Radboud University - Nikhef

High-Energy Physics department - ATLAS team

Novel Approaches for ML-Assisted Particle Track Reconstruction

dr. ir. Uraz Odyurt

2023-10-12

Connecting the Dots Workshop

The collaboration

Radboud University

- Institute of Computing and Information Sciences
- High-Energy Physics

Nikhef

- ATLAS team

University of Twente, University of Amsterdam

SURF

Uraz Odyurt
Sascha Caron
Ana-Lucia Varbanescu
Nadezhda Dobreva
Zef Wolffs
Roel Aaij
Yue Zhao

University of Valencia

- Institute of Corpuscular Physics
- Intelligent Data Analysis Laboratory

Valencian Graduate School and Research Network of Artificial Intelligence

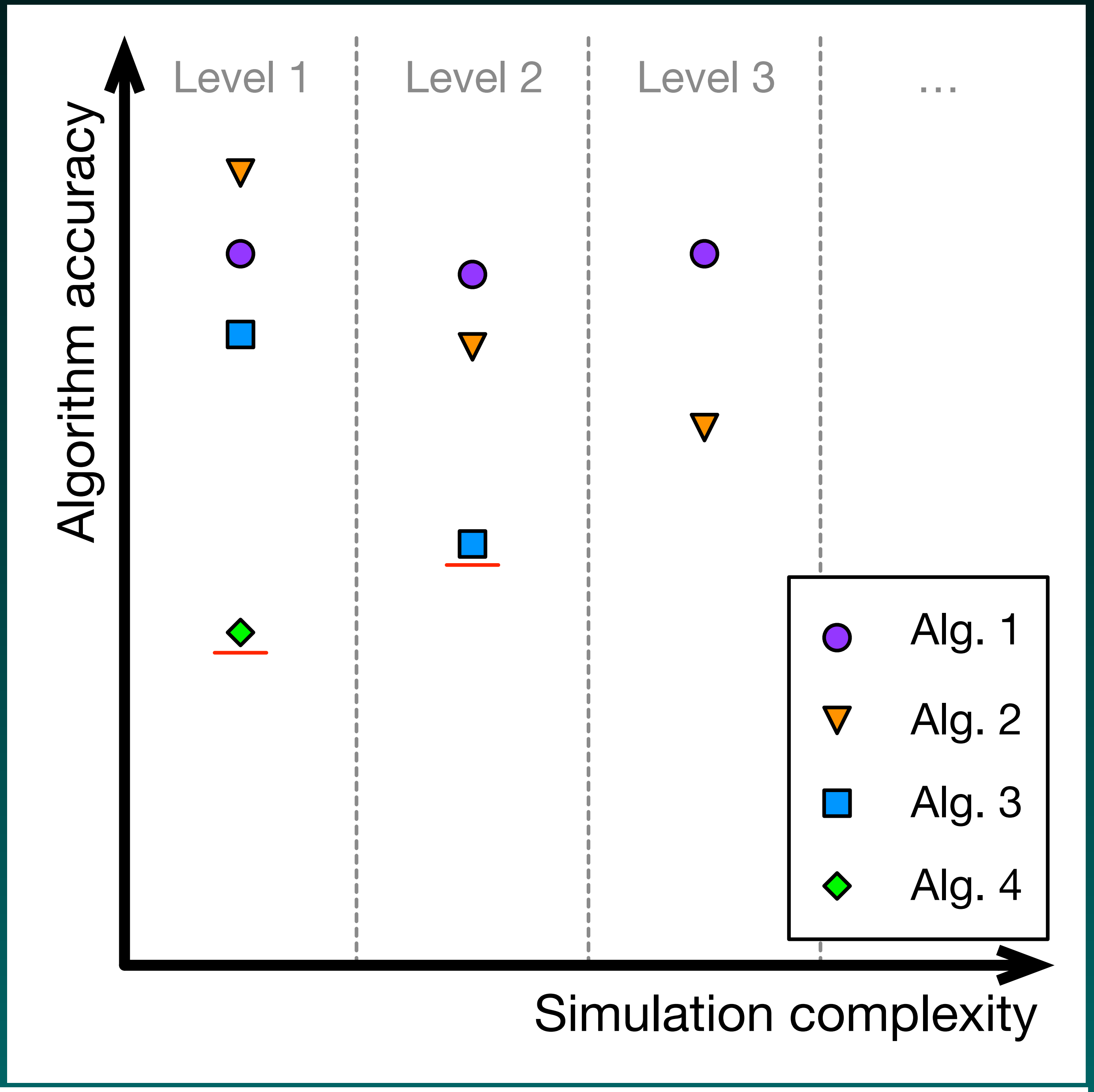
Antonio Ferrer-Sánchez
José D. Martín-Guerrero
Roberto Ruiz de Austri
José Salt

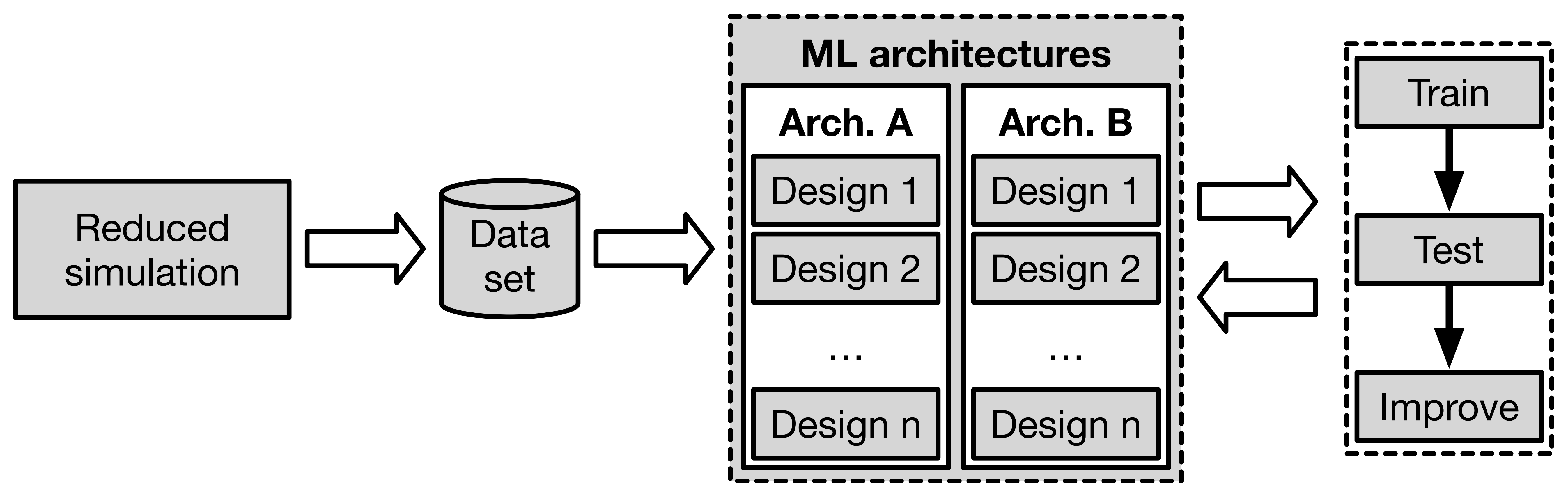
The idea: High-level view

- We want to design ML-assisted solutions

- We want to design and train best ML model(s) for tracking
  => Eliminate designer bias
  => Better corner case response
  => Detector agnostic

- What is the best way to do it?
  => Reduced simulations
  => Synthetic data
  => And more …





Part 1:
Reduced Simulations

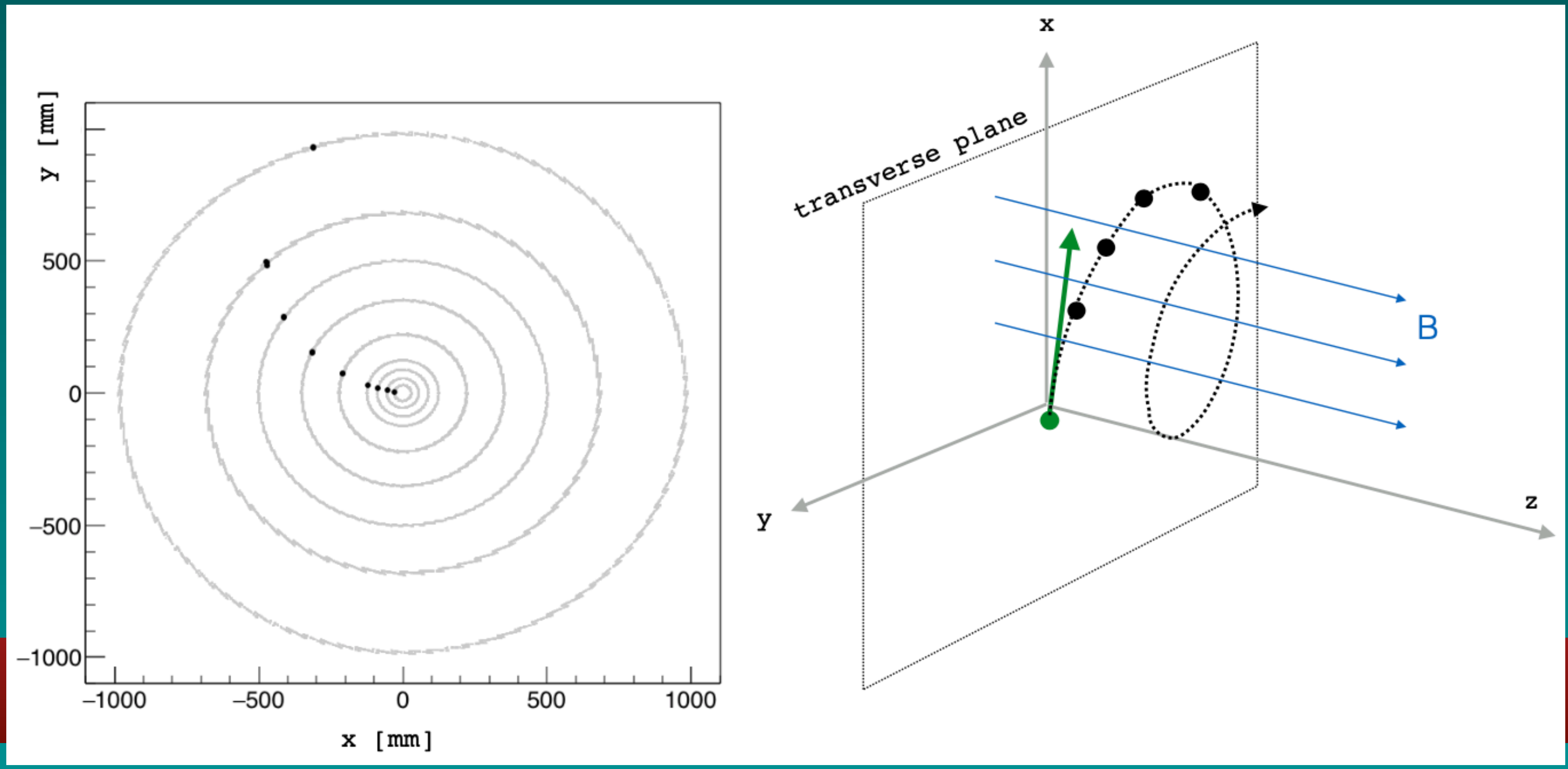
Motivation

- Scientific use-cases
  => Data-intensive: Processing capability is limited, **data deluge**
  => **Time-critical**: Low latency is desired/required

- ML-assisted solutions could be an answer
  => Emphasis on “assisted” …
  => Enables higher data capacity
  => Enables the use of **specialised hardware**
  (GPUs, TPUs, FPGAs, Neuromorphic HW?)

Use-case: The task of particle tracking

- Every **event** releases particles
  => Particles go through the sensors of detectors, e.g., ATLAS
- Every interaction with each sensor is recorded as a **hit**
  => Detector **geometry**
- The real apparatus is quite complex
  => Noise, deviation from the absolute origin
  => Not following an exact arc of helix
  => Secondary particles …
  => Missed detections …



ML-model design approaches

- Primarily **ad hoc** efforts
  => Is it the best model? Most robust?
  => Is it the best data-processing approach?
  => Is it the best data representation? Enough corner case data?

- Automated and multi-objective **Design-Space Exploration (DSE)**
  => Hyperparameter search
  => Neural-Architecture Search (NAS)
  => Expensive …

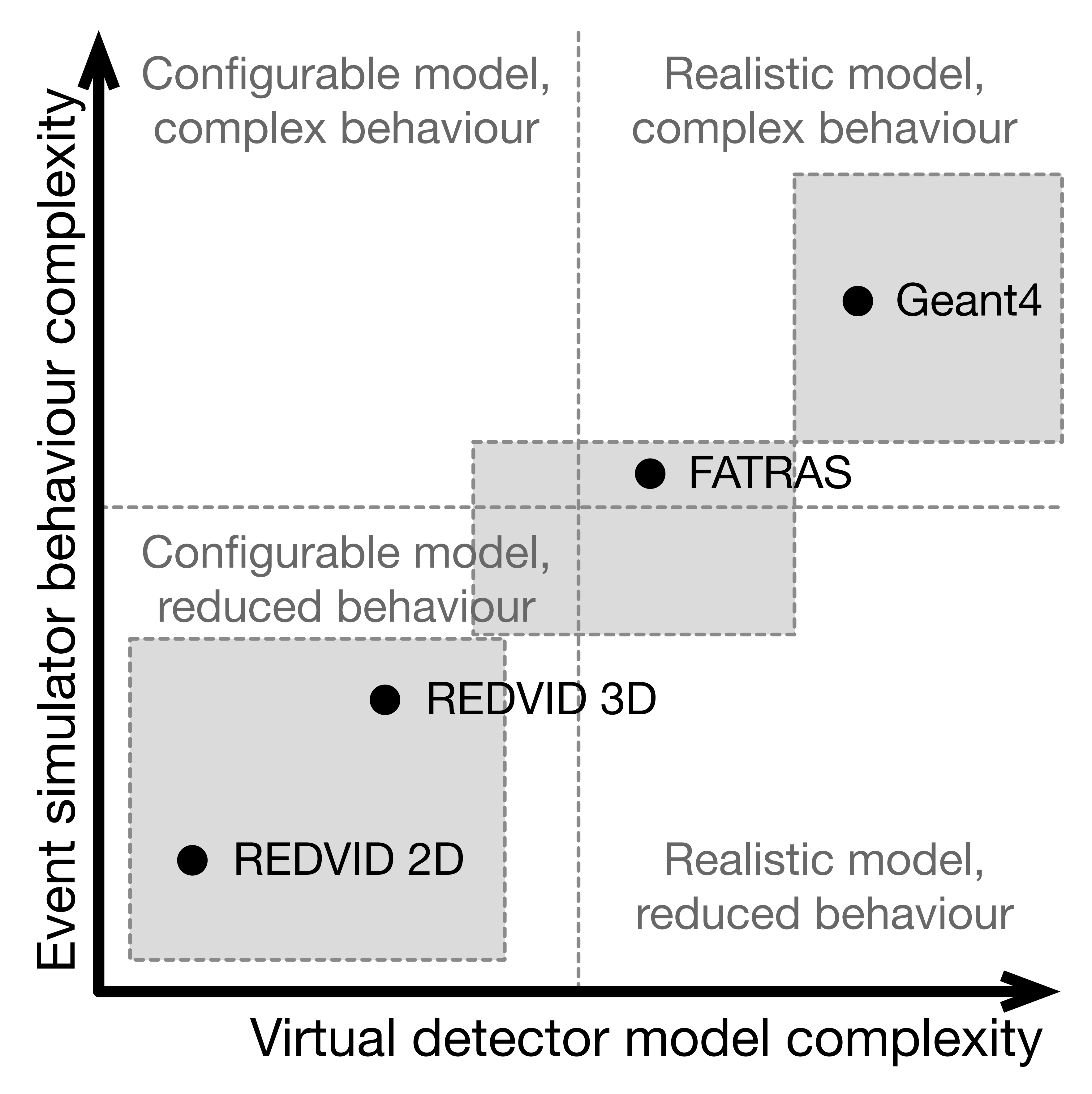
- Very important to minimise => DSE **time** and **computational cost**

DSE: Systematic analysis and pruning of unwanted design points based on parameters of interest.

