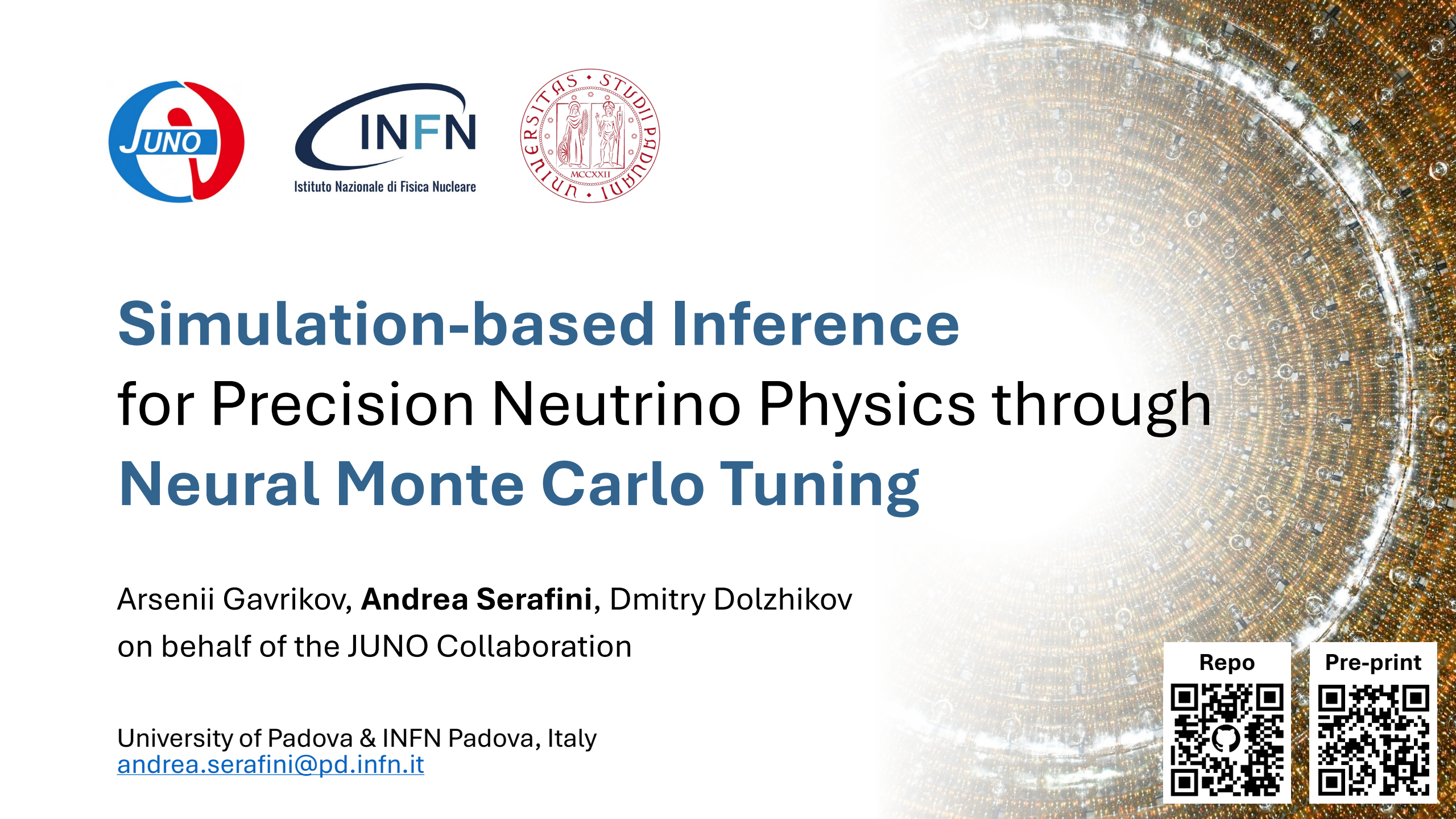


Simulation-based Inference for Precision Neutrino Physics through Neural Monte Carlo Tuning

Arsenii Gavrikov, **Andrea Serafini**, Dmitry Dolzhikov
on behalf of the JUNO Collaboration

University of Padova & INFN Padova, Italy
andrea.serafini@pd.infn.it

Repo

Pre-print

Simulation-based Inference for Precision Neutrino Physics through Neural Monte Carlo Tuning

Our ML guru

JUNO analysis fit/statistics enthusiasts

Arsenii Gavrikov, **Andrea Serafini**, Dmitry Dolzhikov

on behalf of the JUNO Collaboration

University of Padova & INFN Padova, Italy
andrea.serafini@pd.infn.it

Repo

Pre-print

The JUNO experiment

What is JUNO (Jiangmen Underground Neutrino Observatory)? [1]

- Large-scale, next-generation **liquid scintillator** (LS) detector
- It features a massive **20-kiloton LS target**
- Monitored by over **43,000 Photomultiplier Tubes (PMTs)** to detect light

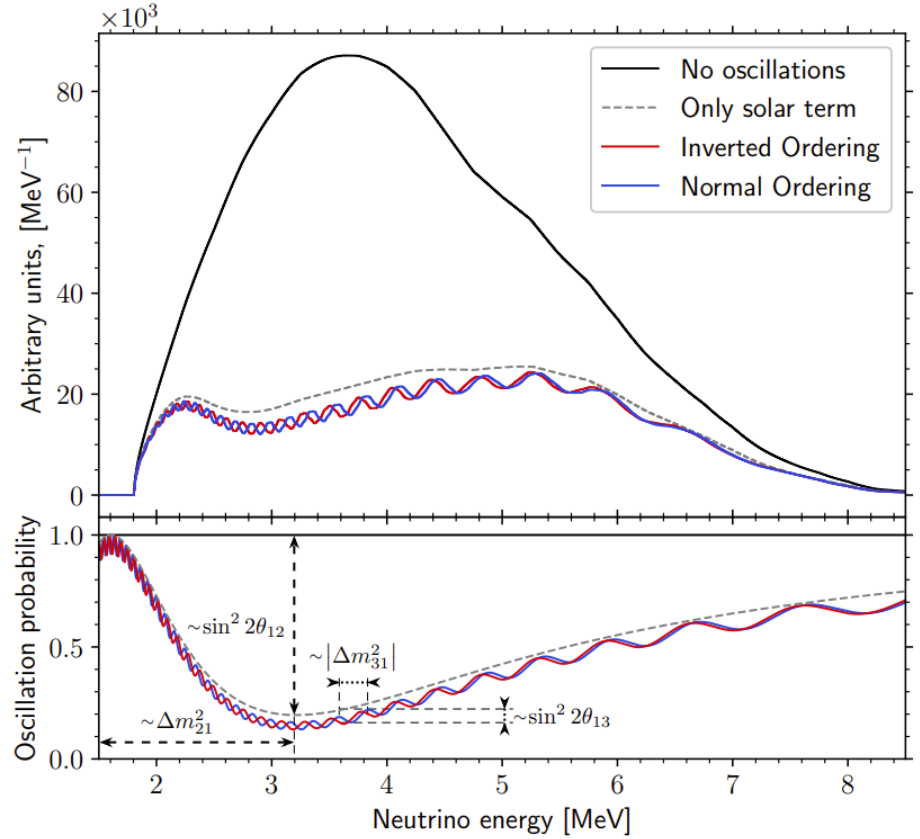
Primary Physics Goals [2, 3]

- Determine the **Neutrino Mass Ordering** (NMO).
- Measure three **neutrino oscillation parameters** (Δm^2_{21}, Δm^2_{31},$\sin^2\theta_{12}$) with **sub-percent precision**
- Broad physics program including geoneutrinos, supernovae, and more

The Core Challenge

- Unprecedented <3% @ 1 MeV energy resolution and **<1% control over energy-scale systematic** uncertainties

→ need for on highly accurate and **well-tuned Monte Carlo** (MC) simulations!

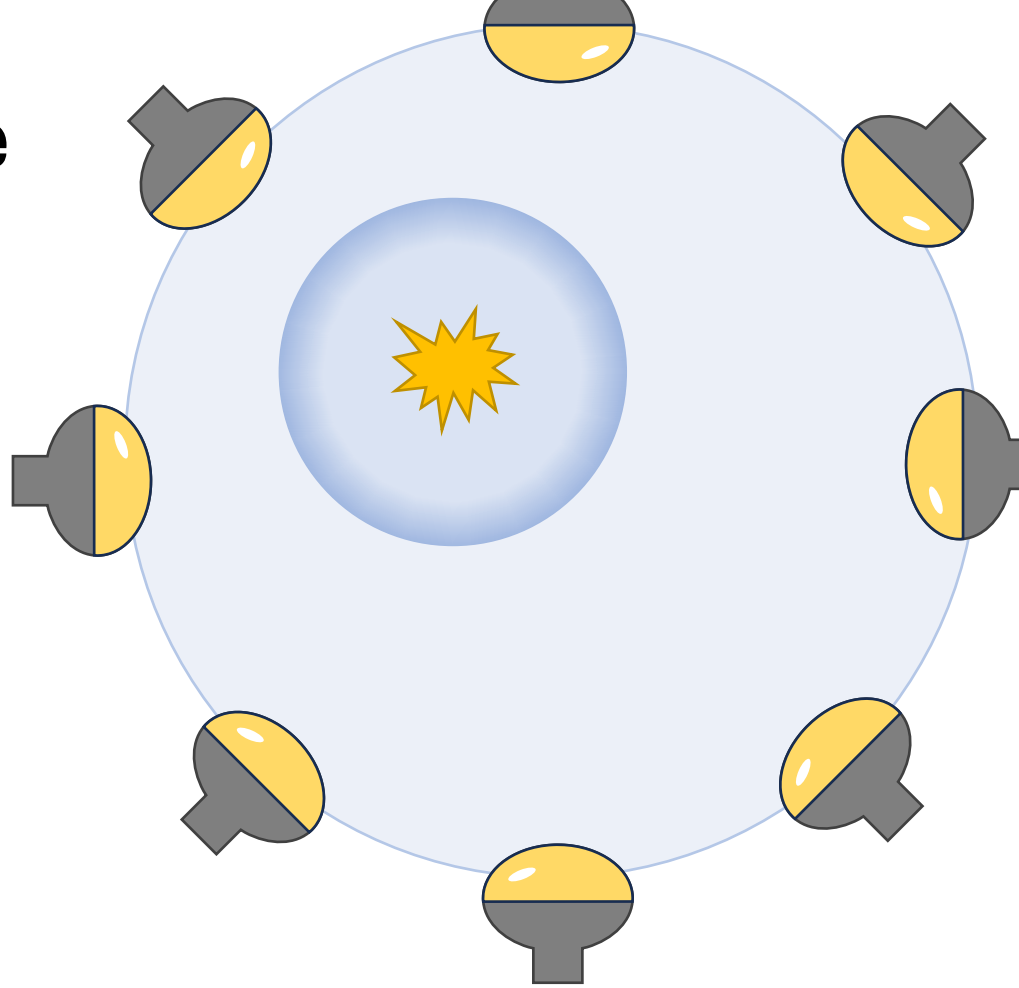


[1] JUNO Collaboration 2016 *J. Phys. G: Nucl. Part. Phys.* **43** 030401
[2] JUNO Collaboration 2022 *Chinese Phys. C* **46** 123001
[3] JUNO Collaboration 2025 *Chinese Phys. C* **49** 3, 033104

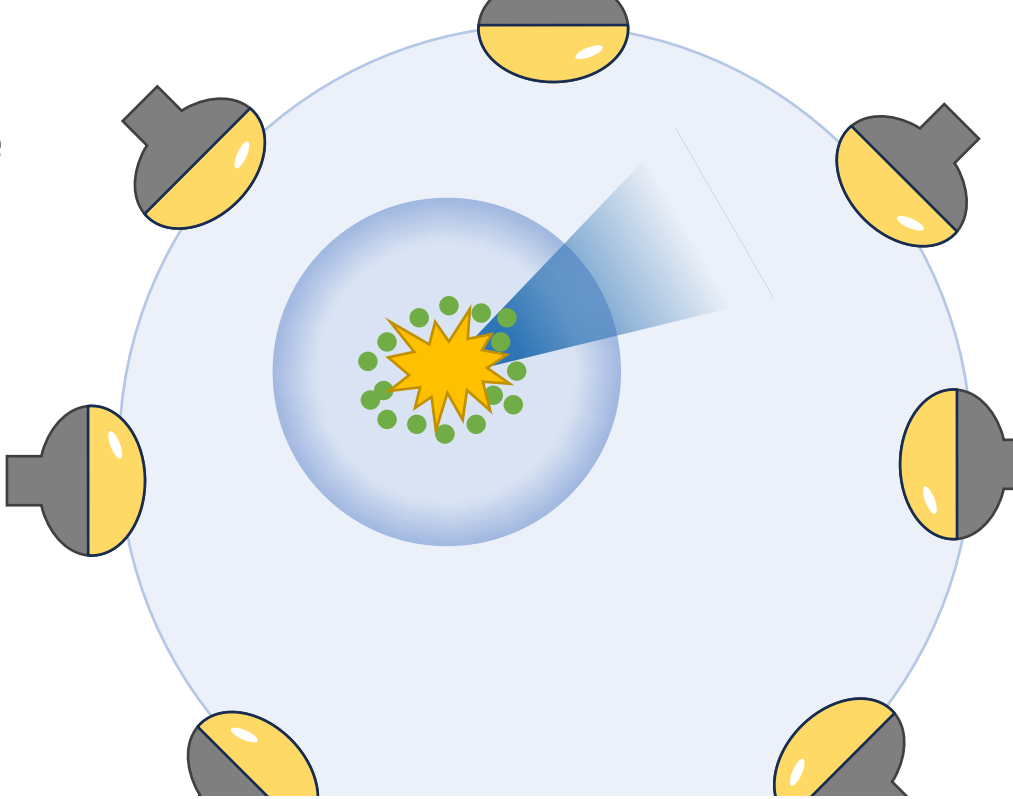
The Energy Response Challenge

How JUNO Detects Events

- JUNO is a **calorimeter**: a particle deposits energy in the scintillator.
- The scintillator emits **light collected by PMTs** and measured as a total **number of photo-electrons (NPE)** → our proxy for the visible energy



The Energy Response Challenge

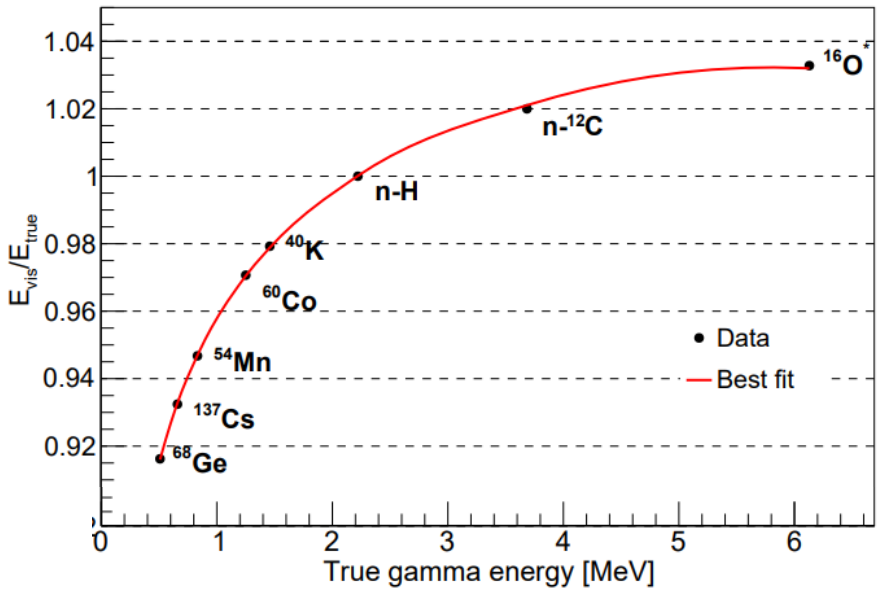


How JUNO Detects Events

- JUNO is a **calorimeter**: a particle deposits energy in the scintillator.
- The scintillator emits **light collected by PMTs** and measured as a total **number of photo-electrons (NPE)** → our proxy for the visible energy

The Problem: A Non-Linear Response

- The relationship between true deposited energy and NPE is **not linear**.
- Non-linearity in the MC modeled by three key physical parameters:
  - **Birks' Coefficient (k_B)** ↓: models quenching (energy lost as heat instead of light at high ionization) reducing the signal.
  - **Cherenkov Factor (f_C)** ↑ **:** models the fraction of Cherenkov light produced by secondary particles absorbed and re-emitted.
  - **Light Yield (Y)** ↑: models the overall scaling factor for the number of scintillation photons emitted per unit energy (after quenching).



