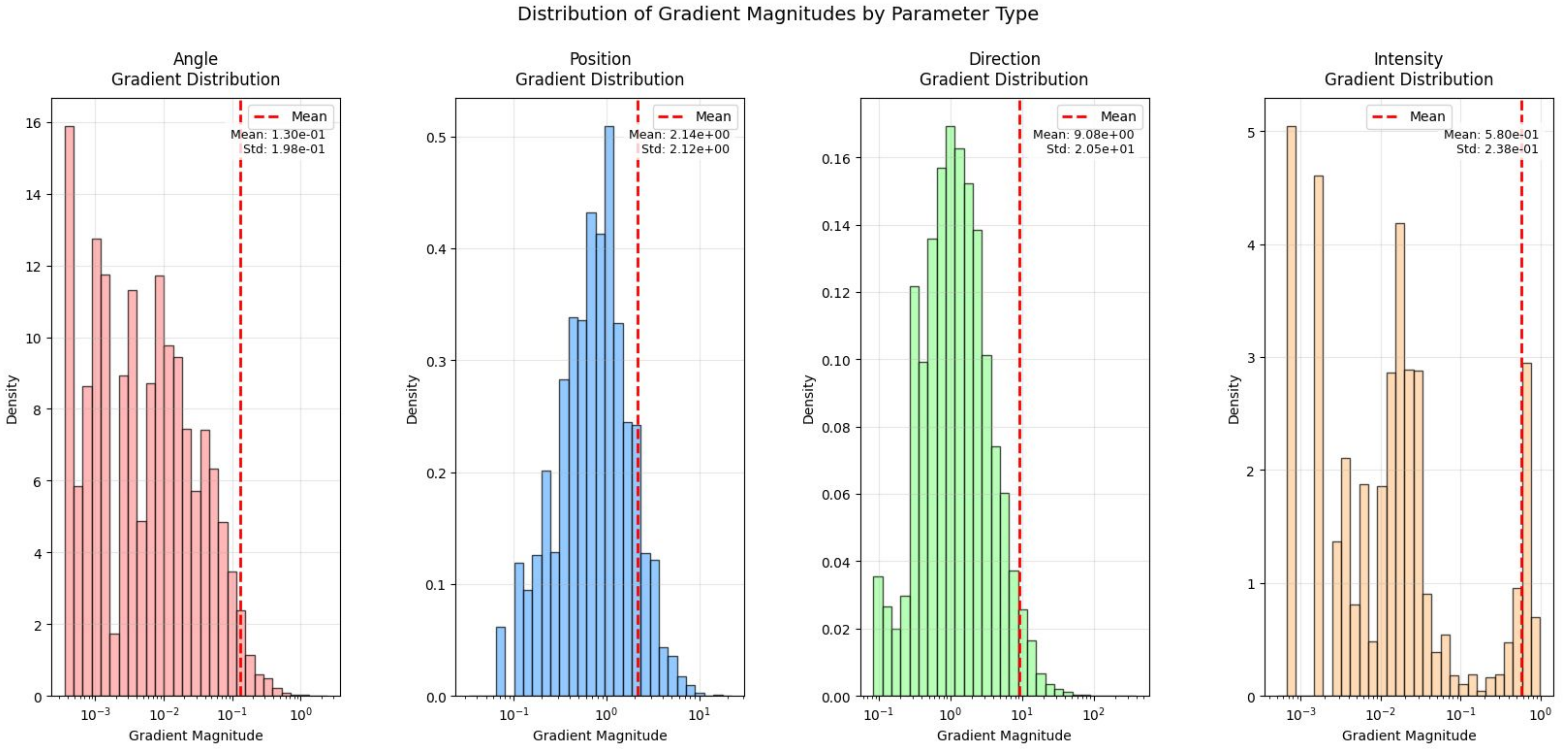


Gradient Descent: Normalizing Gradients

Need to normalize the gradients we get relative to the average gradient magnitudes

```
gradient_scales = {
    'angle': 7.706,
    'position': 0.466,
    'direction': 0.110,
    'intensity': 1.723,
}
```