Different simulations

- Depending on the
  => Detector model complexity +
  => Simulator behaviour complexity

- Two types could be considered
  => Parametric/(re)configurable simulations
  REDVID
  => Physics-accurate simulations
  Geant4, FATRAS, ATLFAST



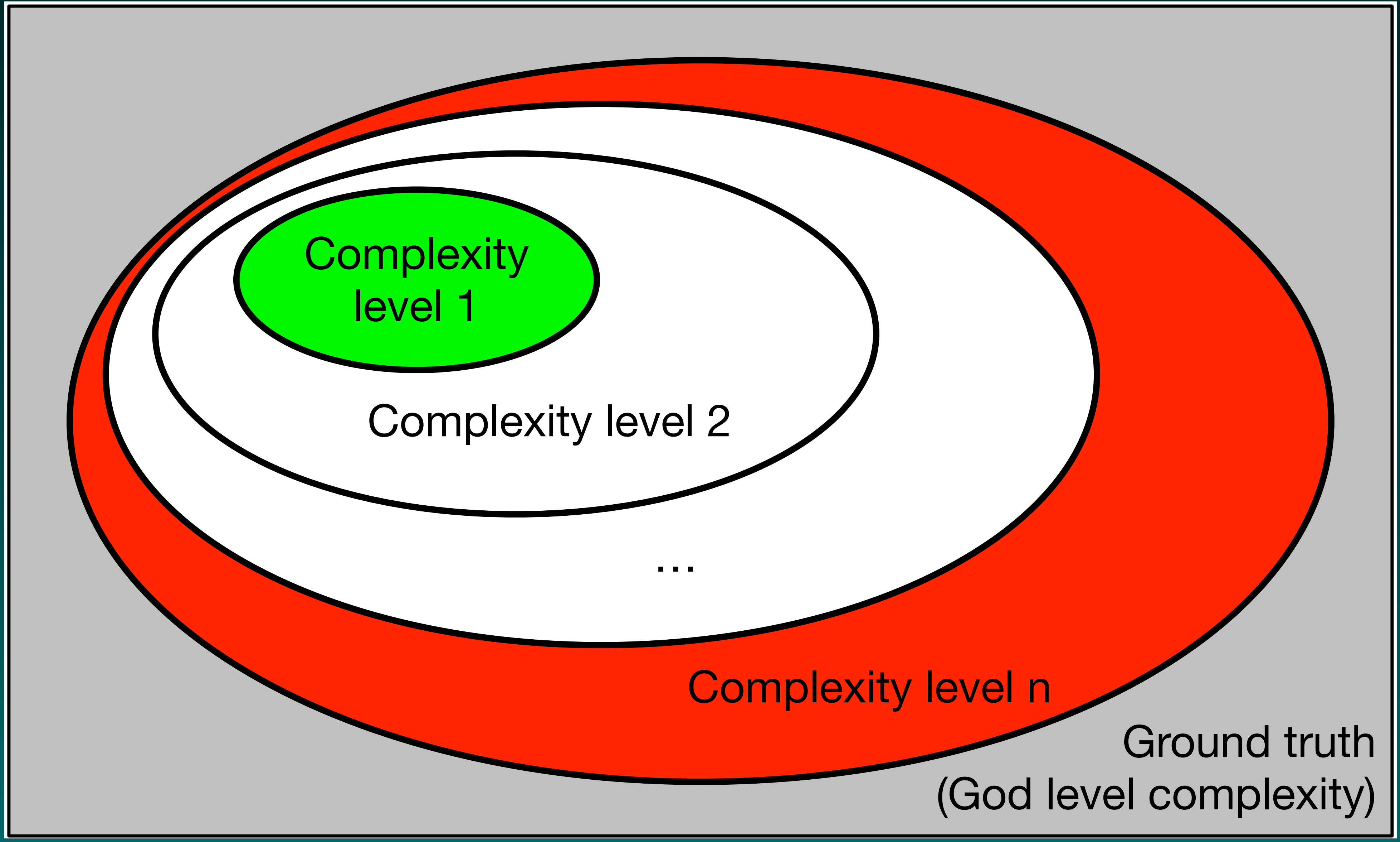
Simulation: **Model of the system** (characteristics) + **Simulator** for events/actions/environment (behaviour)

Layered approach to complexity

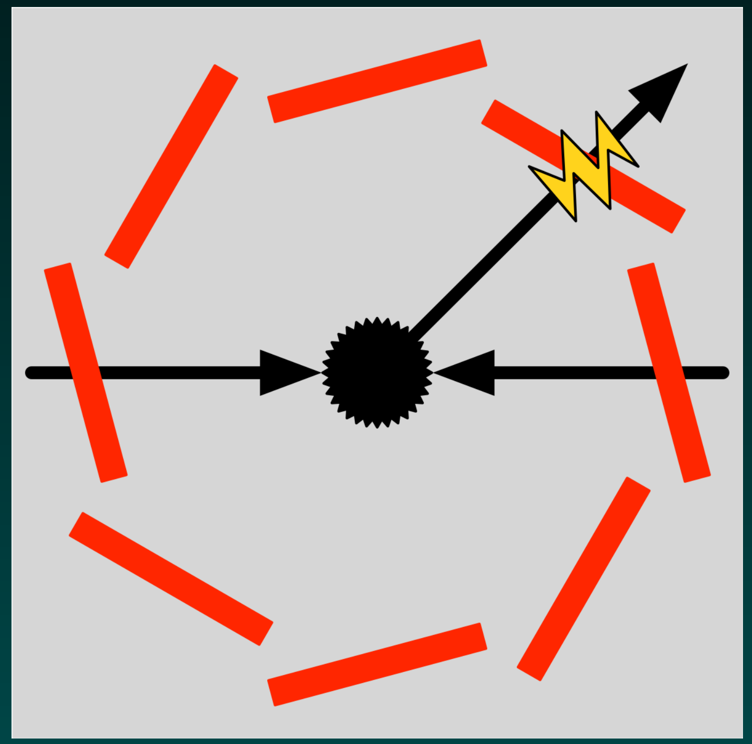
- For a problem that is too complex:
  => The likelihood of finding a solution is lower
  => The time it takes is longer (?)
  => The likelihood of an ad hoc solution is higher

- There are important **secondary goals**
  => Computational performance

- Different **complexity levels** from the **ground truth**
  => Better understanding of the problem
  => Speeding up the automated search



Reduced simulations for HEP



- **REDuced VIrtual Detector (REDVID)**
  => Fully (re)configurable and modular
  => Reduced-Order Models (ROM) for detectors
  => Event simulator with reduced complexity behaviour
  => Generates synthetic data -> Tracks and associated hits

- *“Reduced Simulations for High-Energy Physics, a Middle Ground for Data-Driven Physics Research”*
  => ACM/SIGAPP Symposium On Applied Computing (SAC)
  => Pre-print on arXiv [arXiv:2309.03780]

- REDVID code and reference data sets available online
  => https://VirtualDetector.com/redvid

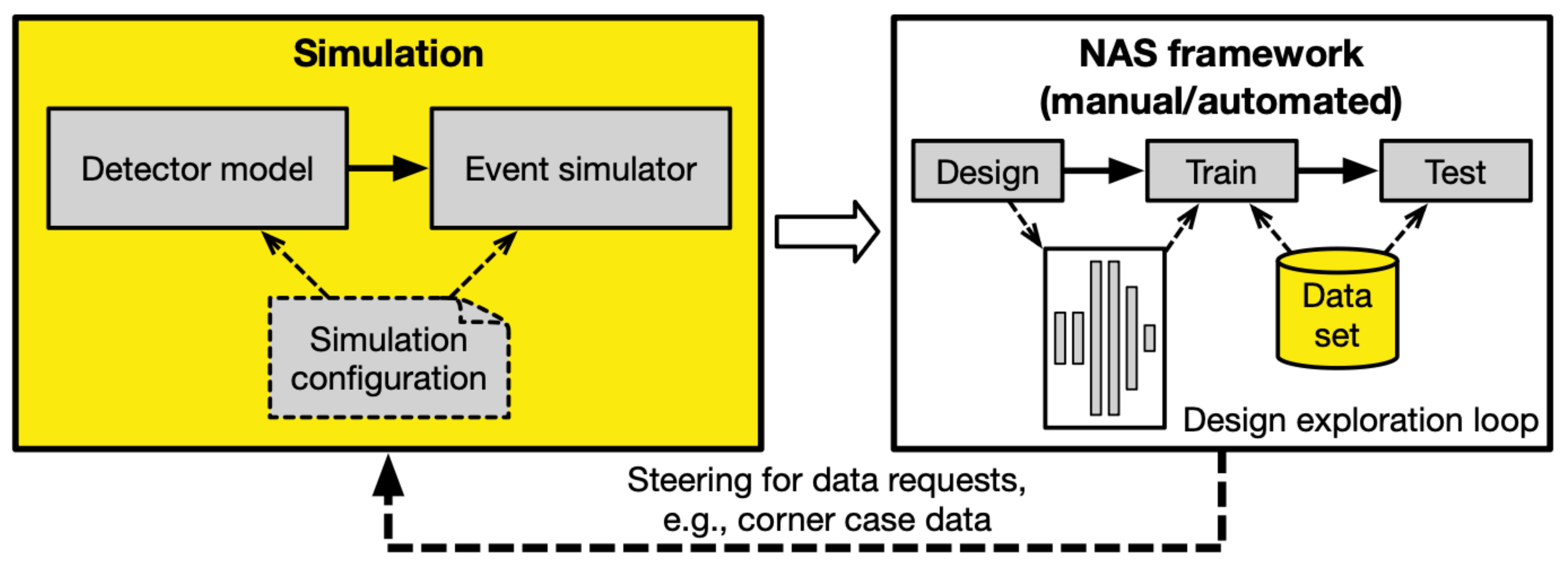
An enabler for ML model design and search

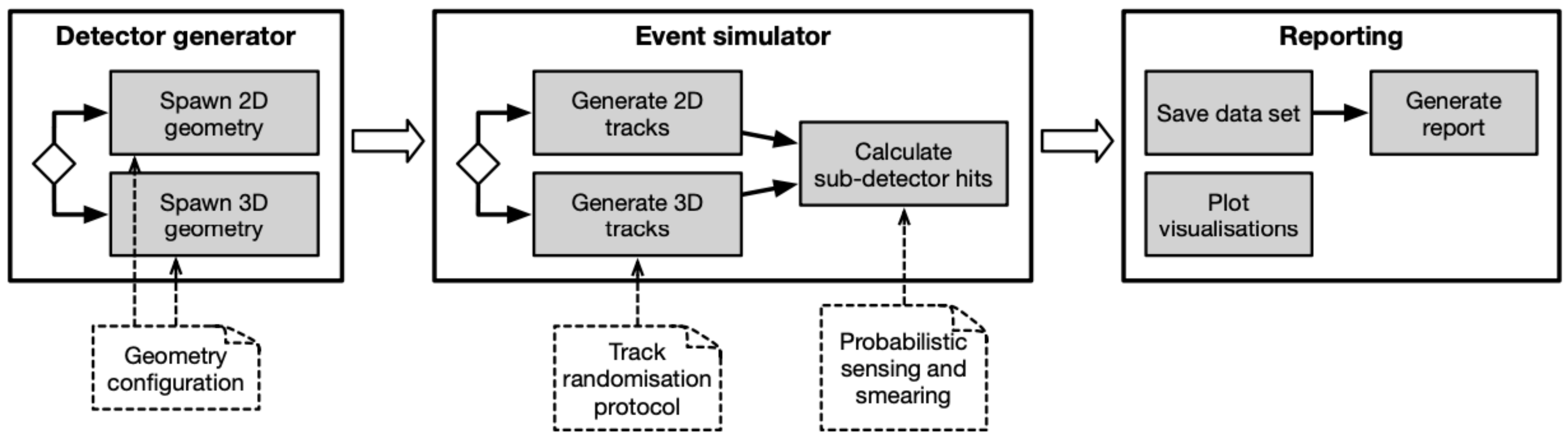
- Drastically **simplifies the problem** for designers

- Runs fast, can be embedded in loops (**simulation-in-the-loop**)
  => Other tools run fast too …

- Fully parametric and (re)configurable
  => Can be **steered** with new configuration
  => Can be used for **corner case** data generation
  => Can be a part of automated exploration/NAS loops

- Track randomisation protocols as a knob to steer simulator behaviour



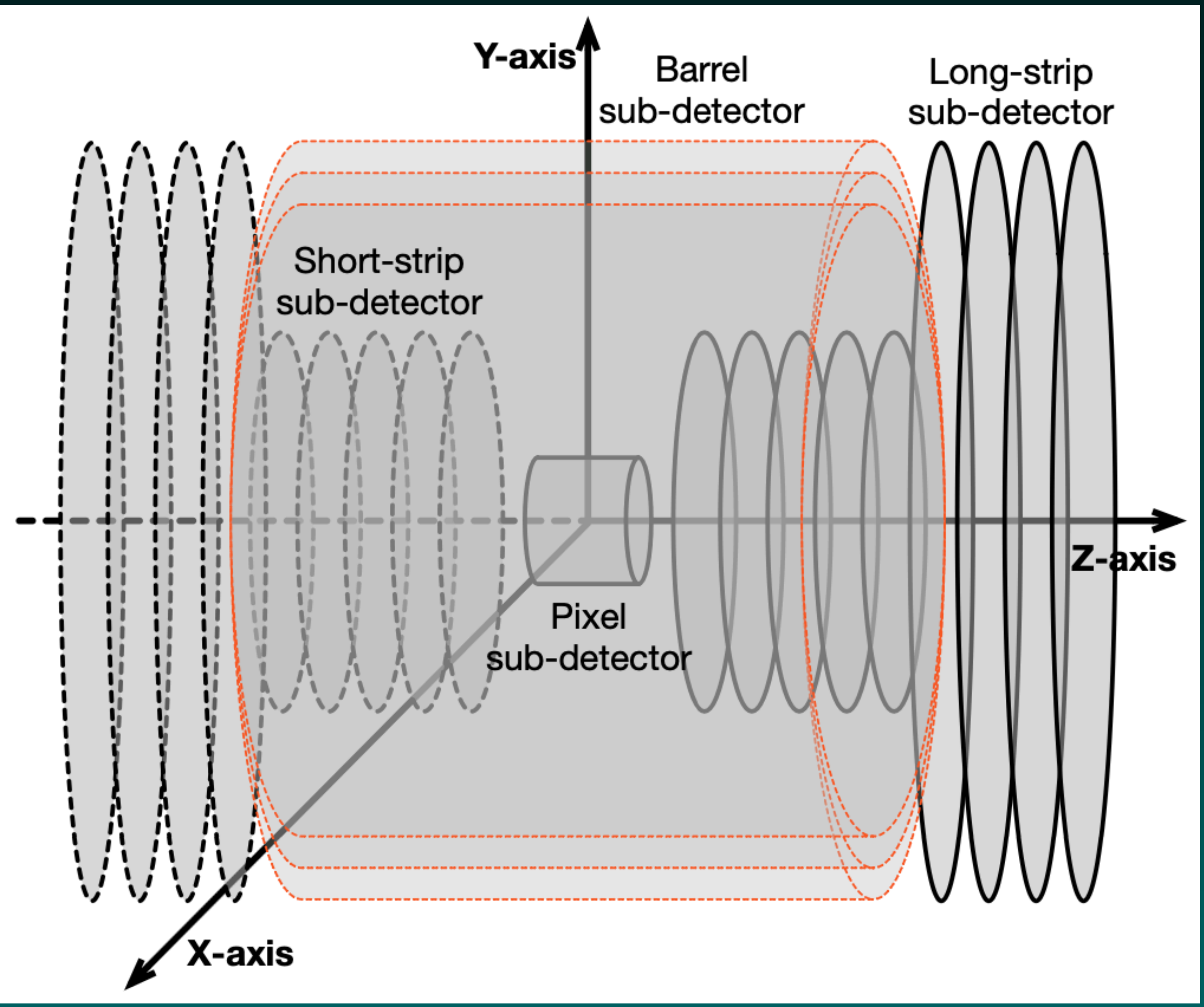


Available configuration

- Detector geometry, sub-detectors
  => Sizes, counts, placements, thickness

- Track randomisation:
  => Linear, helical uniform, helical expanding, origin smearing, randomisation protocol
  => Main source of **non-deterministic behaviour**

- Hit calculation:
  => Hit sensing probability, hit smearing



Performance metrics

- System resource utilisation
  => Execution time
  => CPU-time
  => Internal probes: Granular information per functionality
  => Fairly good results for a Python tool

- Parallelisation is trivial => Just distribute events over threads

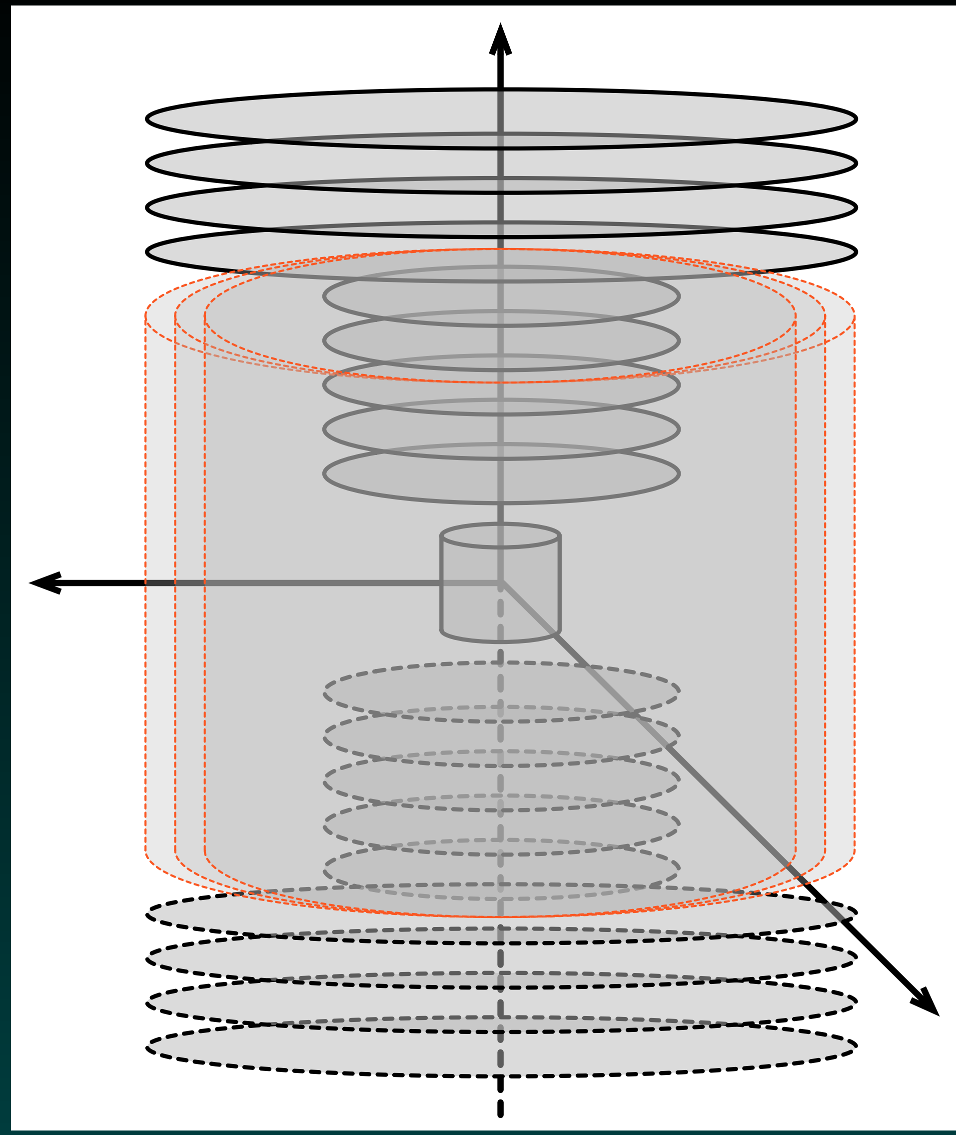
- **Scales linearly**
  => Very desirable!

Table 1: REDVID execution CPU-time cost for simulations of 1 000 events with various track concentrations. All values are in milliseconds. Full simulation times are provided in minutes as well. Even though REDVID is developed in Python, computational cost figures indicate efficiency for frequent executions.