The Energy Response Challenge

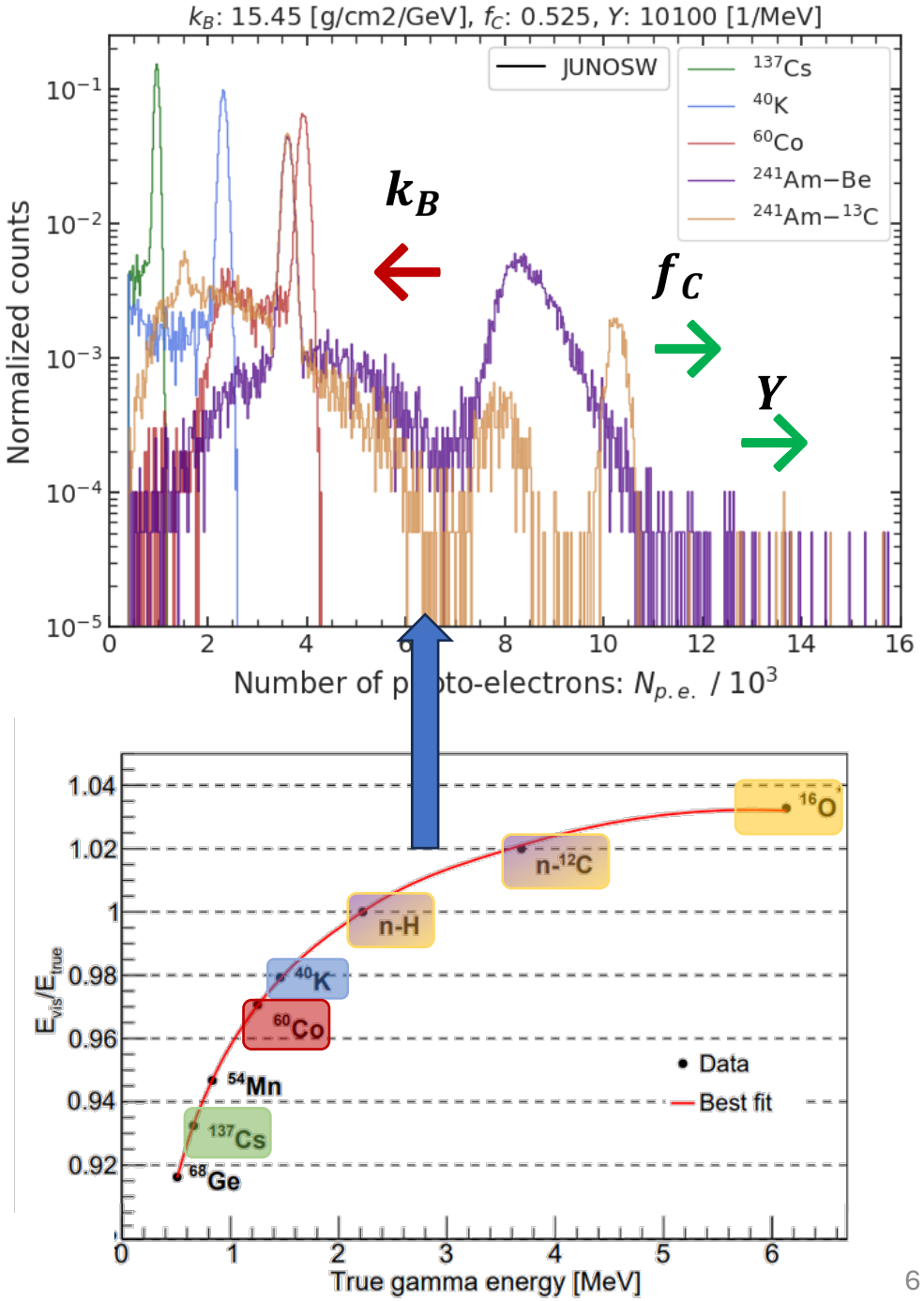
How JUNO Detects Events

- JUNO is a **calorimeter**: a particle deposits energy in the scintillator.
- The scintillator emits **light collected by PMTs** and measured as a total **number of photo-electrons (NPE)** → our proxy for the visible energy

The Problem: A Non-Linear Response

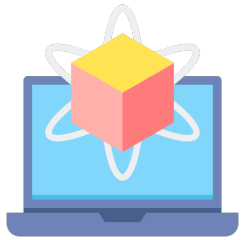
- The relationship between true deposited energy and NPE is **not linear**.
- Non-linearity in the MC modeled by three key physical parameters:
  - **Birks' Coefficient (k_B)** ↓: models quenching (energy lost as heat instead of light at high ionization) reducing the signal.
  - **Cherenkov Factor (f_C)** ↑ **:** models the fraction of Cherenkov light produced by secondary particles absorbed and re-emitted.
  - **Light Yield (Y)** ↑: models the overall scaling factor for the number of scintillation photons emitted per unit energy (after quenching).

Our Task: tune (k_B, f_C, Y) so that our MC matches real calibration data.



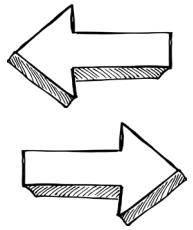
Monte Carlo tuning strategy

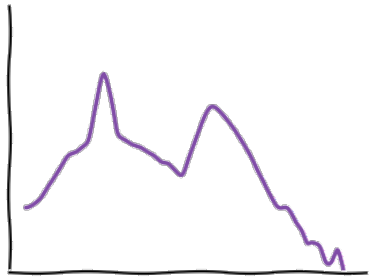
The **most straightforward approach** would be…



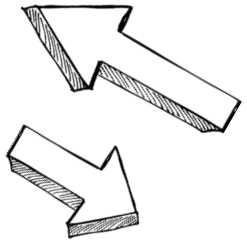
MC

$\phi = \{k_B, f_C, Y\}_{\mathrm{g}}$



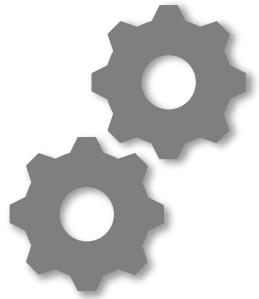


Simulated data M



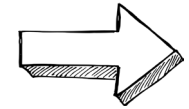
$\mathcal{L}(M(\phi)|D)$

Likelihood



Fitter

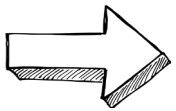
- Minuit
- MCMC
- Nested sampling

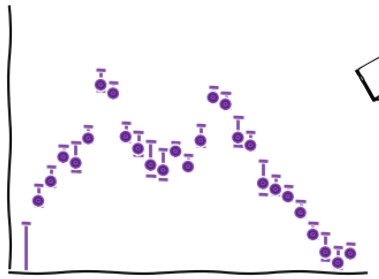


Results

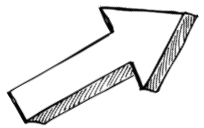
best fit $\{k_B, f_C, Y\}$

JUNO





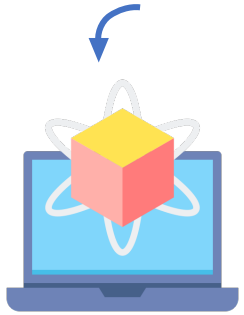
Calibration data D



Monte Carlo tuning strategy

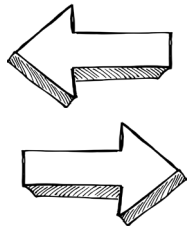
The **most straightforward approach** would be…

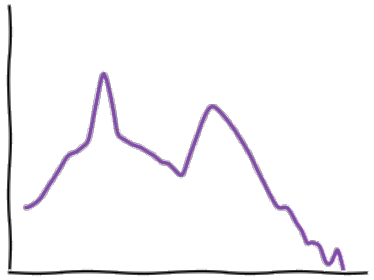
too impractical and slow! Can we replace it with a surrogate model?



MC

$\phi = \{k_B, f_C, Y\}_g$





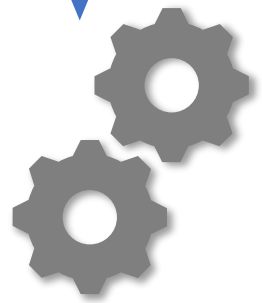
Simulated data M

At **each step** of the fit, **we generate a lot of events with full MC** to build spectra of the sources with the corresponding kB_g fC_g LY_g

$\mathcal{L}(M(\phi)|D)$

Likelihood

Analytically intractable



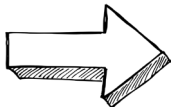
Fitter

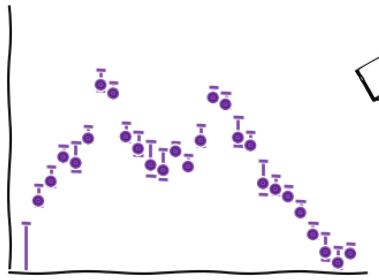
- Minuit
- MCMC
- Nested sampling

Results

best fit $\{k_B, f_C, Y\}$

JUNO





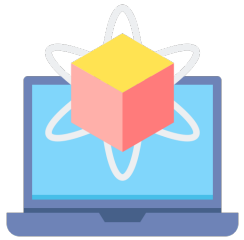
Calibration data D

Monte Carlo tuning strategy

The **most straightforward approach** would be…

too impractical and slow! Can we replace it with a surrogate model?

Neural Likelihood Estimation (NLE)



MC

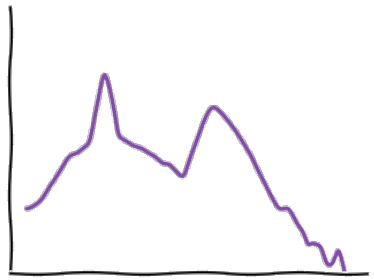
$\phi = \{k_B, f_C, Y\}_{\mathrm{g}}$

Training

Fast ML model to get spectra for ϕ

$p(x|\phi)$

(able to interpolate in param space)



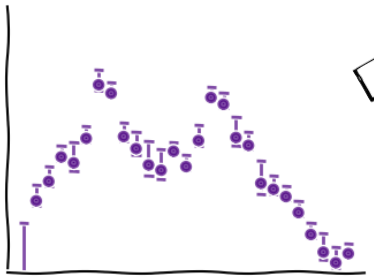
ML generated data M

$\mathcal{L}(M(\phi)|D)$

Likelihood

Approximated with SBI

JUNO



Calibration data D

Fitter

- Minuit
- MCMC
- Nested sampling

Results

best fit $\{k_B, f_C, Y\}$

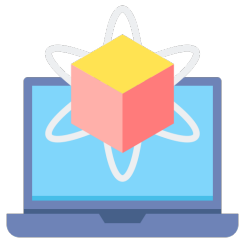
Monte Carlo tuning strategy

The **most straightforward approach** would be…

too impractical and slow! Can we replace it with a surrogate model?

We wanted to:

1. keep model **under control**
2. use **std.** inference **techniques**
3. eventually use NLE as **spectra generator** (MC surrogate)



MC

$\phi = \{k_B, f_C, Y\}_{\mathrm{g}}$

Training

Fast ML model to get spectra for ϕ

$p(x|\phi)$

(able to interpolate in param space)

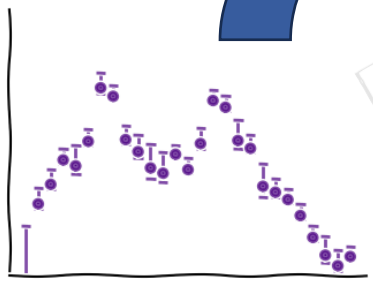
WHY NOT?

