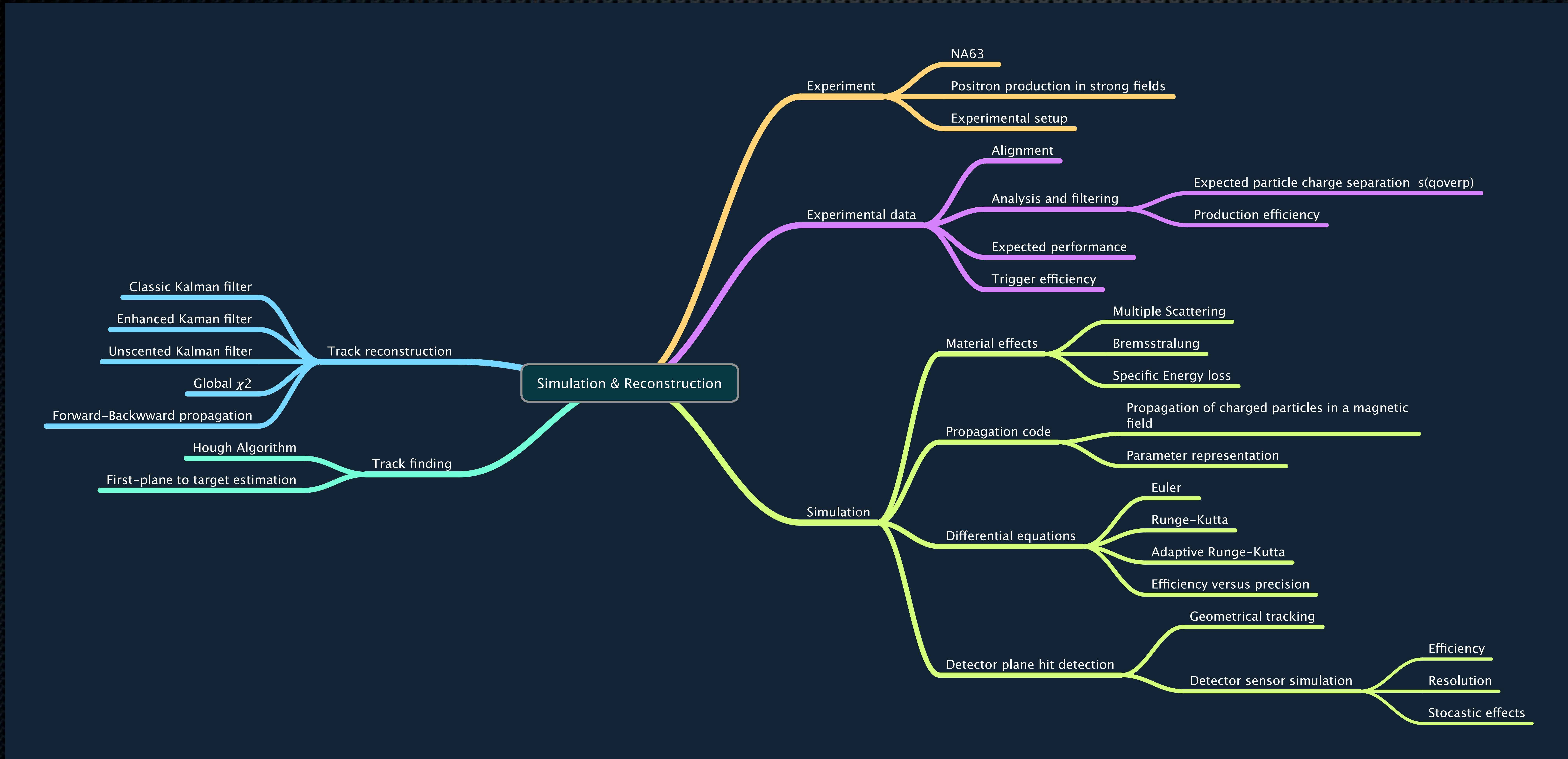
Simulation & Reconstruction

Or: Discovering aspects of computational [experimental] particle physics

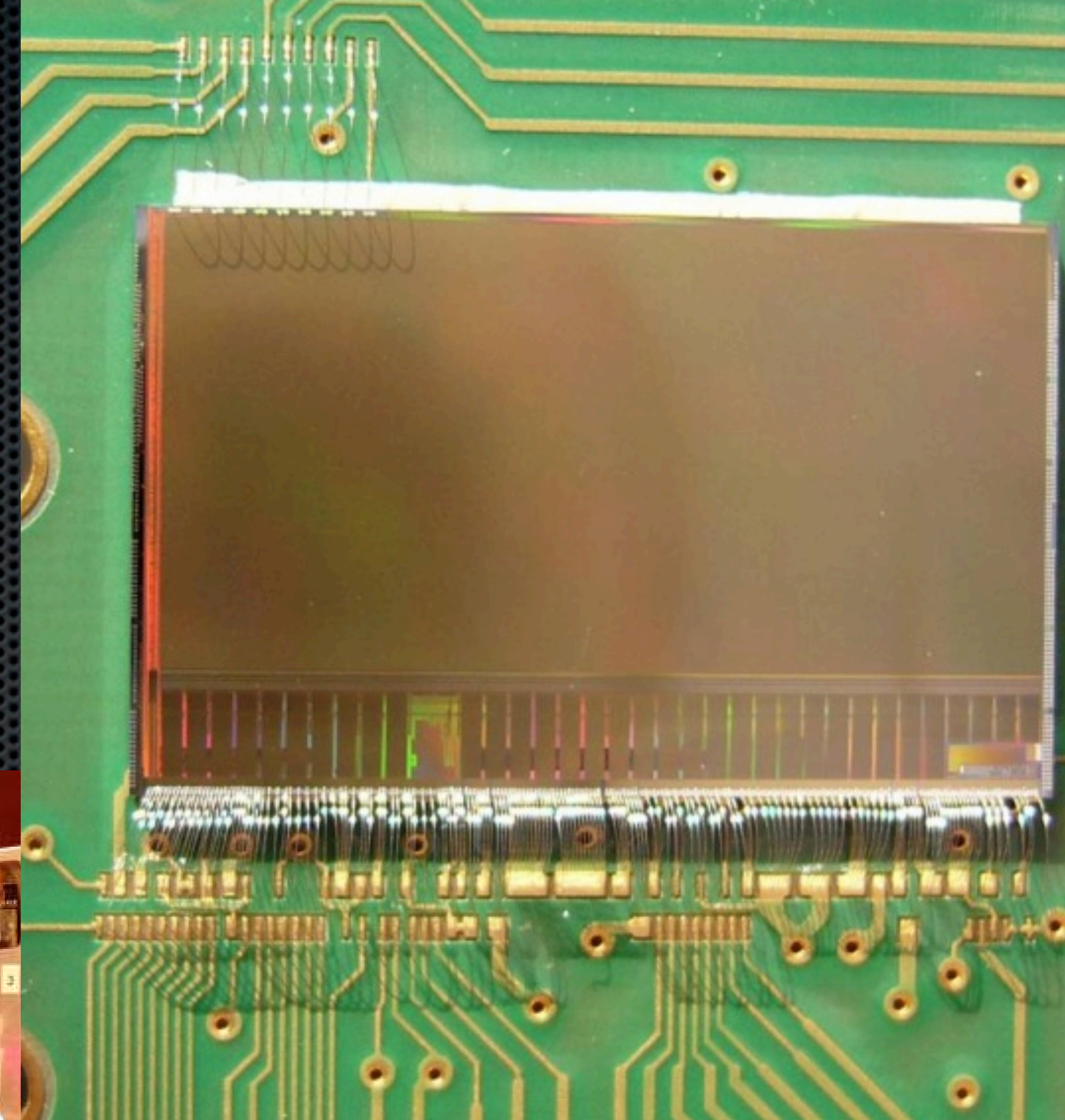
Morten Dam Jørgensen, 2013

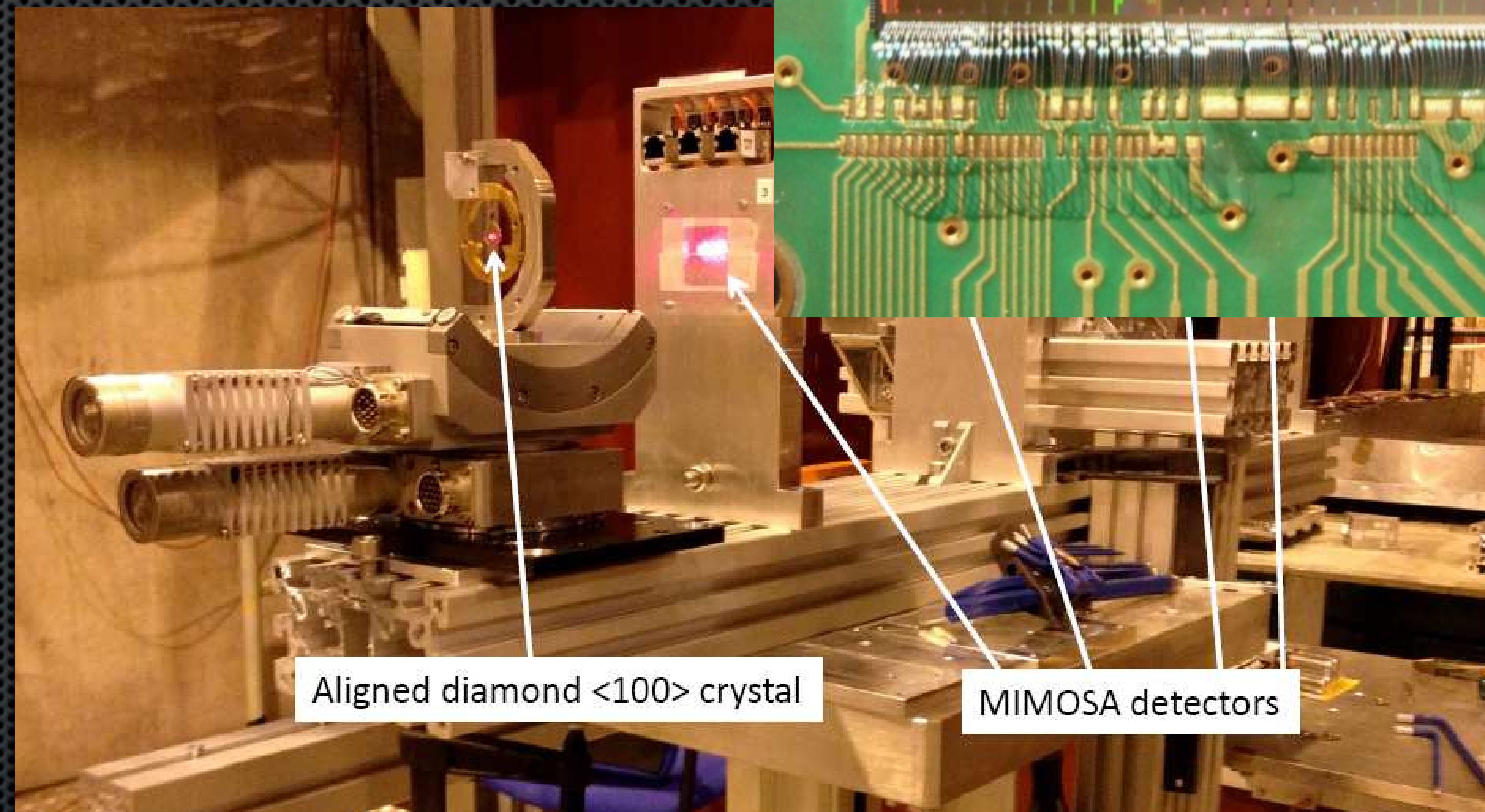
Overview


Simulation & Reconstruction
Experiment
NA63
Positron production in strong fields
Experimental setup
Experimental data
Alignment
Analysis and filtering
Expected particle charge separation s(qoverp)
Production efficiency
Expected performance
Trigger efficiency
Simulation
Material effects
Multiple Scattering
Bremsstralung
Specific Energy loss