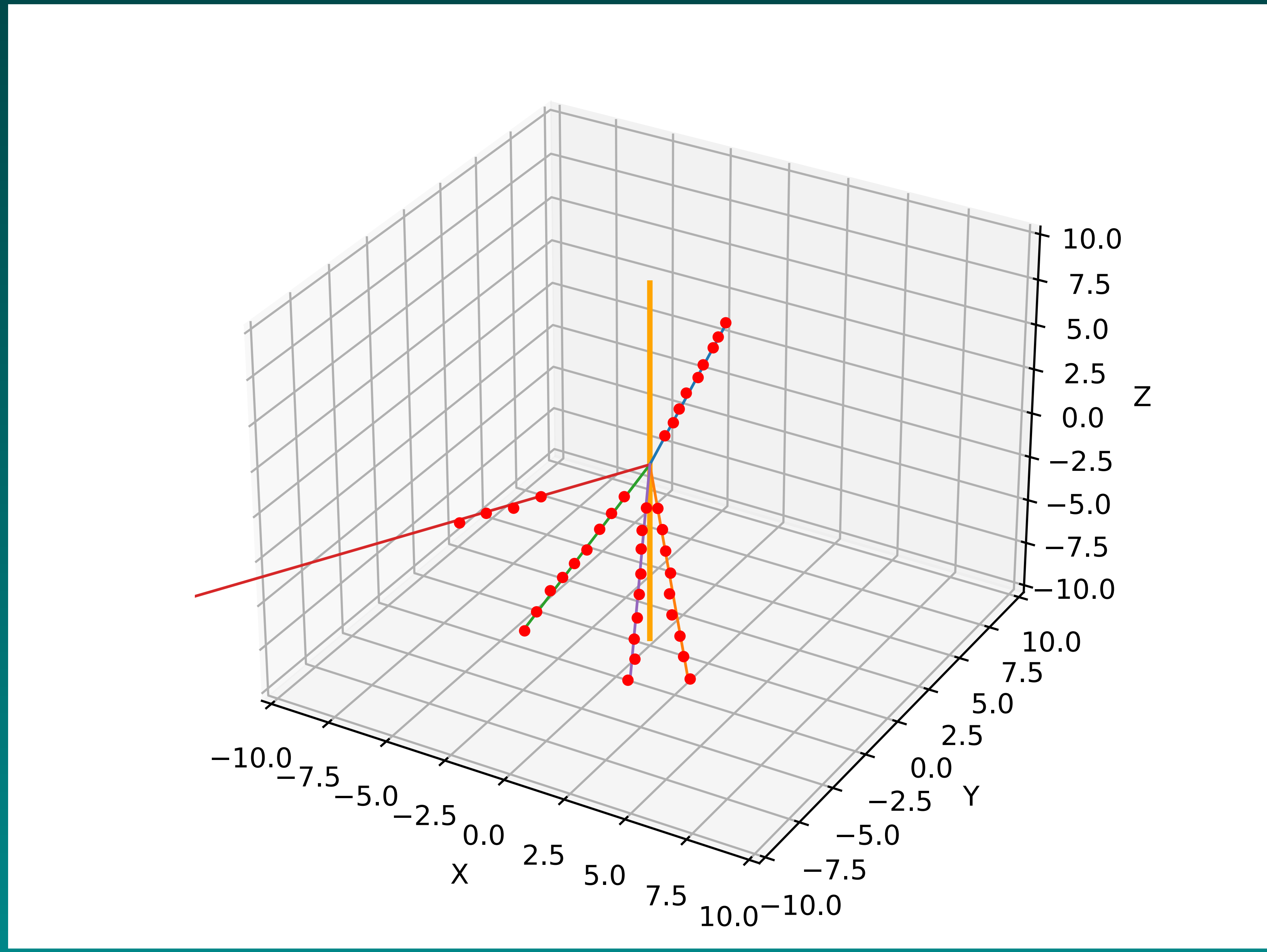
Recipe for 1 000 events	3D detector spawning	Track randomisation per event - Mean	Hit discovery per event - Mean	Full simulation of 1 000 events (minutes)
1 track per event	0.025	0.043	1.463	2 731.17 (0.05)
10 tracks per event	0.025	0.083	13.429	15 418.589 (0.26)
100 tracks per event	0.025	0.465	129.864	137 623.954 (2.29)
1 000 tracks per event	0.025	4.582	1 285.989	1 353 396.641 (22.56)
10 000 tracks per event	0.024	43.765	12 496.208	13 591 628.526 (226.53)