ML generated data M

Neural Posterior Estimation (NPE) directly outputting the posterior for ϕ, $q(\phi|D)$

Approximated with SBI

JUNO



Calibration data D

Results

best fit $\{k_B, f_C, Y\}$

Fitter

- Minuit
- MCMC
- Nested sampling

Two Neural Likelihood Estimators

+ straight and reliable model
- requires pre-defined binning

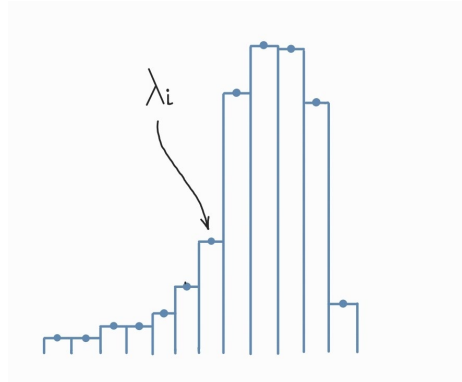
Multi-output regressor

Aims to directly learn **a mapping** from the parameters and a source type to the histogram representing the **spectra PDF**

We use a **Transformer encoder-based** model [1] as the density estimator (**TEDE**)

*Provides **PDF estimation** by outputting a histogram*

TEDE



Conditions:
kB, fC, LY + source type **S**

Normalizing Flow-based model

Aims to learn the exact **conditional probability** of the energies* for a given set of parameters and a source type:

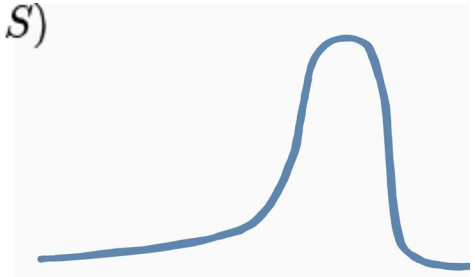
$P(E \mid k_B, f_C, L_y, S)$

We use **Normalizing Flows**-based model [2] to evaluate the exact conditional probability for the events and build the PDF

*Provides the **exact conditional PDF** for any energy value:*

$P(E \mid k_B, f_C, L_y, S)$

NFDE



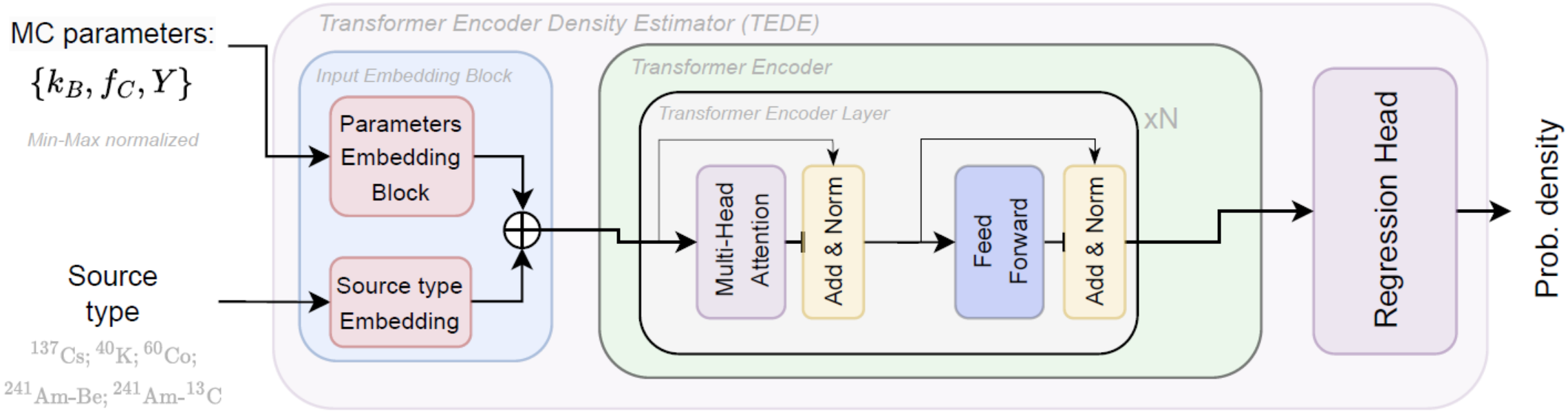
+ exact probability density function
+ no pre-defined binning
+ opens a possibility of an unbinned fit

**represented by amount of light collected*

[1] Vaswani A et al, arxiv: 1706.03762

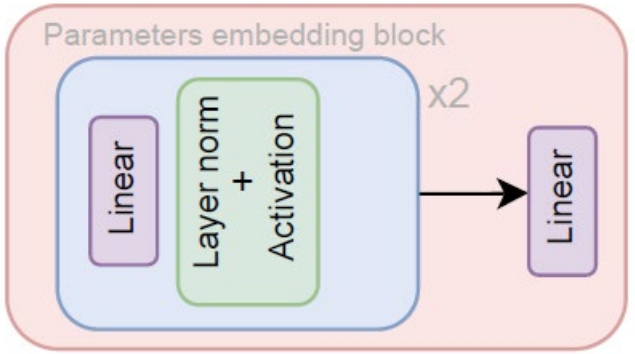
[2] R. J. Rezende et al, arxiv: 1505.05770

Transformer Encoder-based Density Estimator (TEDE)



Multi-output regressor is based on the **Transformer's [1] encoder** :

- A powerful architecture, well known for NLP and CV applications
- Each condition (MC parameter or source type) is treated as a token
- Regressor Head block to convert outputs to the spectra PDF with pre-defined binning (**bin size of 20 PEs**)

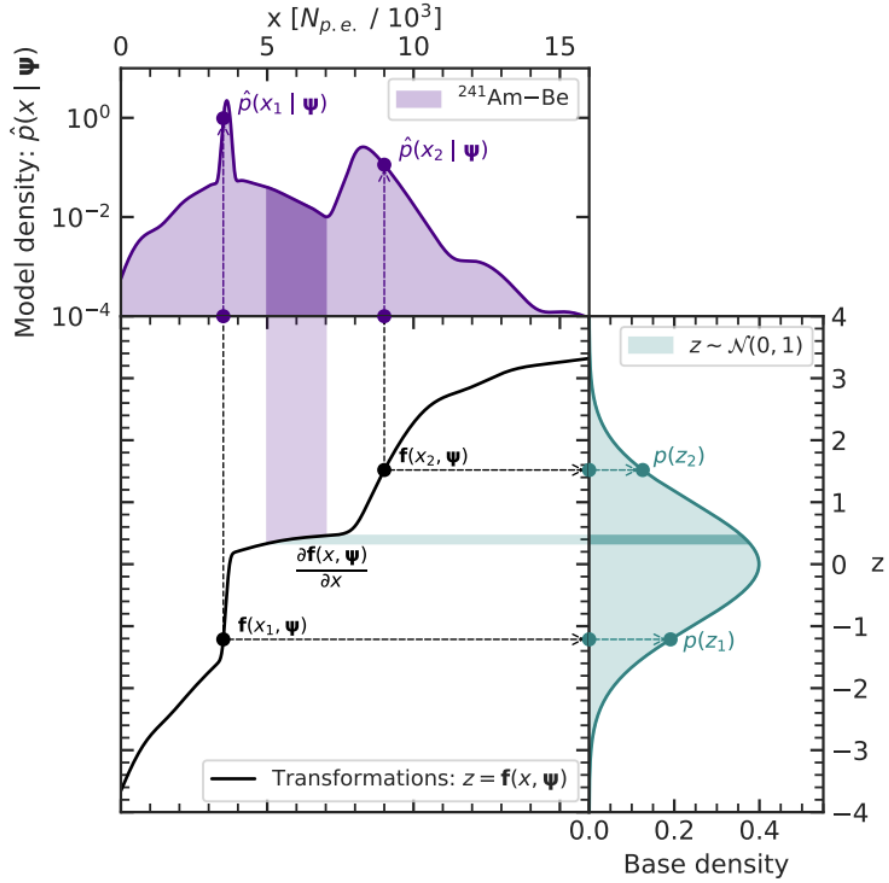


[1] Vaswani A et al, arxiv: 1706.03762

Normalizing Flows-based Density Estimator (NFDE)

Normalizing flows [1]:

- **generative** ML model aimed to **explicitly model a density estimation**
- generalizable for modeling **a conditional density estimation** as well
- based on an **idea of transforming a simple known distribution** (*e.g. standard Gaussian*) **to the complex, desired distribution** by applying multiple learnable (*using data*) and invertible transformations called **flows.**

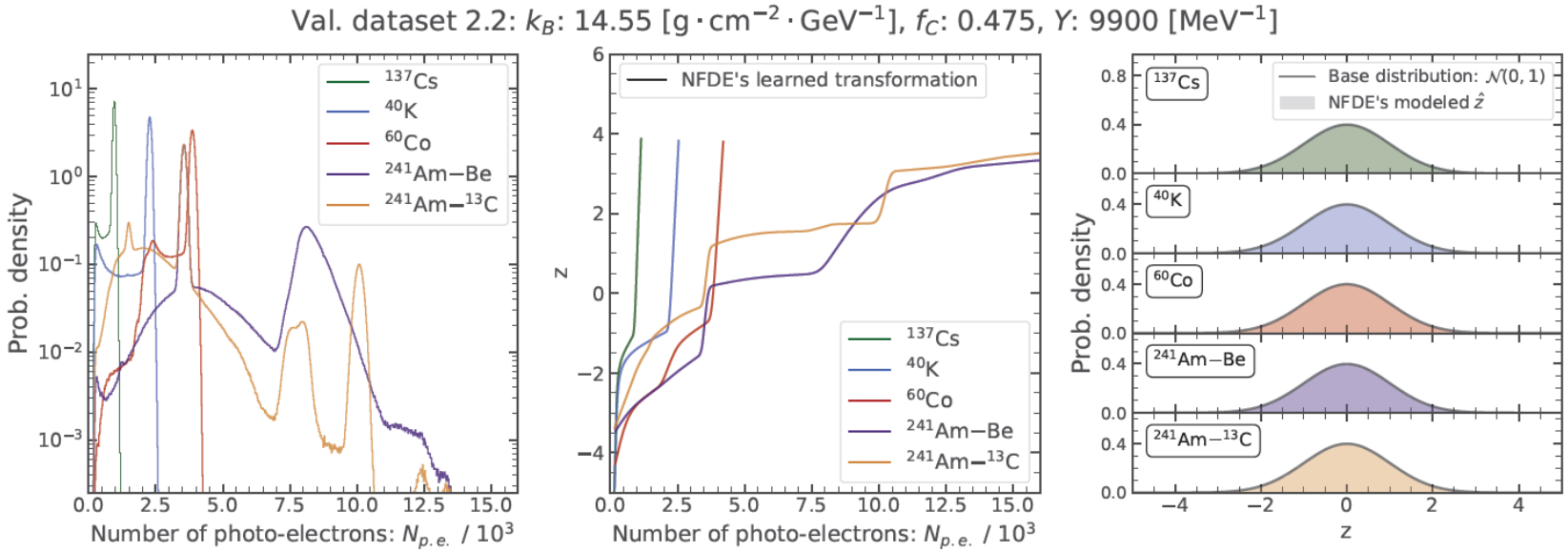


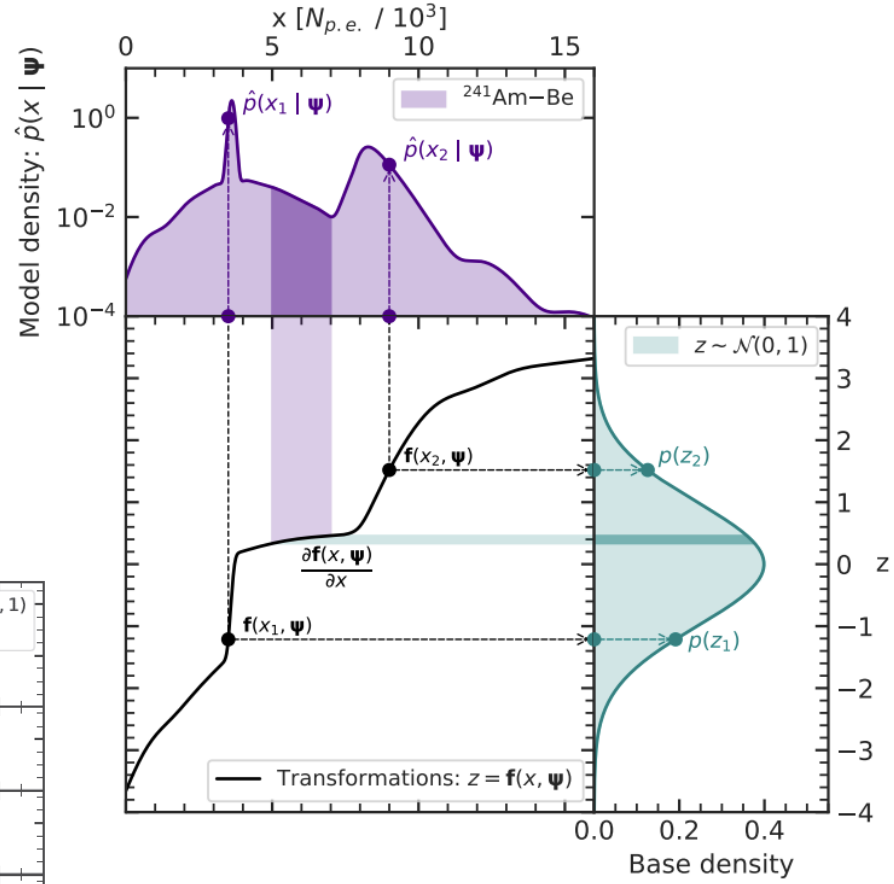
[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing Flows-based Density Estimator (NFDE)

Normalizing flows [1]:

- **generative** ML model aimed to **explicitly model a density estimation**
- generalizable for modeling **a conditional density estimation** as well
- based on an **idea of transforming a simple known distribution** (*e.g. standard Gaussian*) **to the complex, desired distribution** by applying multiple learnable (*using data*) and invertible transformations called **flows.**