Propagation code
Propagation of charged particles in a magnetic field
Parameter representation
Differential equations
Euler
Runge-Kutta
Adaptive Runge-Kutta
Efficiency versus precision
Detector plane hit detection
Geometrical tracking
Detector sensor simulation
Efficiency
Resolution
Stocastic effects
Track reconstruction
Classic Kalman filter
Enhanced Kaman filter
Unscented Kalman filter
Global $\chi 2$
Forward-Backwward propagation
Track finding
Hough Algorithm
First-plane to target estimation

The Experiment

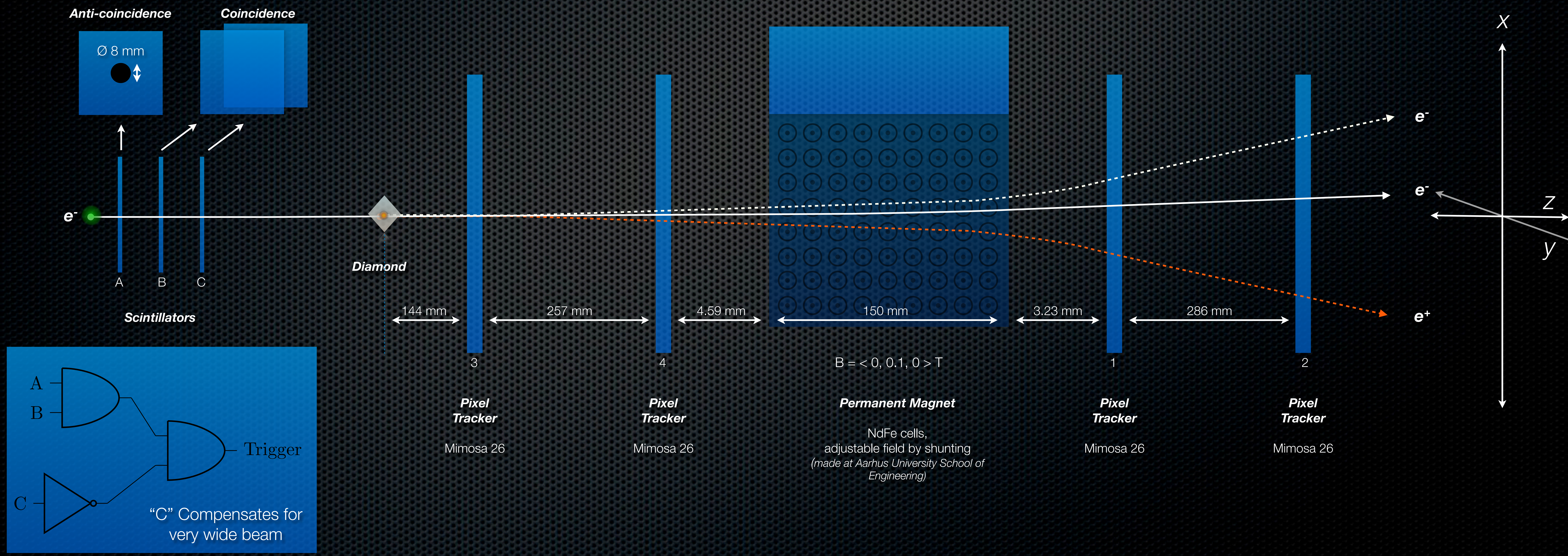
- Positron production due to strong electric fields...