scaled_gradient=
gradient * gradient_scales

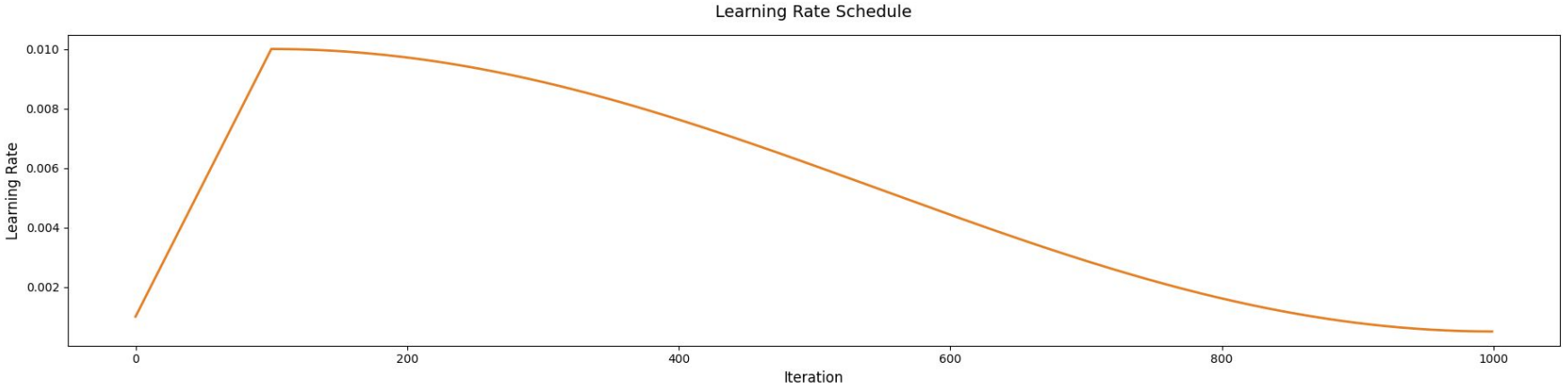


Optimizer Used

```
parameter_scales = {
    'angle': 90.0, # max opening angle
    'position': 5.0, # detector has height 6
    'direction': 1.0,
    'intensity': 8.0 # scale of random generator
}
```