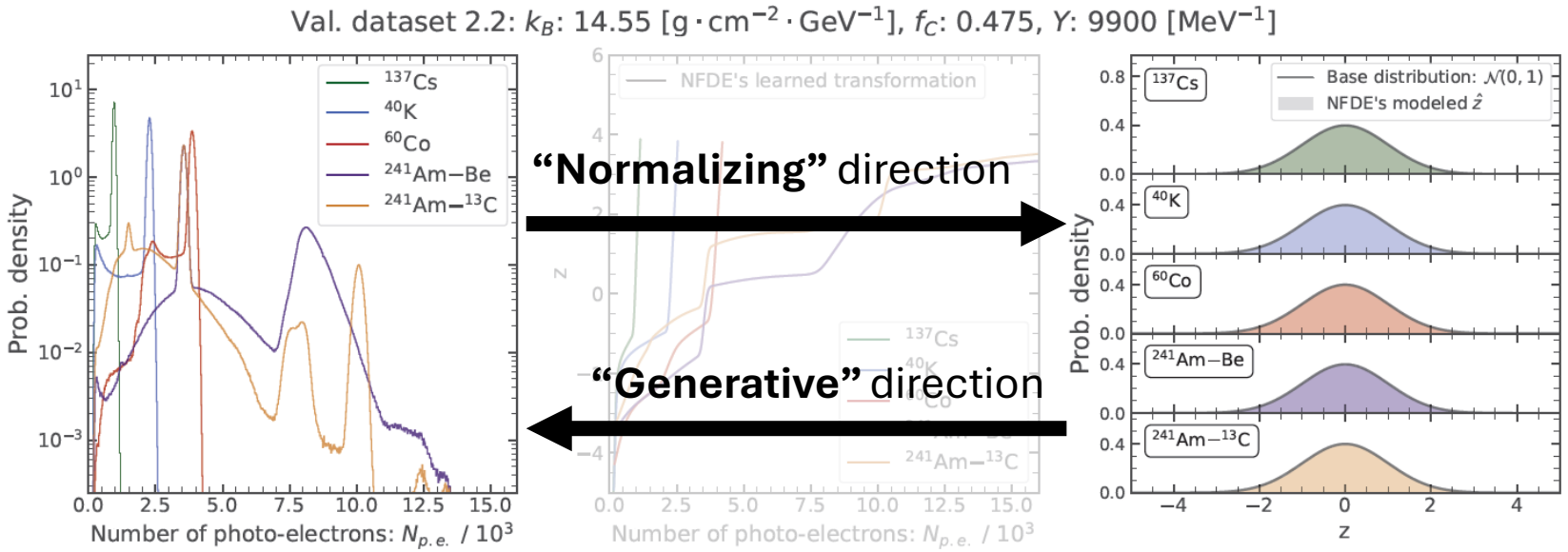


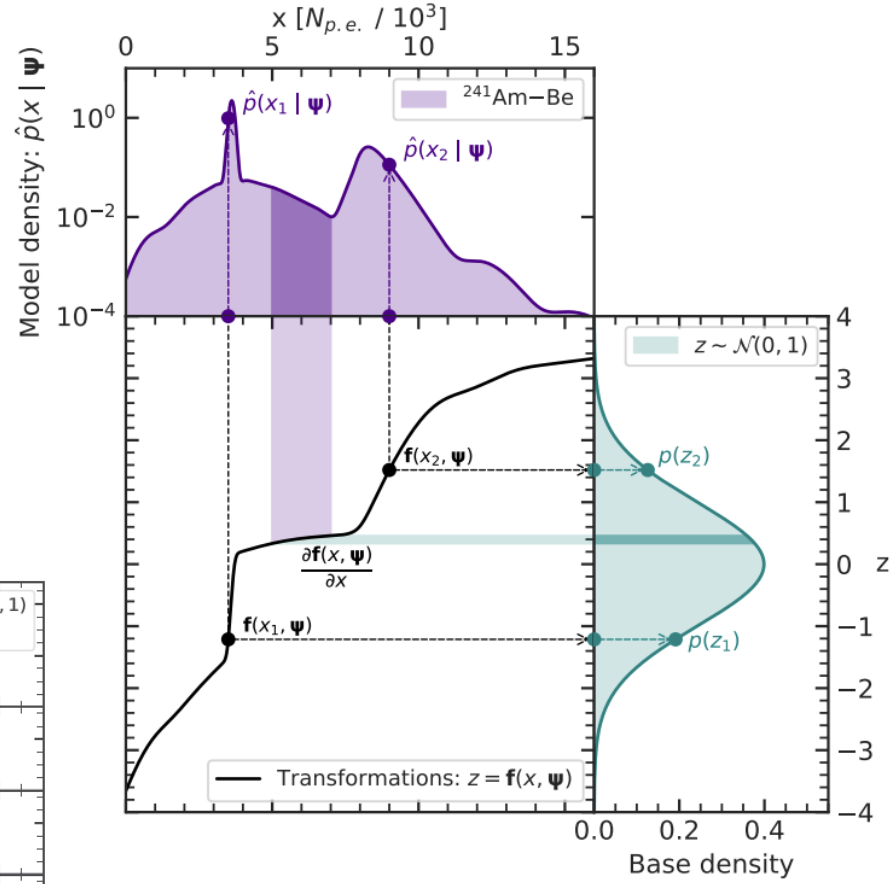
[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing Flows-based Density Estimator (NFDE)

Normalizing flows [1]:

- **generative** ML model aimed to **explicitly model a density estimation**
- generalizable for modeling **a conditional density estimation** as well
- based on an **idea of transforming a simple known distribution** (*e.g. standard Gaussian*) **to the complex, desired distribution** by applying multiple learnable (*using data*) and invertible transformations called **flows.**





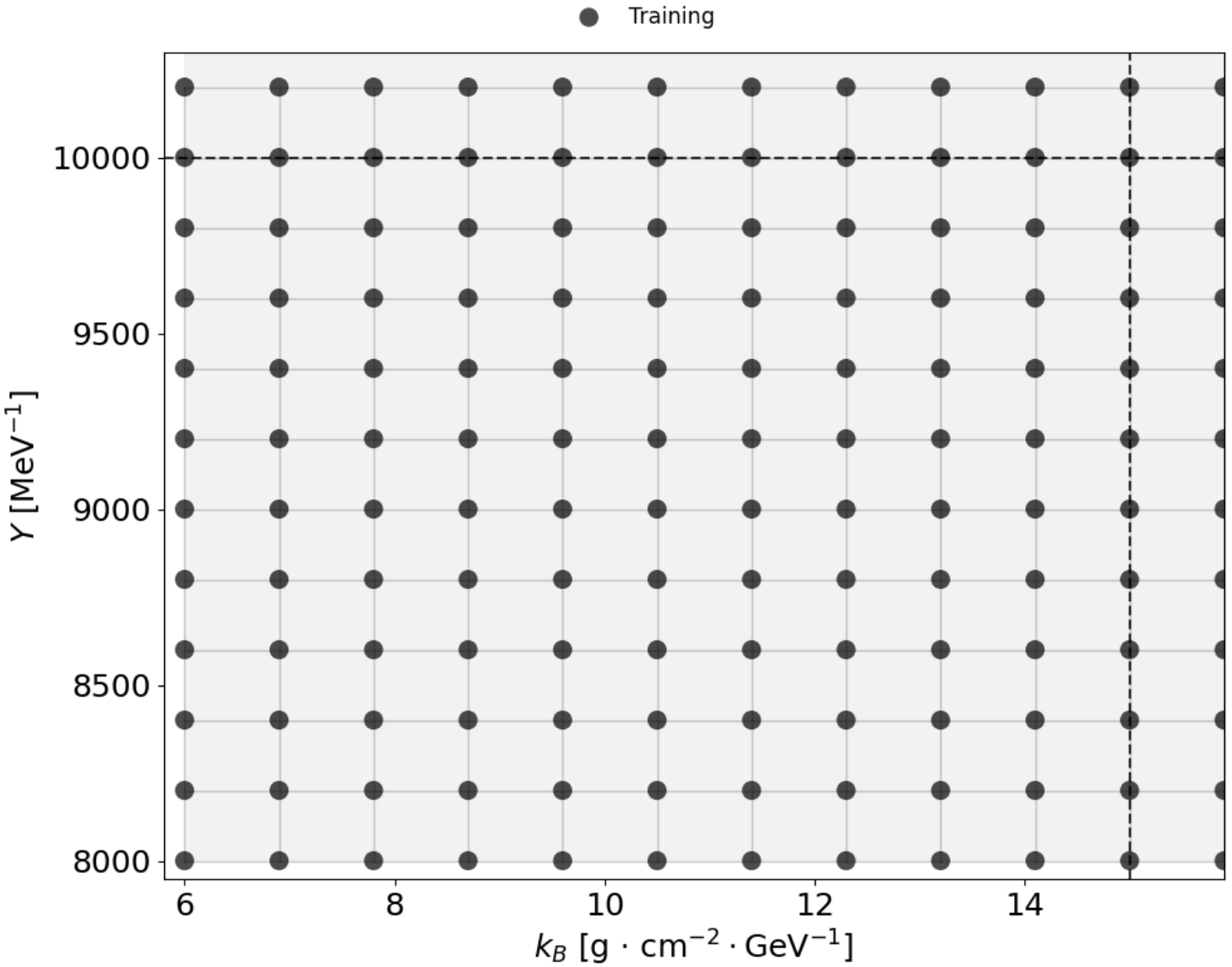
[1] R. J. Rezende et al, arxiv: 1505.05770

Datasets: training/validation strategy

Y and k_B example

Training data

- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

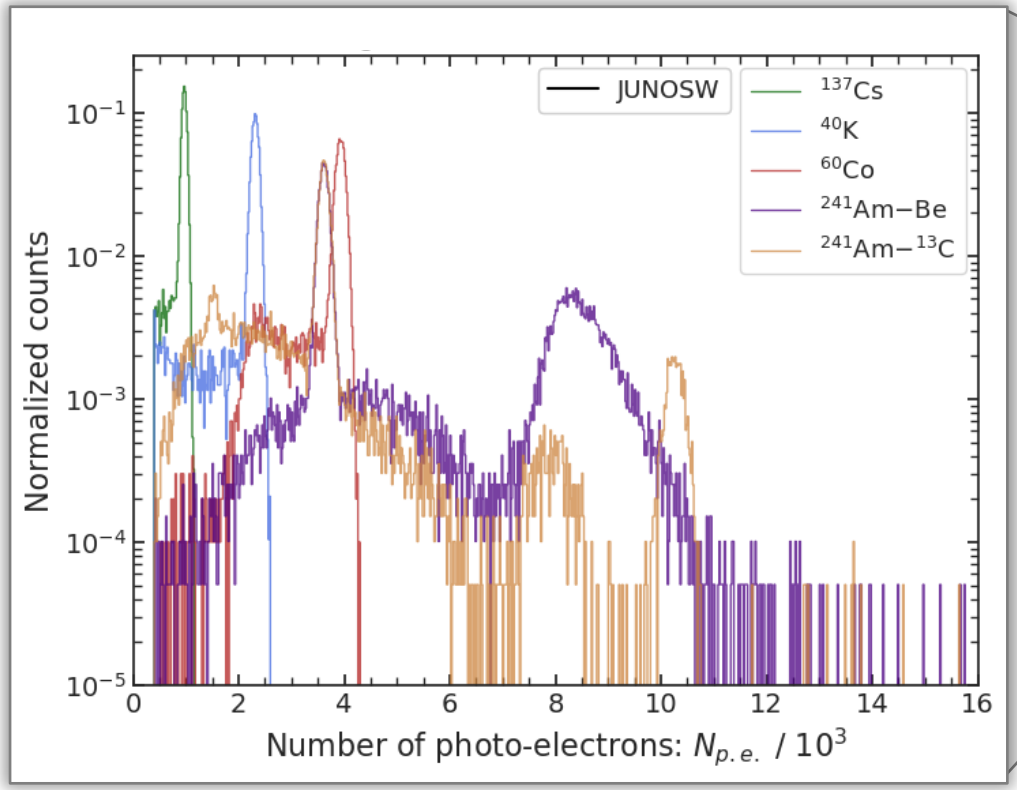


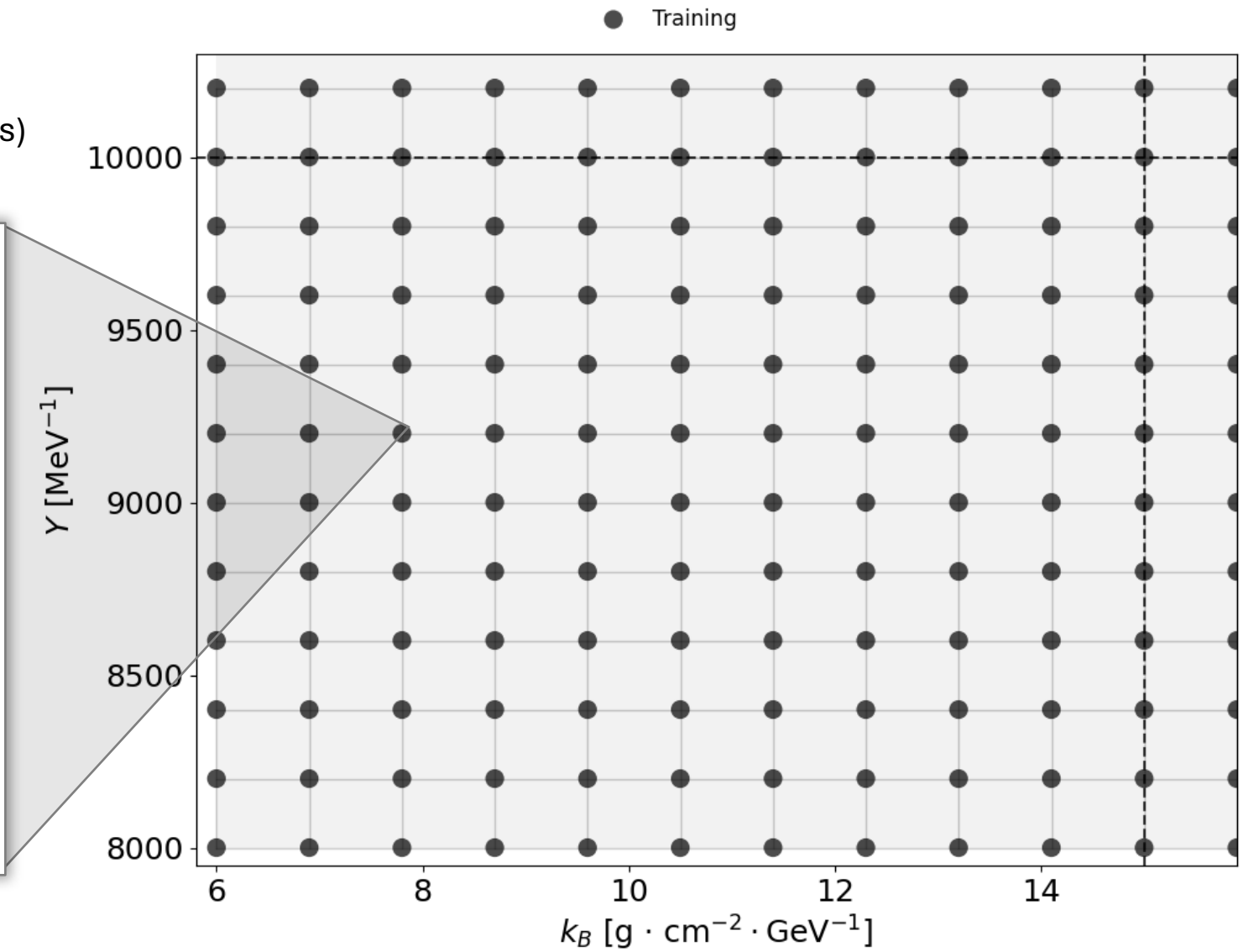
Datasets: training/validation strategy

Y and k_B example

Training data

- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)





