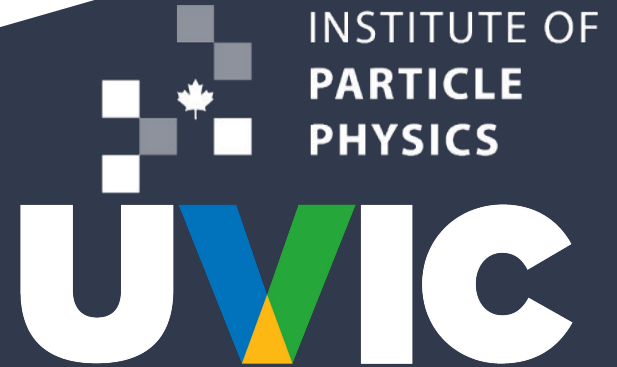


# Not So Straightforward: Linear Fits in Real Life

Caleb Miller

IPP 50th Anniversary Connect Postdoctoral Fellow

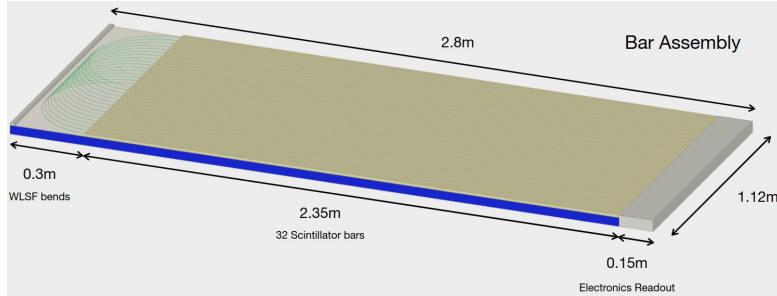
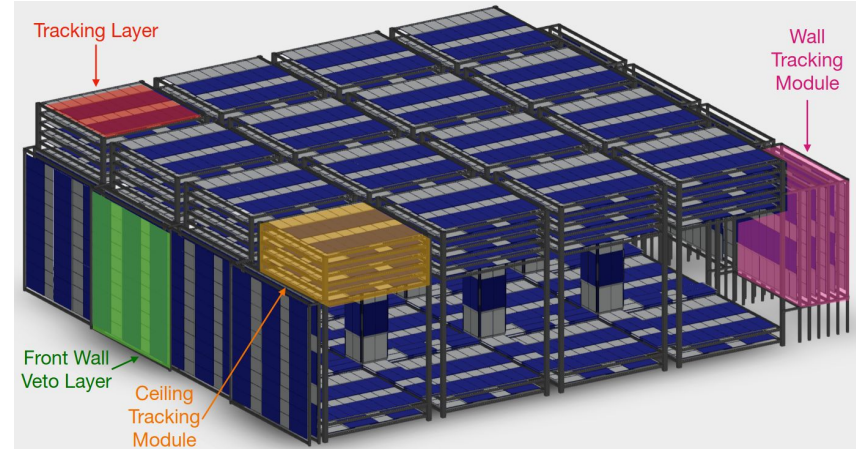
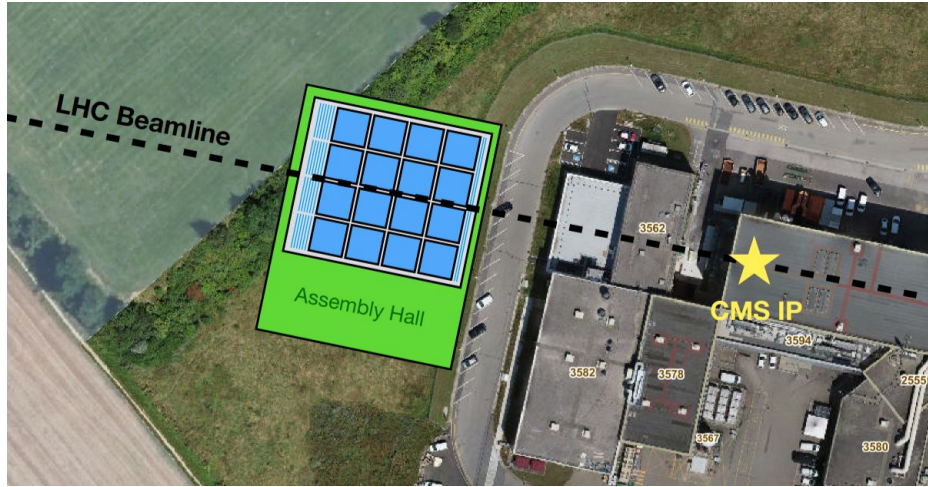
CAP 2025  
University of Saskatchewan