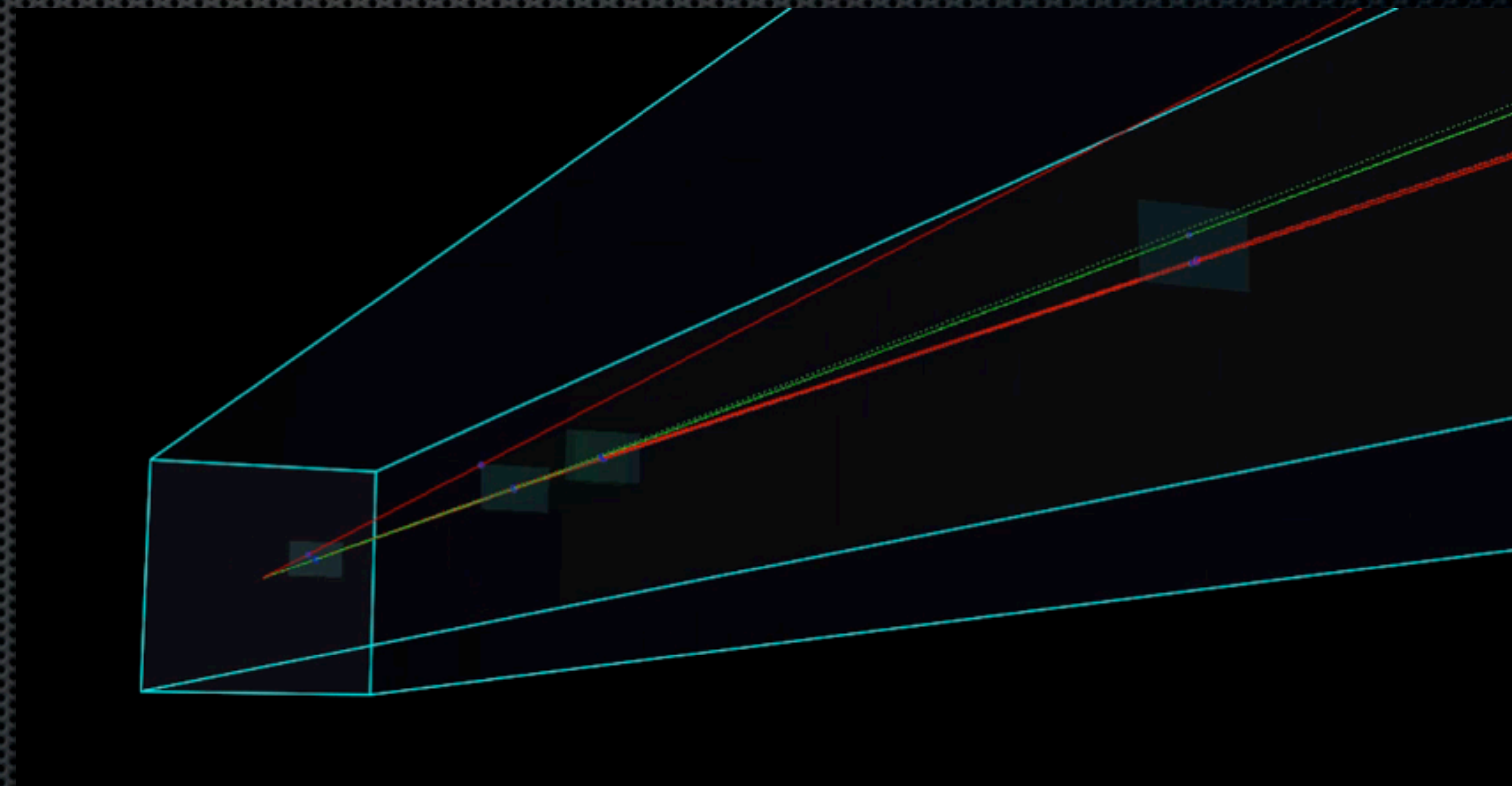
Mimosa NA63 Diamond Setup

Mimosa NA63 Diamond Setup



The Simulation

- Write a simulation to understand how to reconstruct tracks in the events...



Particle Propagation

- The problem
- Magnetic Field
- Scattering and Energy Loss
- Track parameter representation
- Differential Solvers
- Detection modelling

The problem

- Solve the propagation equation in an arbitrary magnetic field, effectively handle the Lorentz force:

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = \frac{\mathrm{d}(m\gamma \ \mathrm{d}\mathbf{x}/dt)}{\mathrm{d}t} = c^2\kappa \ q \ \mathbf{v}(t) \times \mathbf{B}(\mathbf{x}(t))$$

- The complete description of a particle's state as a function of distance *z* at a given point in space can be described by a 5-vector with local coordinates, directions and charge/ momentum,

