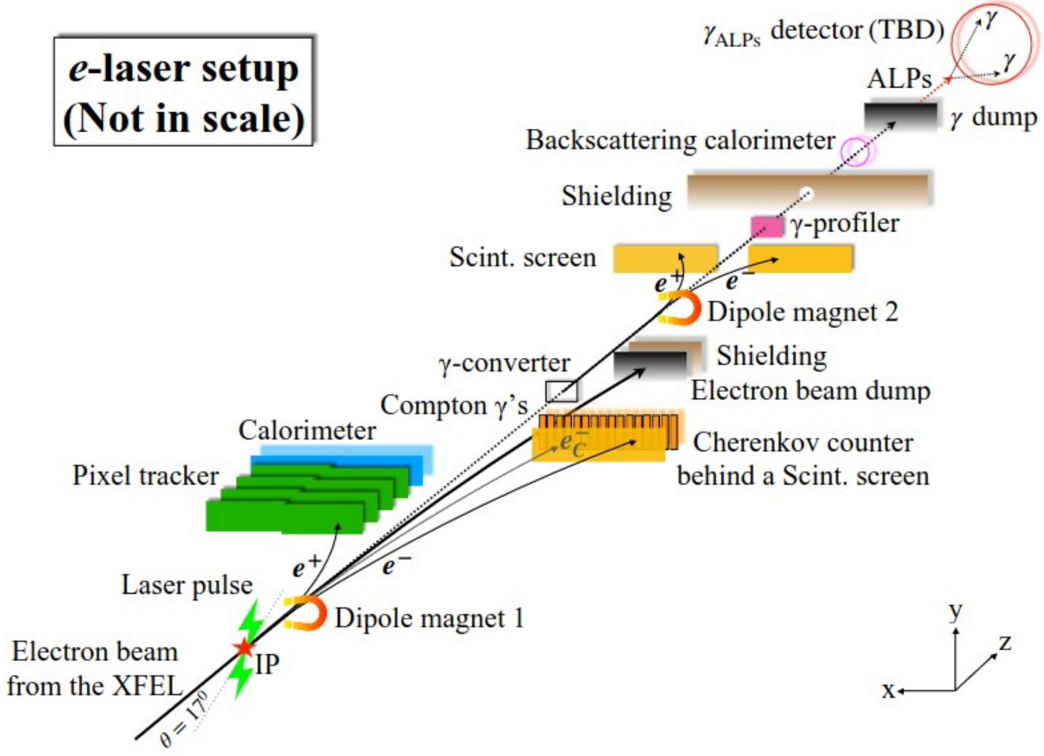
Tracking in the LUXE experiment

Michal Elad
24/08/2023

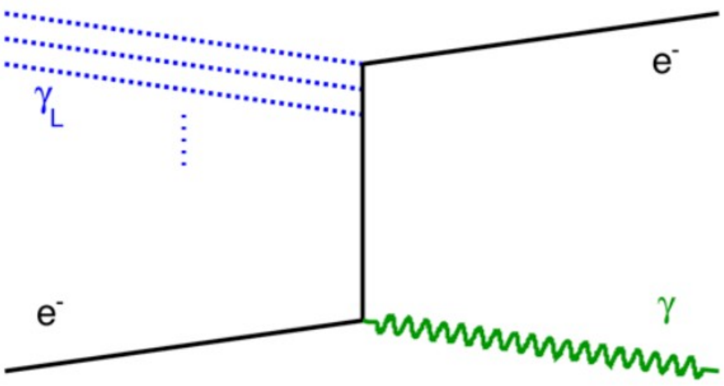
The Laser Und XFEL Experiment (LUXE)