SFU

SIMON FRASER  
UNIVERSITY

# MATHUSLA

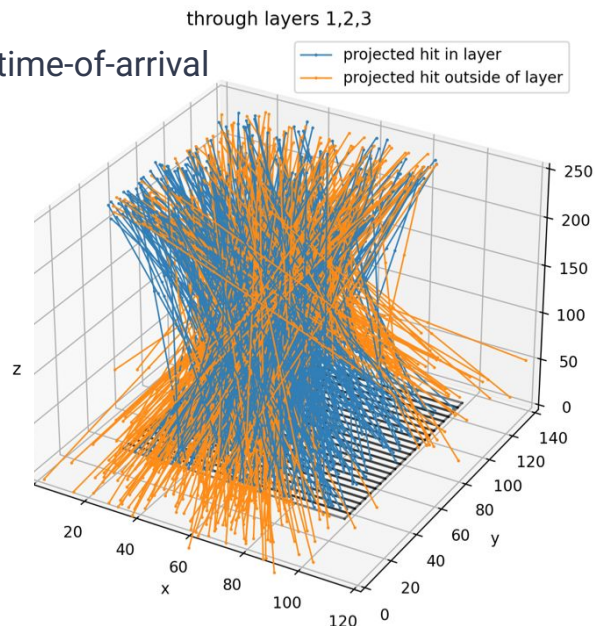
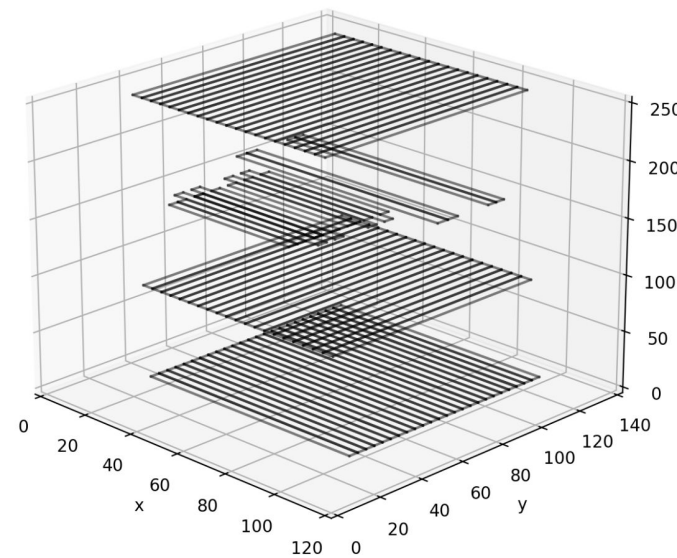


**MATHUSLA Conceptual Design Report:**

<https://arxiv.org/abs/2503.20893>

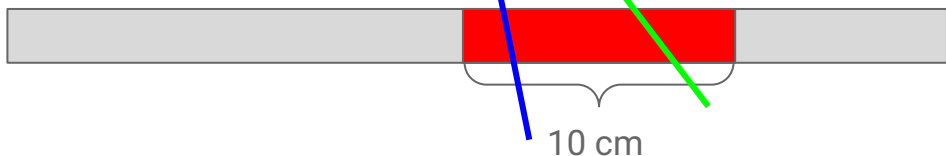
# UVic Teststand

- Tracking layer prototype built at UVic
- Routes fibers to SiPM array
- Detects cosmic muons
- Position reconstruction enabled by time-of-arrival differences



# Tracking

- DAQ timing resolution (0.5 ns) results in a 10cm position resolution
- What is the best fit line?