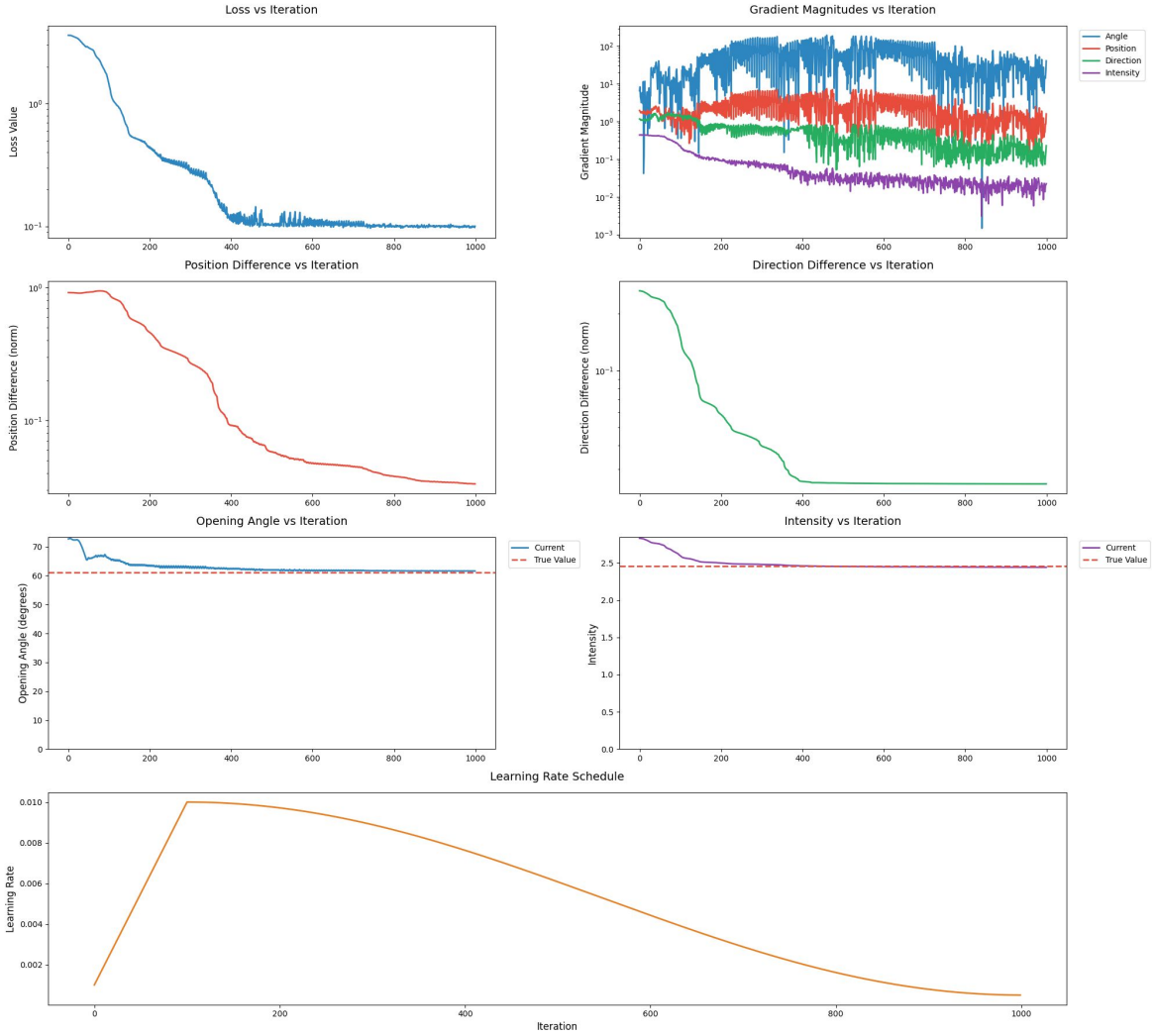
normalized_parameter = parameter/parameter_scales

- Normalize Parameters to get a similar scale.
- Used an SGD optimizer with momentum and gradient clipping
- Used a learning rate schedule of cosine annealing with warmup



