Fourier analysis to improve calorimeter fast simulation: The 6 Seasons Model

By Francis Beckert

What is FastSim and why does it exist?

Simulating entire particle showers with Gauss is extremely computationally intensive

- >50% of CPU time taken up by calorimeter simulation

FastSim seeks to address this by storing some data + properties from the full simulation

- Emulates the results from the full simulation without running the full particle-matter interactions

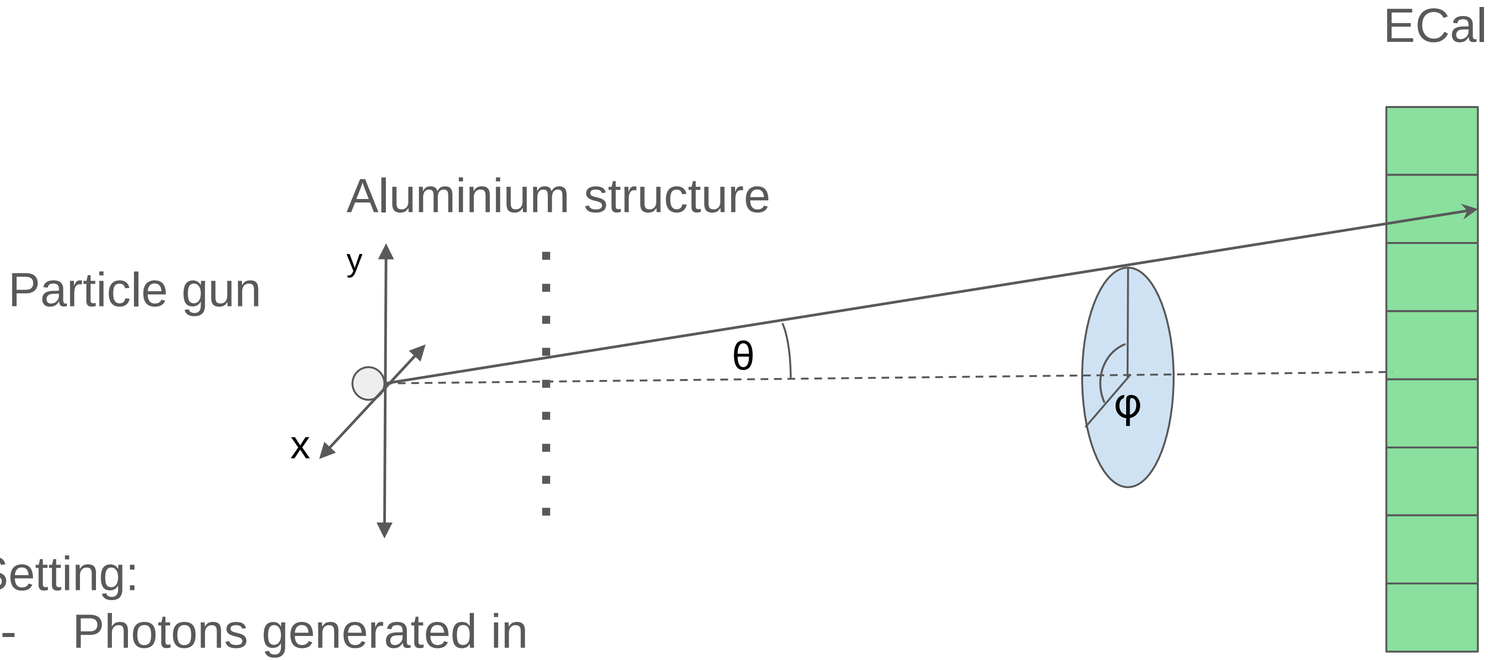
Why is building a good FastSim for Calo hard?

Limited data storage

- n^5 different input combinations for particle gun alone (n = number of bins per dimension)
- Up to 6 different particle types (photon, e+/-, pi+/-, K+/-, p, n)
- Needs to be loaded on every job

Needs to be *FAST*

- Must be significantly faster than full simulation to warrant slight decrease in data quality



Setting:

- Photons generated in front of spd
- Run2 Calo

(Not to scale!)

Particle gun setting: Generating showers

Properties of a shower:

$$E_i = \text{deposited energy in } i\text{th cell}$$

$$x_i = x\text{-coordinate of } i\text{th cell}$$

$$y_i = y\text{-coordinate of } i\text{th cell}$$

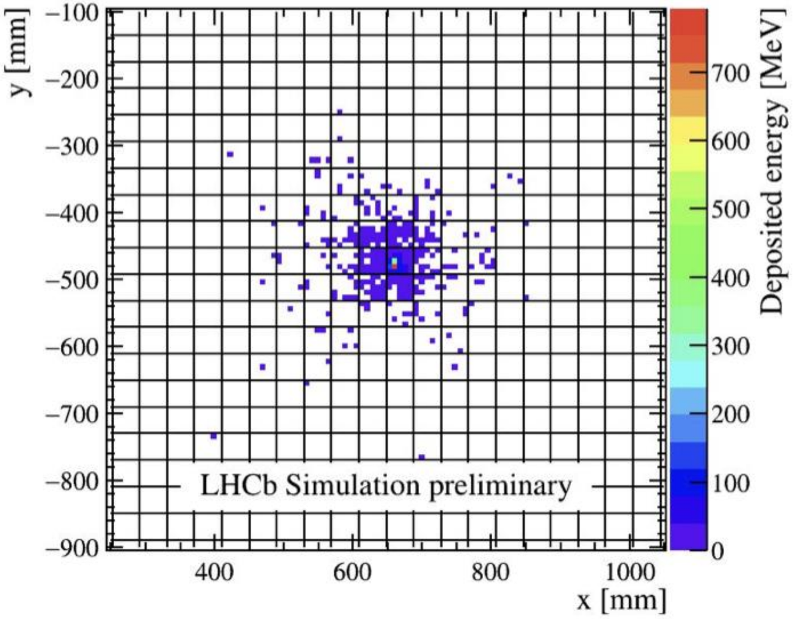
$$x_{\text{clust}} = \frac{\sum_i E_i \cdot x_i}{\sum E_i}$$

$$y_{\text{clust}} = \frac{\sum_i E_i \cdot y_i}{\sum E_i}$$

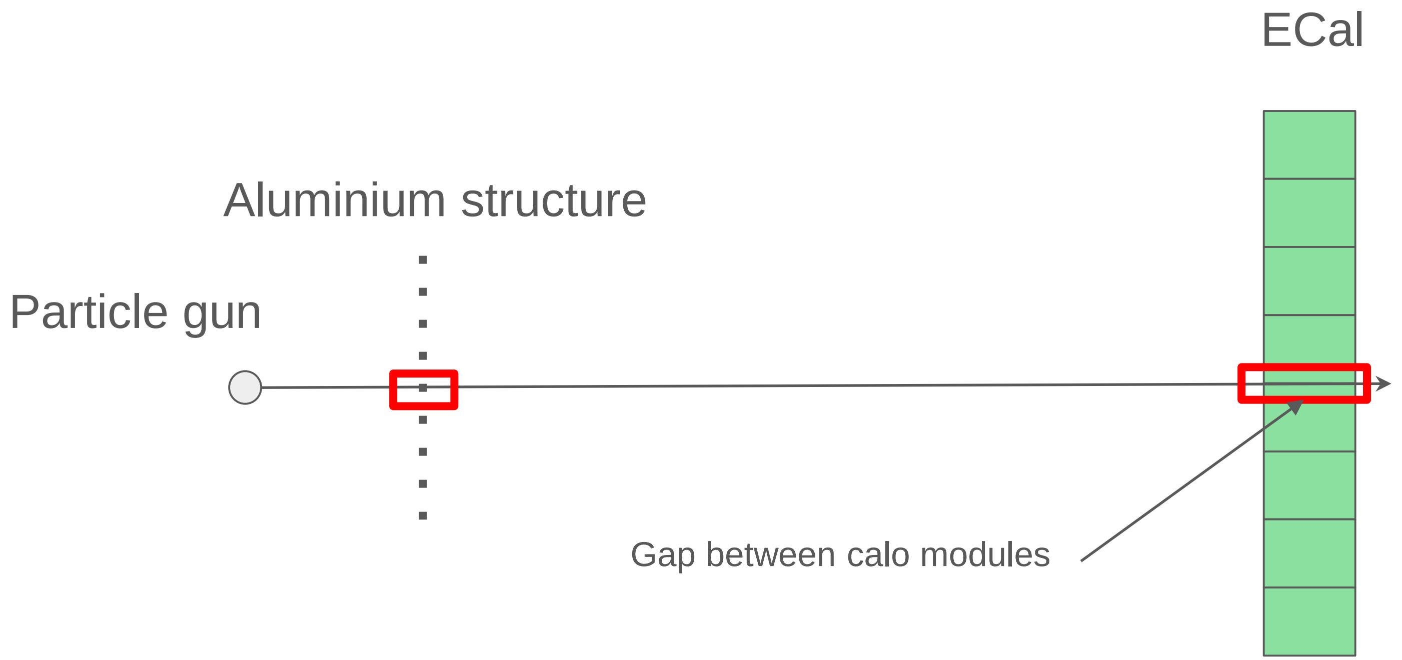
$$\sigma(y_{\text{clust}}) = \sqrt{\frac{\sum_i E_i (y_i - y_{\text{clust}})^2}{\sum_i E_i}}$$