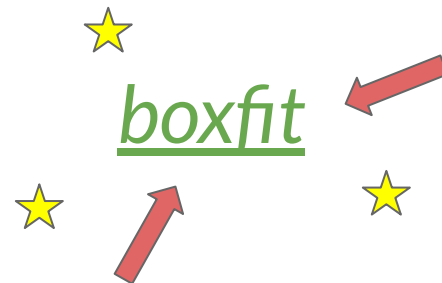


## The "establishment" way



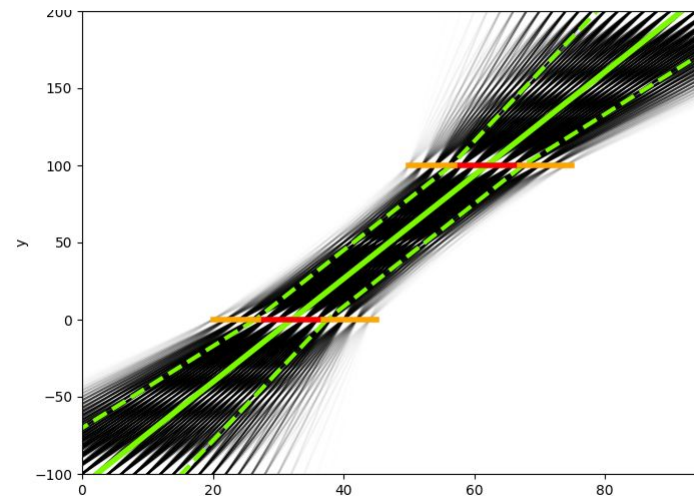
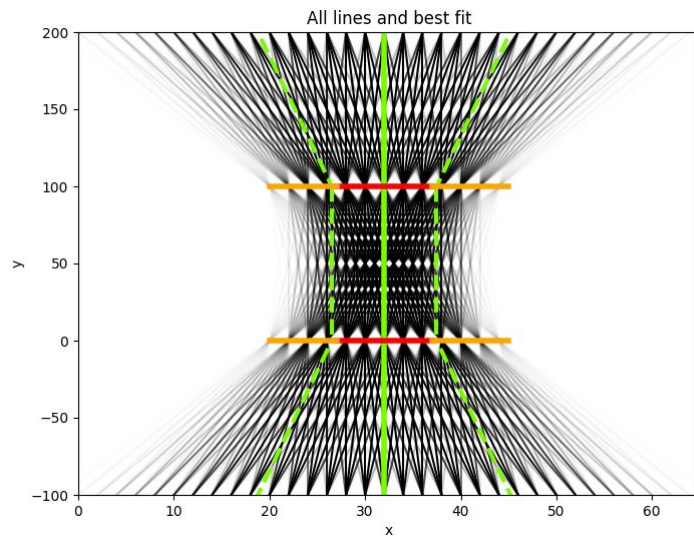
gaussian errors, chi2 minimization,  
singular value decomposition, etc..

OR



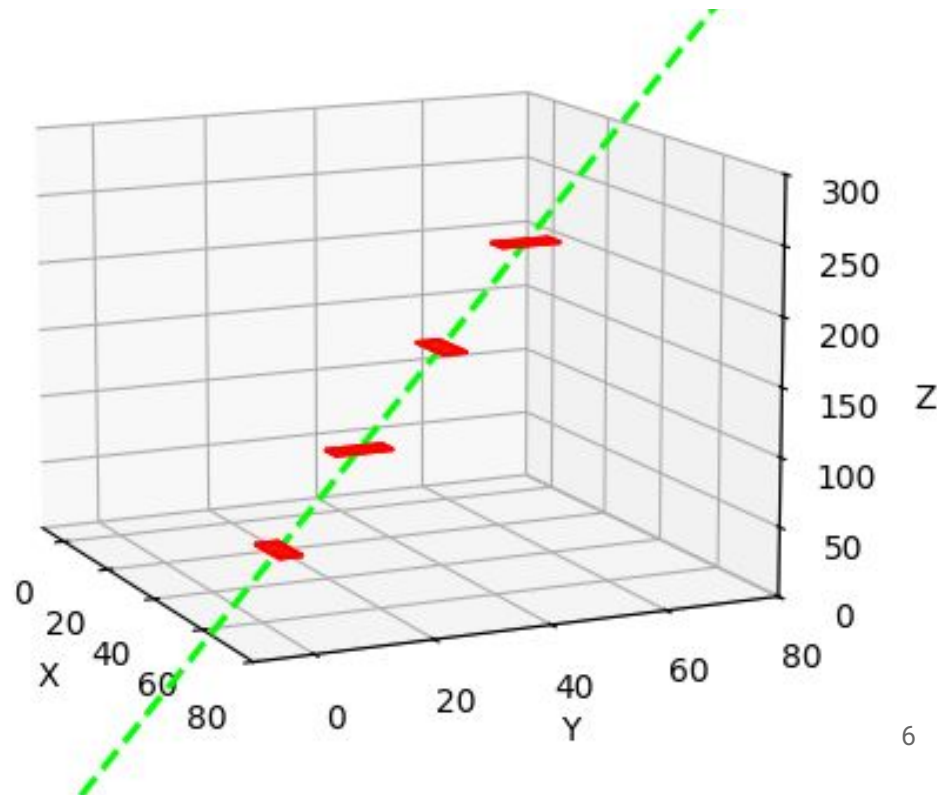
# boxfit

- Plan:
  - Segment the detector into smaller “boxes”
  - Assign a physically representative weight to each box
  - Draw every possible line
  - Take a weighted average