<u>LUXE main goals:</u>

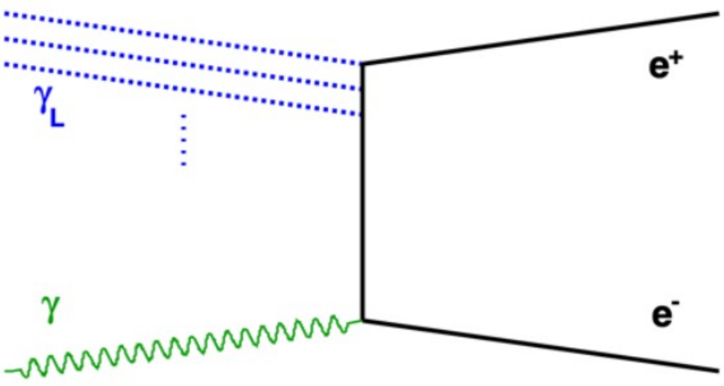
- Precision measurements in the **non-pertubative** regime of **QED**
- Search for **new particles** beyond the SM



The Laser Und XFEL Experiment (LUXE)



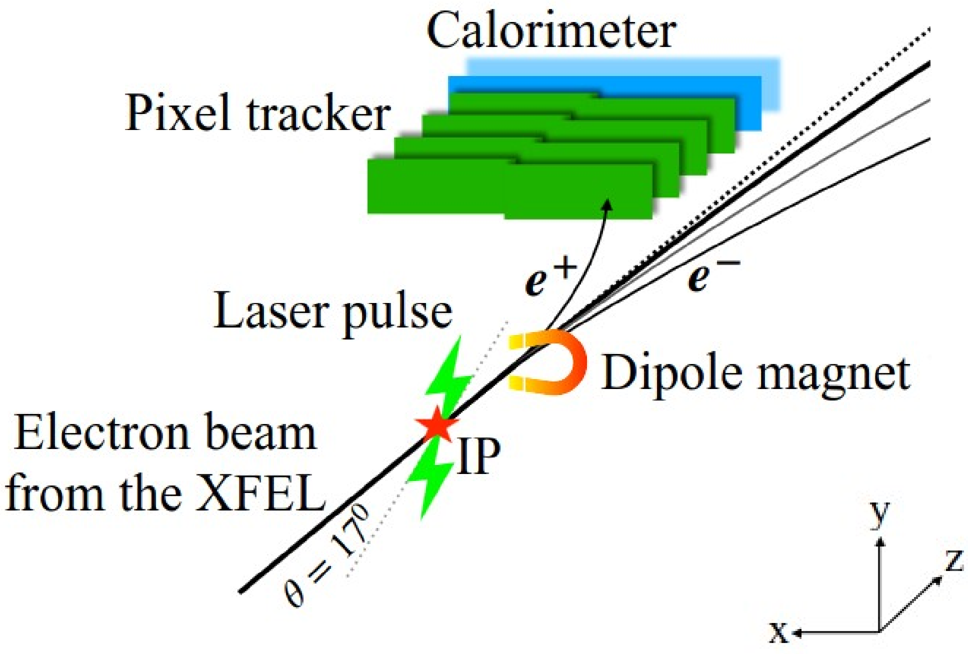
Non-linear Compton scattering



Breit-Wheeler process

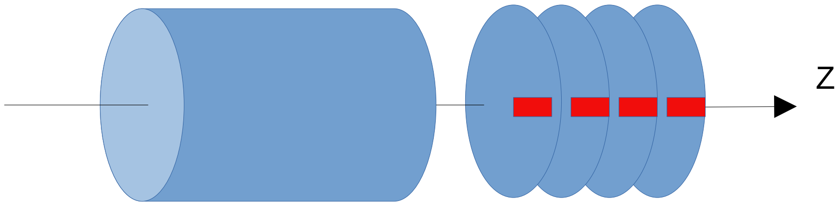
Tracking

- 4 Layers, 100 mm apart
- Each layer – 2 sensors slightly overlapping
- Hits position is currently given in **global** coordinates: (x, y, z)
- No magnetic field in the tracker → Trajectories are straight lines (up to scattering)



A common tracking software (ACTS)