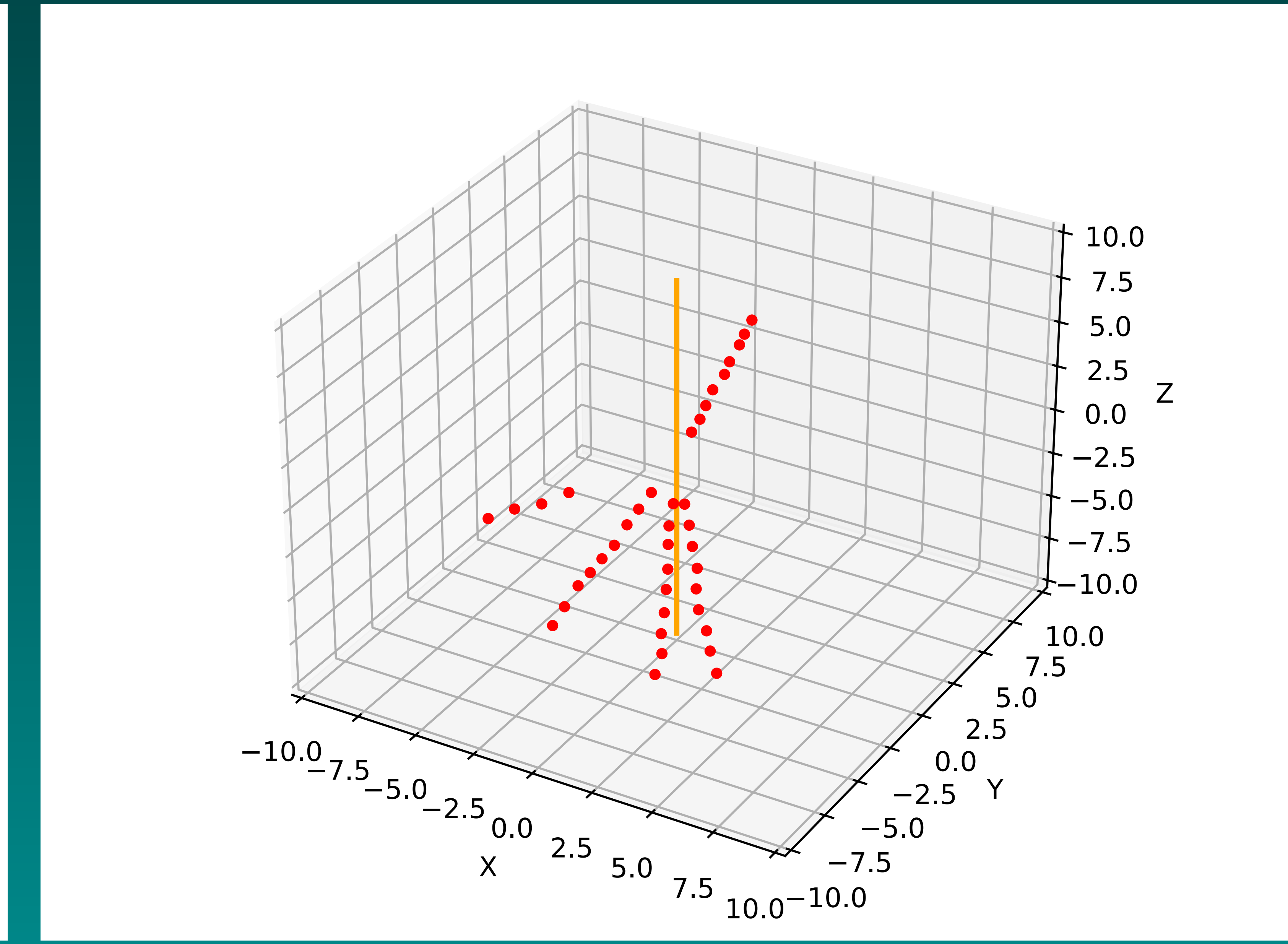
A few examples

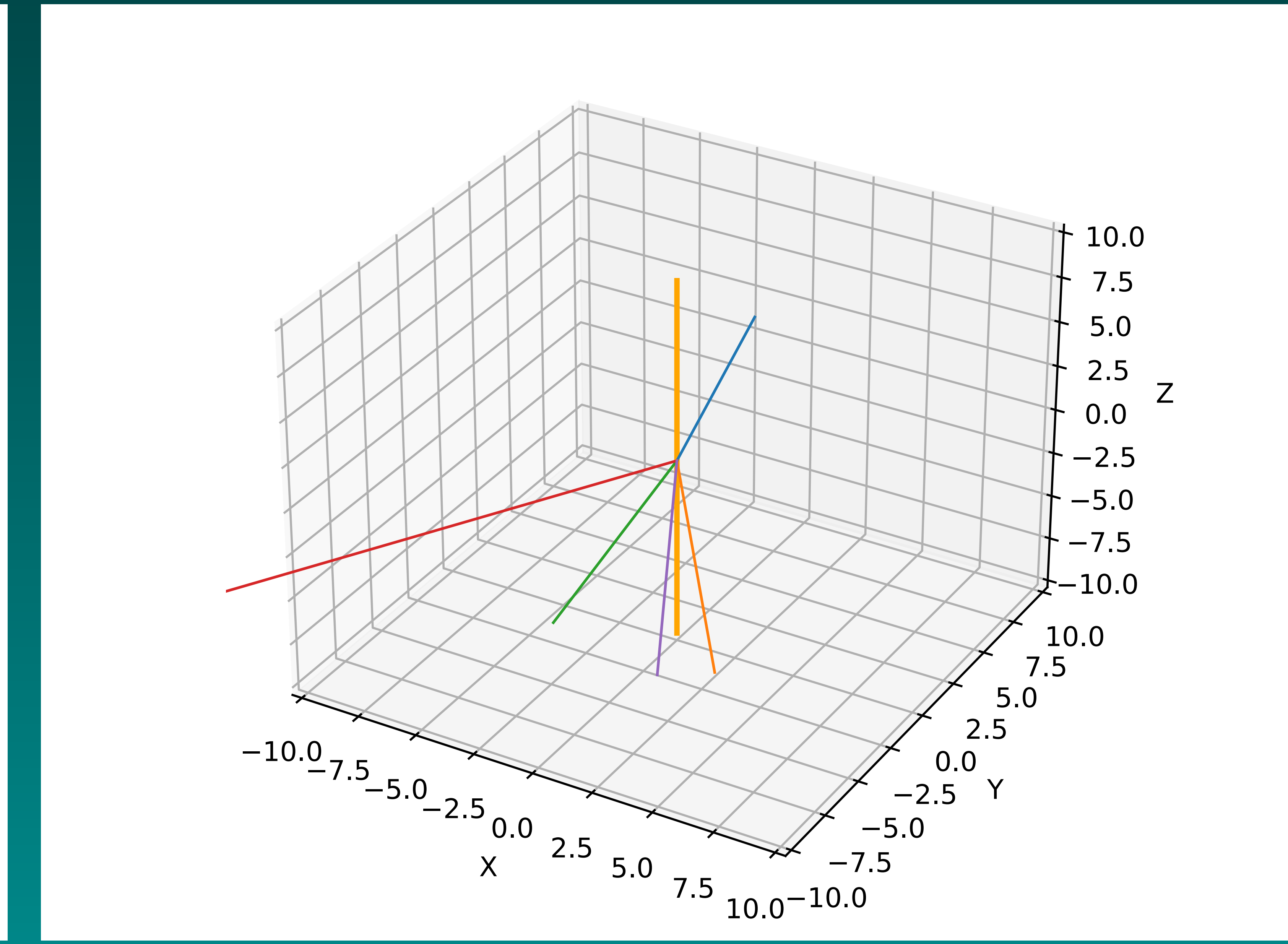
Track propagation



5 tracks, linear

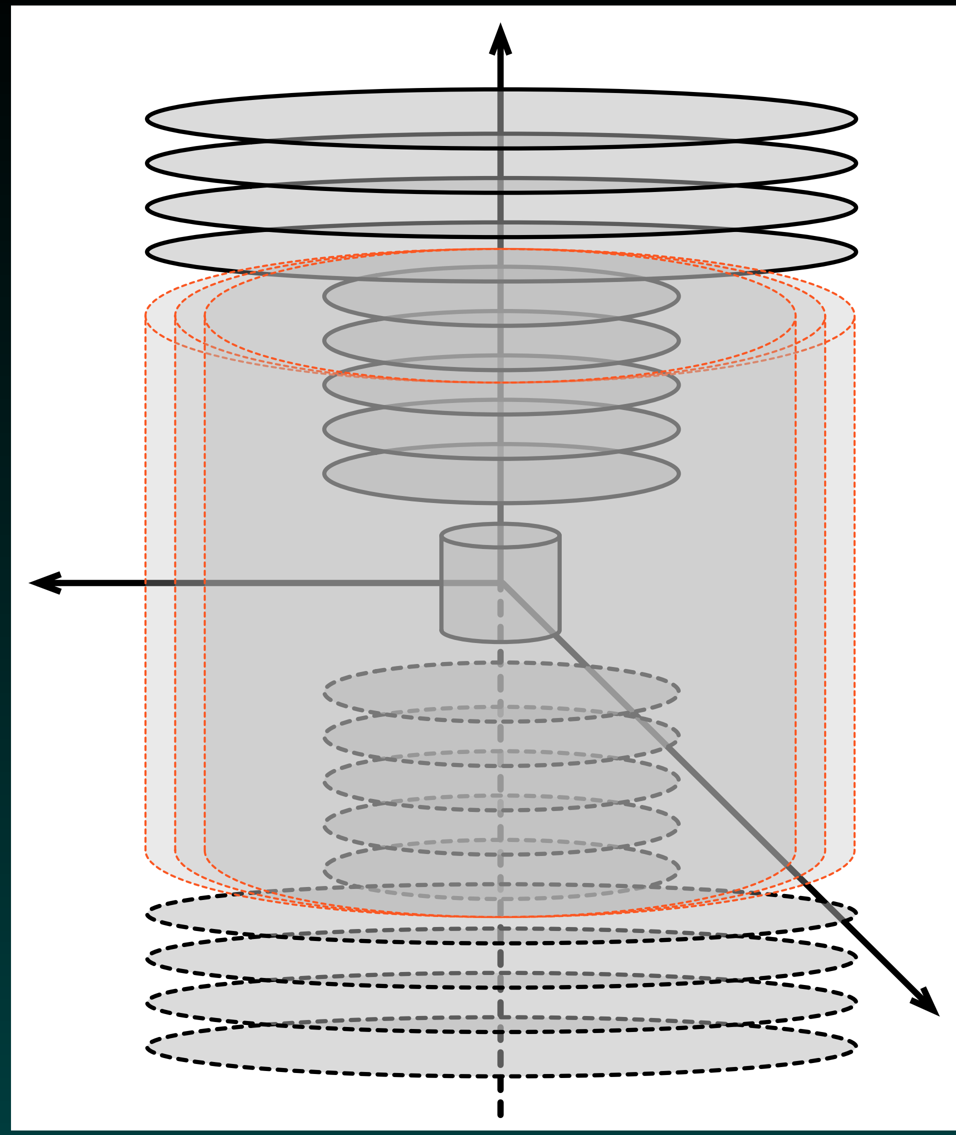




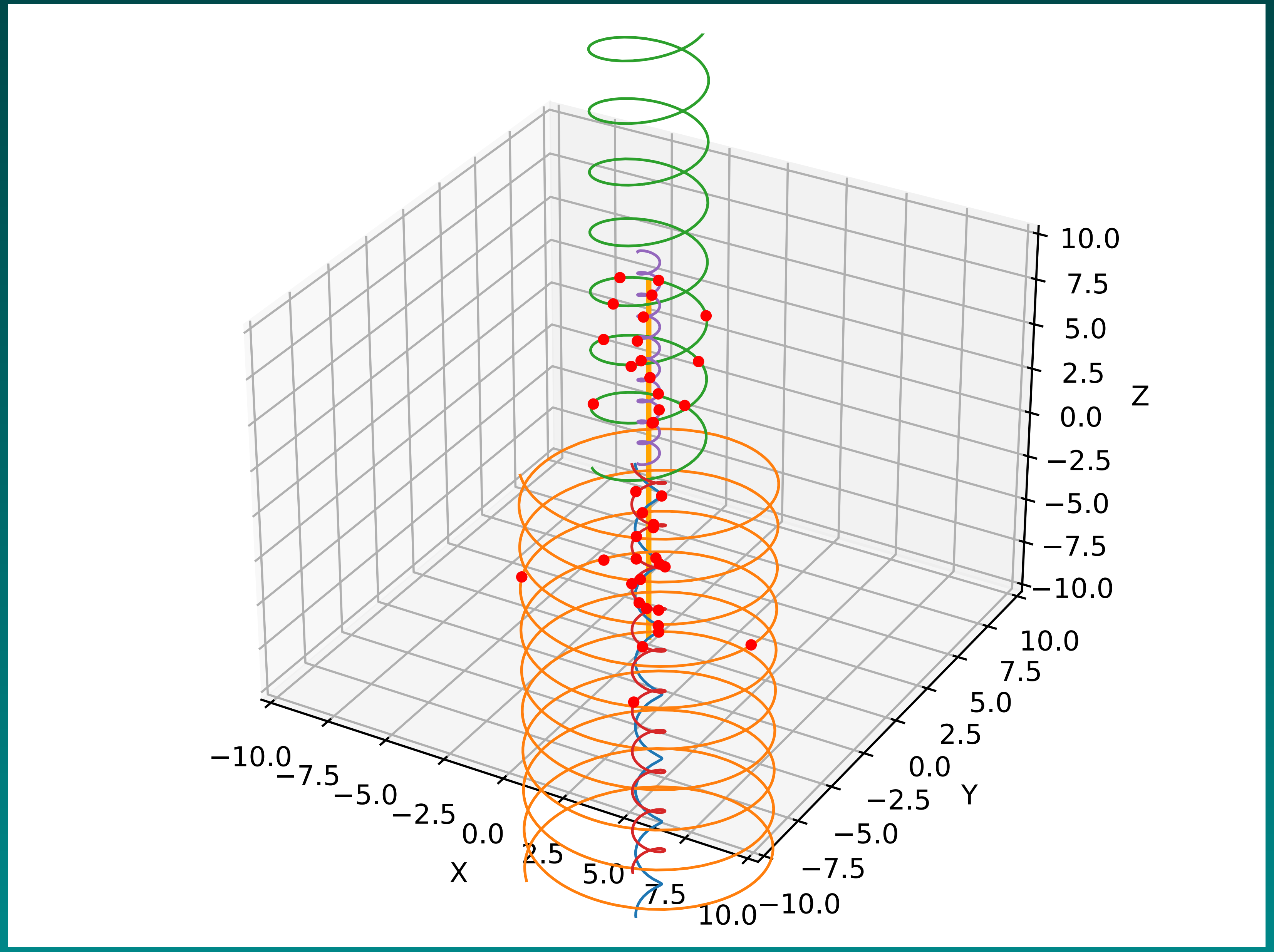


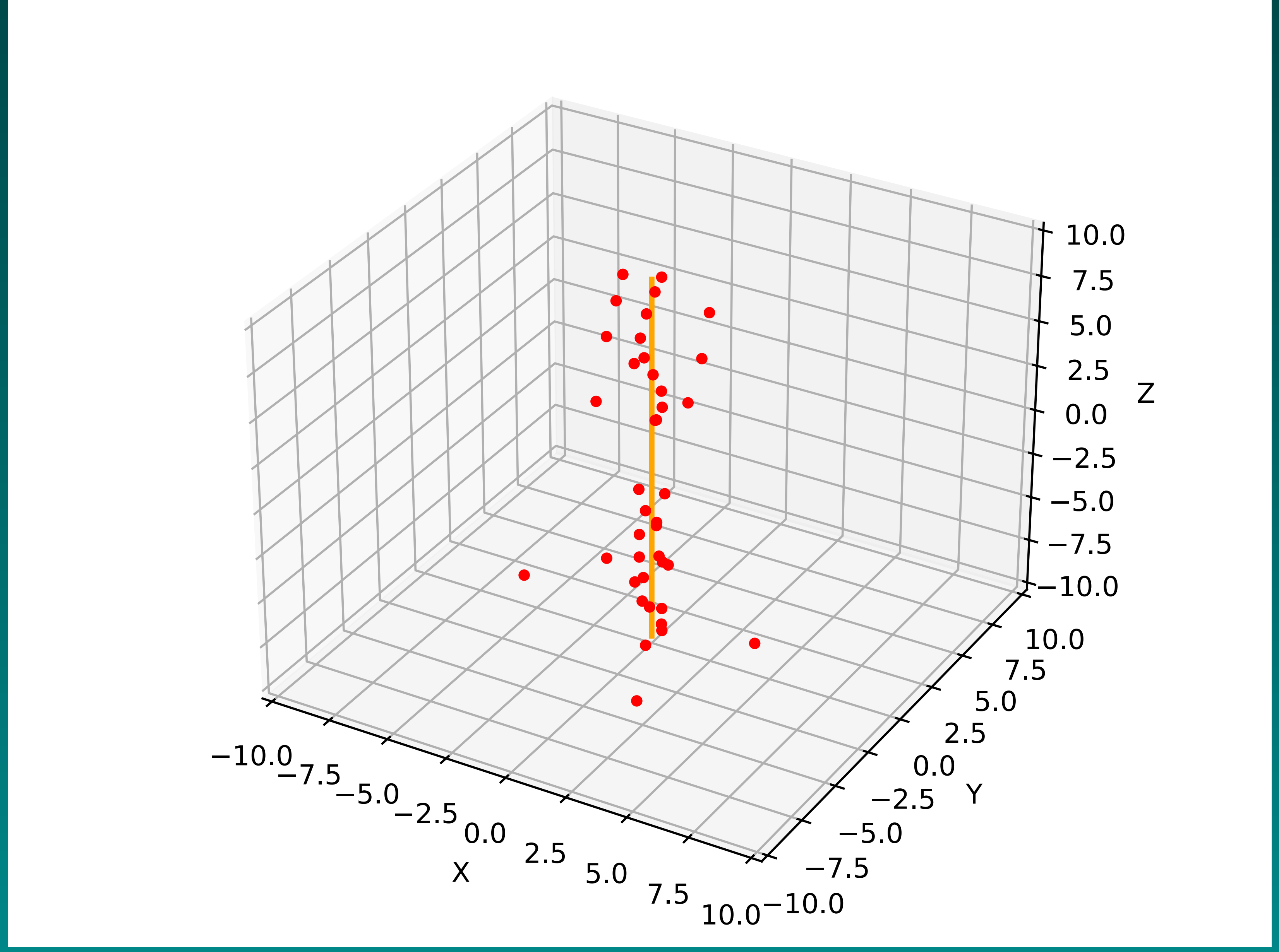
A few examples

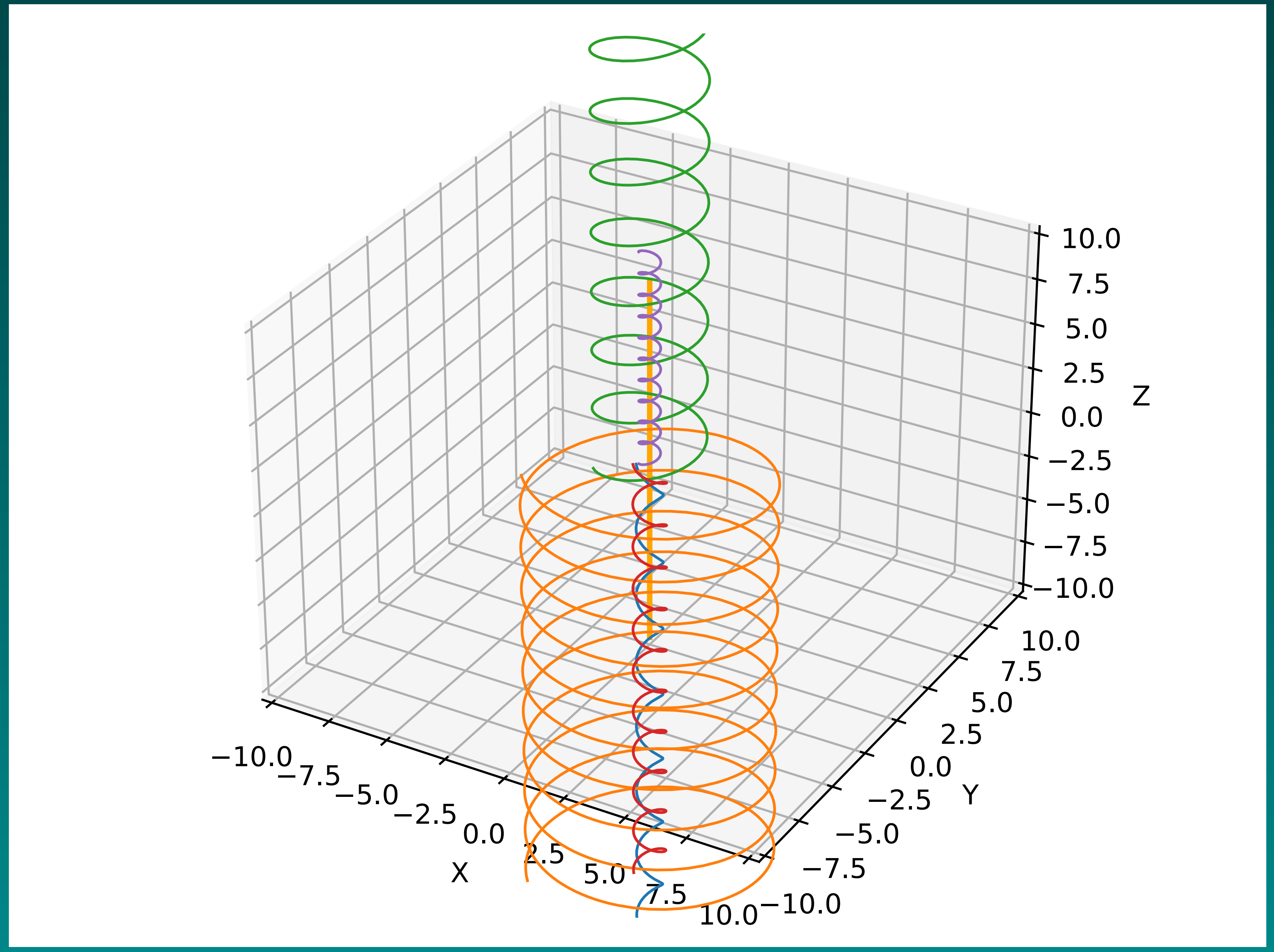
Track propagation



5 tracks, helical uniform

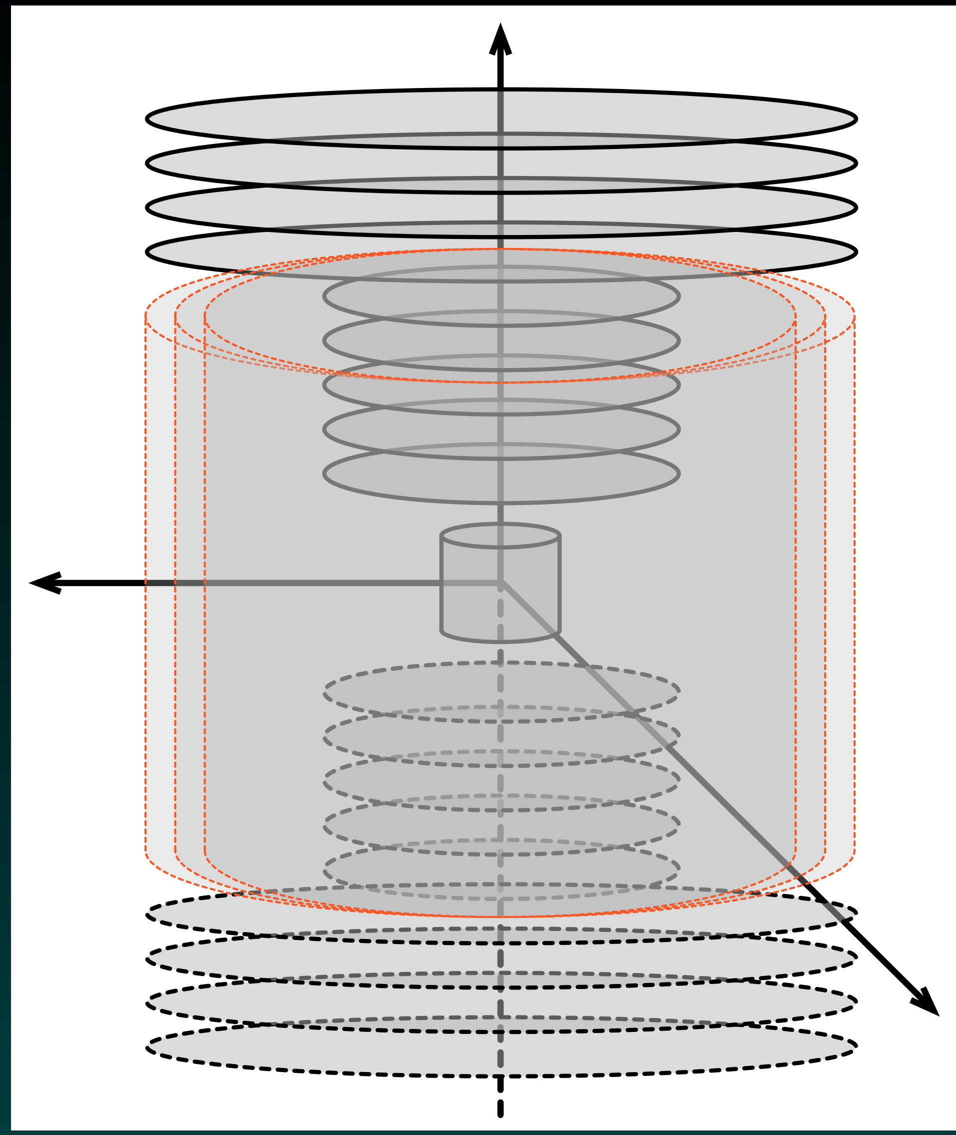




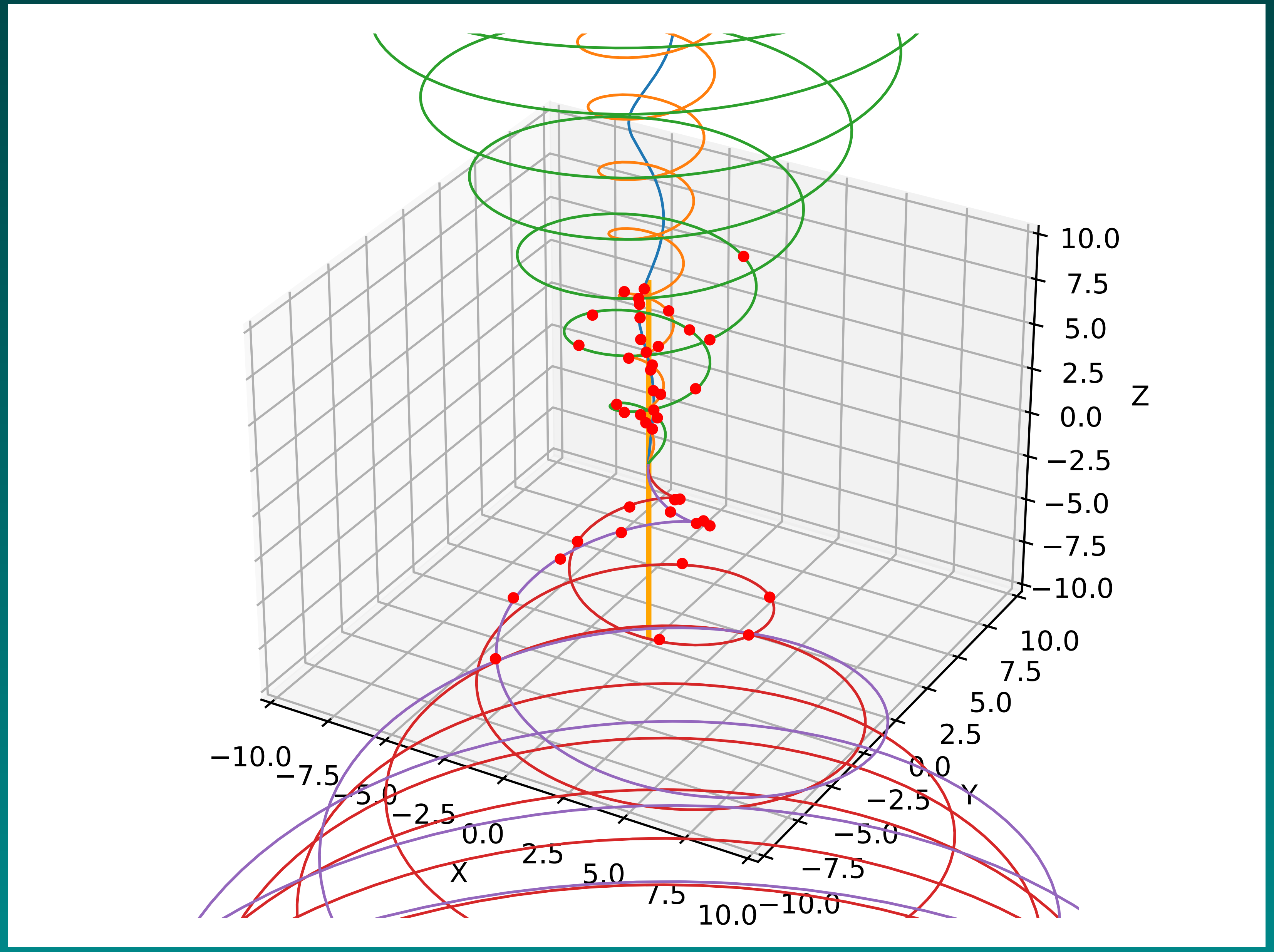


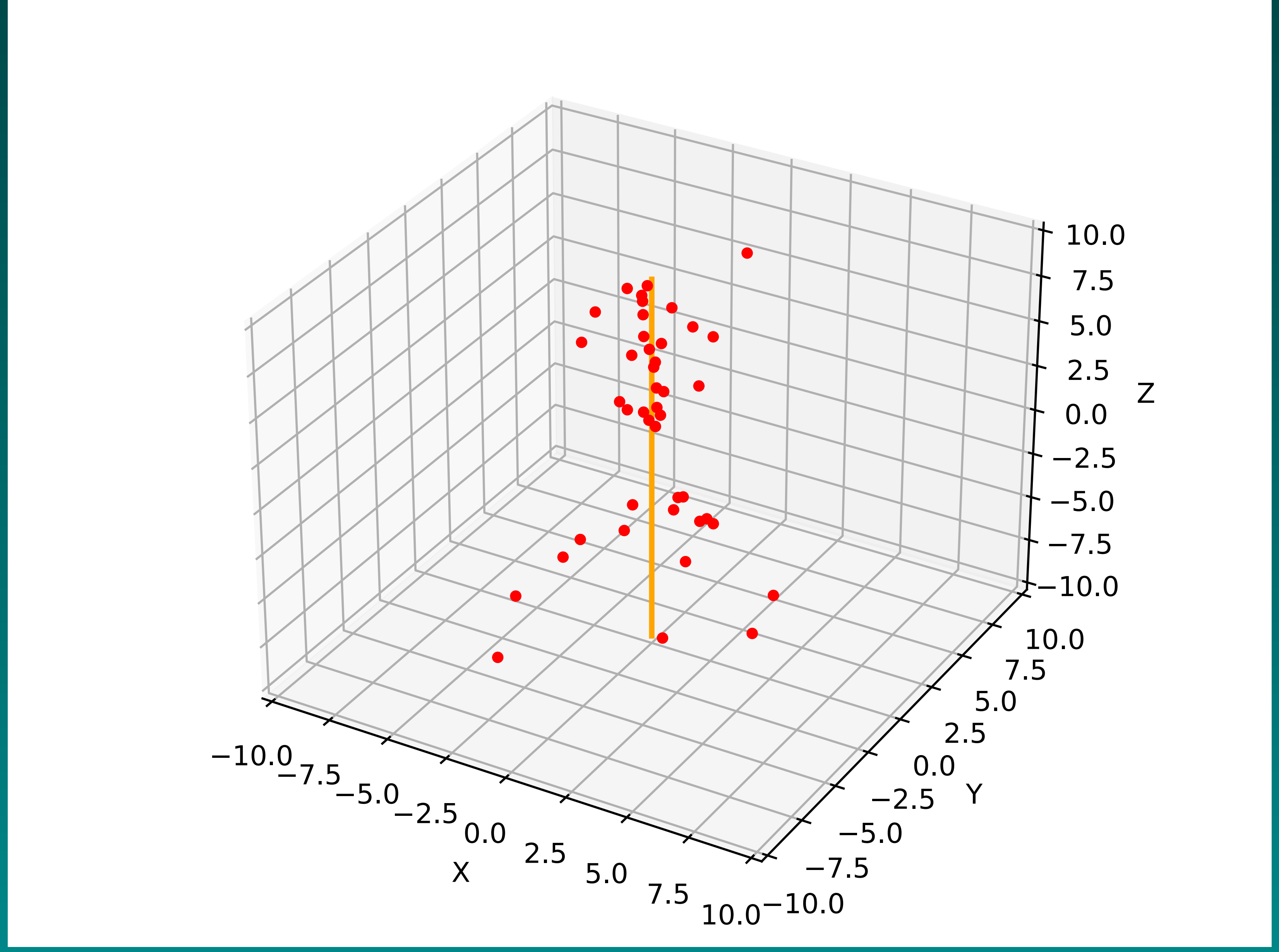
A few examples

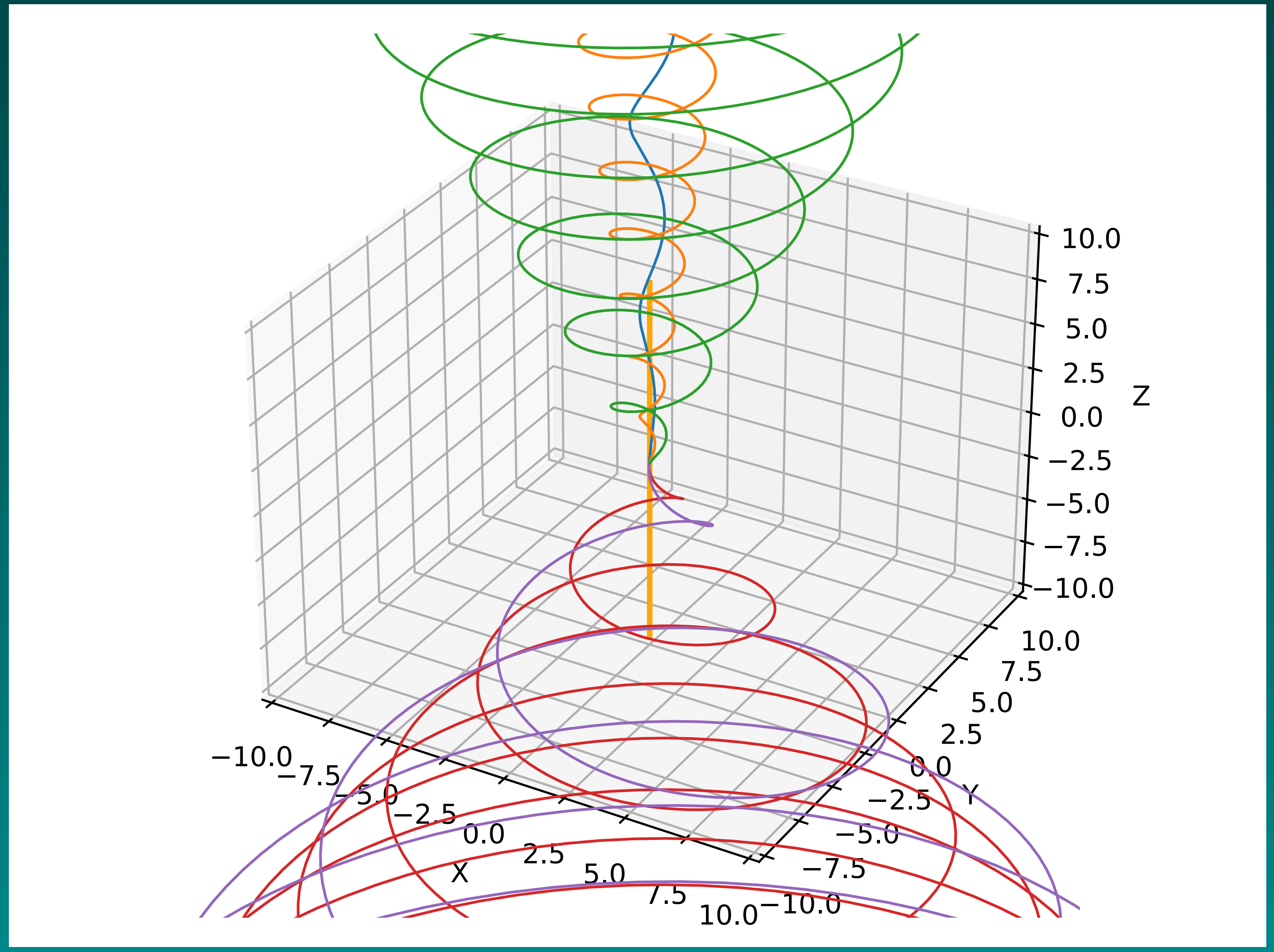
Track propagation



5 tracks, helical expanding