Training Results

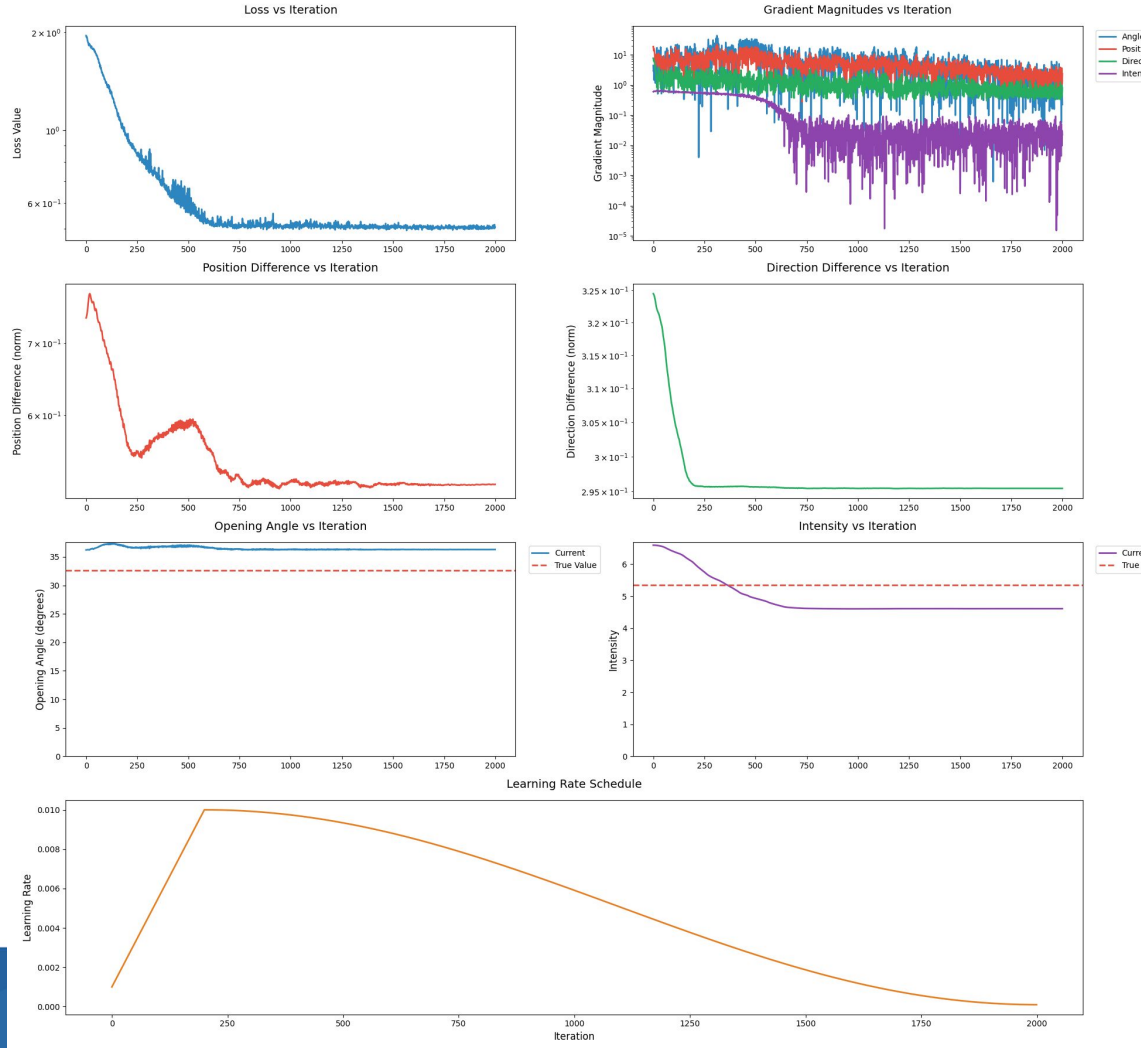
Some initial parameters were able to train completely.



Training Results

Some did not converge, this depended on the initial guess. The trainings that got stuck were almost always related to opening angle.