$$\sigma(x_{\text{clust}}) = \sqrt{\frac{\sum_i E_i (x_i - x_{\text{clust}})^2}{\sum_i E_i}}$$

$$\boxed{E_{clust} = \sum_i E_i}$$

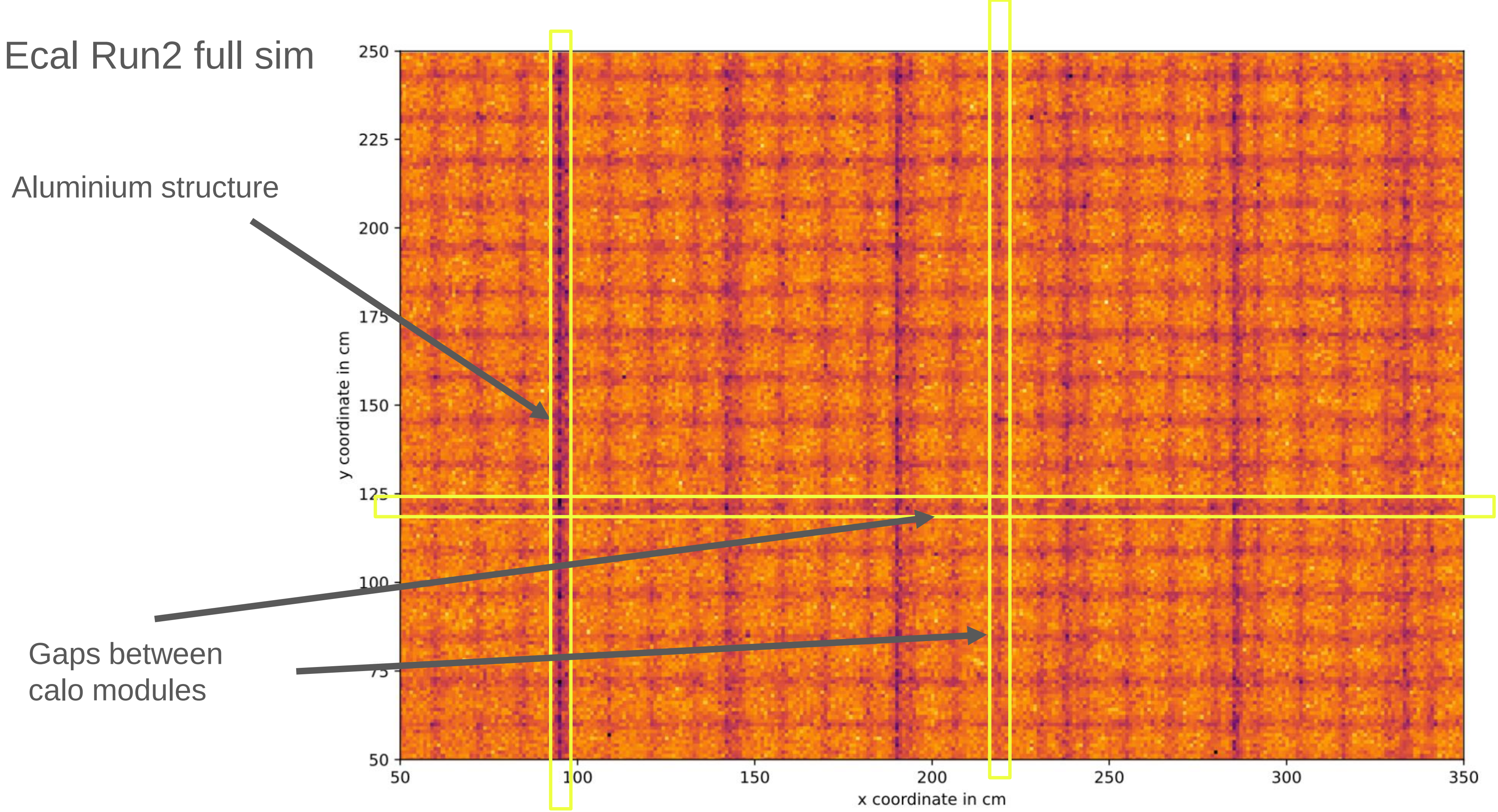


Time for abusive notation: we will say E = E_clust


ECal
Aluminium structure
Particle gun
Gap between calo modules

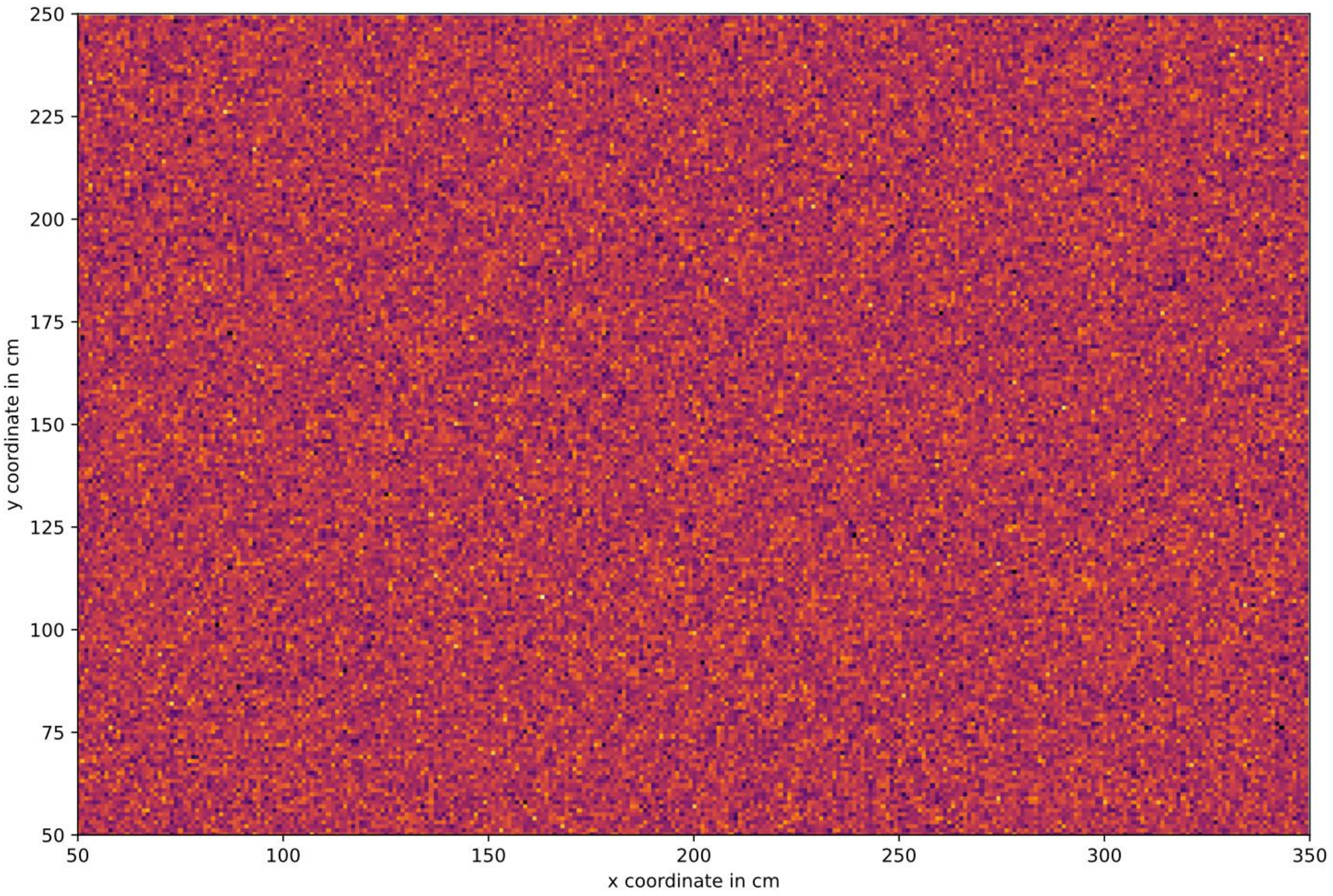
(Not to scale!)