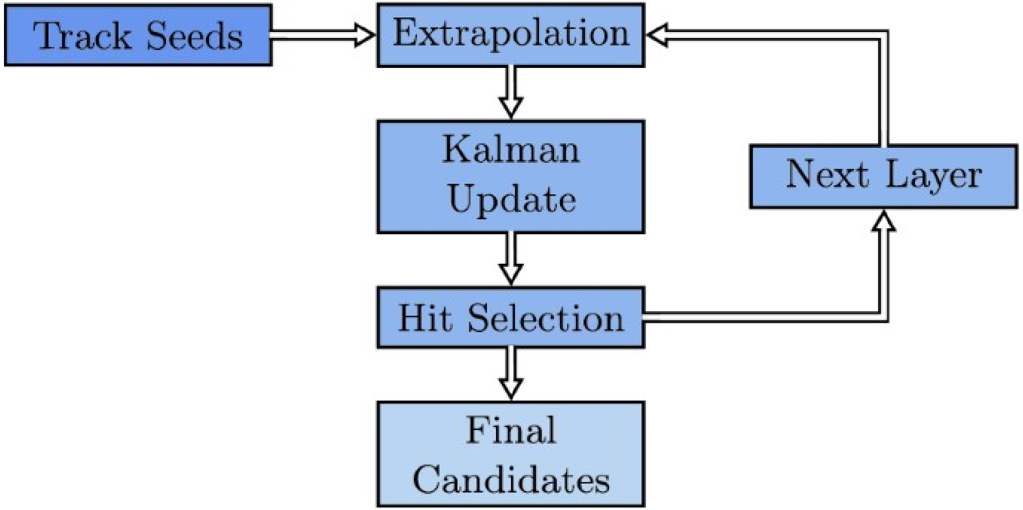
- Experiment-independent toolkit for track reconstruction in HEP
- Assumes a barrel-shaped detector with end-cap disks
- The 4 layers are mapped to a rectangle in 4 end-cap disks:

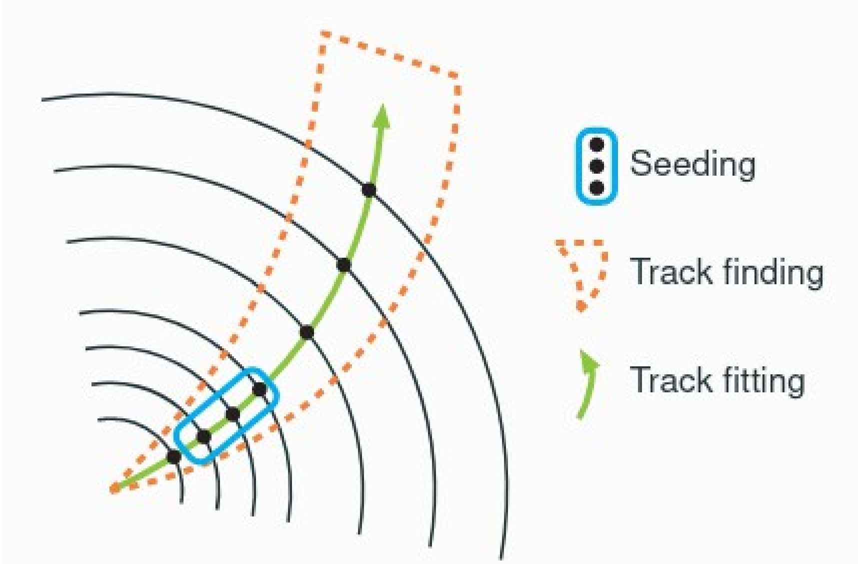


(Apologies for the unprofessional sketch)

Combinatorial Kalman Filter (CKF)

- Preforms the track finding and track fitting altogether.
- If more than one hit is found, the procedure splits for both possibilities





Images from: https://acts.readthedocs.io/en/latest/tracking.html , and: https://link.springer.com/chapter/10.1007/978-3-030-24997-7_6

Kalman Filter

- Preforms track fitting only, thus, requires track candidates
- At each step – adds the next hit and updates the fit