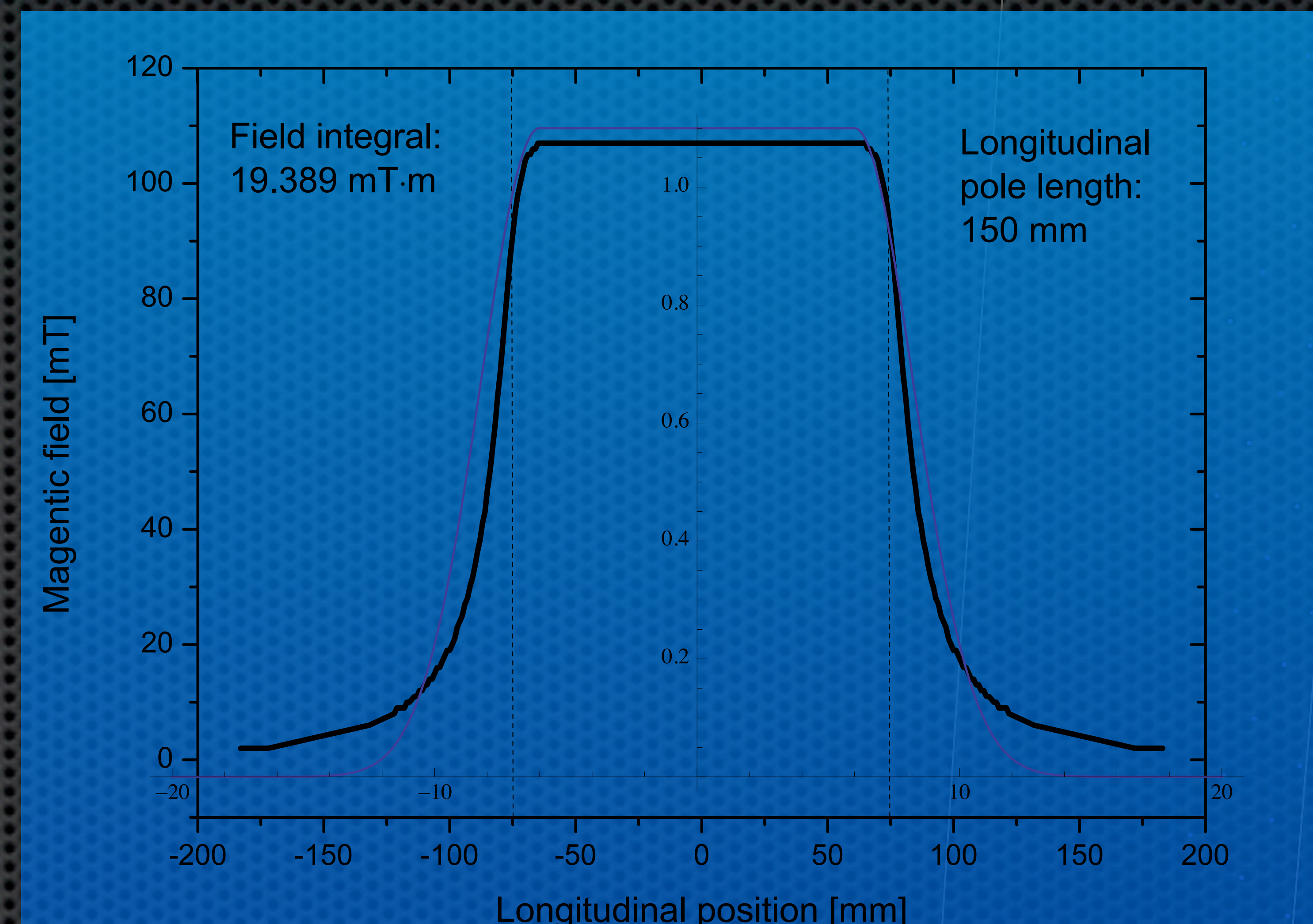
$$\mathbf{r} = (x, y, t_x, t_y, q/p)^T$$

- That makes ti possible to solve the dynamical equation as a function of distance along *z*,

$$\frac{\mathrm{d}\mathbf{r}(z)}{\mathrm{d}z} = \begin{pmatrix} \mathrm{d}x \\ \mathrm{d}y \\ \mathrm{d}t_x \\ \mathrm{d}t_y \\ \mathrm{d}(q/p) \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \\ \kappa\cdot(q/p)\cdot\sqrt{1+t_x^2+t_y^2}\cdot(t_x t_y\cdot B_x-(1+t_x^2)\cdot B_y+t_y\cdot B_z) \\ \kappa\cdot(q/p)\cdot\sqrt{1+t_x^2+t_y^2}\cdot((1+t_y^2)\cdot B_x-t_x t_y\cdot B_y-t_x\cdot B_z) \\ 0 \end{pmatrix} \equiv \mathbf{f}(z,\mathbf{r})$$

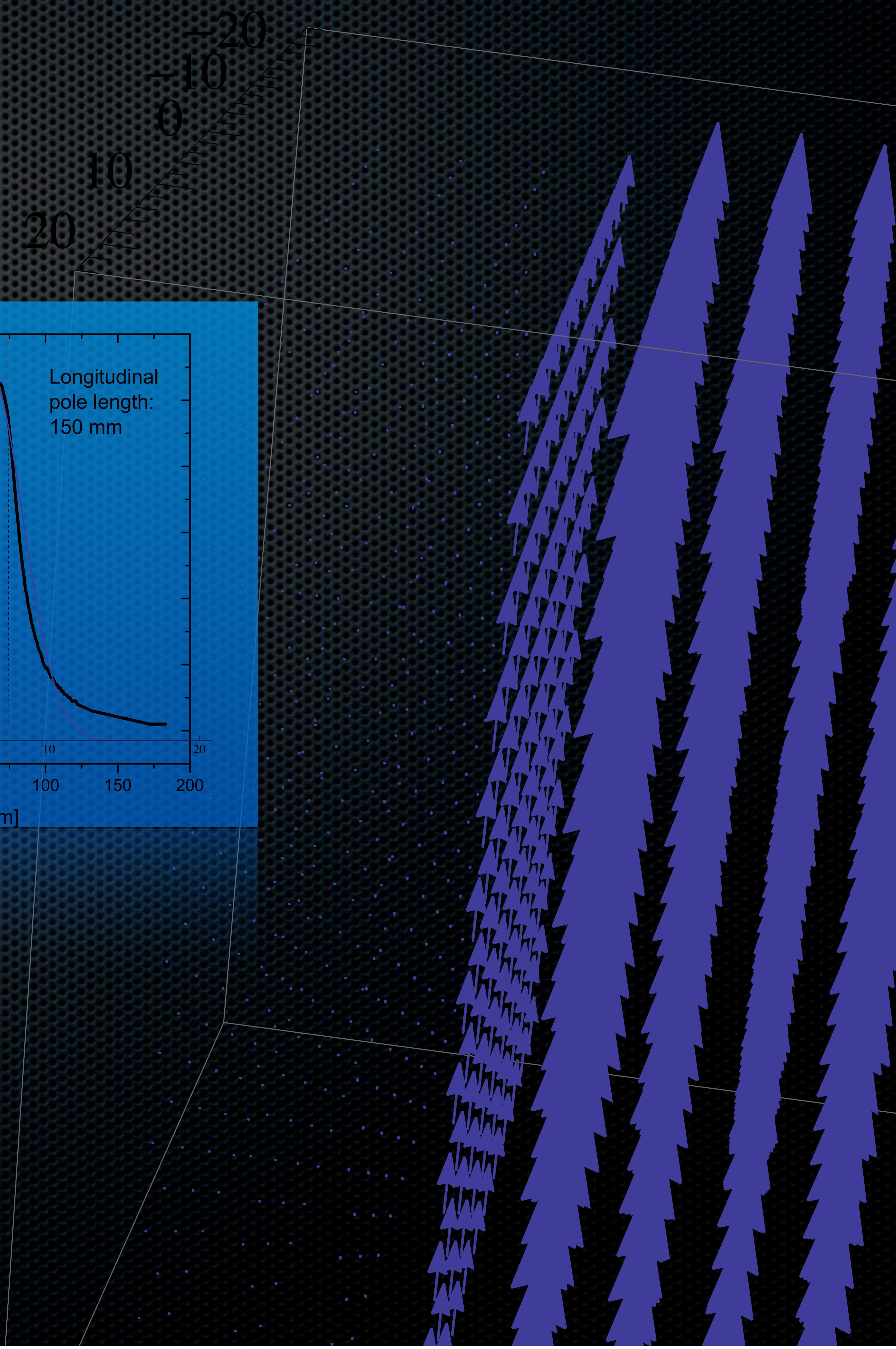
Magnetic Field Modelling

- Natural permanent magnets arranged to form a fairly homogeneous field along the longitudinal beam direction