Datasets: training/validation strategy

Y and k_B example

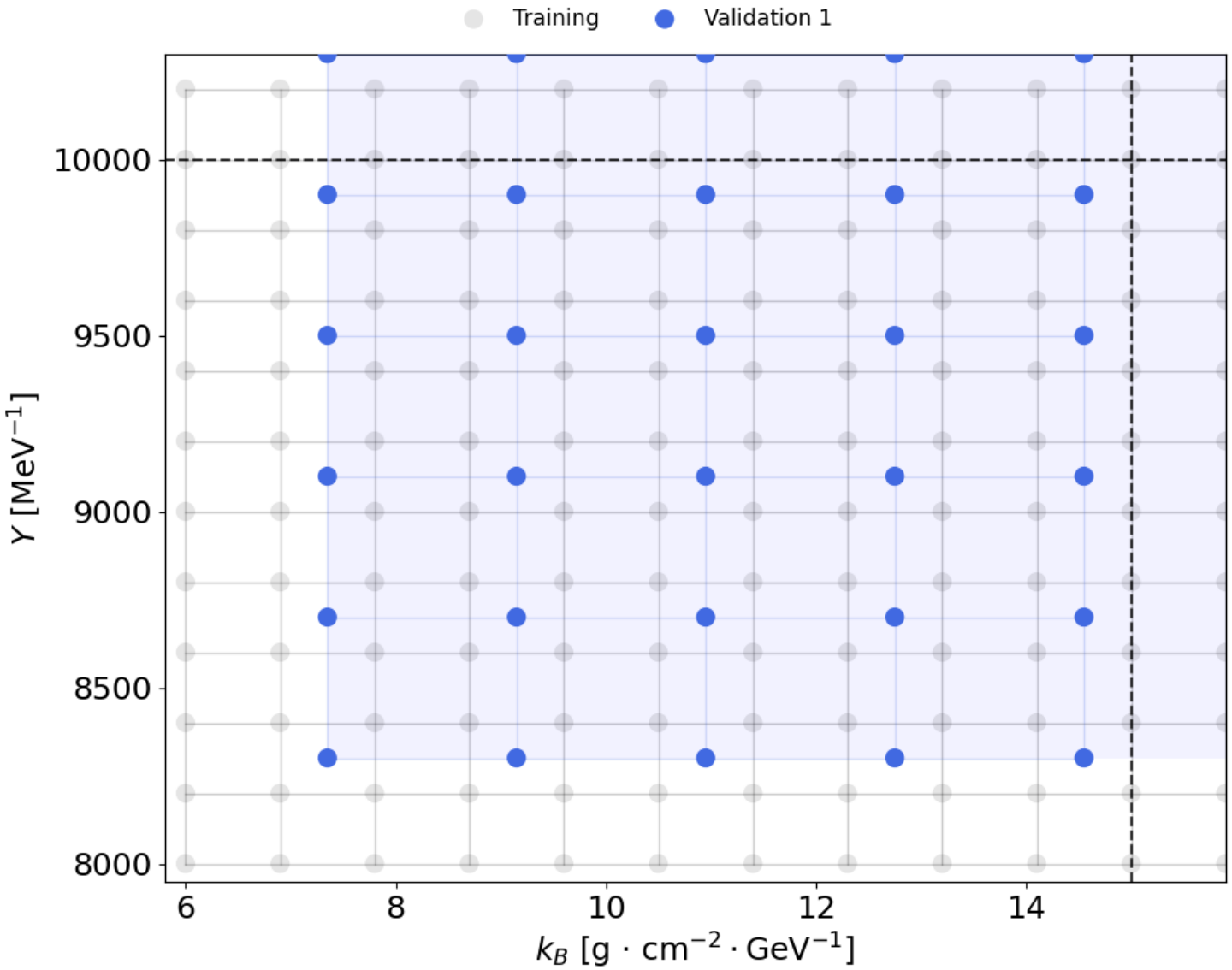
Training data

- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1

- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space



Datasets: training/validation strategy

Y and k_B example

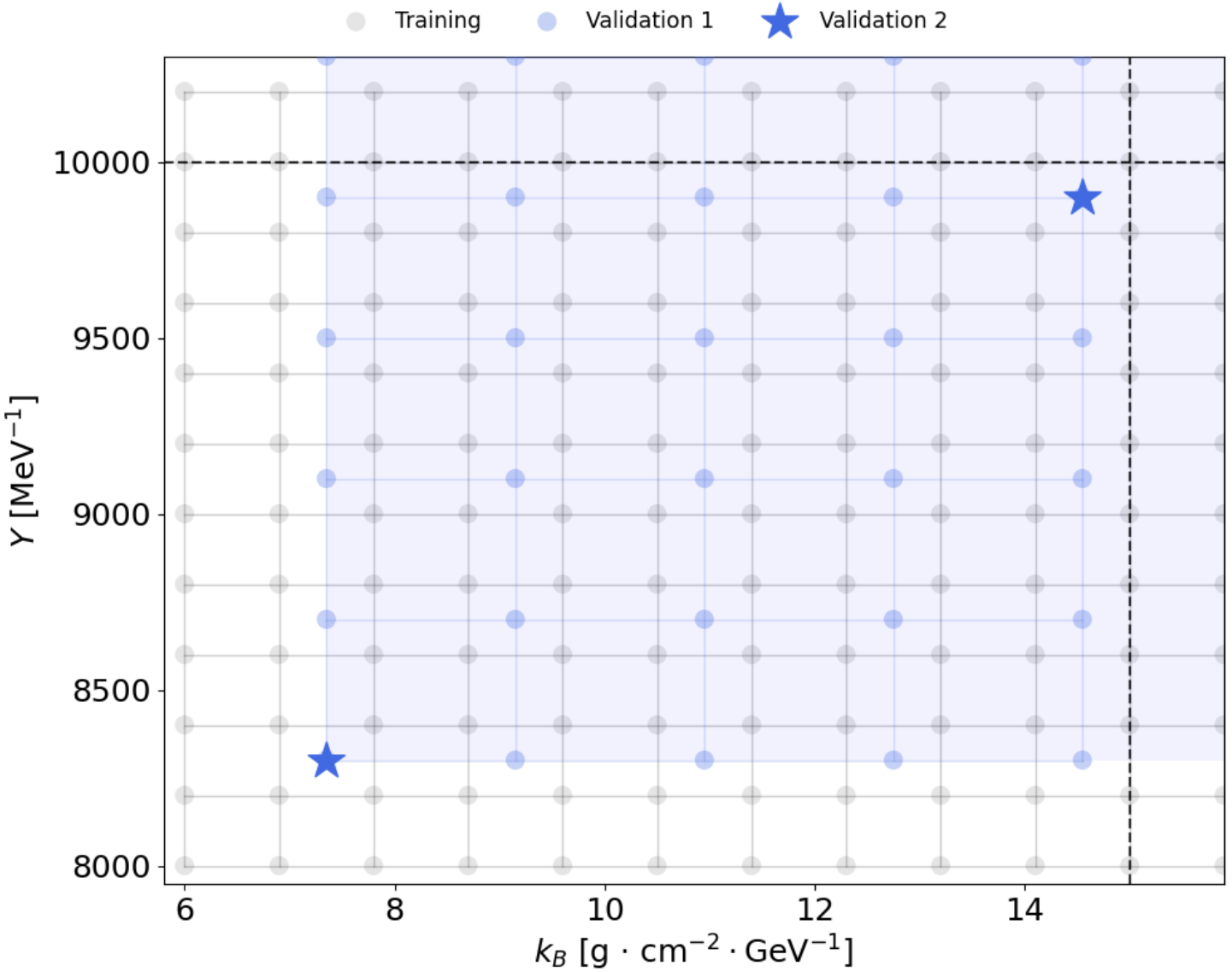
Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

→ anchor PDF to high-stat spectra



Datasets: training/validation strategy

Y and k_B example

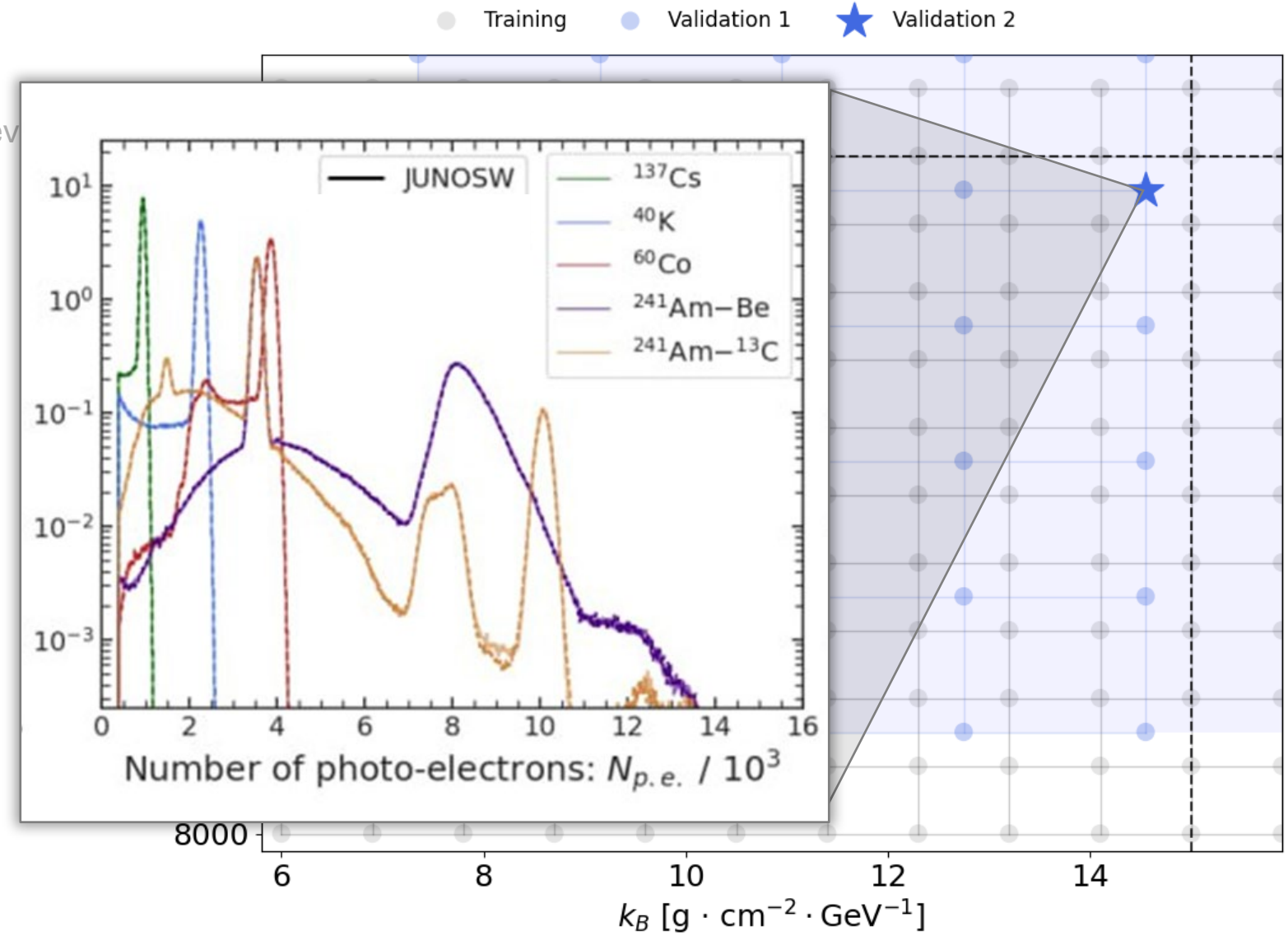
Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M ev

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

→ anchor PDF to high-stat spectra



Datasets: training/validation strategy

Y and k_B example

Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

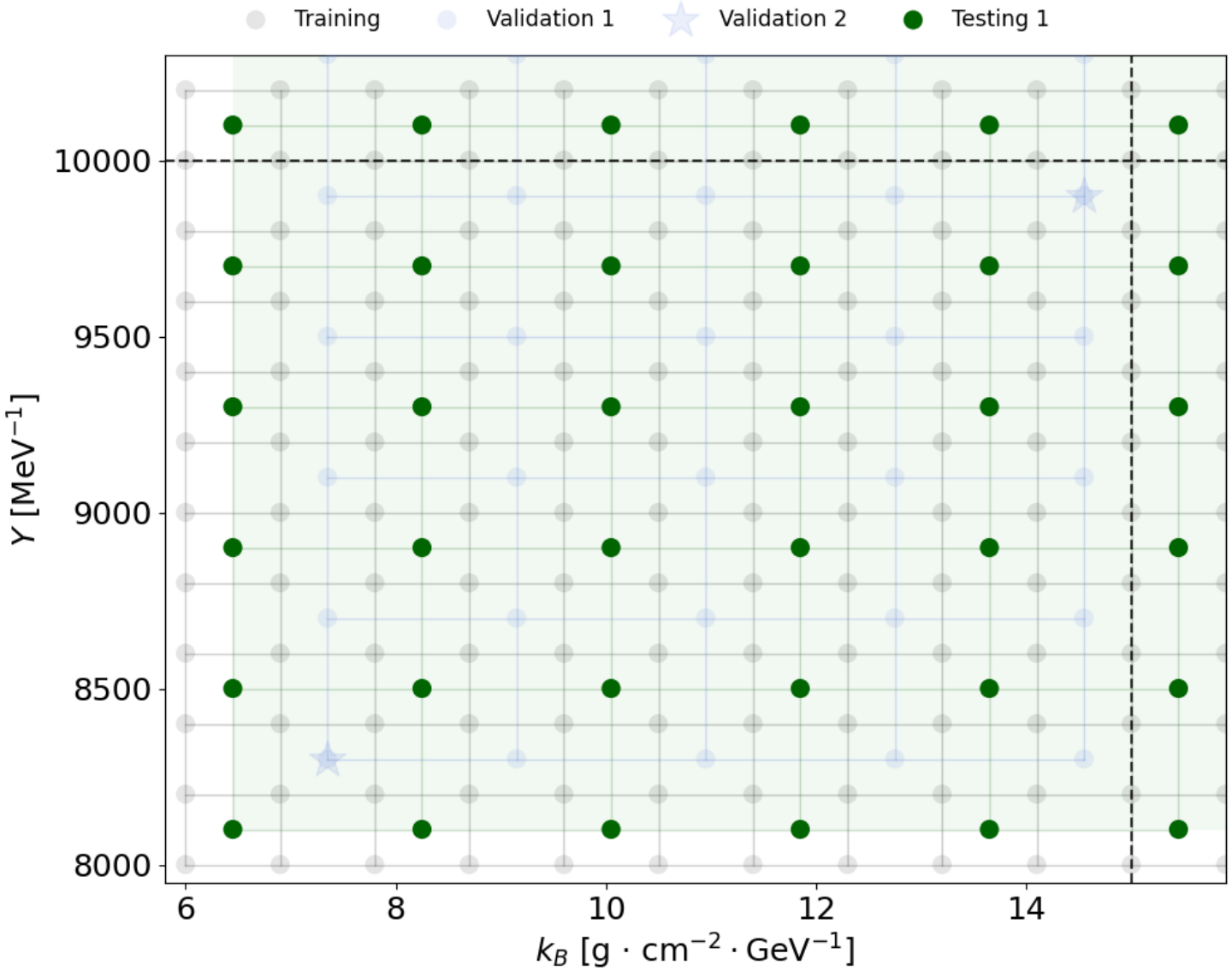
→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

→ anchor PDF to high-stat spectra

Testing data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ check possible **biases over parameter space**