# 3D boxfit

- Extending to 3D adds a few more variables to track but the principle remains the same
- All hits are segmented in a similar way (as described for the 2D fit)
- For simplicity lines are only drawn between box segments in the top and bottom layers and then the overall weight calculated
  - $W_{\text{line}} = W_1 * W_2 * W_3 * W_4$
- Requires a control script to remove noise or iterate over a sub-selection of hits



# Performance

- So far this has been fun, but how well does boxfit do?
- Uvic Honours Project student, **Trystan Humphrey**, put together a MC generator and a number of fit algorithms to compare performance
- **Step 1: Generate tracks**
  - randomly draw lines through the detector, only keep ones that hit all 4 layers
- **Step 2: Apply detector resolution**
  - hit points are moved to the center of the volume determined by the resolution

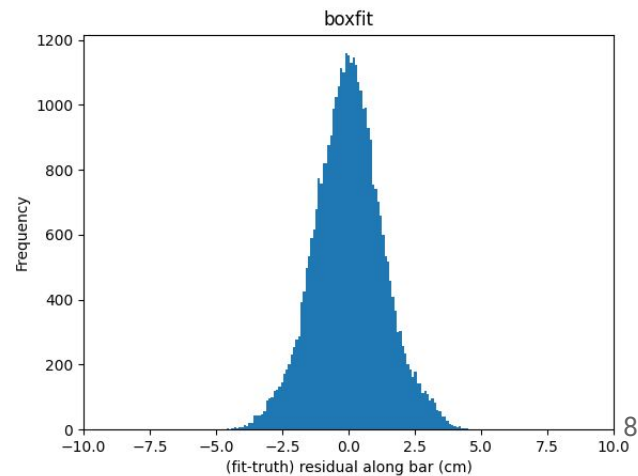
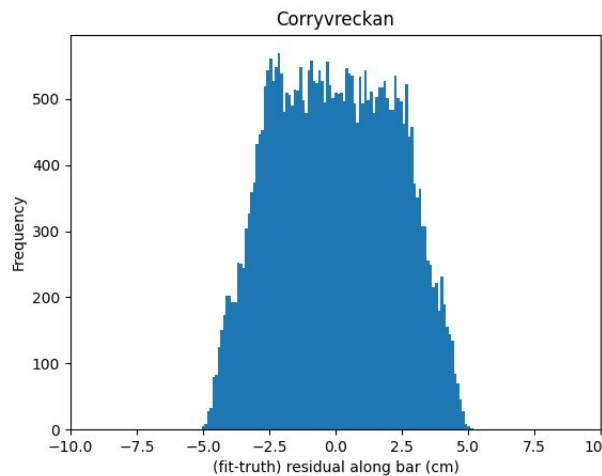
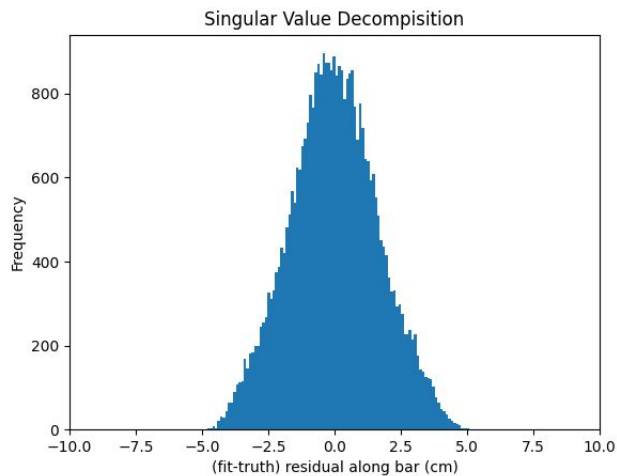


- **Step 3: Run fits on reconstructed hit points**
  - SVD, does not account for any uncertainties, only uses the hit locations
  - Corryvreckan, software package developed for test beam analysis, part of which is straight line tracking
    - <https://gitlab.cern.ch/corryvreckan/corryvreckan>,
  - boxfit
- **Step 4: Calculate fit-truth residuals**