- The code is written based on the CKF for LUXE
- ACTS requires a propagator, updater, and a smoother
- Currently – assume a single particle and consider all hits as a track candidate
- Next step – get multiple track candidates and loop over them

Global to local coordinates

- Current given coordinates are global (x, y, z) where (0, 0, 0) is the interaction point (IP).
- ACTS requires a surface + local coordinates (u, v).
- The variance is given in term of du, dv.
- Conversion required:

  u, v + surface $\rightarrow$ x, y, z $\rightarrow$ ρ, z

```
// compute Jacobian from global coordinates to rho/z
//
//         rho = sqrt(x² + y²)
// drho/d{x,y} = (1 / sqrt(x² + y²)) * 2 * {x,y}
//             = 2 * {x,y} / r
//       dz/dz = 1 (duuh!)
//
double x = globalPos[Acts::ePos0];
double y = globalPos[Acts::ePos1];
double scale = 2 / std::hypot(x, y);
Acts::ActsMatrix<2, 3> jacXyzToRhoZ = Acts::ActsMatrix<2, 3>::Zero();
jacXyzToRhoZ(0, Acts::ePos0) = scale * x;
jacXyzToRhoZ(0, Acts::ePos1) = scale * y;
jacXyzToRhoZ(1, Acts::ePos2) = 1;
// compute Jacobian from local coordinates to rho/z
Acts::ActsMatrix<2, 2> jac =
    jacXyzToRhoZ * rotLocalToGlobal.block<3, 2>(Acts::ePos0, Acts::ePos0);
// compute rho/z variance
Acts::ActsVector<2> var = (jac * localCov * jac.transpose()).diagonal();
```

Quadratic unconstrained binary optimization (QUBO)

- Input for the Kalman Filter is the output of the QUBO
- The objective function to minimize:

$$O = \sum_{i}^{N} \sum_{j<i} b_{ij} T_i T_j + \sum_{i=1}^{N} a_i T_i,$$

T_i – triplets of consecutive hits, assume binary values.