Datasets: training/validation strategy

Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

→ anchor PDF to high-stat spectra

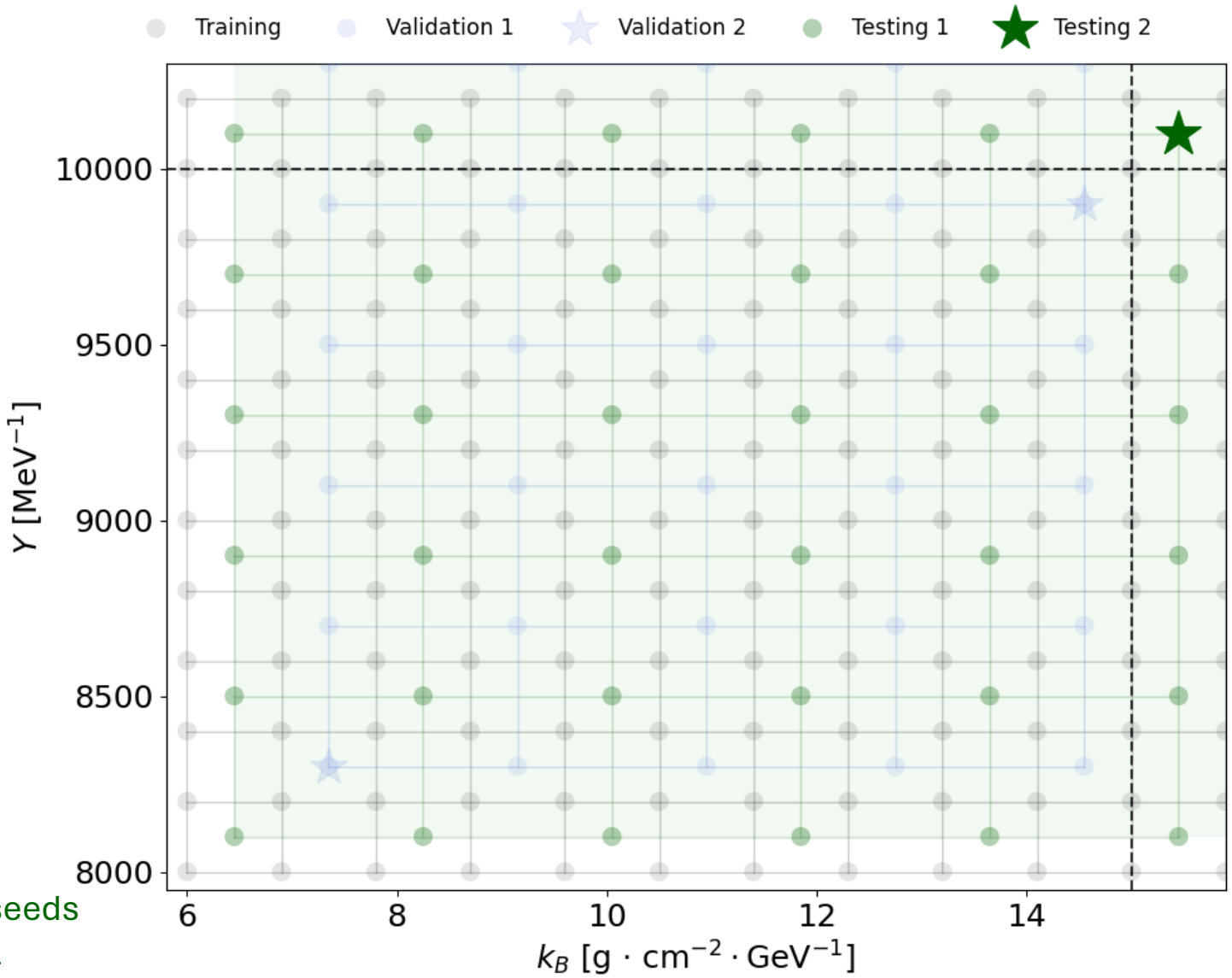
Testing data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ check possible **biases over parameter space**

Testing data #2
- 1 point
- 1000 x [1k; 2k; 5k; 10k; 25k] events x 5 with diff. seeds

→ Analysis of **systematic uncertainty** of the model

Y and k_B example



Datasets: training/validation strategy

Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

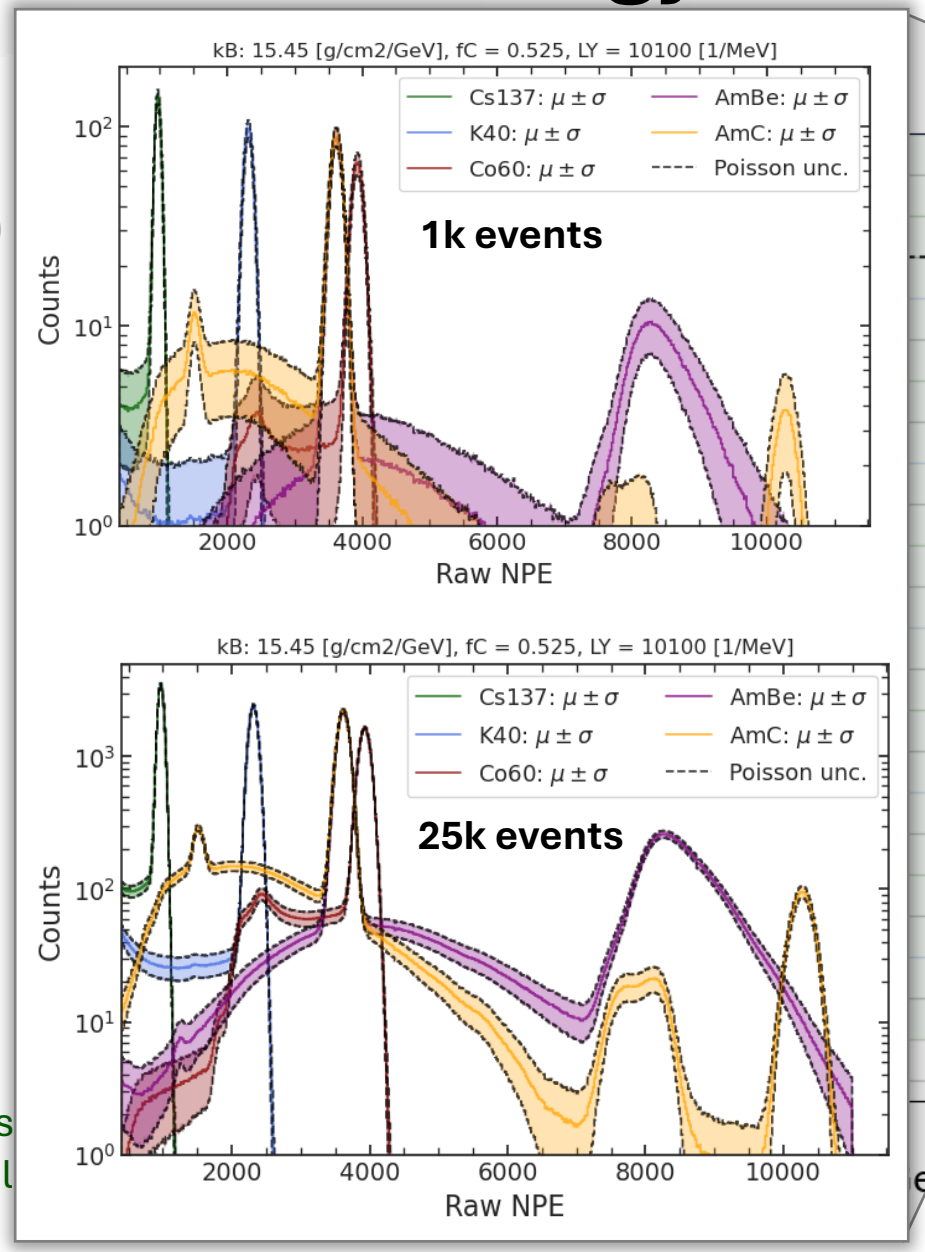
→ anchor PDF to high-stat spectra

Testing data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

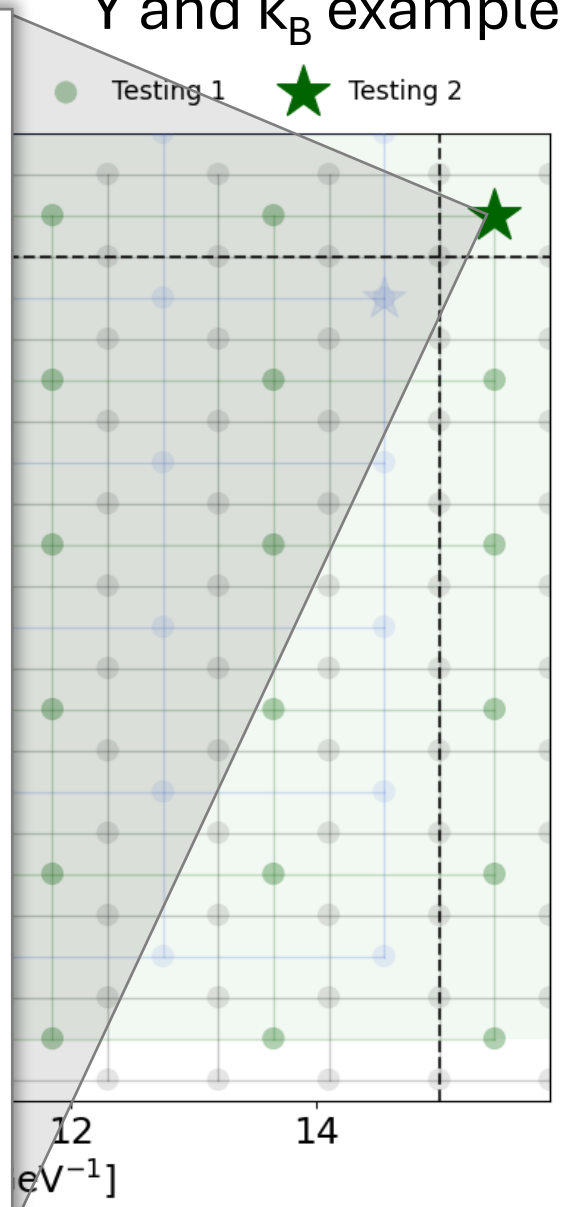
→ check possible **biases over parameter space**

Testing data #2
- 1 point
- 1000 x [1k; 2k; 5k; 10k; 25k] events x 5 with diff. s

→ Analysis of **systematic uncertainty** of the model



Y and k_B example



Datasets: training/validation strategy

Y and k_B example

Training data
- 21 points per param (21^3 combs)
- 10k events x 5 sources per point (~500M events)

Validation data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ cover most of parameter space

Validation data #2
- 3 points in total
- 10M x 5 events per point (150M events)

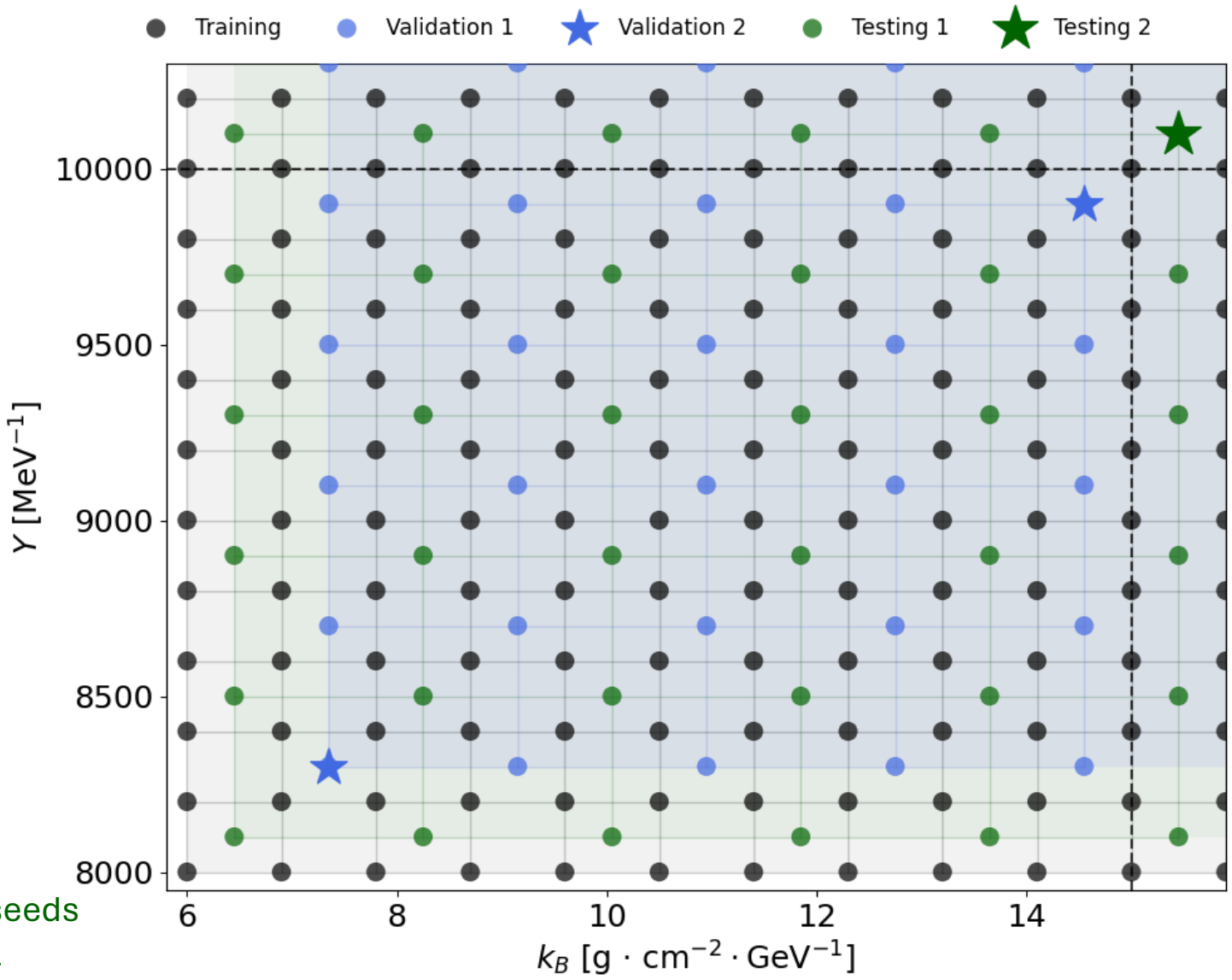
→ anchor PDF to high-stat spectra

Testing data #1
- 10 points per param (10^3 combs)
- 10k x 5 events per point (50M events)

→ check possible **biases over parameter space**

Testing data #2
- 1 point
- 1000 x [1k; 2k; 5k; 10k; 25k] events x 5 with diff. seeds