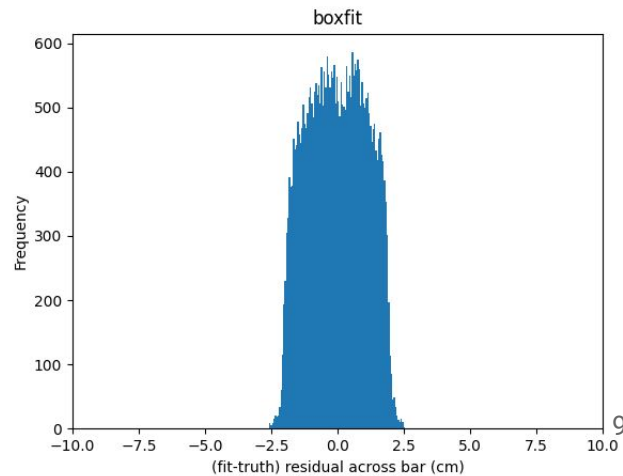
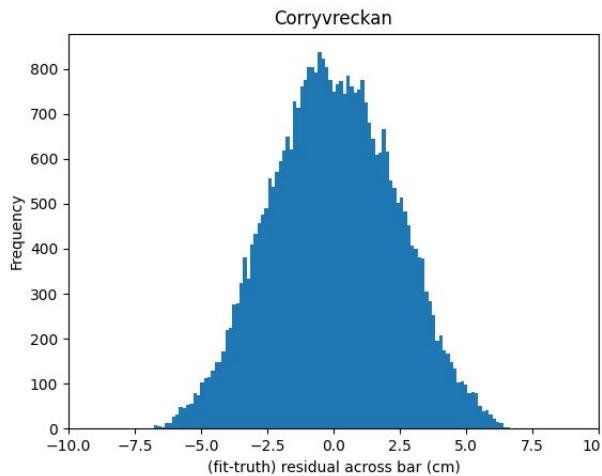
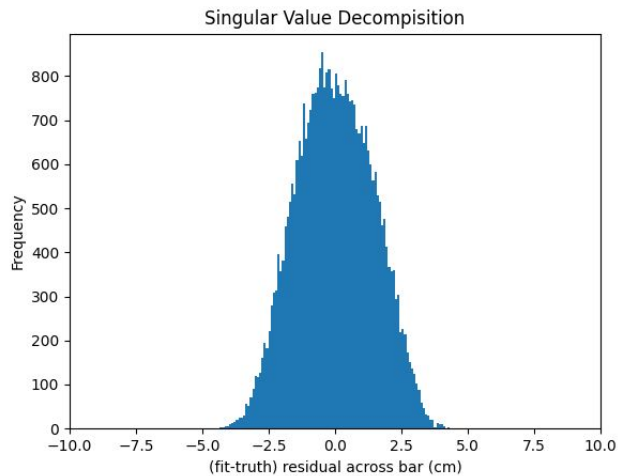
# Residuals ‘along’ the bars

- “Along” the bar is the time-resolved dimension ( $0.5\text{ns} = 10\text{cm}$ )
- SVD performs better than expected, likely the lack of errors doesn’t bias it towards the center
- Corryvreckan seems to be strongly biased towards the center-point, leading to a near-uniform residual
- boxfit looks slightly better than SVD, likely helped by a non-biasing knowledge of the uncertainties



# Residuals ‘across’ the bars

- “Across” the bar is physically constrained in the detector (4cm width)
- SVD performs very similarly to the ‘along’ dimension
- Corryvreckan seems to be worse than it should be. Currently either don’t understand either the result or a bug exists
- boxfit shows roughly uniform behaviour strongly limited to the 4cm width. Coarse box binning across the bars may be not giving enough flexibility in this dimension



# Try it yourself

**Guaranteed to:**

1. run in less than a Ms/event!
2. scale better than  $O(\text{TREE}(n))$ !
3. use less than a PB of memory!

- Available as a python package: `pip install boxfit`
- Only tested on MATHUSLA style geometry to data, likely needs some debugging for other geometries
  - contribute? 🙏
- Code available on github: <https://github.com/calebmiller/boxfit/>
  - Basic example available
  - default operation returns:
    - average x
    - average y
    - average z
    - average theta
    - average phi
    - 5x5 covariance matrix

# Conclusions

- Sometimes simple problems don't always have existing solutions
- boxfit was/is a fun side project that works well for our use-case but is in no way polished or optimized
  - maybe one day?
- Established tools always come with underlying assumptions that don't necessarily capture reality
- Future work on boxfit:
  - extend to arbitrary geometries. Currently very MATHUSLA biased in implementation
  - optimize ray drawing. I suspect an order of magnitude improvement is easy
  - visualization of 3D 1-sigma surface
  - return a p-value associated with a proposed source vertex

# Thank You!