Ecal Run2 full sim

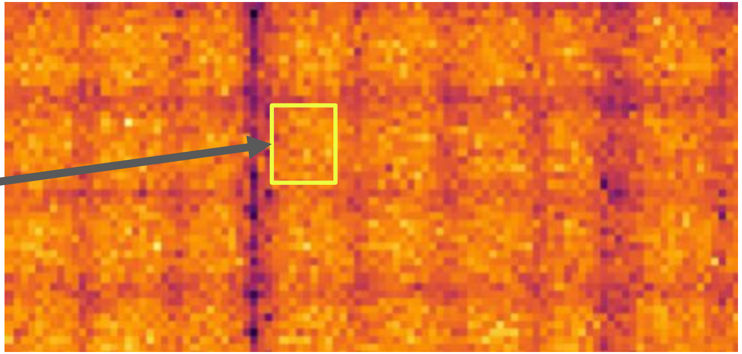


Issue with current version of fast sim:

Everything is uniform

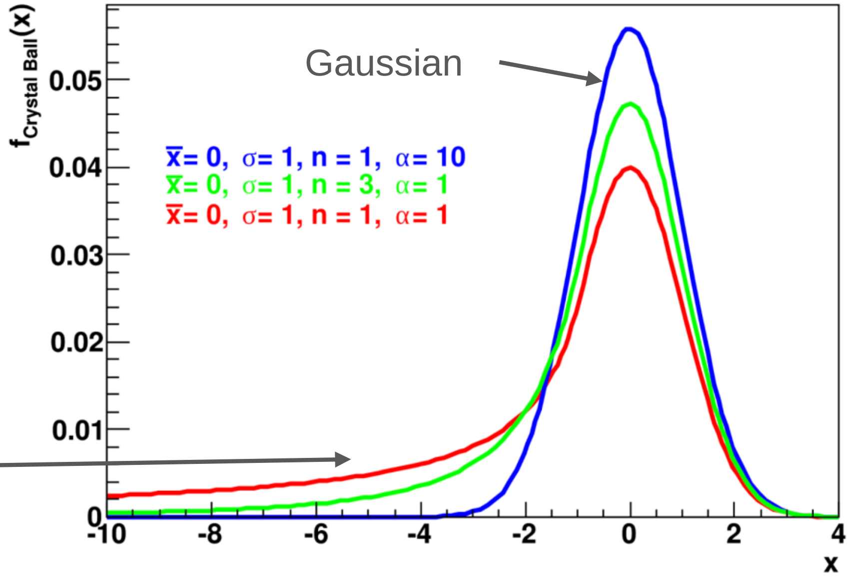


Statistical fluctuations in uniform regions



log(E) modeled by crystal ball function:

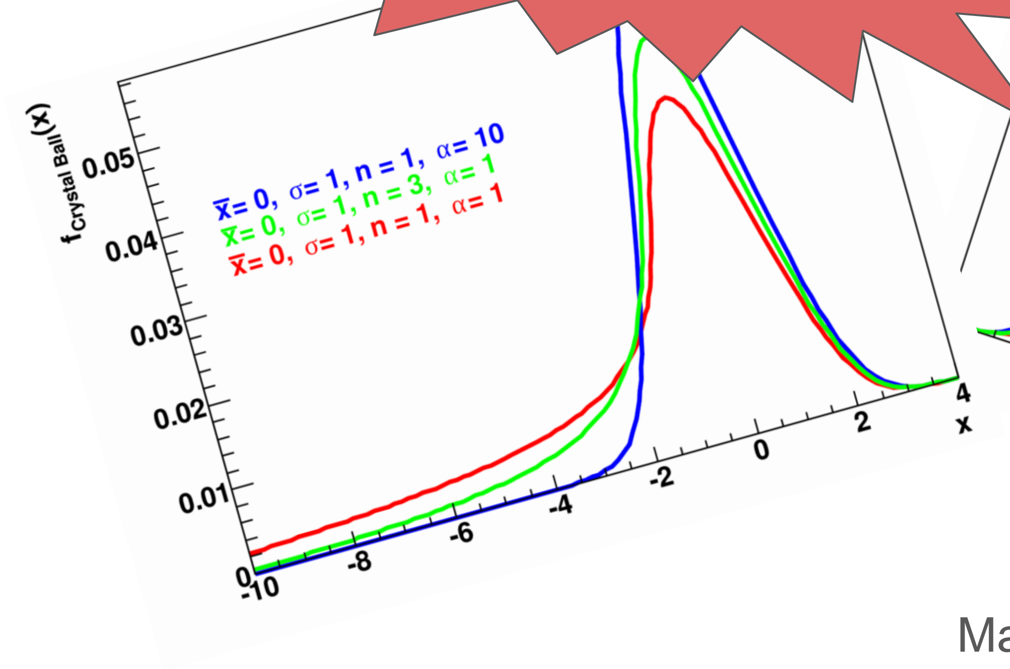
$$f(x;\alpha,n,\bar{x},\sigma) = N \cdot \begin{cases} \exp(-\frac{(x-\bar{x})^2}{2\sigma^2}), & \text{for } \frac{x-\bar{x}}{\sigma} > -\alpha \\ A \cdot (B - \frac{x-\bar{x}}{\sigma})^{-n}, & \text{for } \frac{x-\bar{x}}{\sigma} \leqslant -\alpha \end{cases}$$