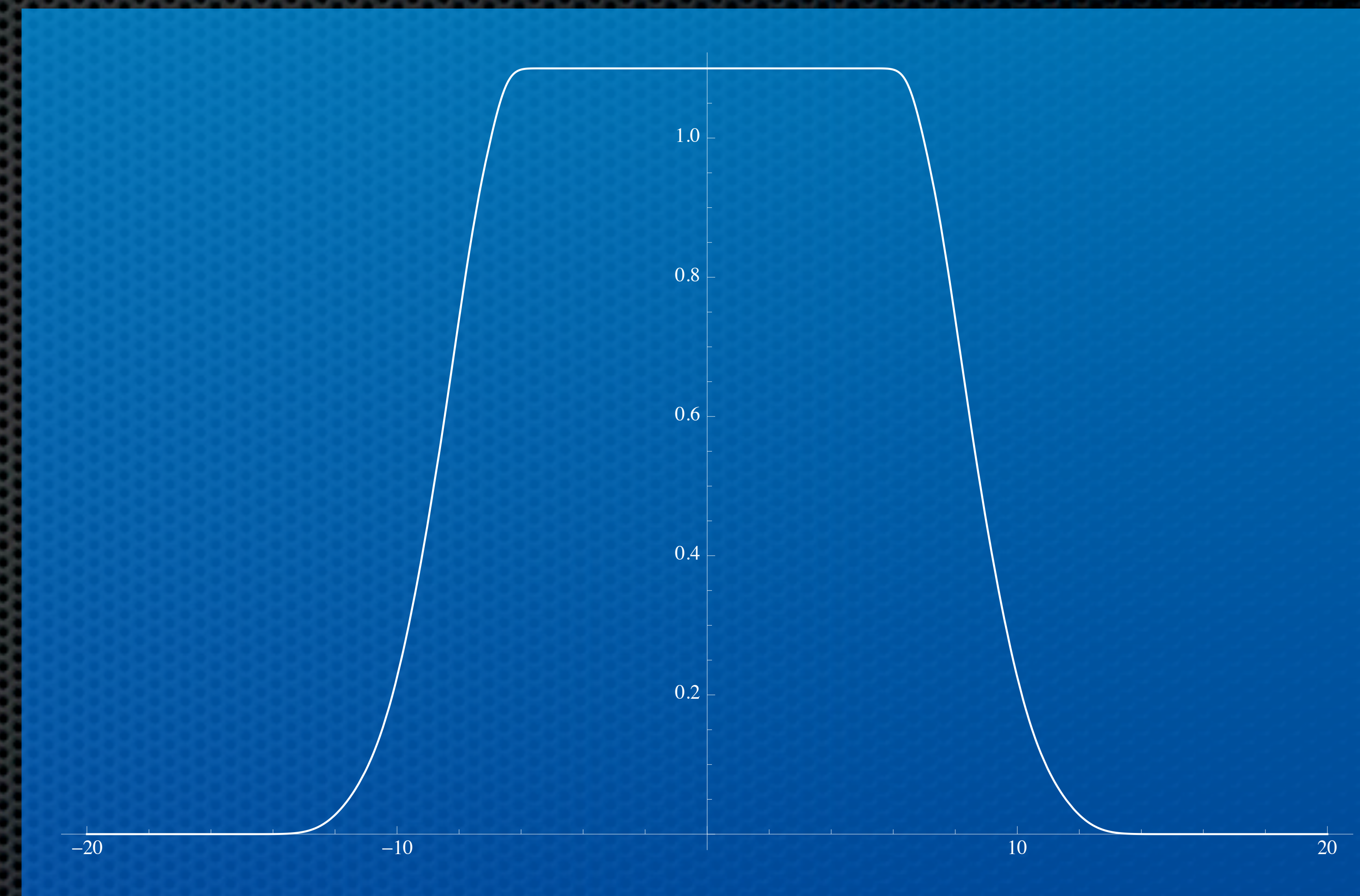
- Super simple model
  (flat top and gaussian tails)

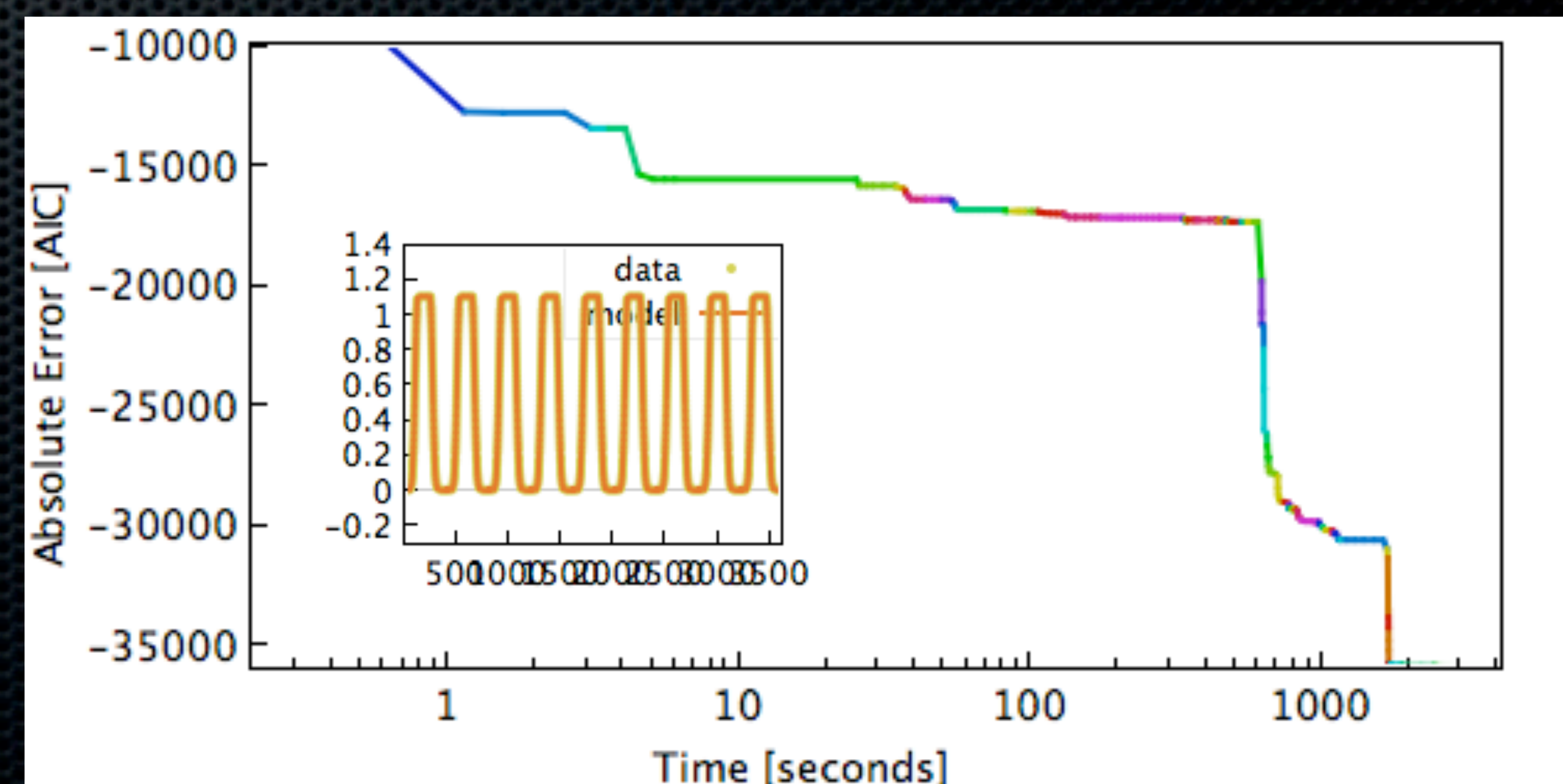
$$\vec{B}(z) = \begin{cases} (B_{z_0}, 0, 0)^{\mathrm{T}}, & \text{if } -60 < z < 60 \text{ mm} \\ (B_{z_0}\, e^{\frac{-(z-60\mathrm{mm})^2}{2c^2}}, 0, 0)^{\mathrm{T}}, & \text{if } z > 60 \text{ mm} \\ (B_{z_0}\, a^{\frac{-(z+60\mathrm{mm})^2}{2c^2}}, 0, 0)^{\mathrm{T}}, & \text{if } z < -60 \text{ mm} \end{cases}$$