b_{ij} – compatibility of triplets

a_i – quality of the triplet

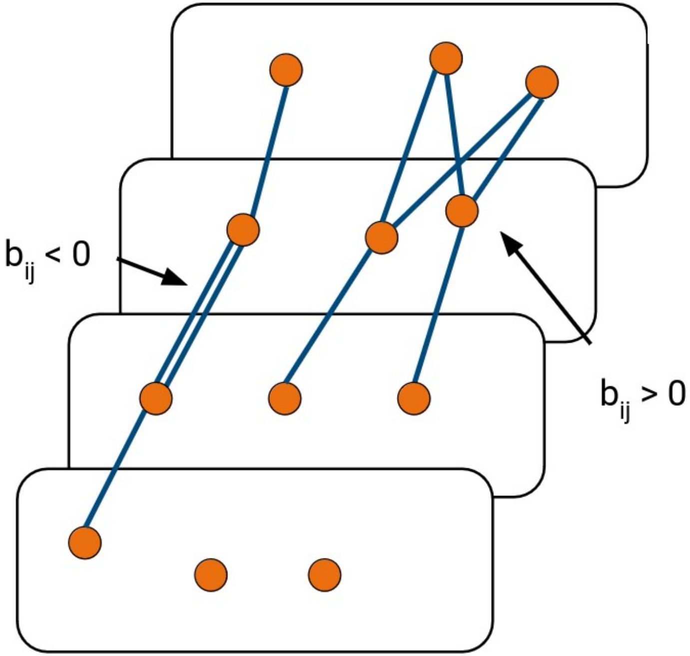
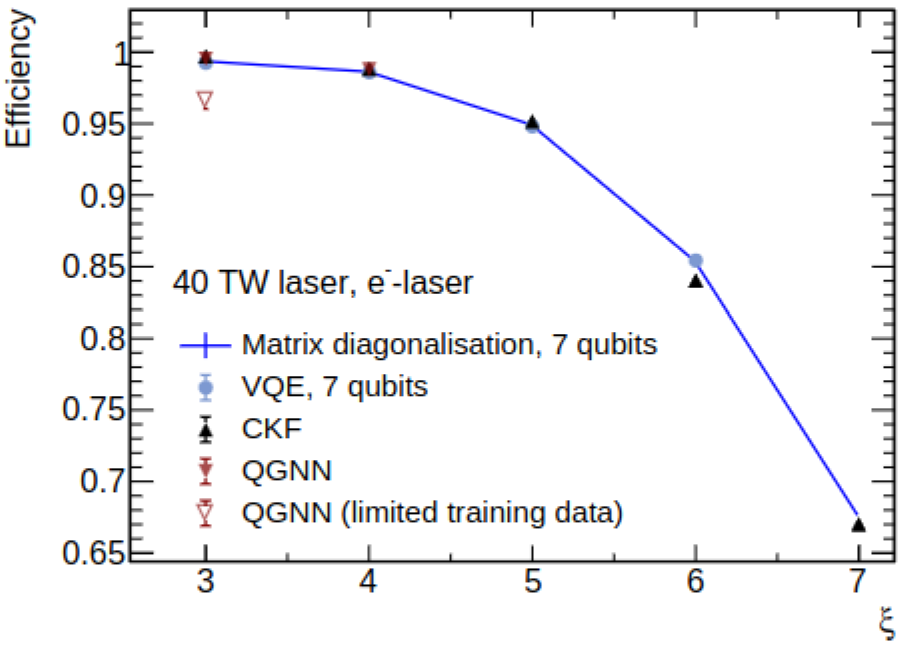
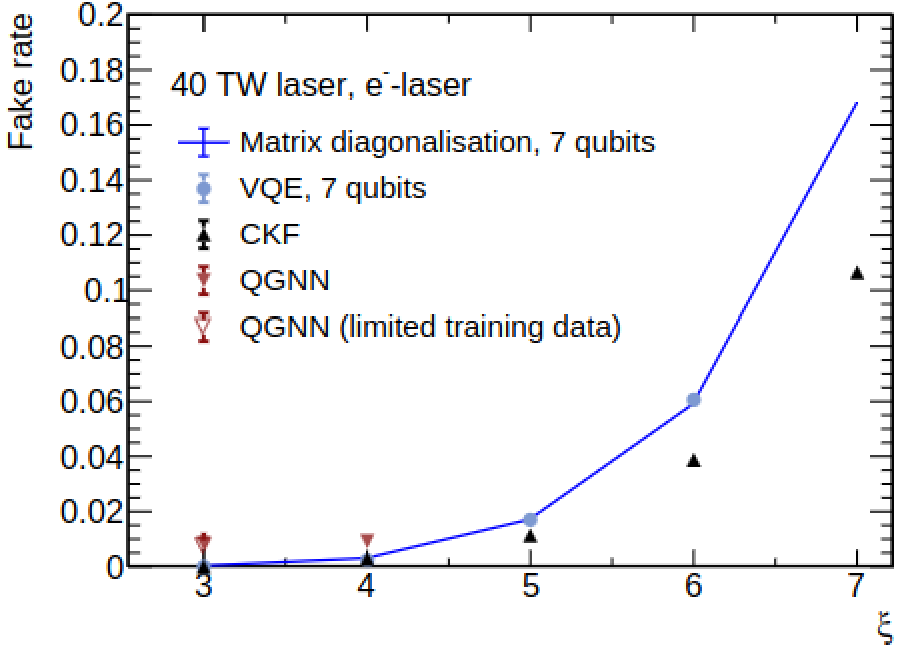


Image by courtesy of David Spataro

CKF efficiency and fake rate





CKF includes the track fitting. For QUBO, the track candidates were fitted to straight lines with the least-square method. Current aim is to fit using the Kalman filter and compare the results.
Image taken from: https://arxiv.org/pdf/2304.01690.pdf