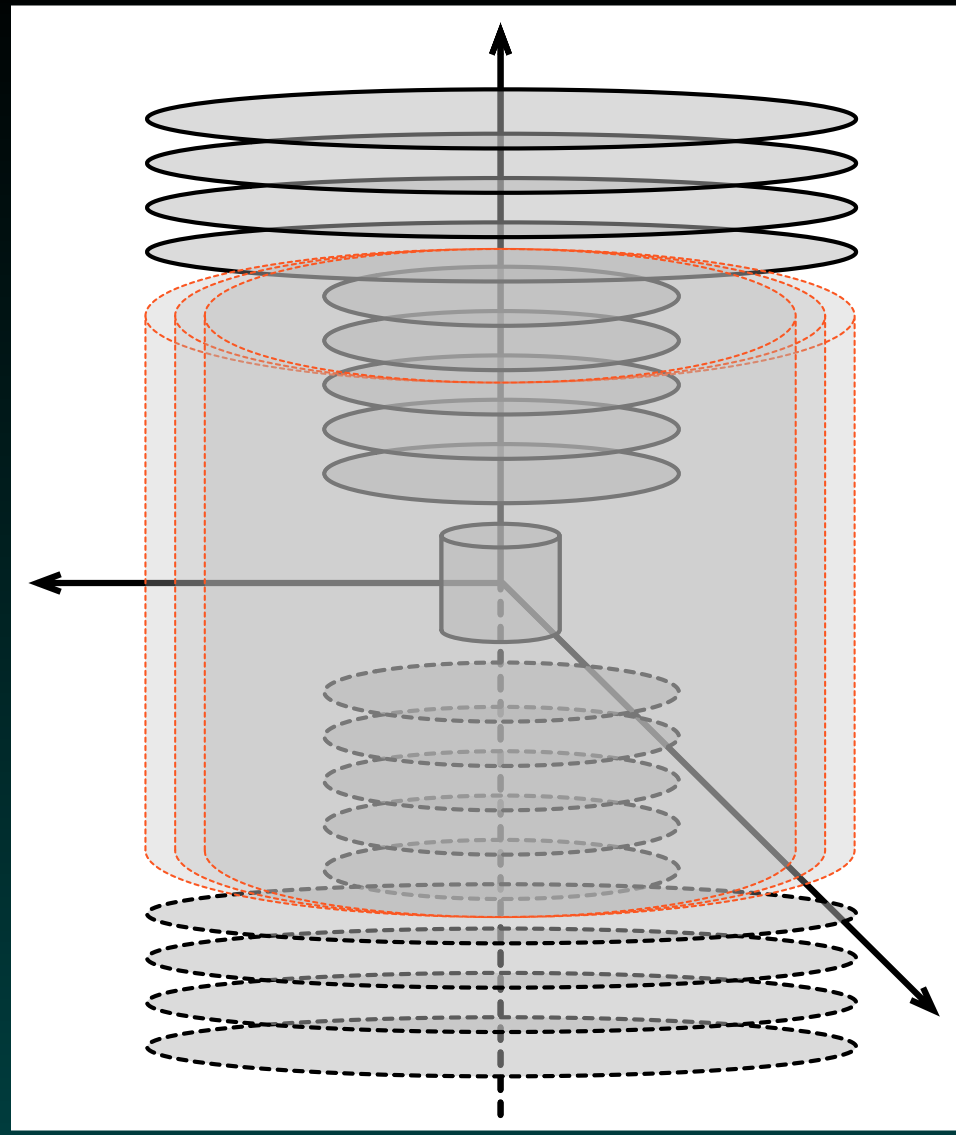




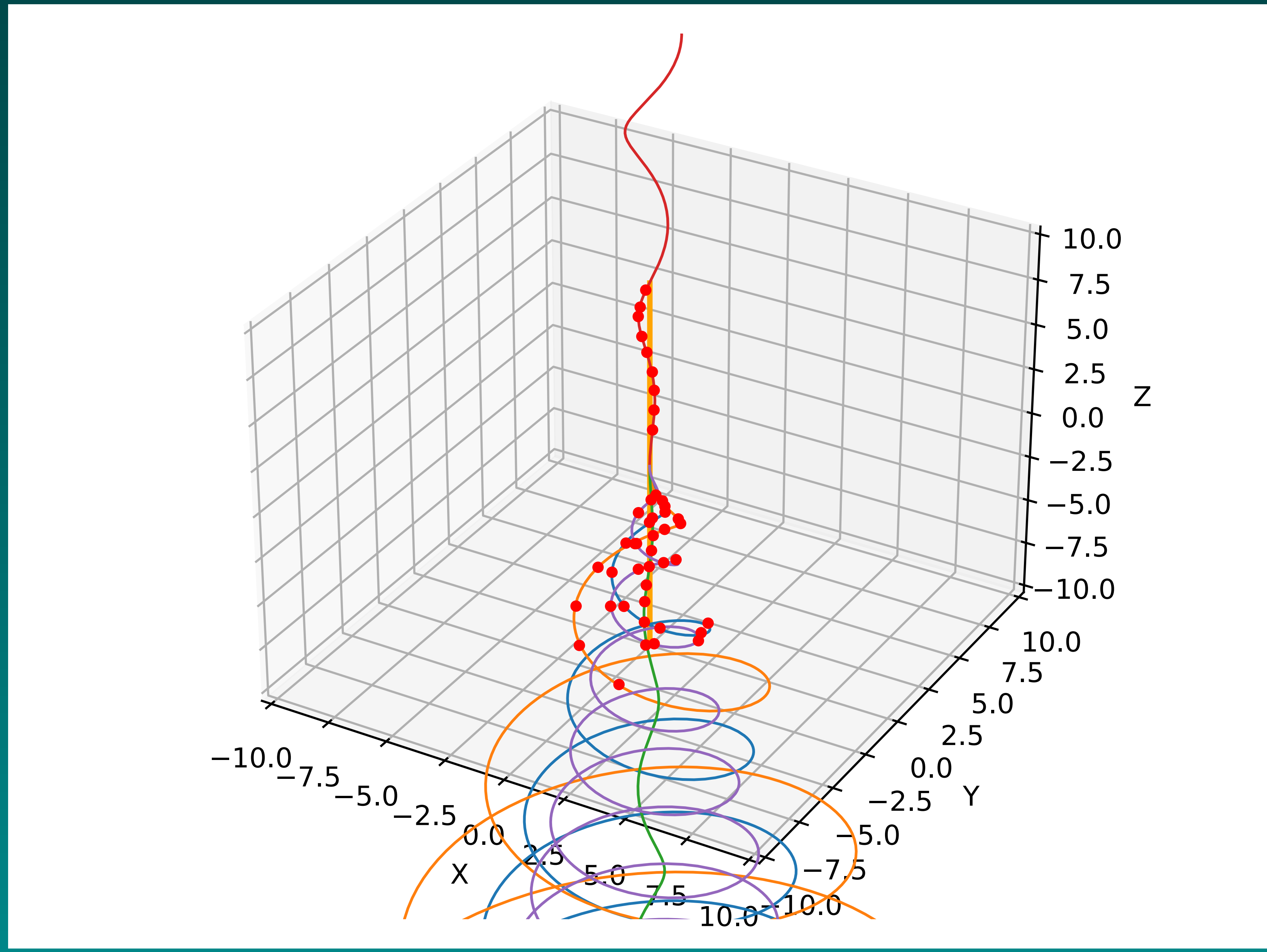


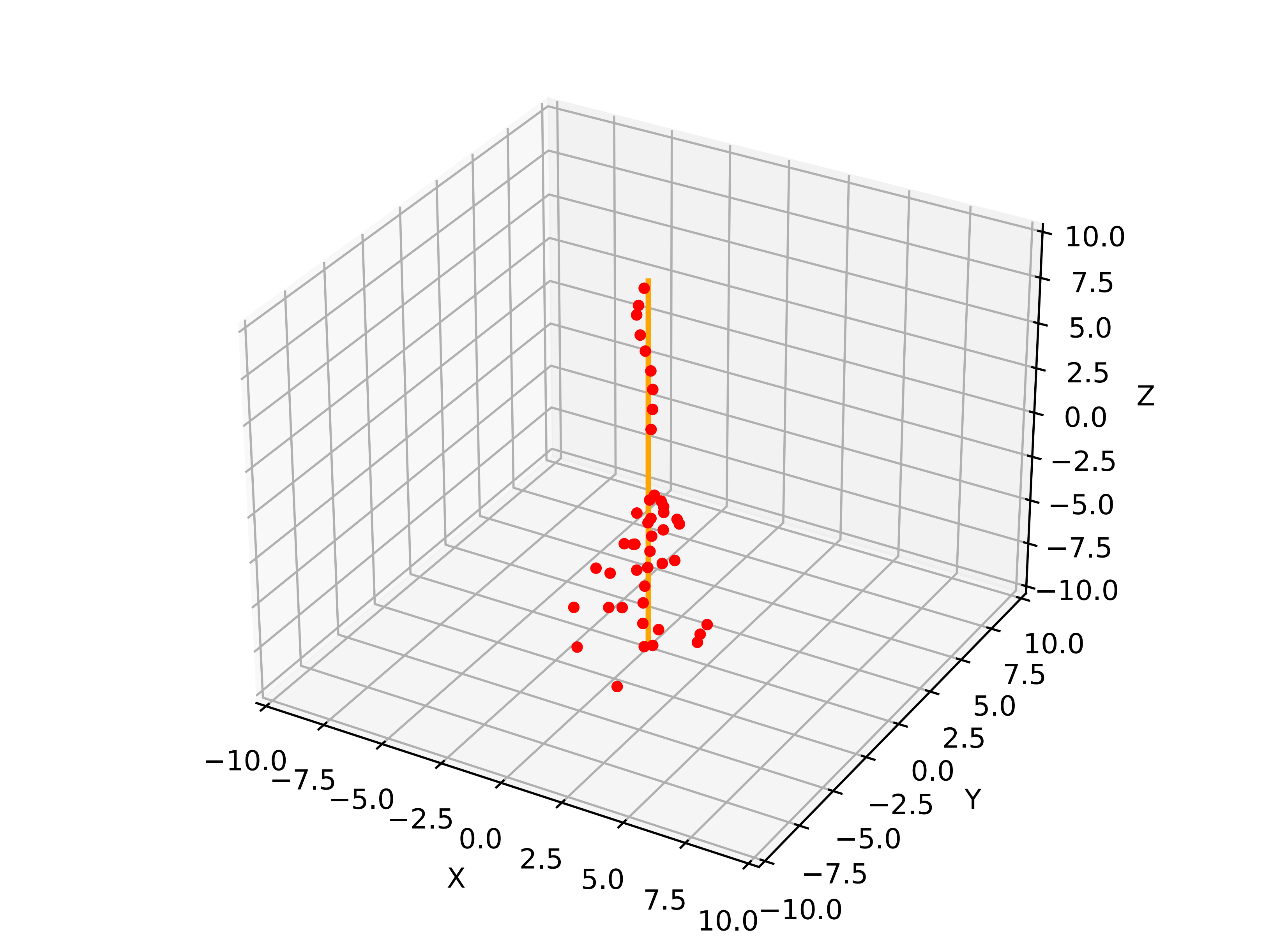
A few examples

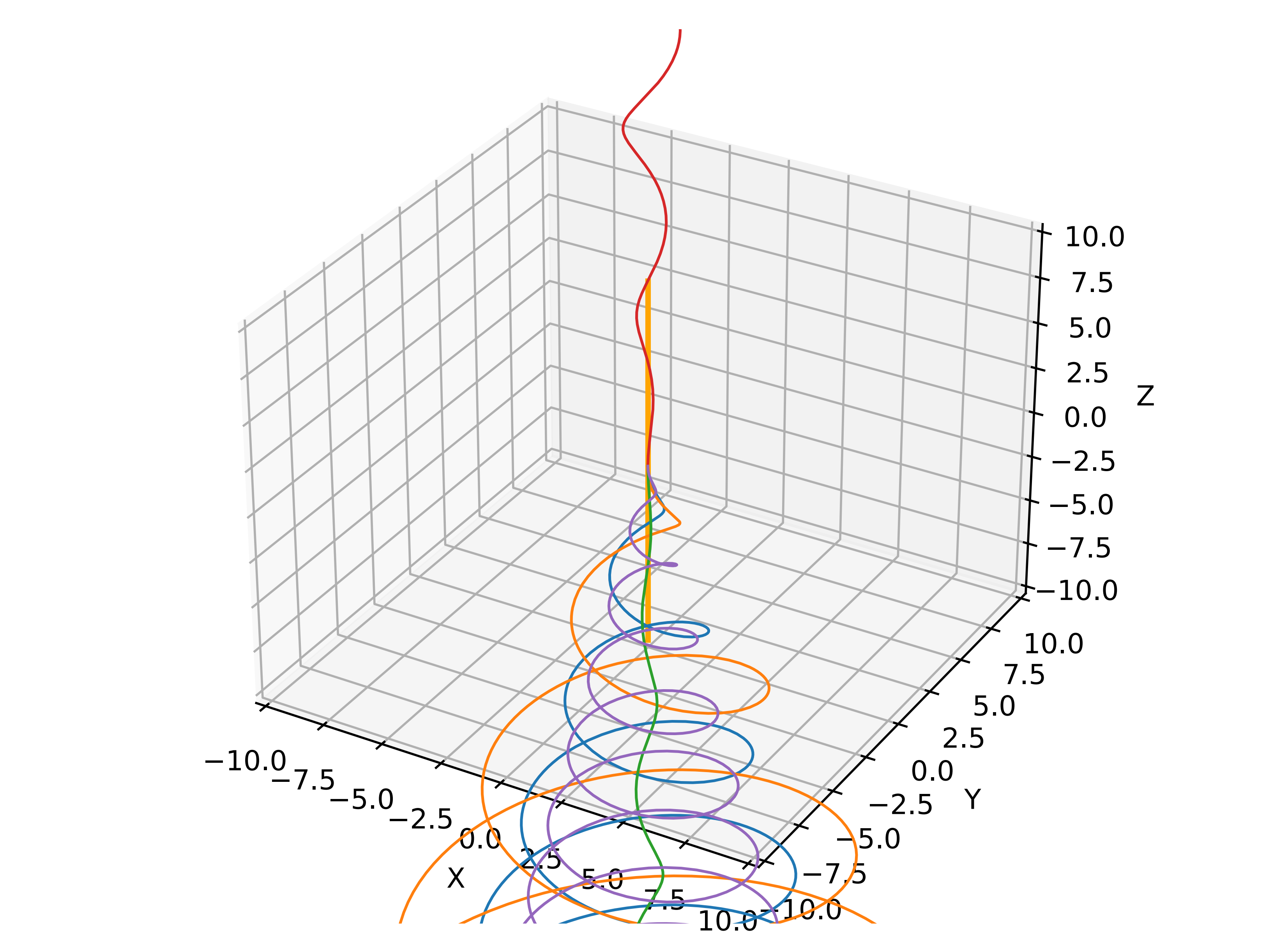
Track propagation



5 tracks, helical expanding







Part 2:
ML Algorithms and Approaches

ML approaches applied to REDVID data

- Two architectures: **U-Nets** and **Transformers**

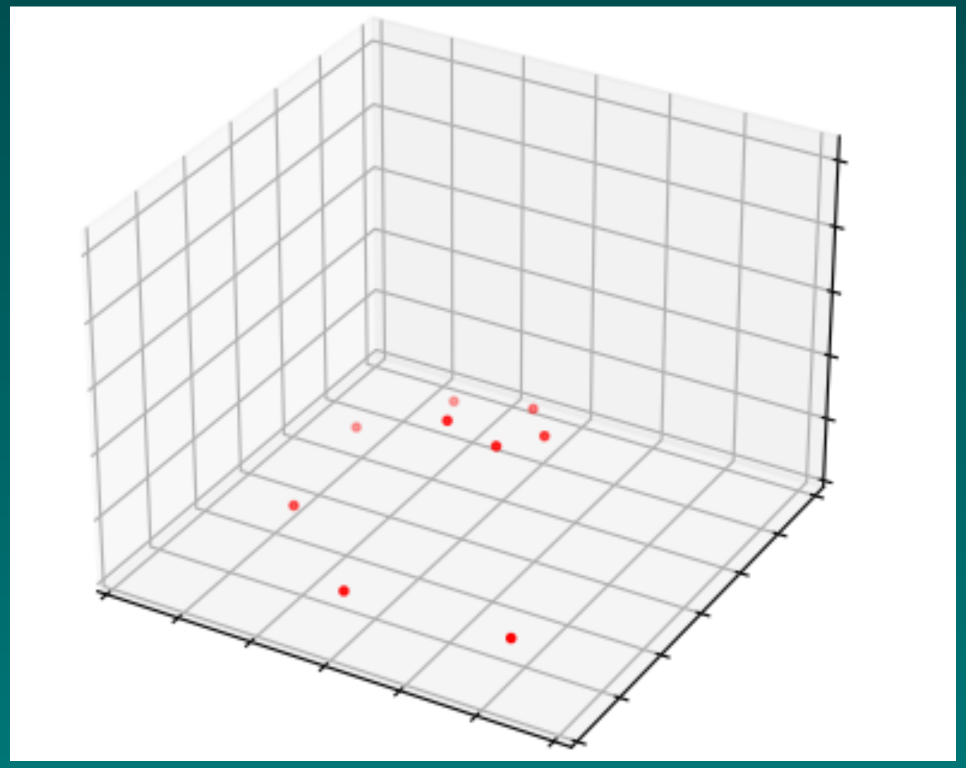
- U-Net strategy:
  => Track discovery with interpolation

- Transformer strategies:
  1. Similar to language translation, hits to tracks
  => Guess the next hit from a seed …