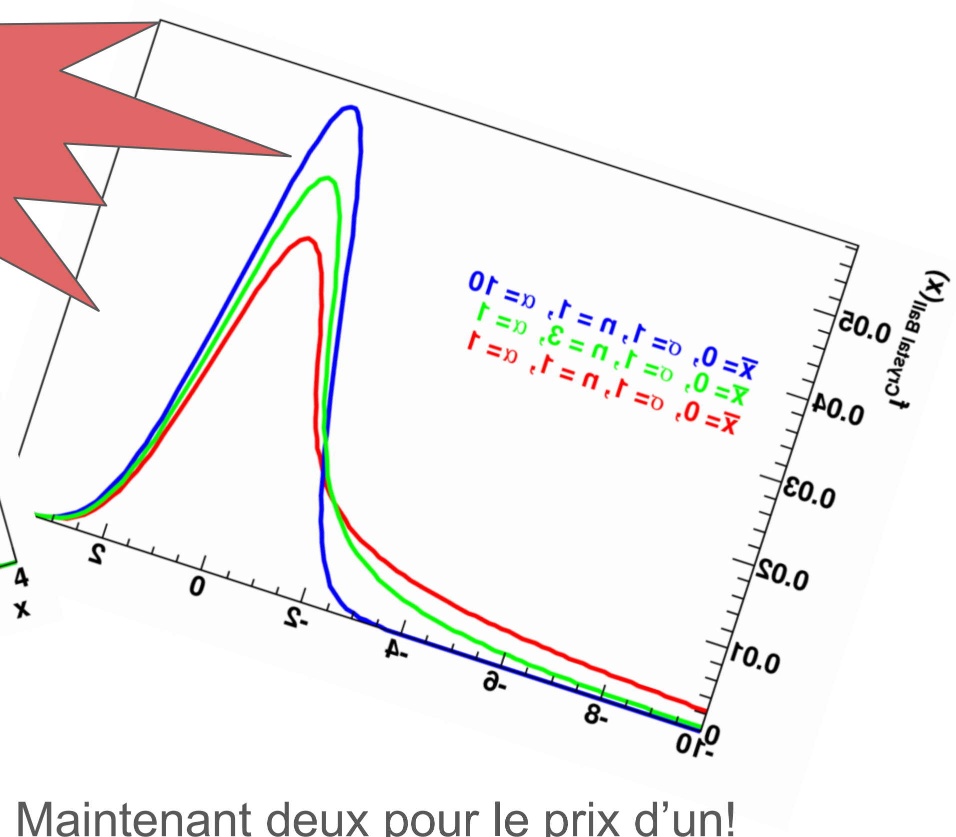


Power law tail

Actually, that’s a lie

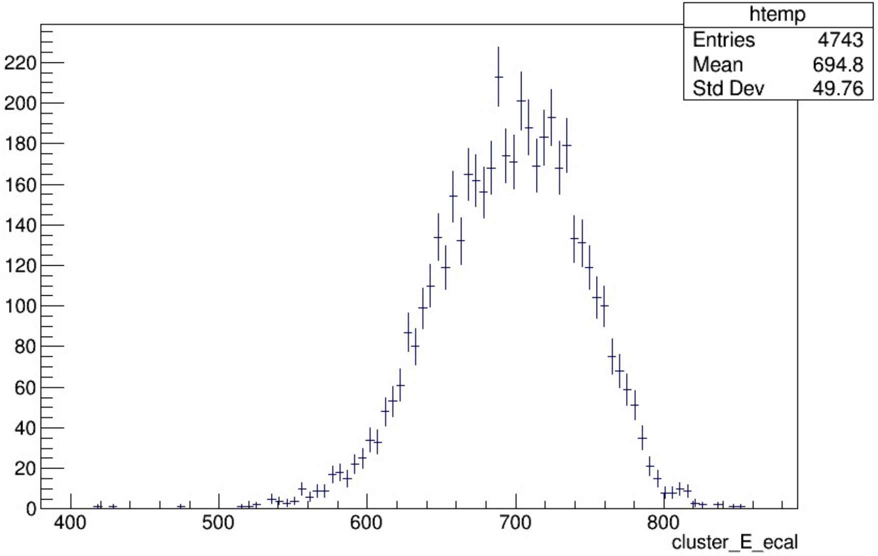
2 Crystal ball functions !!

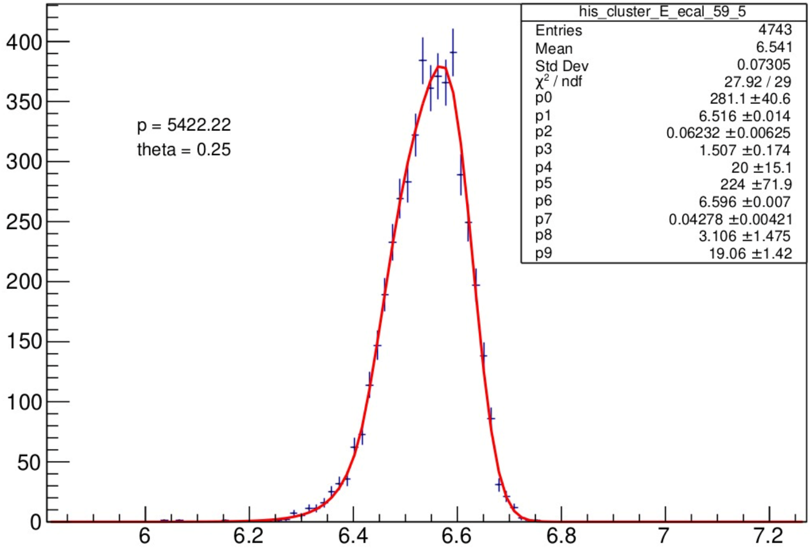




Maintenant deux pour le prix d’un!

Log transform E + fit
double crystal ball





Given (x, y, θ, φ) can we predict cluster Energy in Ecal?

Deterministic problem:

- How can we accurately quantify and model the non-uniform regions of the calorimeter?
- How does varying each input parameter affect the expected cluster energy?

Probabilistic problem:

- Since the cluster energy is determined by a stochastic process, it is also random and follows some distribution
- How can we model this distribution?
- How do the deterministic effects influence this distribution?

Approaches

1. Linear neural network with sigmoid activation layer

Pros:

- Captures linear relations between the inputs
- Sigmoid function allows us to learn the binary relation between inputs and non-uniformities

Cons:

- Small networks fail to learn the complex patterns
- Large networks are … well too large
- Difficult to build an optimizer for data with stochastic component

Approaches

1. Linear neural network with sigmoid activation layer

2. RandomForest regressor

Pros:

- Can encode the decision points which indicate whether or not we are in a non-uniform region

Cons:

- To understand the complex relationships between 4 variables a large number of trees is required

Approaches

1. Linear neural network with sigmoid activation layer

2. RandomForest regressor

3. BallTree + Nearest Neighbors

Pros:

- Fast predictions
- Does not need to encode relationships between inputs

Cons:

- Model size scales linearly with train data set size (and we need >100M data points to get accurate statistics in 4 dims)

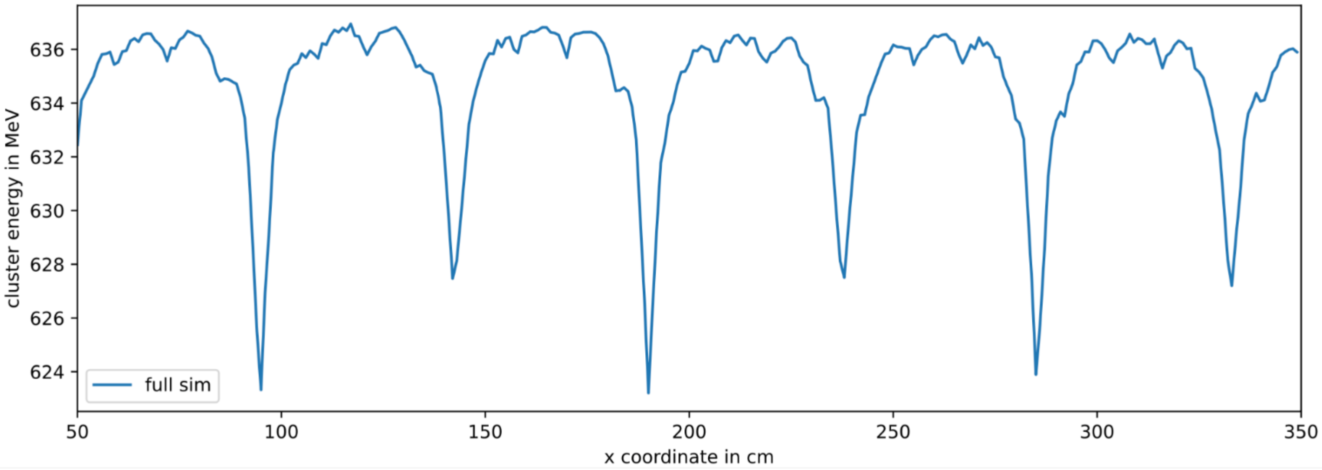
Approaches

1. Linear neural network with sigmoid activation layer

2. RandomForest regressor

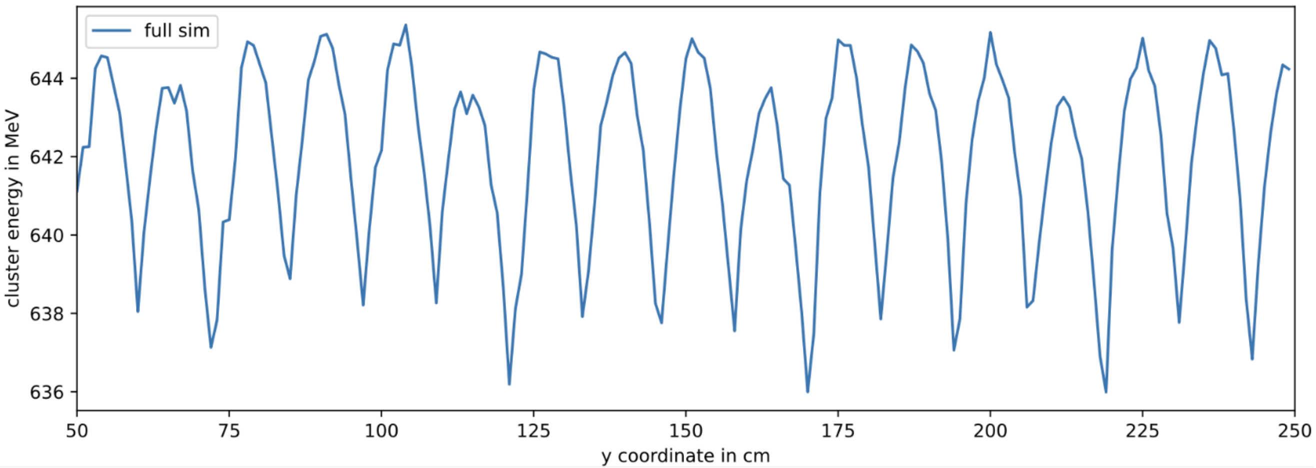
3. BallTree + Nearest Neighbors

Common problem: Either too big or too inaccurate x:



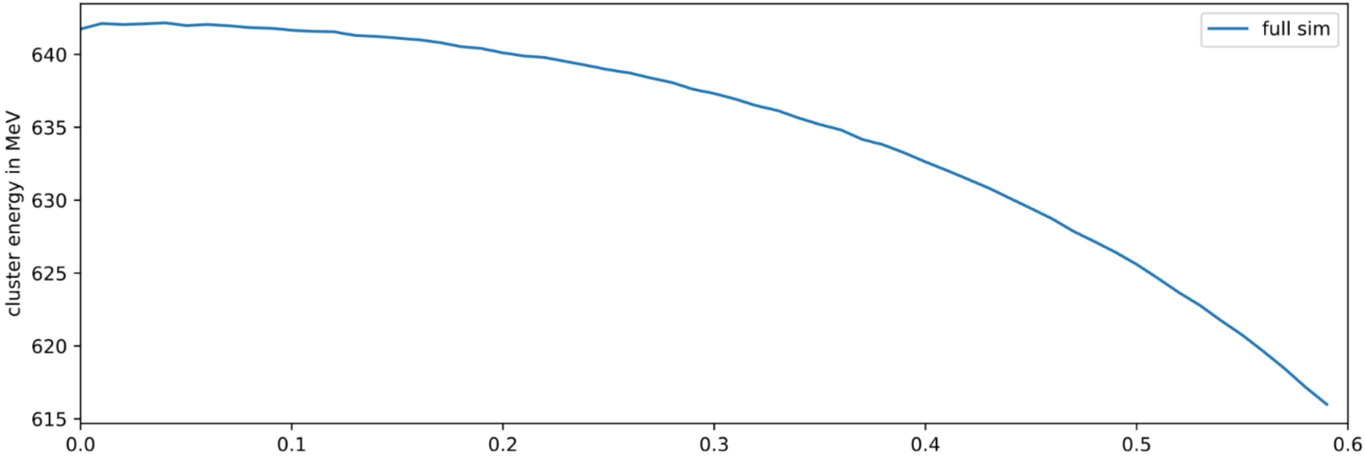
636
634
632
630
628
626
624
cluster energy in MeV
full sim
50
100
150
200
250
300
350
x coordinate in cm

y:



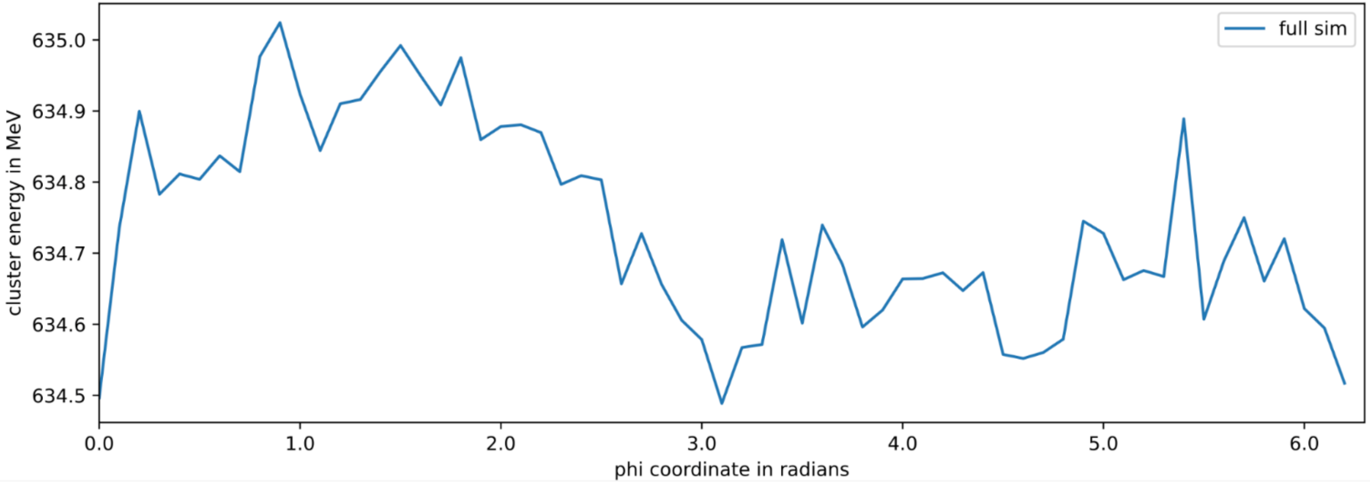
full sim
644
642
640
638
636
cluster energy in MeV
50
75
100
125
150
175
200
225
250
y coordinate in cm

θ:



full sim
cluster energy in MeV
640
635
630
625
620
615
0.0
0.1
0.2
0.3
0.4
0.5
0.6

φ:



full sim
cluster energy in MeV
635.0
634.9
634.8
634.7
634.6
634.5
0.0
1.0
2.0
3.0
4.0
5.0
6.0
phi coordinate in radians

Can we reduce this to a linear problem?

Solution: The 6 Seasons Approach

Data:

$(x,y,\theta,\varphi) \rightarrow E$

x100M physics events

1. Bin data

Bin to 2D "image" for each coordinate pair

For example: 400 x-bins and 300 y-bins, for each bin calculate the avg cluster E

- X vs Y
- X vs θ
- X vs φ
- Y vs θ
- Y vs φ
- θ vs φ

2. Fourier filter

2D Fast Fourier Transform

Filter all signals weaker than $n{\cdot}\sigma$

Save significant signals in sparse format

>5x data compression per coordinate pair bin + additive along input dims

3. Linear Regression

Reconstruct each coordinate pair with inverse Fourier Transform

6-vector of cluster energies **v**

Linear Regression on **v +** (x,y,θ,φ) against E

Assumptions we are making:

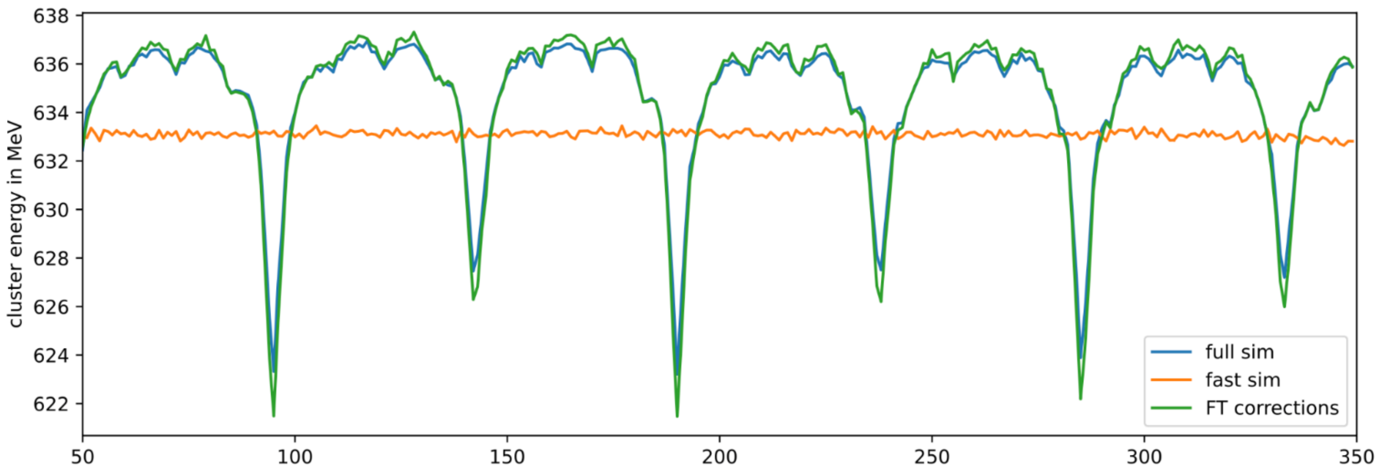
1. The periodicity in the data is additive (i.e. the effects of non-uniformities add)