Parameterisation with Eureqa

- To run track propagator on a GPU a potentially complicated magnetic field map must be parameterised to store in memory
- Powerful symbolic regression fitter
- Finds not only the fit coefficients but also the analytical function
- Example: fit the conditional function from the previous slide





Model:

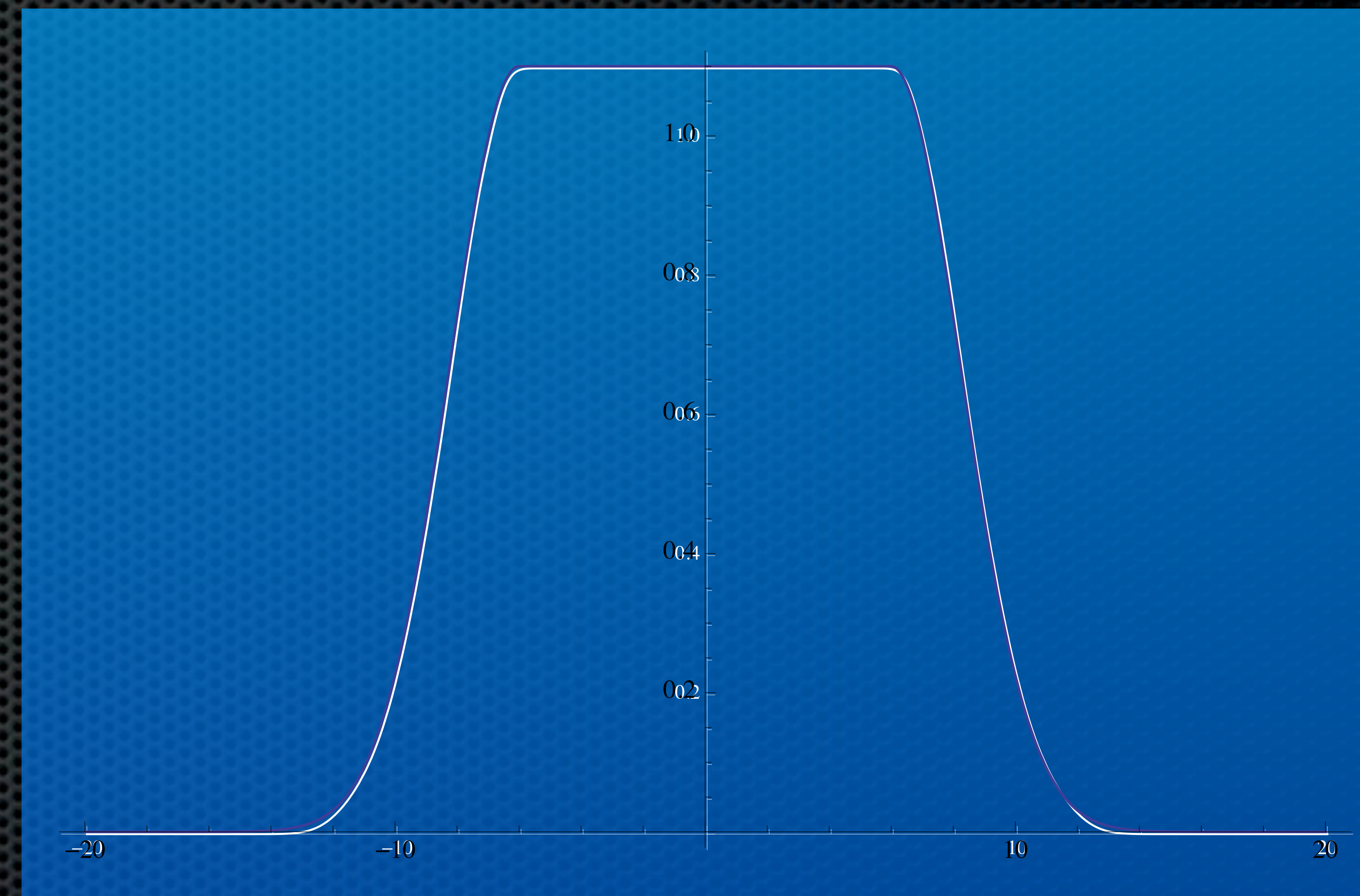
Bx = a gauss(gauss(b $\boldsymbol{z}^4$ + c ***z*** sin(d ***z***)) sin(e ***z*** + f $\boldsymbol{z}^3$ + g sin(sin(h ***z***))) - i $\boldsymbol{z}^3$ - j sin(sin(k ***z***)))

Text:

Bx = 1.10006*gauss(gauss(0.000924456*z^4 + 0.0953314*z*sin(0.824152*z))*sin(0.00114605*z + 0.00114605*z^3 + 0.141278*sin(sin(0.824152*z))) - 0.00114605*z^3 - 0.141278*sin(sin(0.824152*z)))

Parameterisation with Eureqa

- To run track propagator on a GPU a potentially complicated magnetic field map must be parameterised to store in memory
- Powerful symbolic regression fitter
- Finds not only the fit coefficients but also the analytical function
- Example case, fit the conditional function from the previous slide