  2. Encoder-only transformer to regress the track parameters
  3. Encoder-only transformer as a classifier, to assign hits to spatial bins
  => Hit to road classification

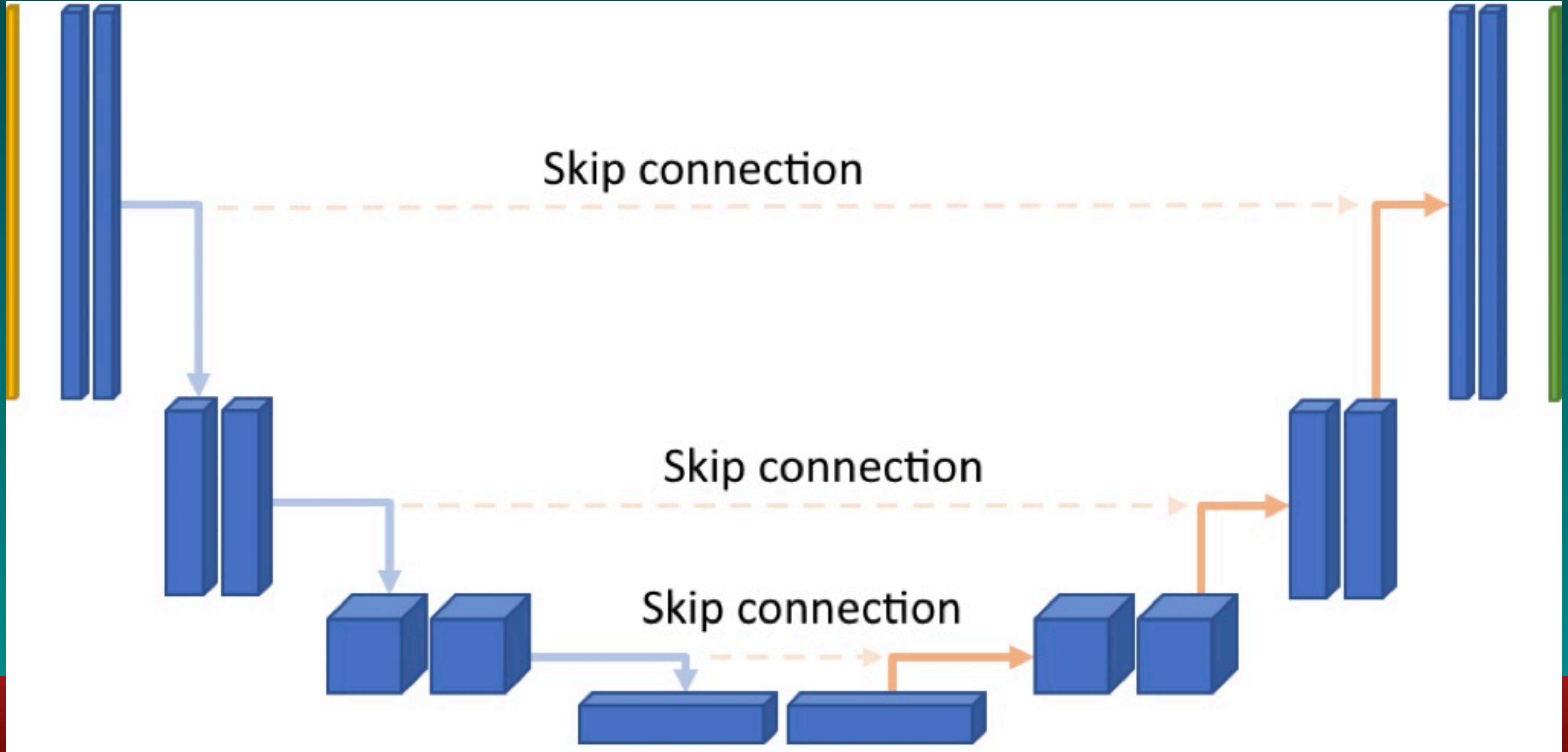
Note: Cost-effective evaluation of different solutions is made possible through complexity reductions …

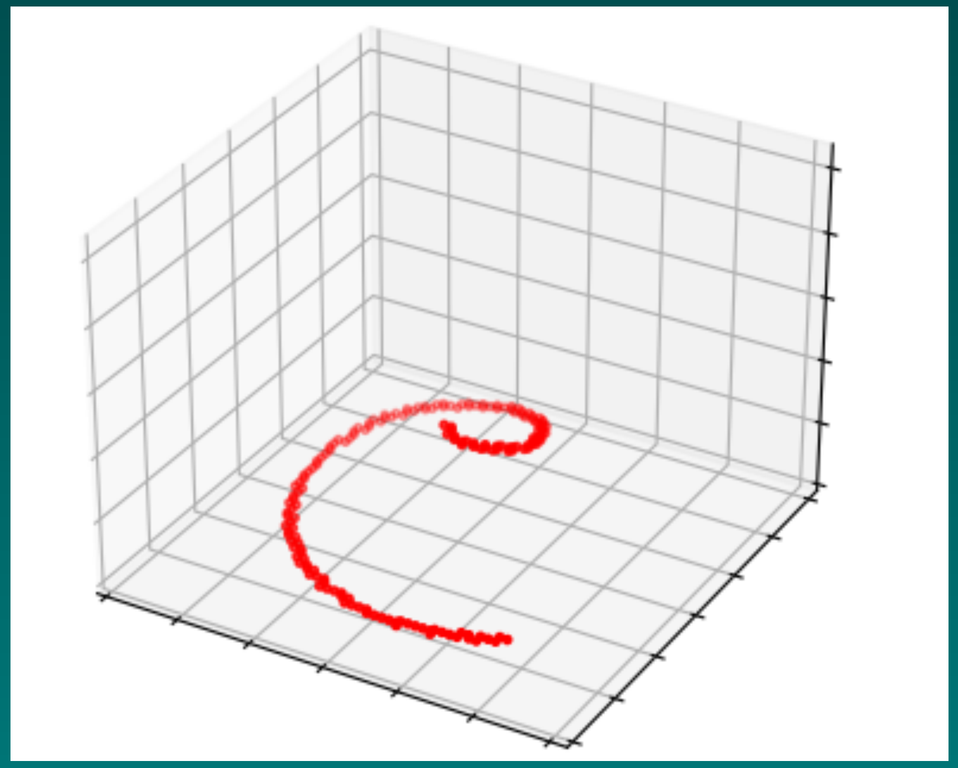
Clustering-based track classification

- Sparse U-Net to interpolate points in 3D space
- Vanilla convolutions are substituted by sub-manifold sparse convolutions
  => Only consider points with information
  => Suitable for our use-case, points in empty space …