1. The data is additive across coordinate pairs

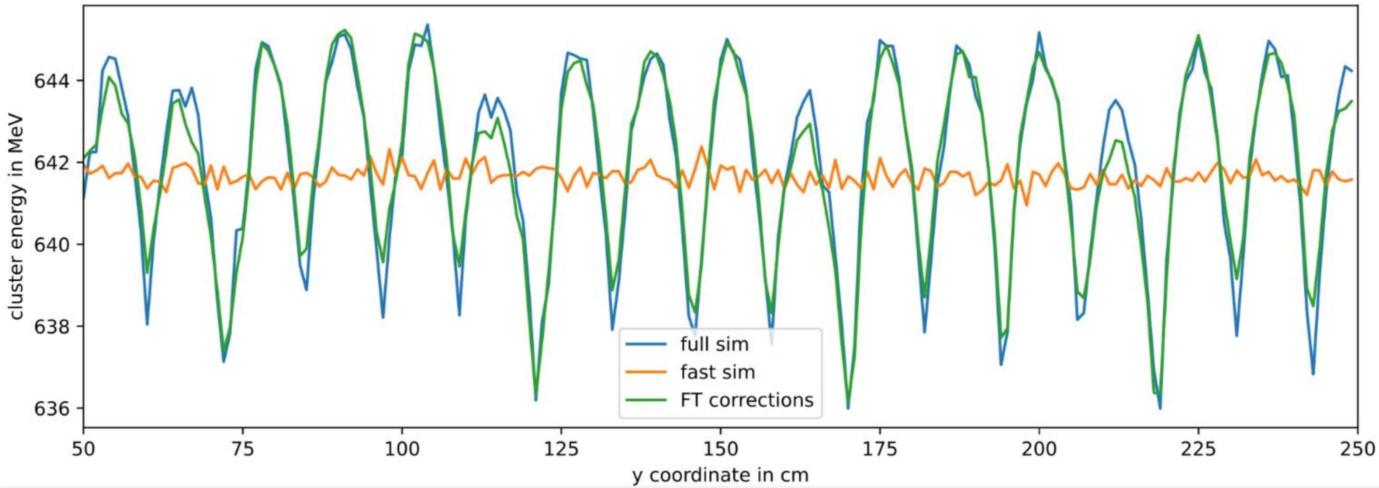
1. The fourier filtered outputs are linearly correlated with cluster energy

Results:

x:

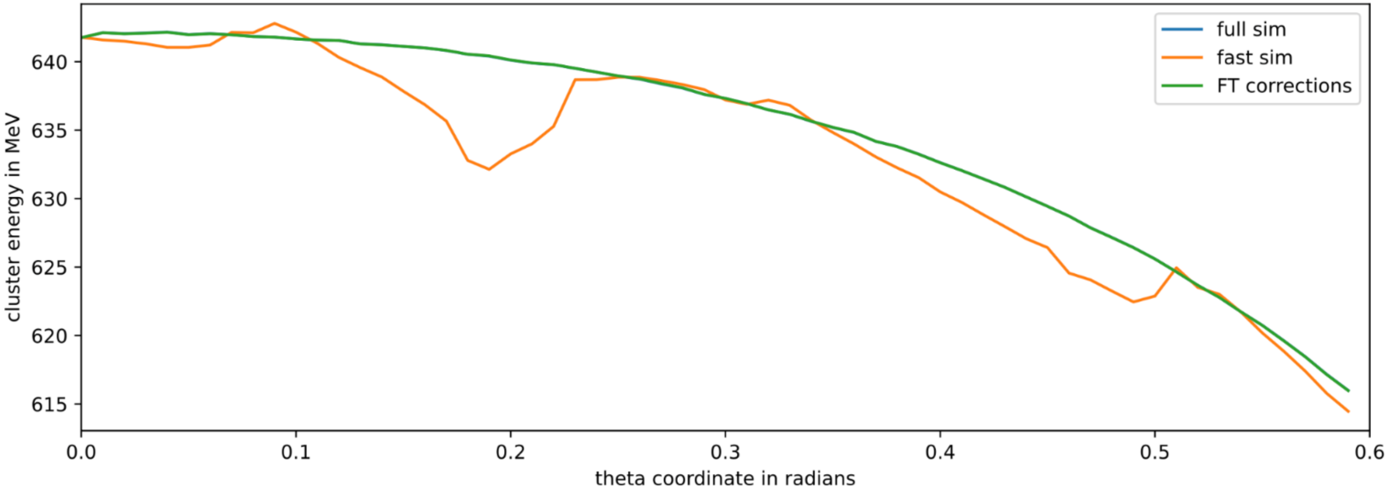

cluster energy in MeV
638
636
634
632
630
628
626
624
622
50
100
150
200
250
300
350
full sim
fast sim
FT corrections

y:


cluster energy in MeV
644
642
640
638
636
50
75
100
125
150
175
200
225
250
y coordinate in cm
full sim
fast sim
FT corrections

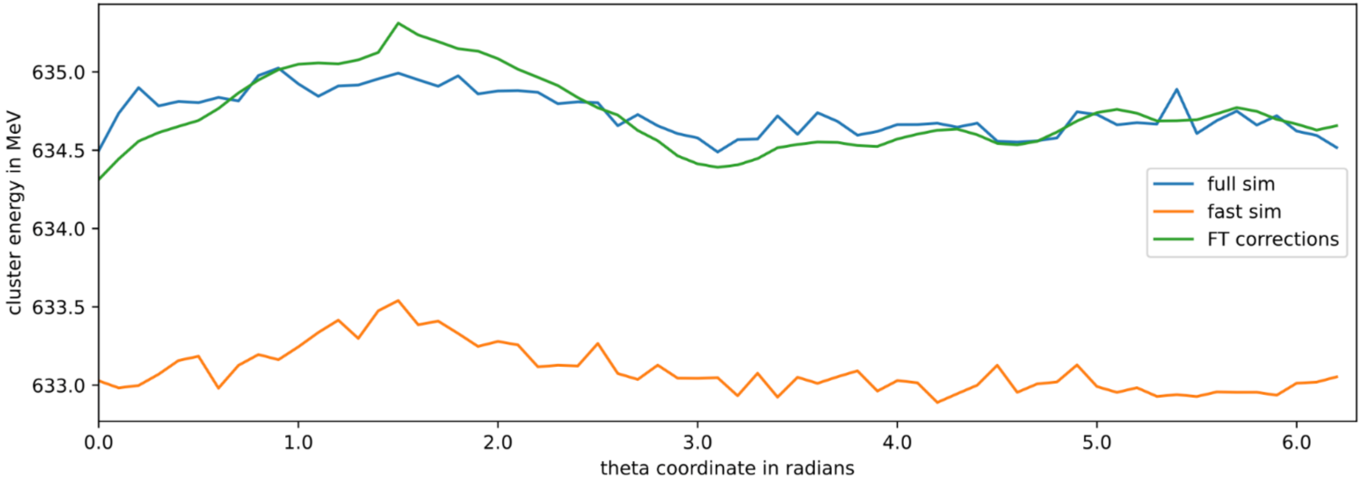
Results:

θ:



full sim
fast sim
FT corrections
cluster energy in MeV
640
635
630
625
620
615
0.0
0.1
0.2
0.3
0.4
0.5
0.6
theta coordinate in radians