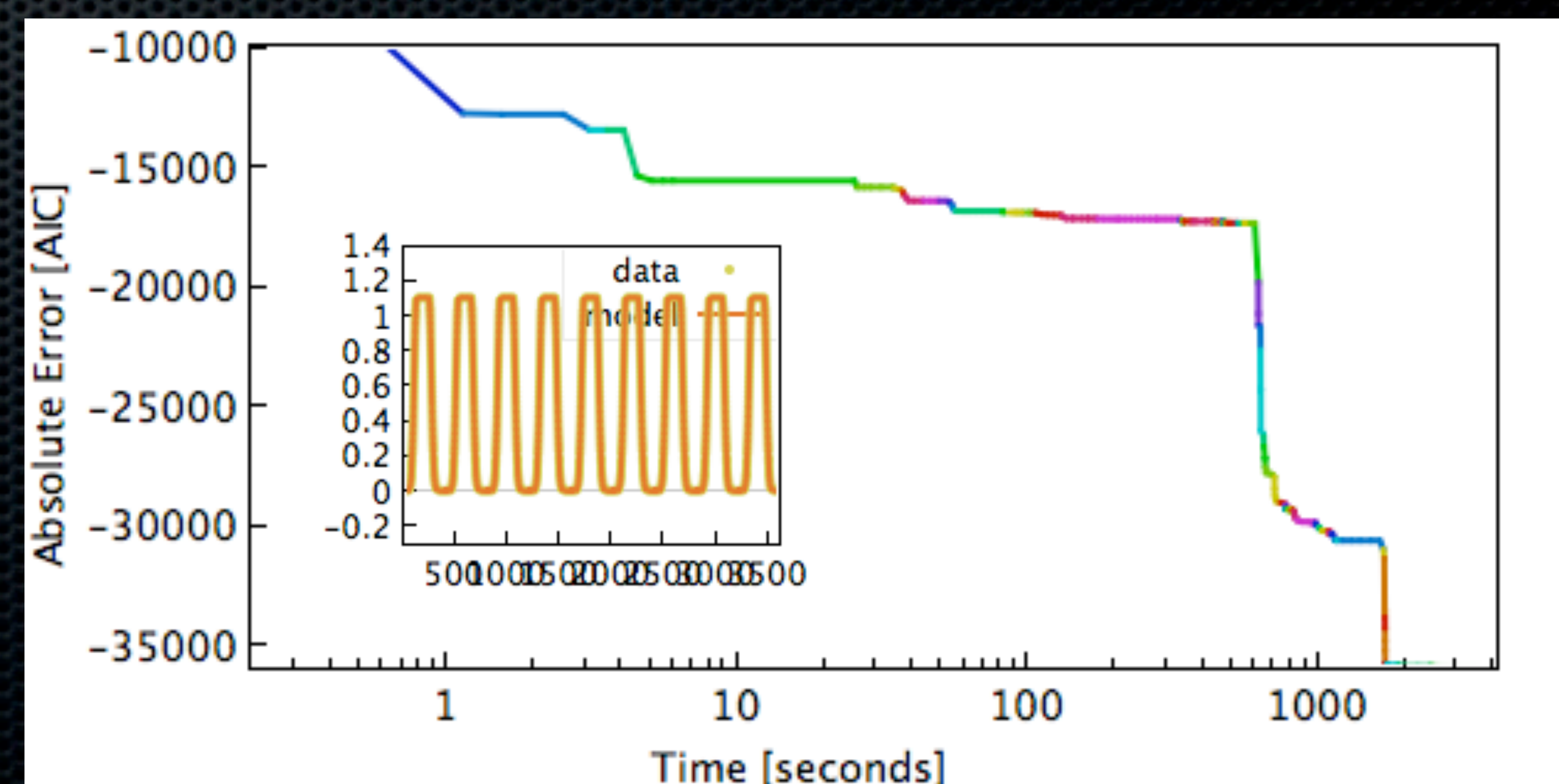


Model:

$\boldsymbol{Bx}$ = a gauss(gauss(b $\boldsymbol{z}^4$ + c $\boldsymbol{z}$ sin(d $\boldsymbol{z}$)) sin(e $\boldsymbol{z}$ + f $\boldsymbol{z}^3$ + g sin(sin(h $\boldsymbol{z}$))) - i $\boldsymbol{z}^3$ - j sin(sin(k $\boldsymbol{z}$)))

Text:

Bx = 1.10006*gauss(gauss(0.000924456*z^4 + 0.0953314*z*sin(0.824152*z))*sin(0.00114605*z + 0.00114605*z^3 + 0.141278*sin(sin(0.824152*z))) - 0.00114605*z^3 - 0.141278*sin(sin(0.824152*z)))