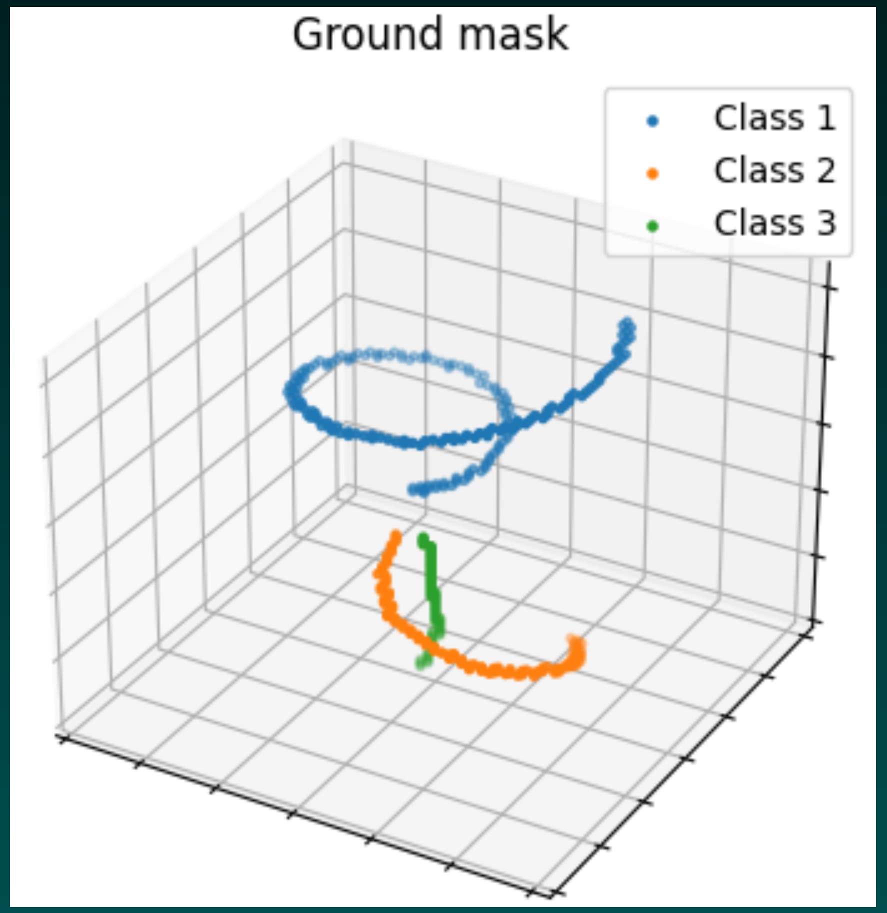
Sparse points





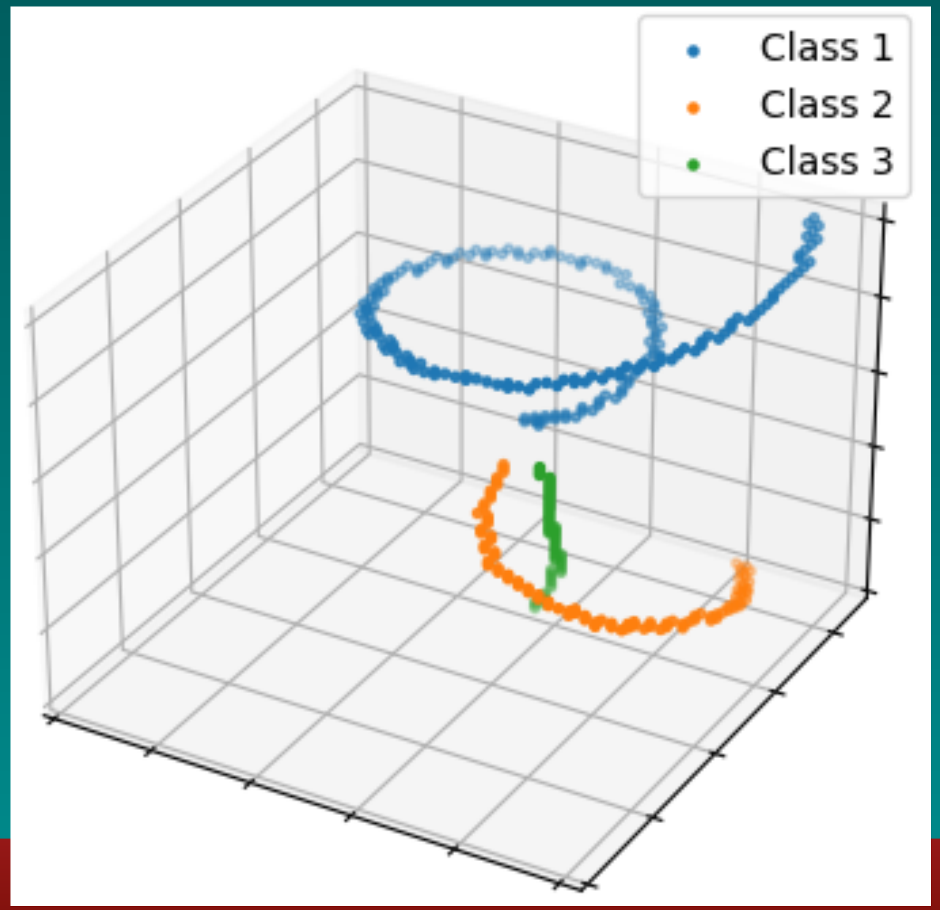
Interpolated track

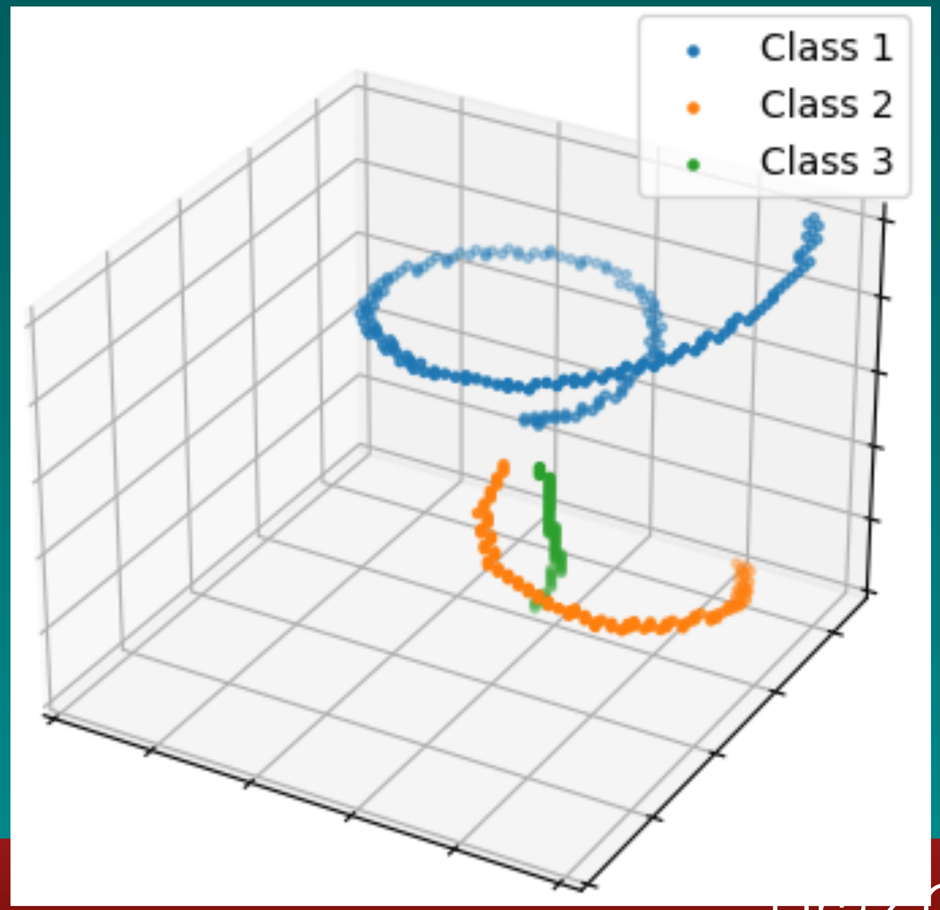
Clustering-based track classification

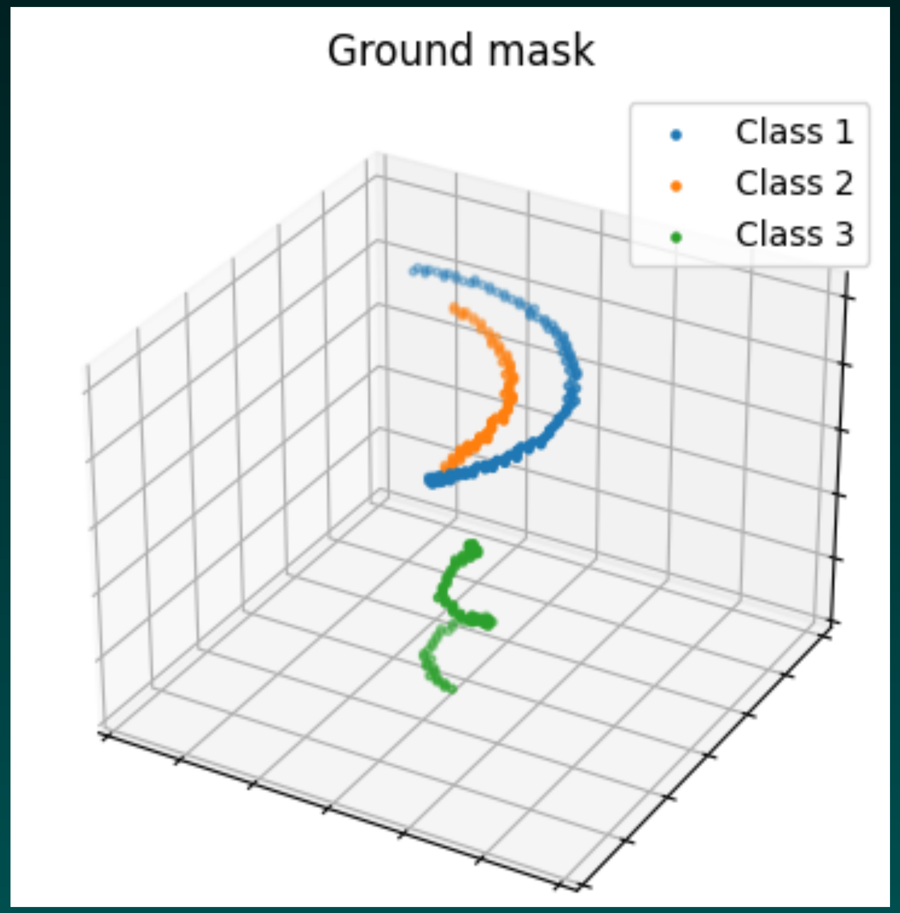


DBSCAN

Spectral clustering

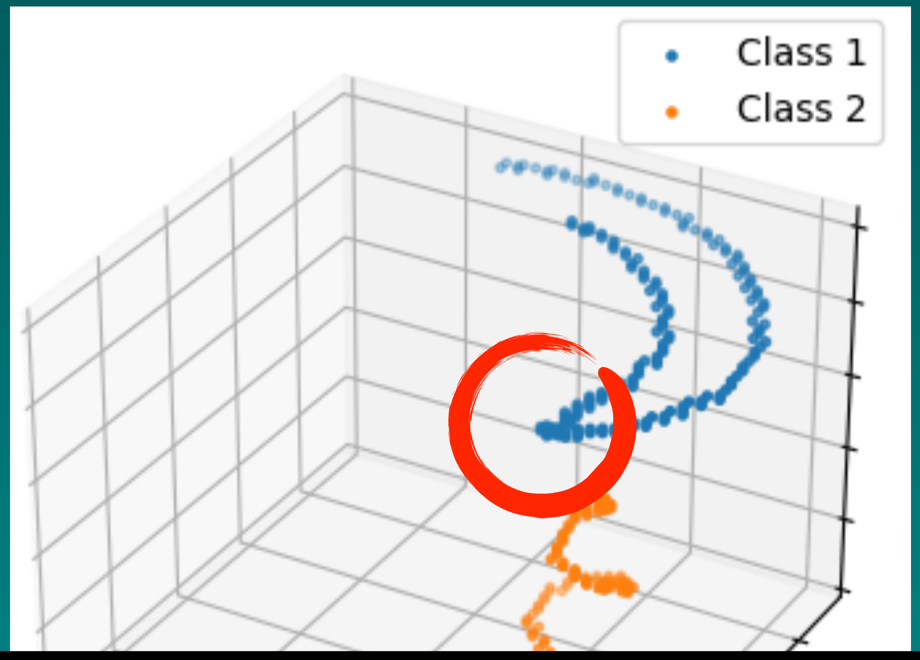


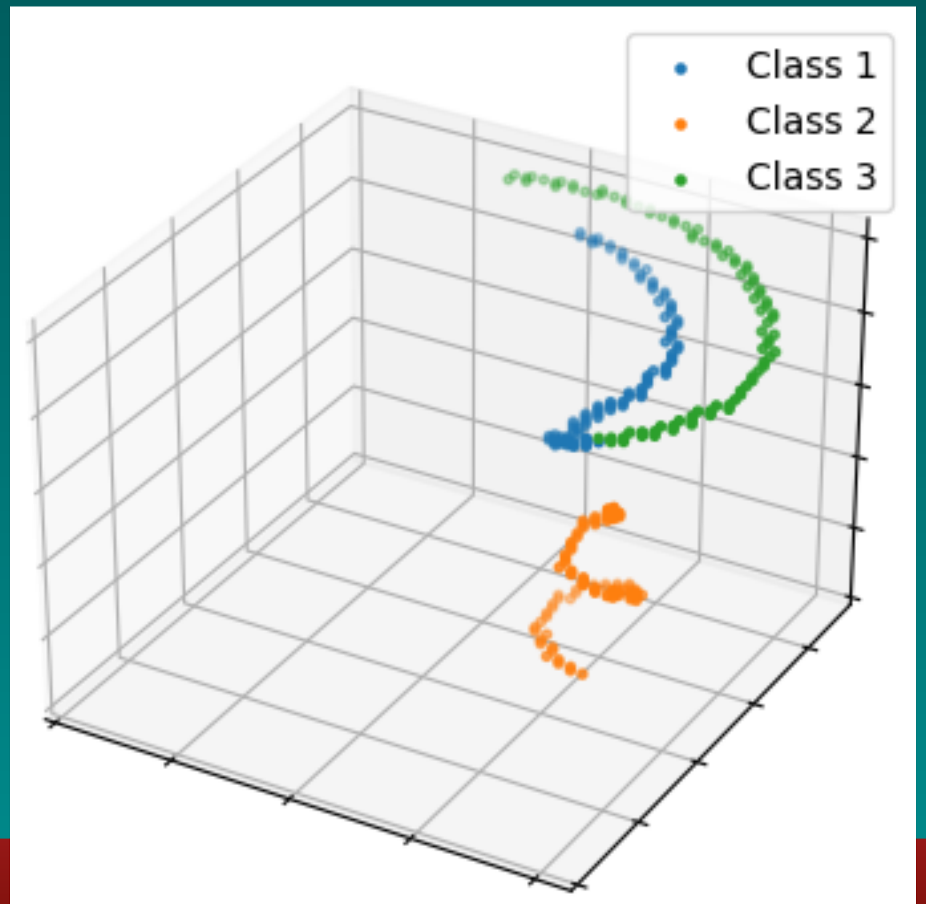




DBSCAN

Spectral clustering



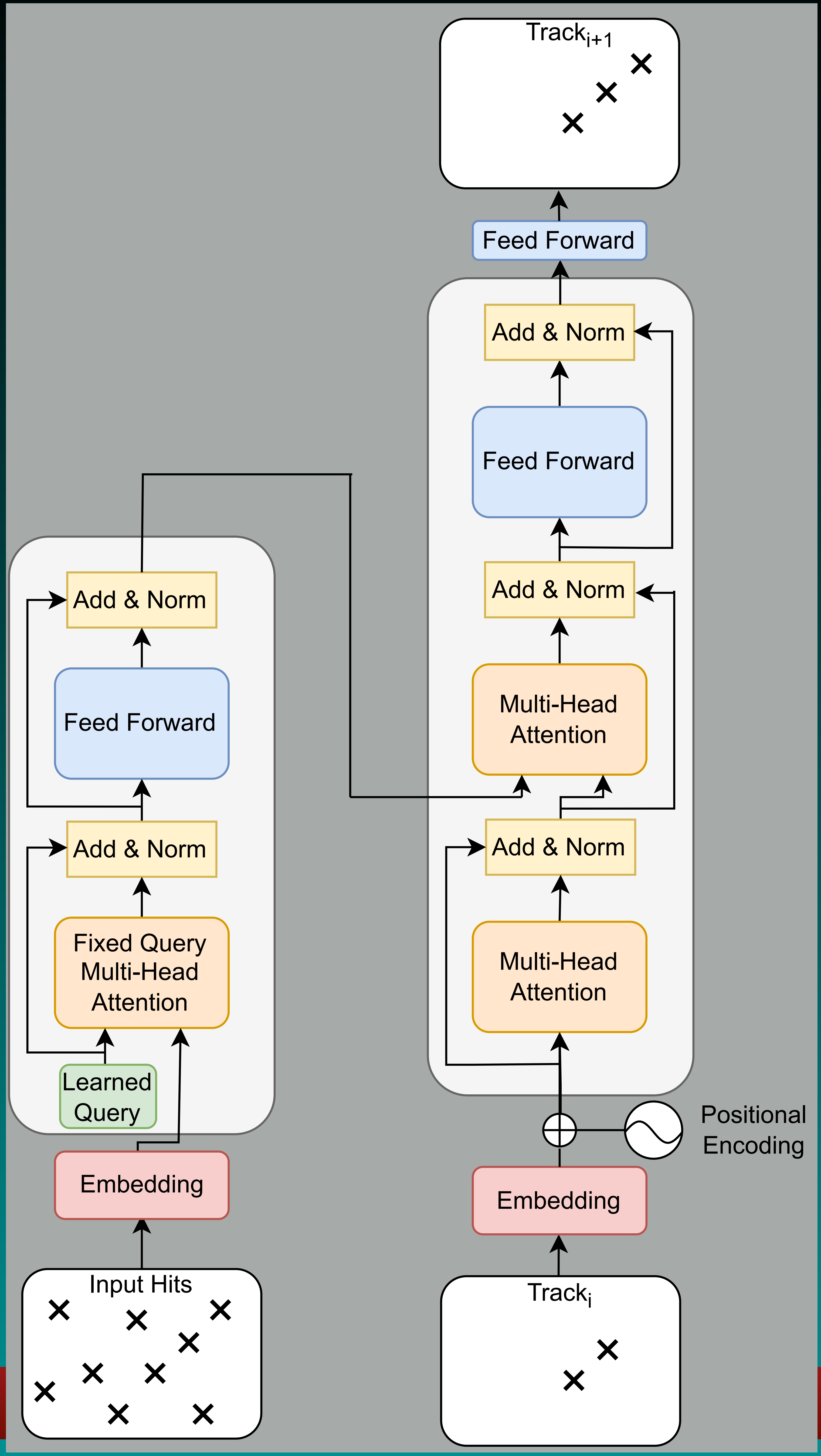


DBSCAN is affected by the available resolution

Transformer arch.

- Similar to the original Transformer paper => Autoregressive track building network => Model translates from hits to tracks (Language A -> Language B)