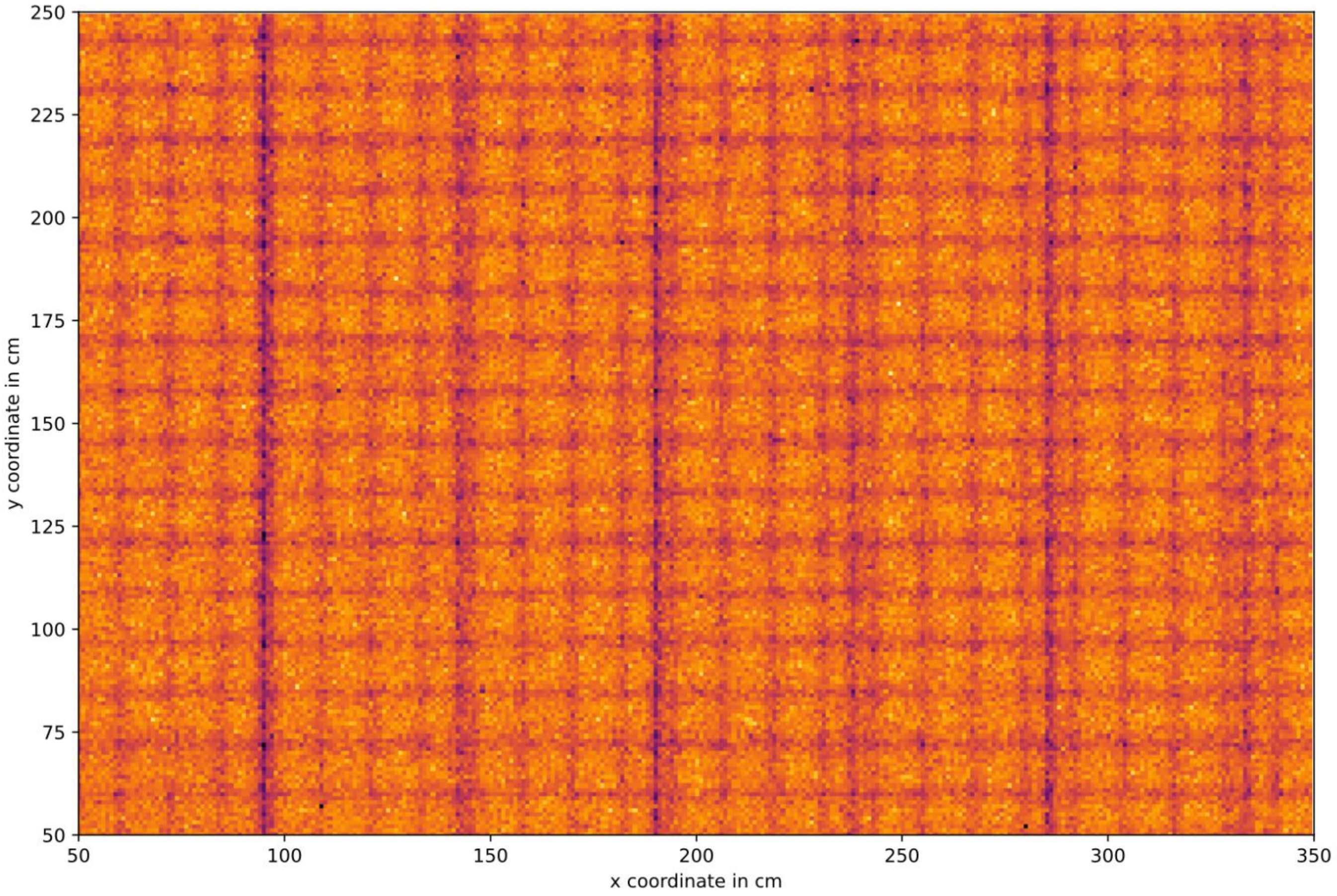
φ:



full sim
fast sim
FT corrections
cluster energy in MeV
635.0
634.5
634.0
633.5
633.0
0.0
1.0
2.0
3.0
4.0
5.0
6.0
theta coordinate in radians

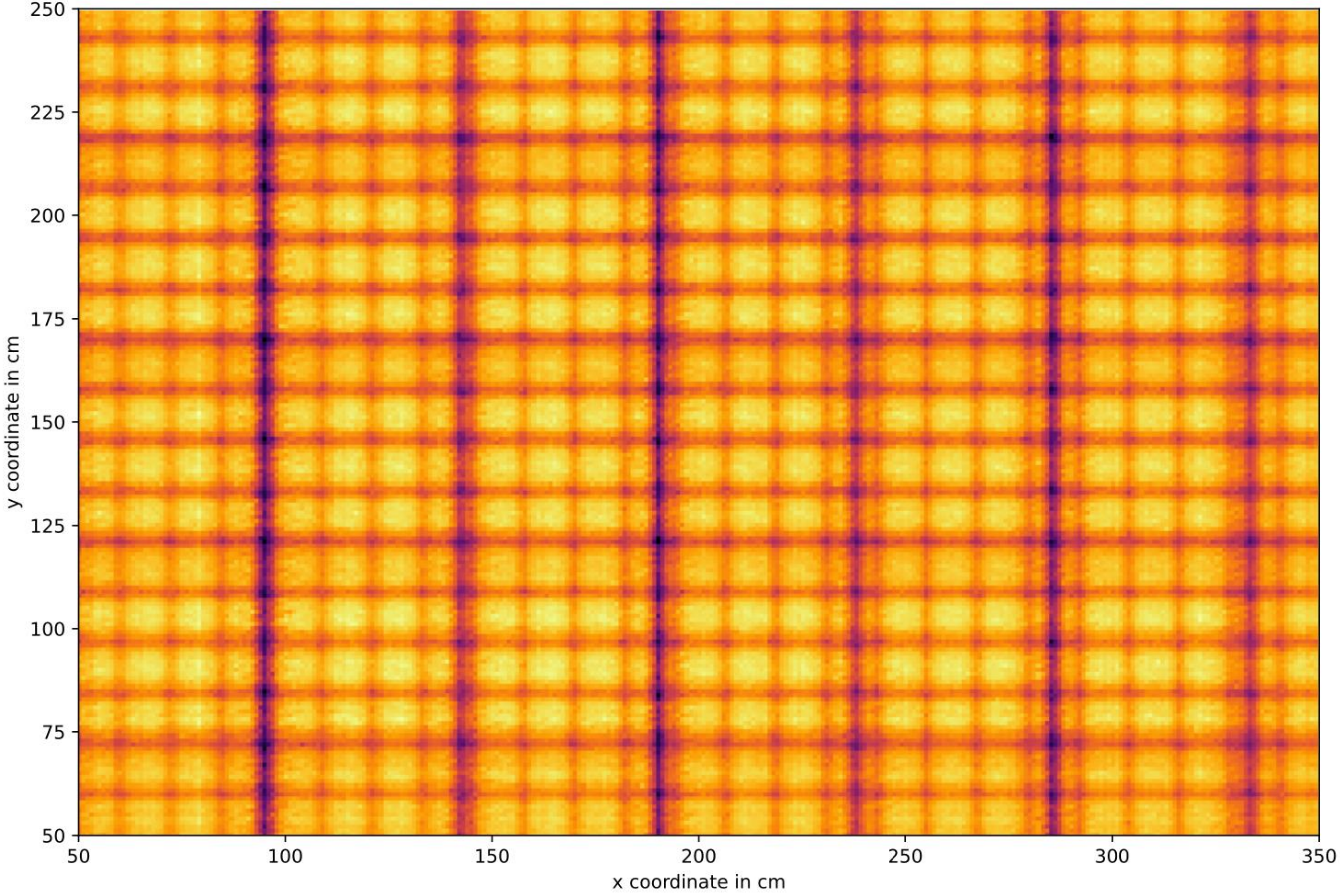
Results:

Full Sim:
X vs Y with
θ<0.05



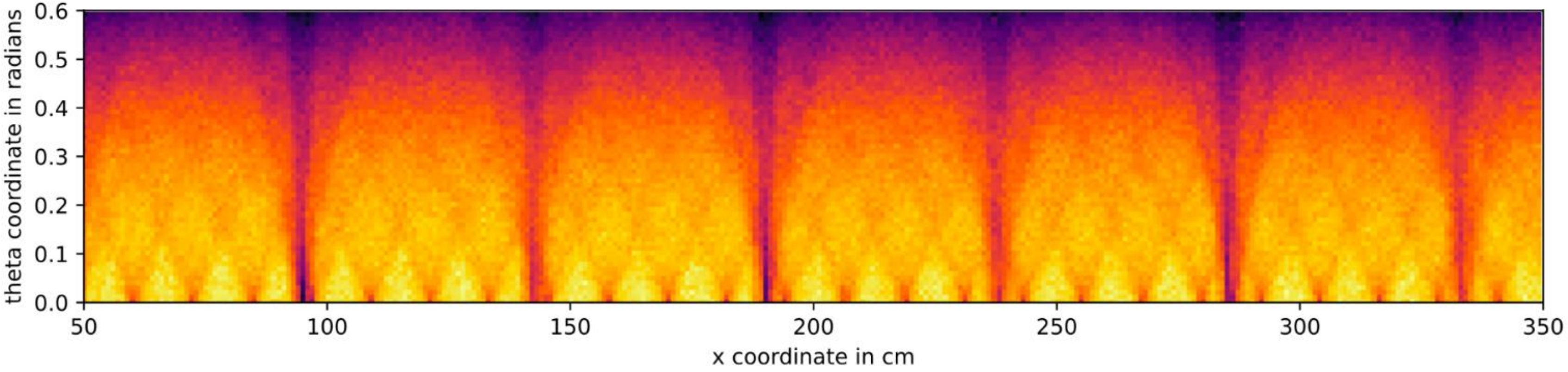
Results:

FT corrections:
X vs Y with
θ<0.05

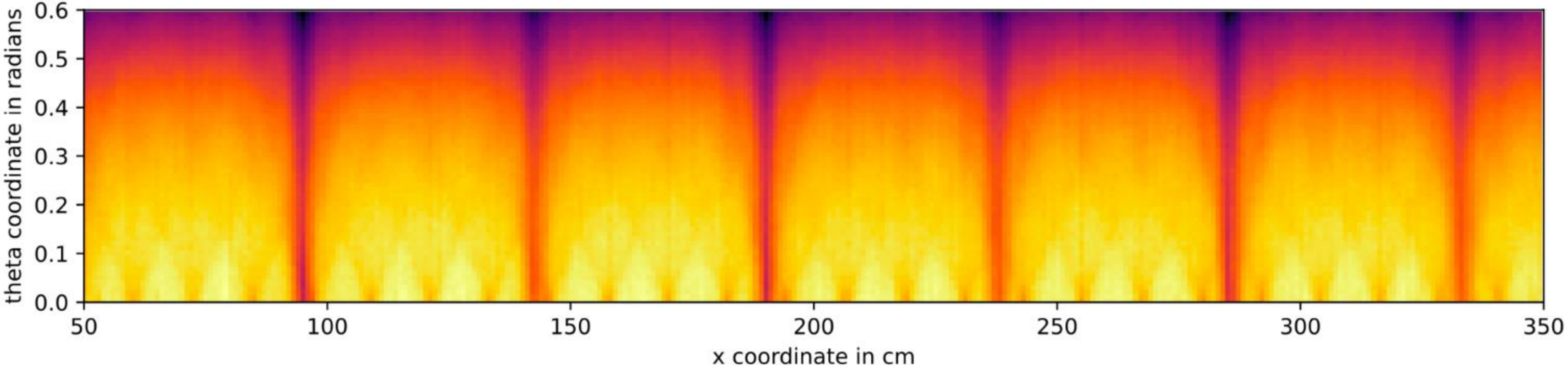


Results:

x vs θ



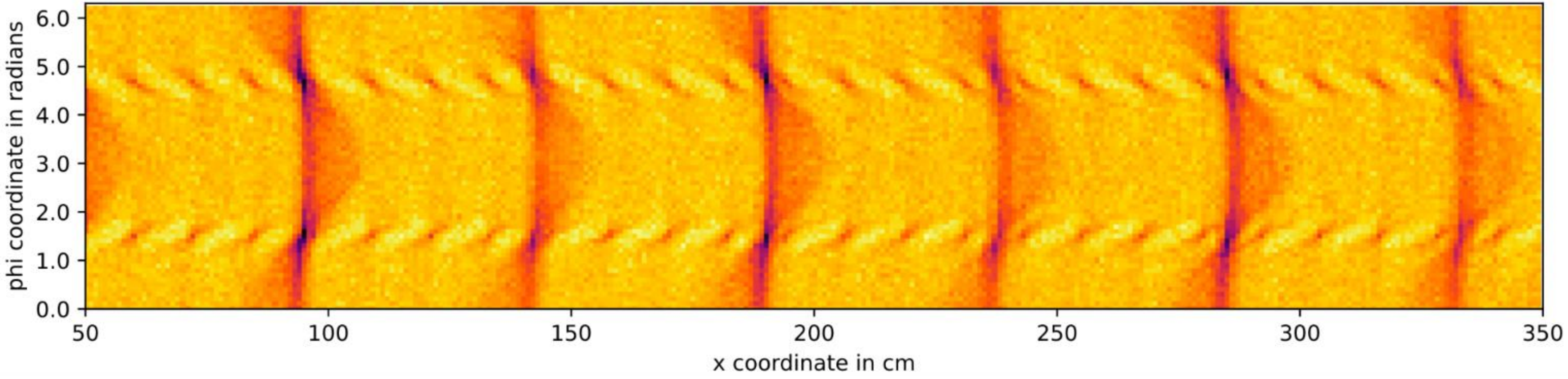
Full Sim Data



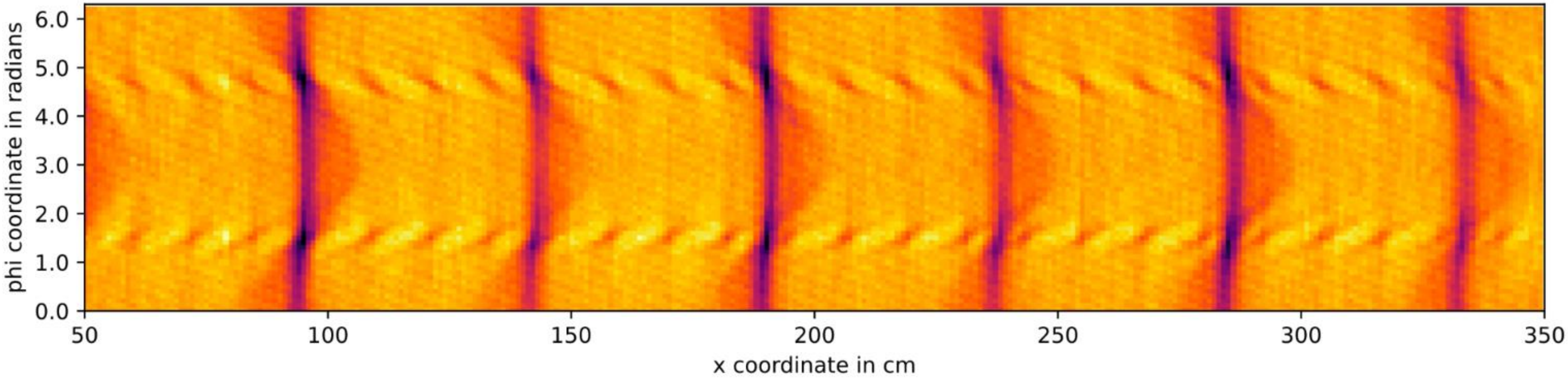
FT corrections

Results:

x vs φ



Full Sim Data



FT corrections

Real Results (not just pictures) :

$$\text{RMS per bin} = \sqrt{\sum_i b_i^2}$$

RMSE per bin: **1.616 MeV**

$$b_i = \int_{B_i} (E(x,y,\theta,\phi) - \hat{E}(x,y,\theta,\phi))d\lambda$$

$$B_i = (x_i, x_i + l_x) \times (y_i, y_i + l_y) \times (\theta_i, \theta_i + l_\theta) \times (\phi_i, \phi_i + l_\phi)$$

$$\lambda = \text{Lebesgue measure on } \mathbb{R}^4$$

Relative error per bin: **1.640 x 10^-3**
[RMSE/(max - min)]

RMSE per prediction: **50.877 MeV**

$$\text{RMS per prediction} = \sqrt{\sum_{x,y,\theta,\phi} (E(x,y,\theta,\phi) - \hat{E}(x,y,\theta,\phi))^2}$$

Standard deviation of fitted crystal ball distribution: **49.76 MeV**

Model Specs :

Size: 500 kb

Prediction speed: >70.000 predictions per second

Next Steps :

How do we apply the corrections?

- Deterministic: apply a correction based on some analytical expression to fast sim prediction
- Probabilistic: use some probability distribution to interpolate between the fast sim data and corrections (beta distribution with parameters determined by prediction values)

How does this scale with momentum?

- We determined an analytical expression for the scaling with momentum, but how does this scaling interact with the periodicity?

Acknowledgements:

My two wonderful supervisors!

Sergey Kholodenko

Matteo Rama

Thank you for your attention!

Any questions?