→ Analysis of **systematic uncertainty** of the model



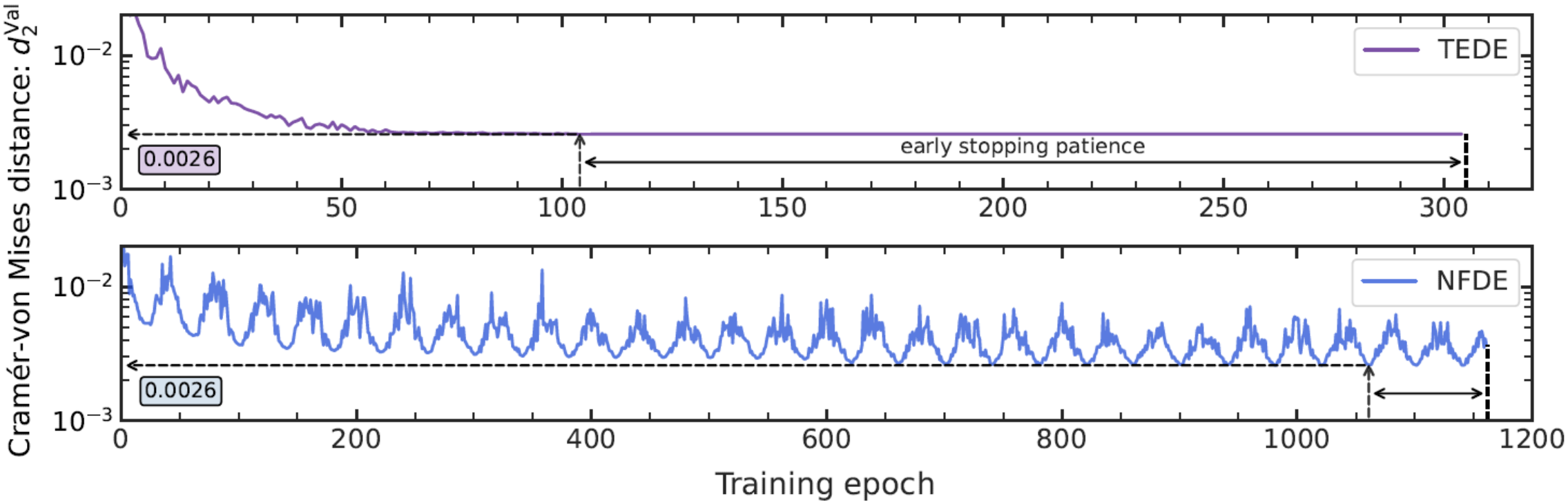
Training and validation of the models

- For both models we use **Kullback–Leibler divergence as the loss function** for training:
- As validation metrics we use p-norm distances between distributions' CDFs:

$$d_p(F_u, F_v) = \left(\int_{-\infty}^{+\infty} |F_u(x) - F_v(x)|^p \, dx \right)^{\frac{1}{p}}$$

as it is applicable for both TEDE and NFDE cases

p = 1: *Wasserstein distance*
p = 2: *Cramer-von Mises distance*
p = ∞: *Kolmogorov-Smirnov distance*



Training and validation of the models

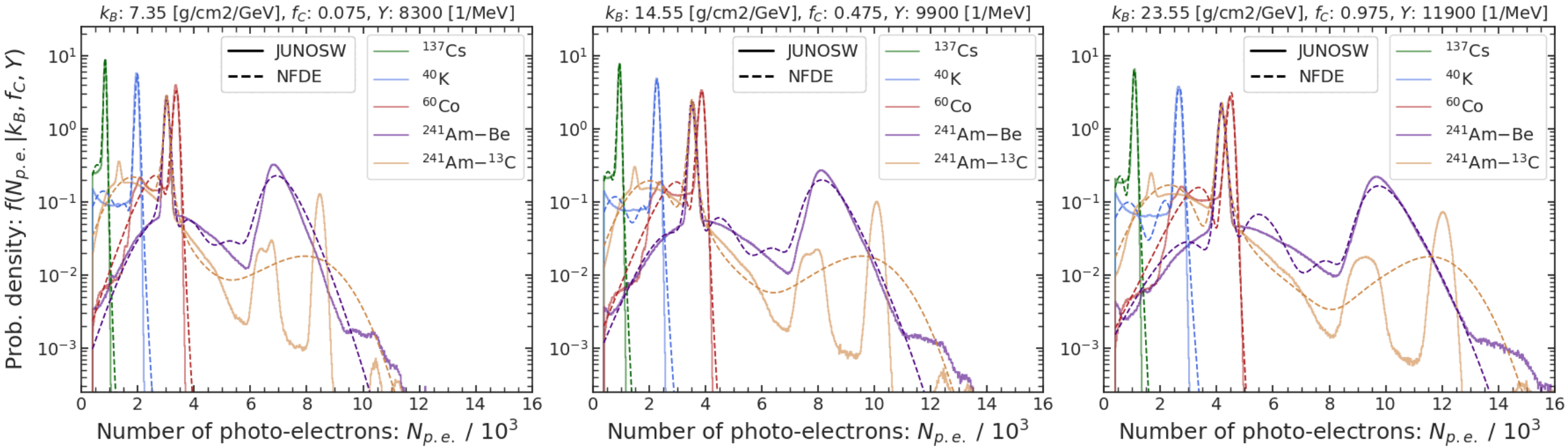
- For both models we use **Kullback–Leibler divergence as the loss function** for training:
- As validation metrics we use p-norm distances* between distributions' CDFs:

$$d_p(F_u, F_v) = \left(\int_{-\infty}^{+\infty} |F_u(x) - F_v(x)|^p \, dx \right)^{\frac{1}{p}}$$

as it is applicable for both TEDE and NFDE cases

p = 1: *Wasserstein distance*
p = 2: *Cramer von Mises distance*
p = ∞: *Kolmogorov-Smirnov distance*

Validation dataset №2. Epoch: 0, Iteration: 927, $d_1^{V_2} = 0.0544$; $d_2^{V_2} = 0.0323$; $d_\infty^{V_2} = 0.0598$



Parameter estimation

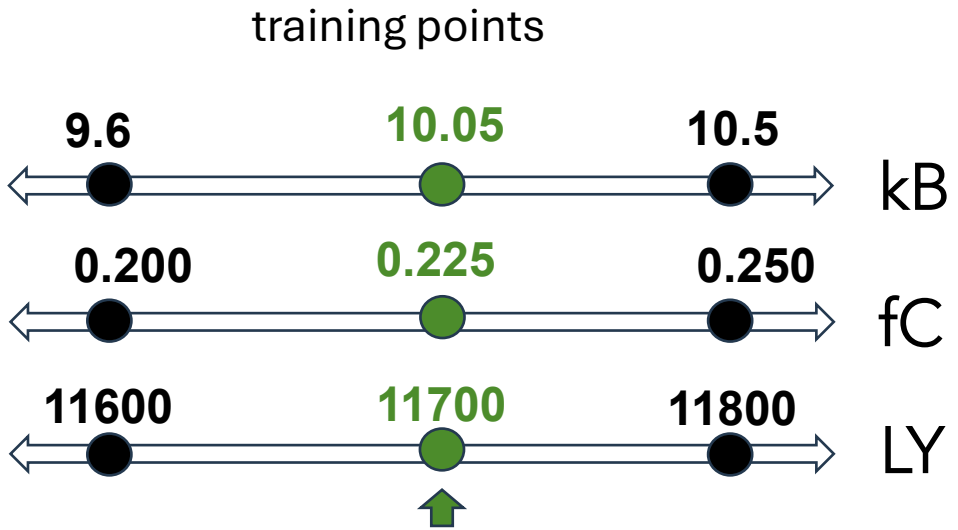
Example with NFDE:

- **Combined fit** on all sources together
- **Cost function:** Extended unbinned likelihood
- **Optimizer:** Nested Sampling (ultranest)*

→ Estimate the **kB, fC, LY best parameters**

→ Explores full phase space, **provides full posterior**

→ **Shows correlations** between the parameters



the testing dataset №1 point is between the points from the training dataset: ***interpolation***

*all fits performed with Minuit, MCMC (emcee), Nested Sampling (ultranest) → consistent results

Parameter estimation

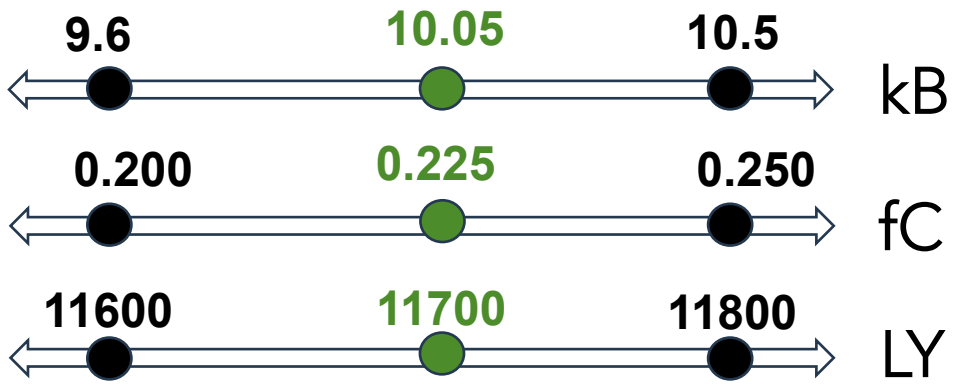
Example with NFDE:

- **Combined fit** on all sources together
- **Cost function:** Extended unbinned likelihood
- **Optimizer:** Nested Sampling (ultranest)*

→ Estimate the **kB, fC, LY best parameters**

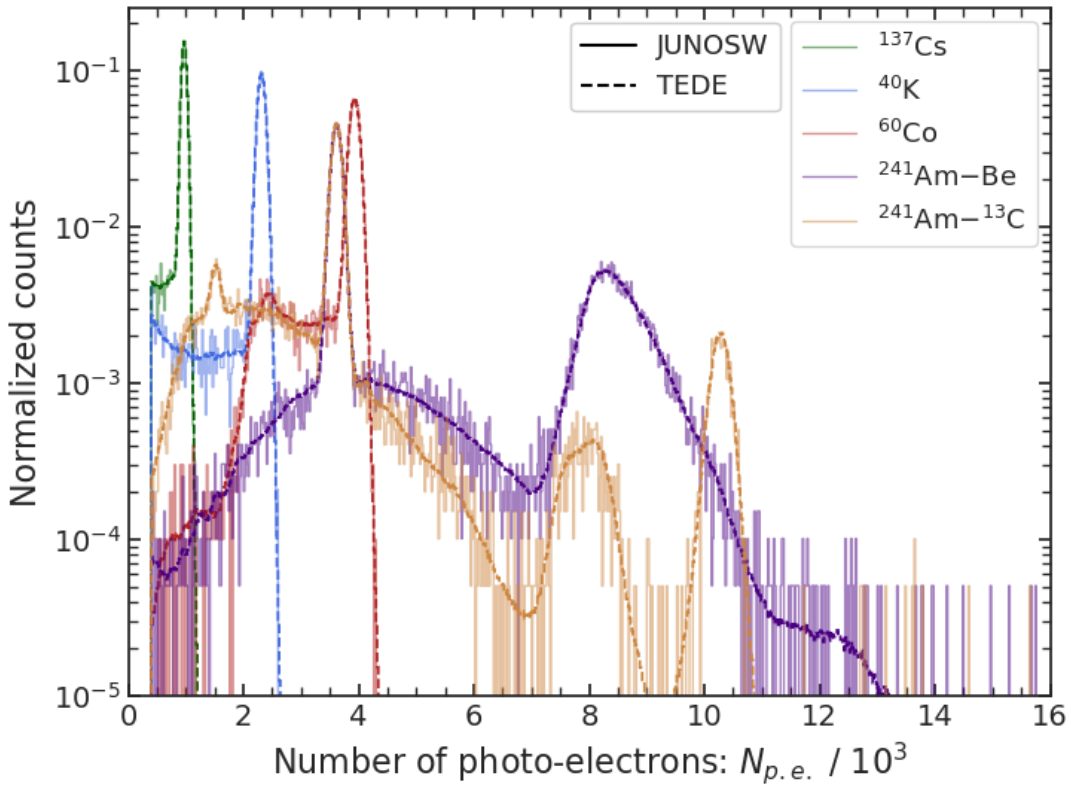
→ Explores full phase space, **provides full posterior**

→ **Shows correlations** between the parameters



Model provides spctra estimates for ***any continuous*** values of the parameters

Model **interpolation** smooth and denoised



*all fits performed with Minuit, MCMC (emcee), Nested Sampling (ultranest) → consistent results

Parameter estimation

Example with NFDE:

- **Combined fit** on all sources together
- **Cost function:** Extended unbinned likelihood
- **Optimizer:** Nested Sampling (ultranest)*

→ Estimate the **kB, fC, LY best parameters**

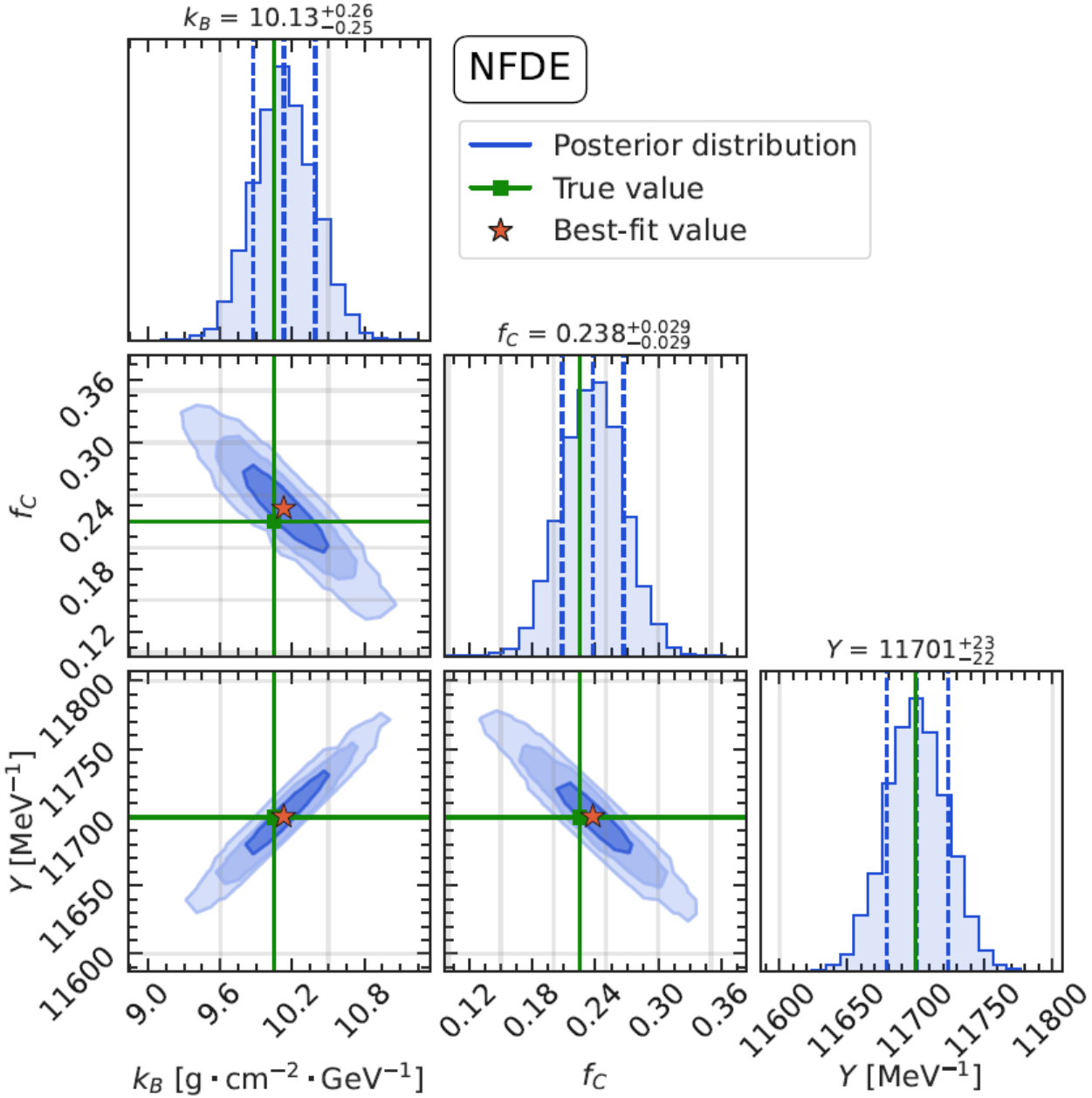
→ Explores full phase space, **provides full posterior**