Example of a bad optimization that gets stuck in a local minimum

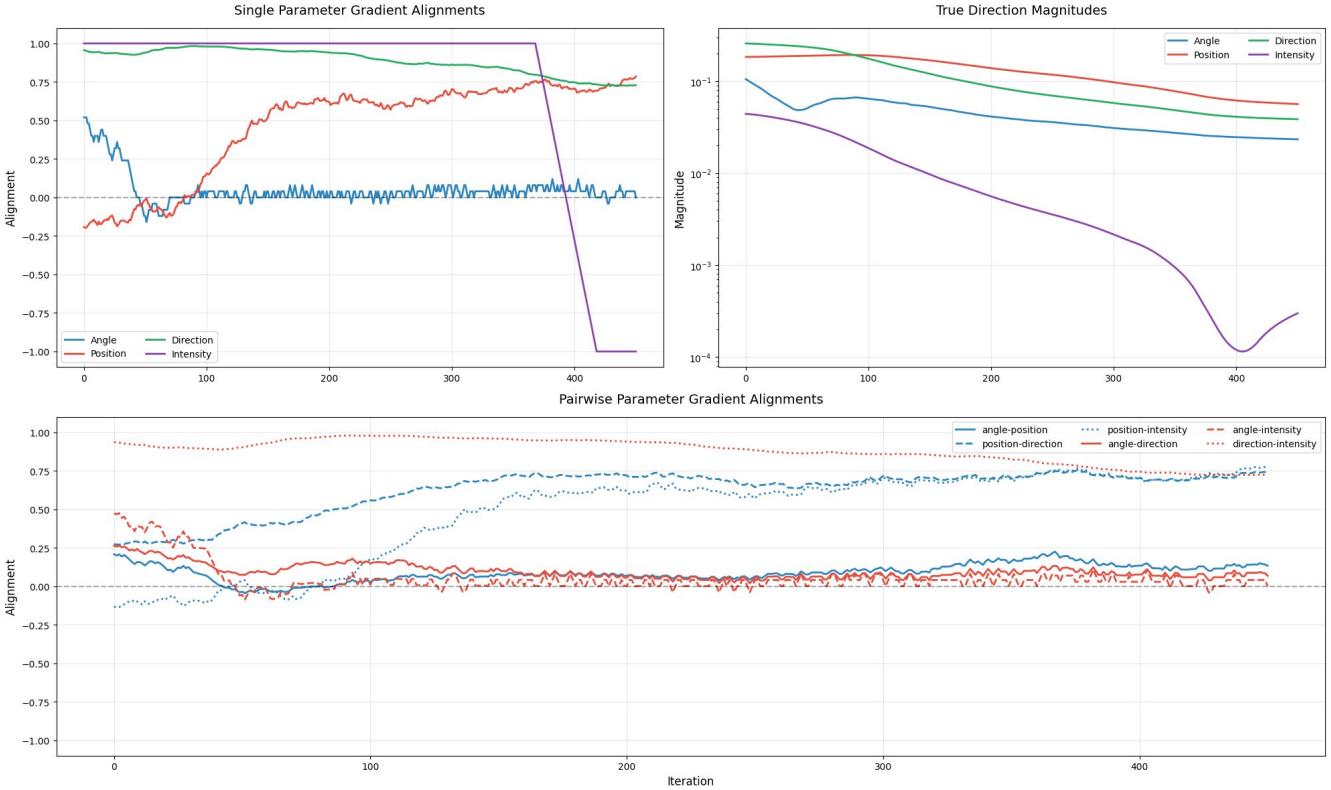


Gradient Analysis

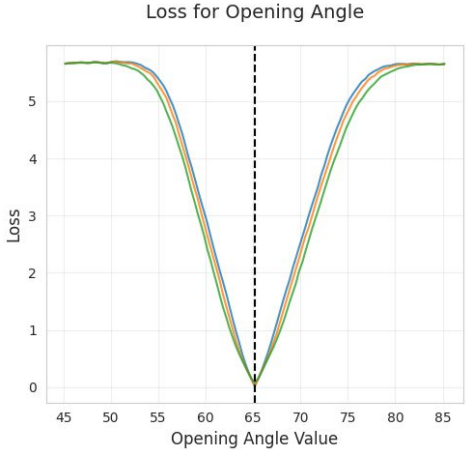
Created a way to visualize how correct the gradients are. This is just the cosine similarity between the current gradient and the direction to the true value.

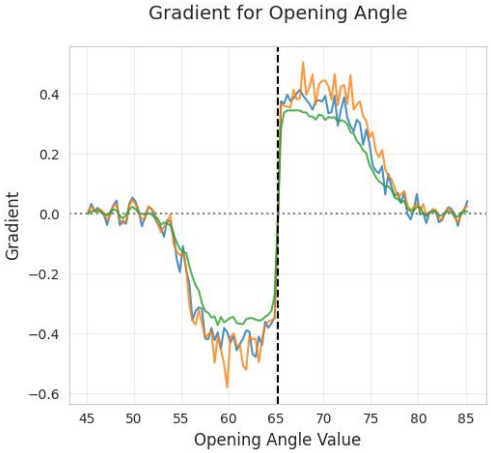
We want all the values to be close to 1.

The opening angle is near 0! This means this is almost random noise



Gradient Validation Opening Angle





Possible Culprits:

- Loss Surface is too harsh. The opening angle gradient for example becomes flat and then gradient descent for opening angle is overcome with noise! Need to consider different losses.
- The generate cone function may be causing some problems. Implementing the siren and a more realistic particle track could reduce this. This ties into Cesar's JAX implementation of the cherenkov siren