- Both encoder and decoder have position information as coordinates of hits

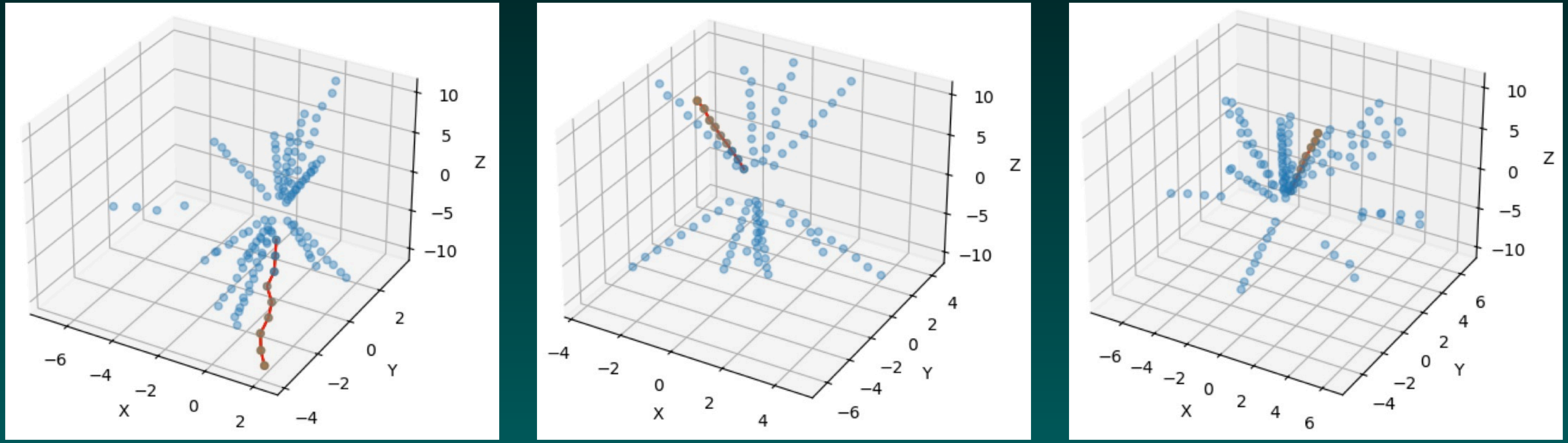


Guess the next hit from a seed …

Hits to tracks translation

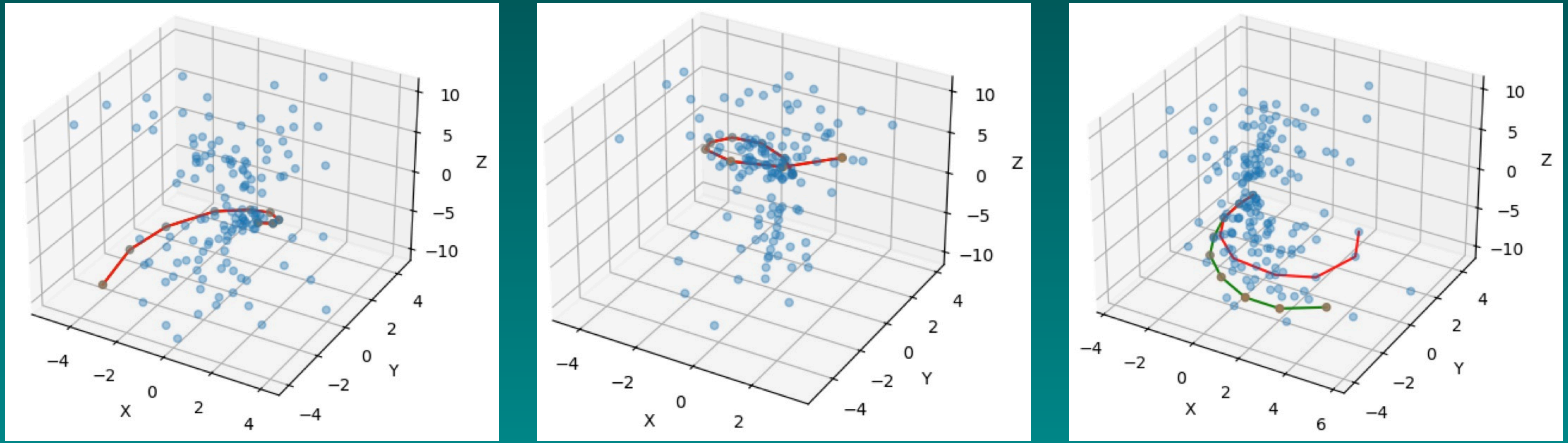
1-20 tracks, linear

correct classification 91.7%



1-20 tracks, helical

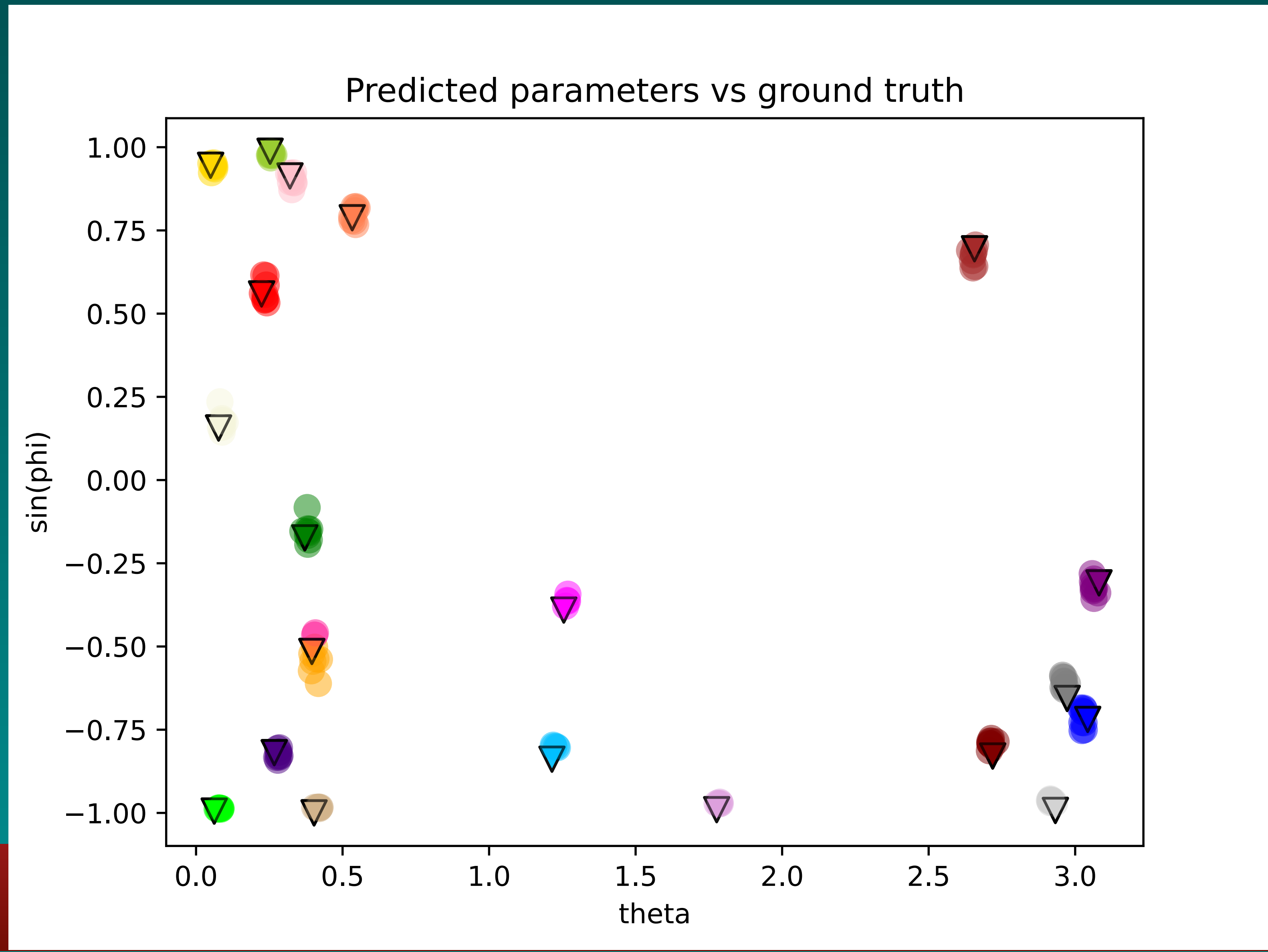
correct classification 72.3%

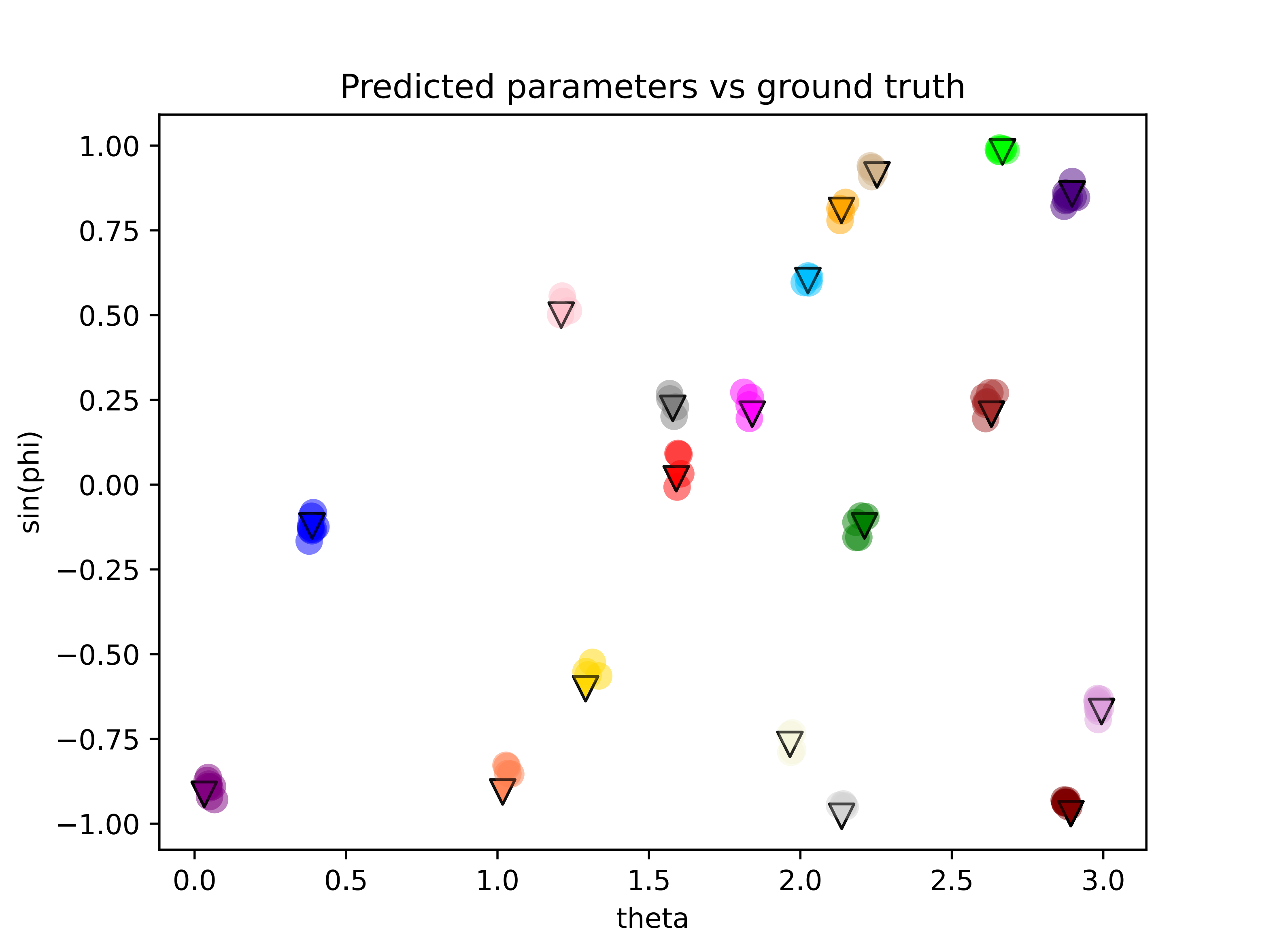


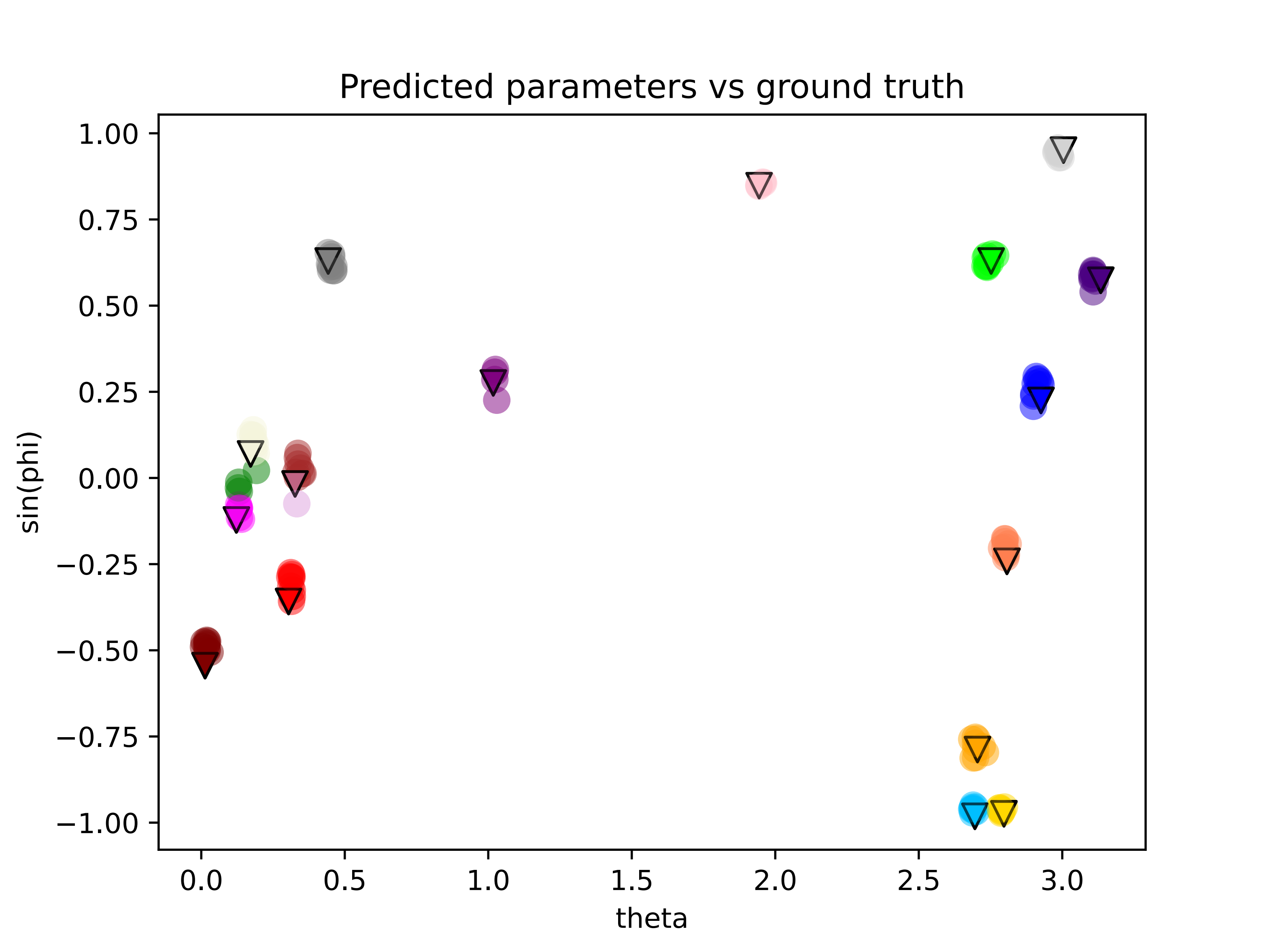
Track parameter regression

- Encoder-only transformer to regress the track parameters
  => Clustering hits based on regressed values

- Sequence of hits per event to sequence of track parameters
  => Agglomerative clustering







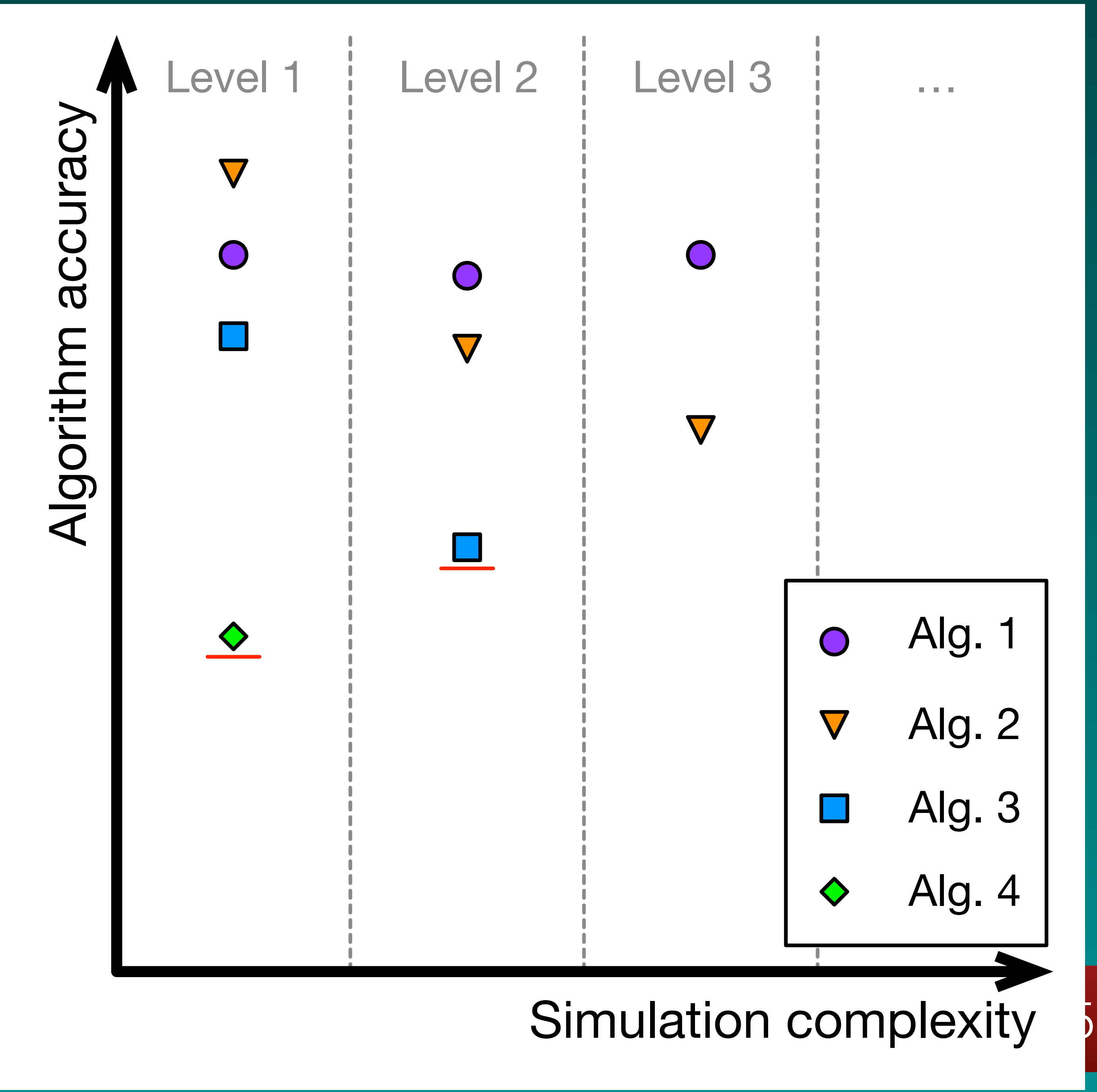
What to take from this?

- We are seafaring heathens!
  => Our approach is unorthodox, **out-of-the-physics-box**

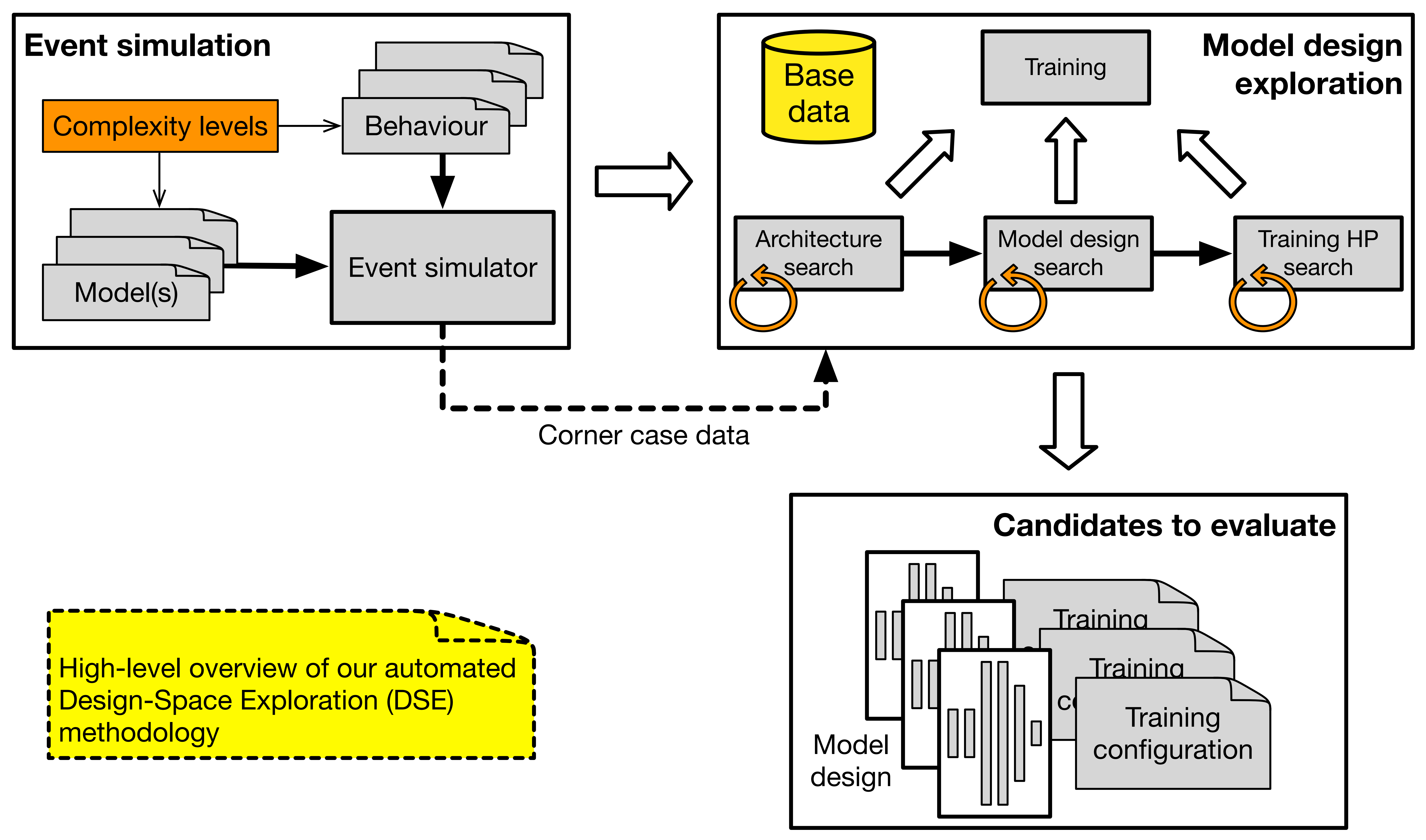
- This is work-in-progress, with a relatively fast pace

- In simple terms:
  => Try algorithms in low complexity levels: **Yay or nay?**
  => Increase the complexity, transfer, improve

- Next steps:
  => Move up to the level of trackML data
  => Move up to the level of ATLAS data
  => Automated DSE, achieve **secondary objectives**



What to take from this?



High-level overview of our automated Design-Space Exploration (DSE) methodology

Thanks!

Questions?