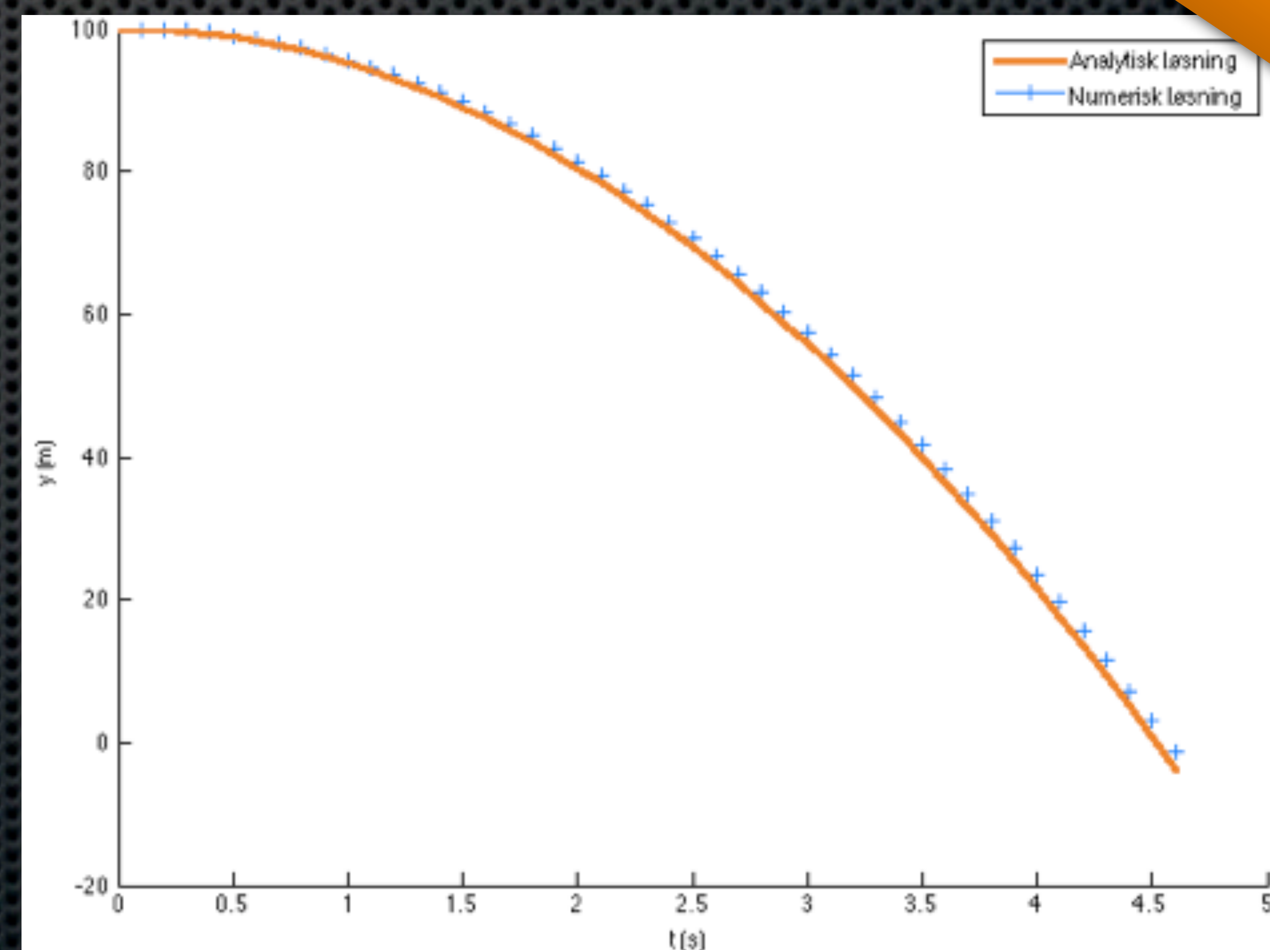
Scattering & Energy Loss

- Multiple Scattering
- Bremsstrahlung & Energy loss
- (Pair production?)

Differential Solvers

- Requirements from solvers
  - Precision
  - Speed/efficiency
- Tested versions
  - Euler
  - Runge-Kutta 4
  - Adaptive RK45

$$y(t + \Delta t) = y(t) + v(t)\Delta t$$
$$v(t + \Delta t) = v(t) - g\Delta t$$




```
g = -9.8;
y0 = 100; v0 = 0; % start
y = y0; v = v0; % initial
t = 0; dt = 0.1; % time
% Euler method
while y > 0 % stop at y=0
    t = t + dt;
    y = y + v .* dt;
    v = v + g .* dt;
    plot(t,y,'b+')
end

% Analytical Solution
t = 0:0.1:t;
y = y0 + v0 + (0.5 .* g .*
t.^2);
plot(t,y,'r')
```

Euler

- Simplest way to solve ODEs (Ordinary Differental Equations)

Runge-Kutta

Adaptive Runge-Kutta

Solver comparison

Detection Modelling

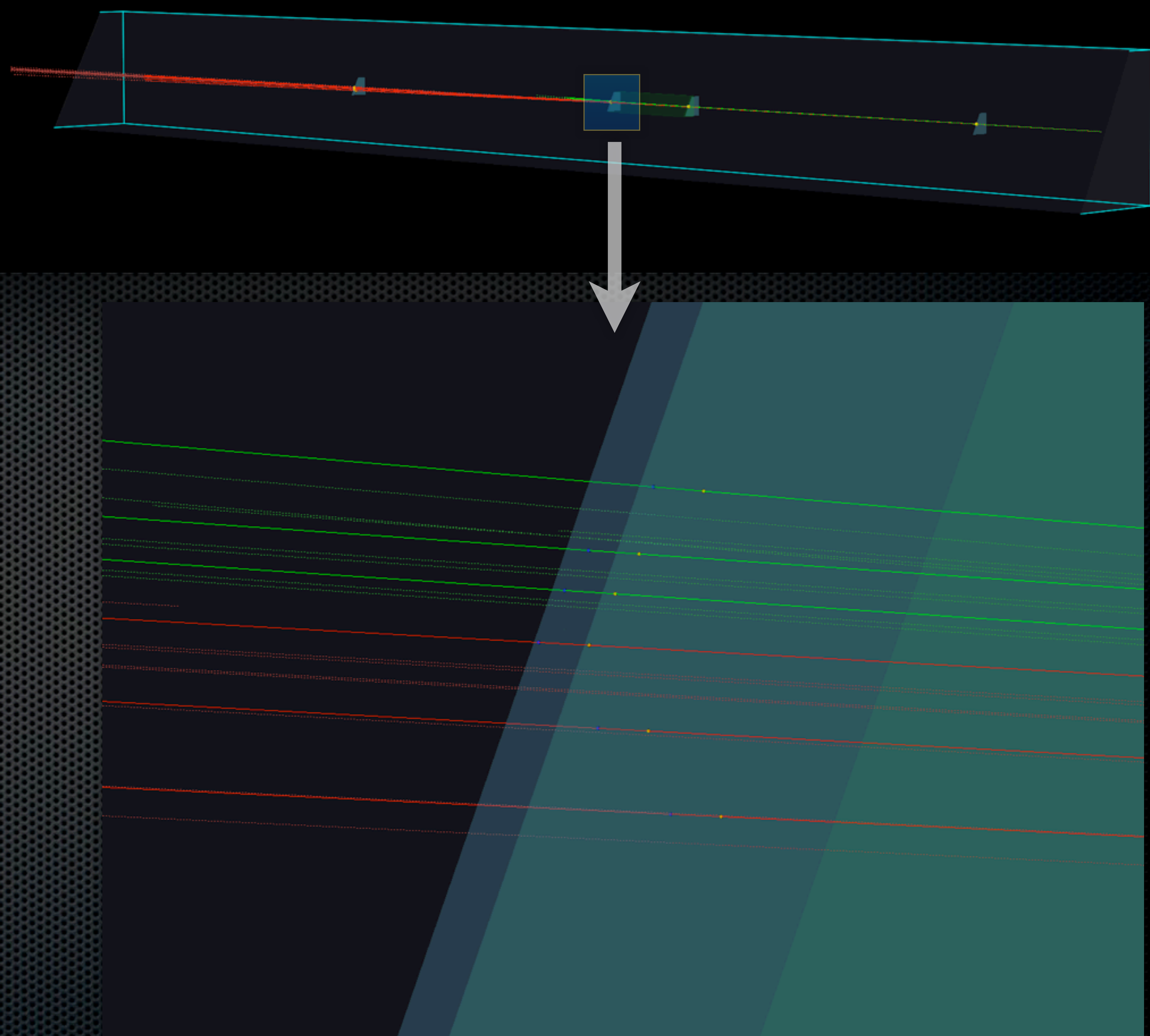
- Detector efficiencies
- Noise
- Resolution
- Alignment

Track Reconstruction

- Fitting with progressive-recursive filters i.e. Kalman Filters
  - Classical (linear) Kalman filters
  - Extended (locally linear) Kalman filters
  - Unscented (non-linear) Kalman filters
- Gaussian Sum Filters
- Deterministic Annealing Algorithms
- Forward-Backward tracking (smoothing)

Errors

- The devilish details, estimating error matrices for the Kalman filter
- Notice distance between dotted (fitted) and solid (truth) tracks to the right...



Analysis & Data