→ **Shows correlations** between the parameters

Parameters estimation combined:

○ kB: **10.13 +- 0.26**	10.05 [g/cm²/GeV]
○ fC: **0.238 +- 0.029**	0.225
○ LY: **11701 +- 23**	11700 [1/MeV]

→ **well within 1σ** (fluctuations expected for 1 fit)

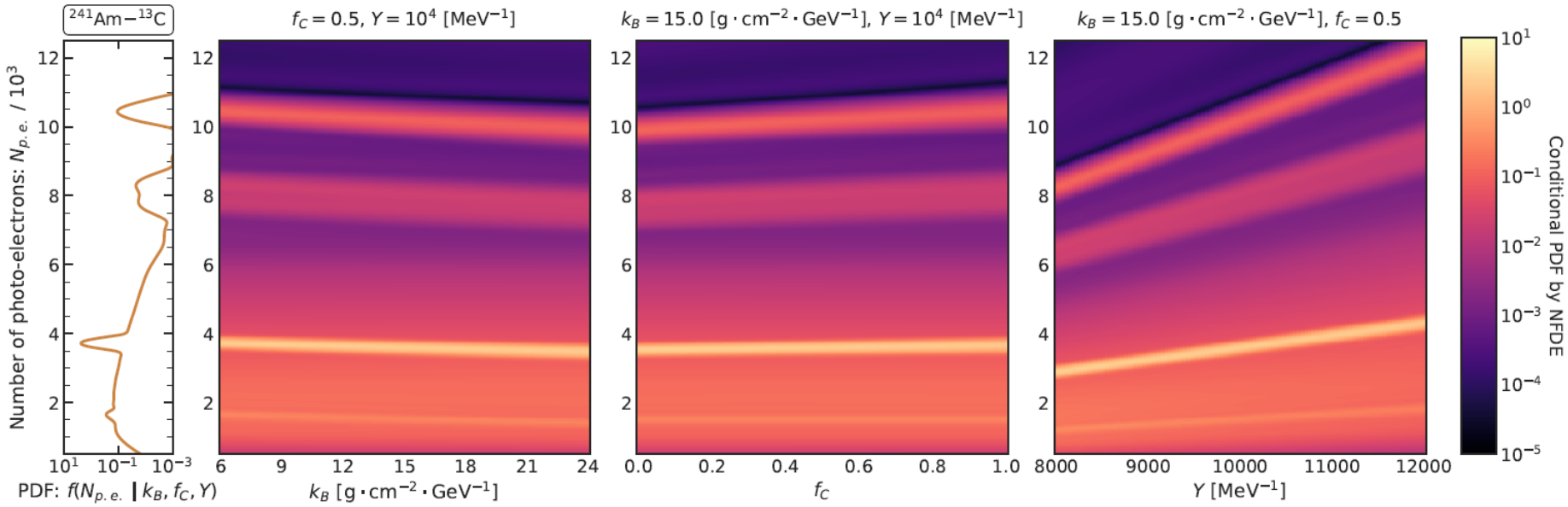


*all fits performed with Minuit, MCMC (emcee), Nested Sampling (ultranest) → consistent results

Learned dependences

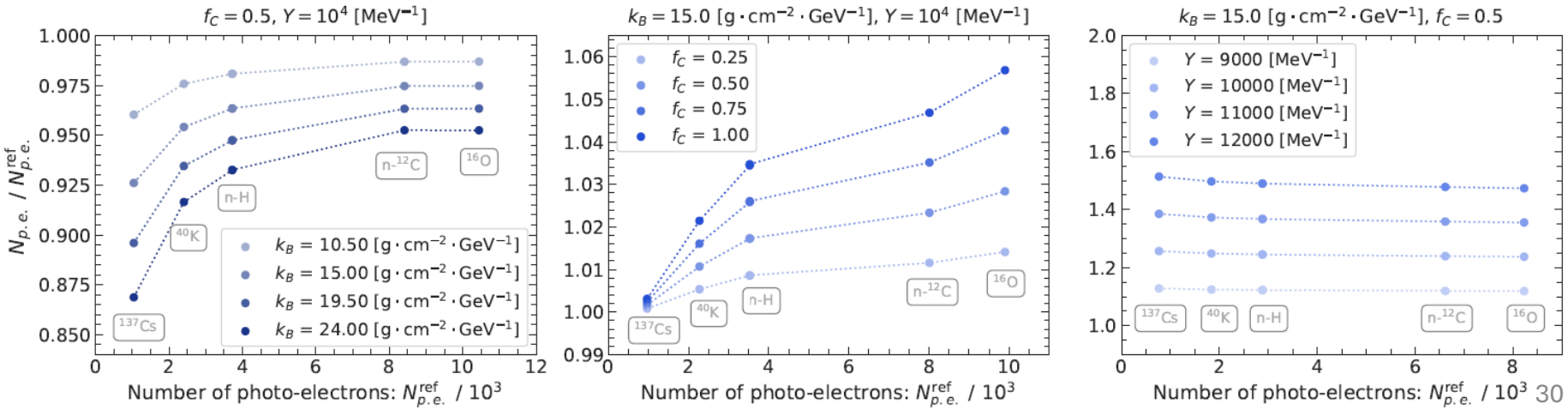
1) PDF vs. parameter

(almost linear)



2) **PDFs vs. energy (NPE)**

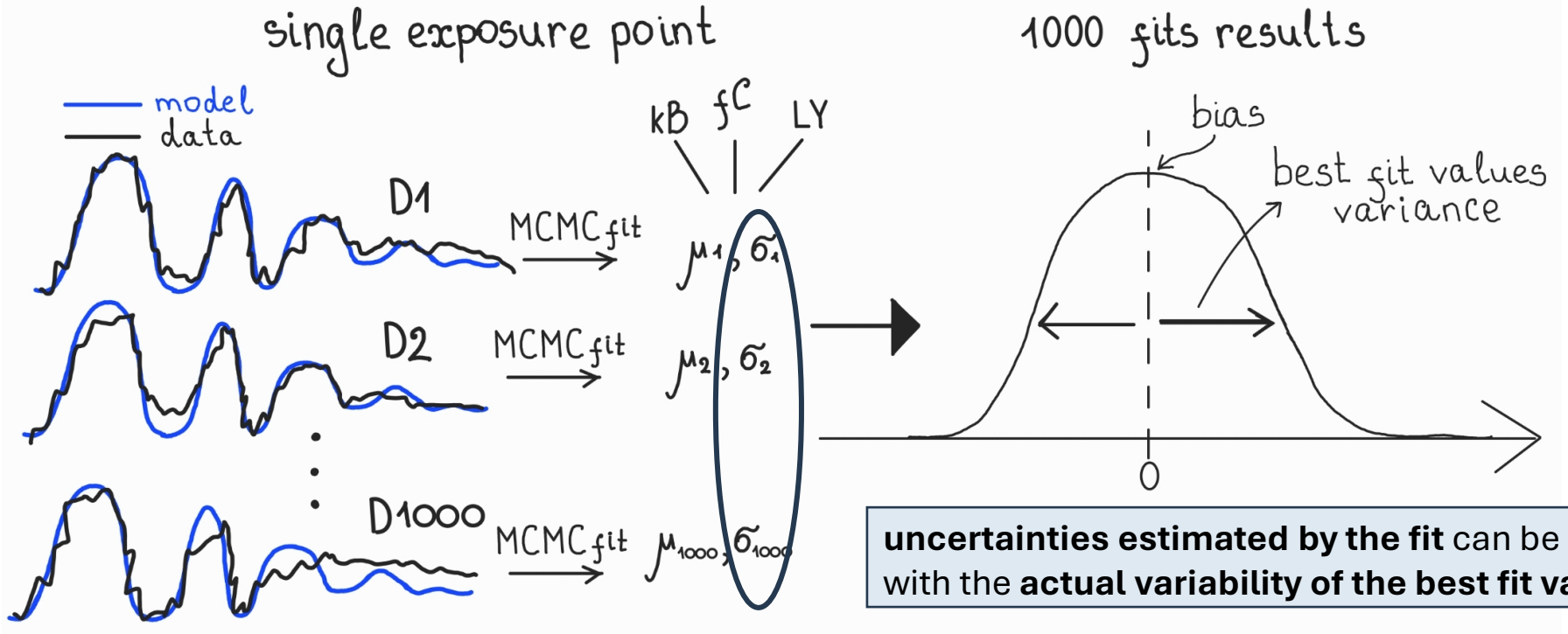
(energy non-linearity)



Evaluate ML-driven systematic uncertainty

To perform systematic uncertainty estimation analysis, we use the testing dataset 2**:**

- **Unseen during training point** in the parameter space: kB, fC, LY = (15.45, 0.525, 10100)
- **5 different exposures**: 1k, 2k, 5k, 10k, 25k events
- **1000 datasets with different MC generator seed** per each exposure



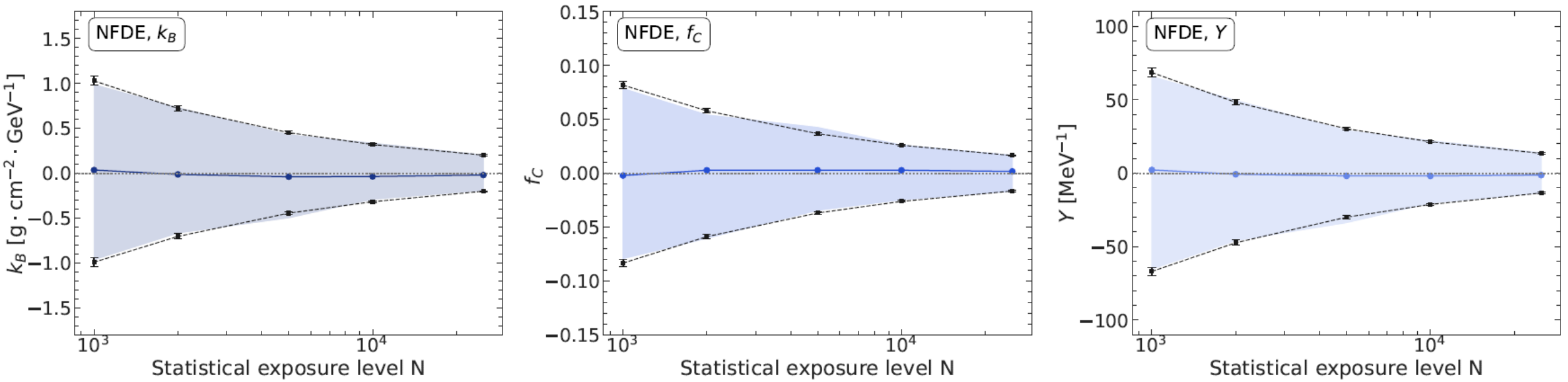
uncertainties estimated by the fit can be compared with the **actual variability of the best fit values**

Estimation of ML-driven systematic uncertainty

To perform systematic uncertainty estimation analysis, we use the testing dataset 2**:**

- **Unseen during training point** in the parameter space: kB, fC, LY = (15.45, 0.525, 10100)
- **5 different exposures**: 1k, 2k, 5k, 10k, 25k events
- **1000 datasets with different MC generator seed** per each exposure

→ **Estimated uncertainty matches observed variability**

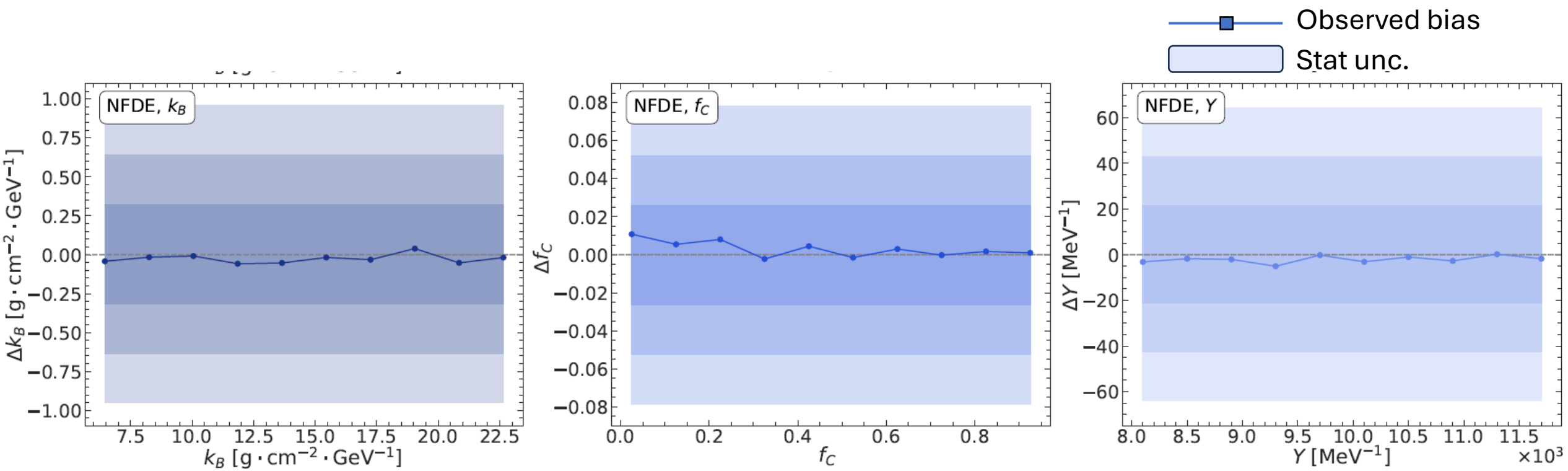


Estimation of ML-driven systematic bias

Using **the testing dataset 1,** one can check the bias across **different points**:

- Run MCMC fits per each testing point of the dataset
- Compare bias with the uncertainty **obtained by the previous analysis** for the 10k exposure point

→ **Observed biases are well within the uncertainty**



Summary

Challenge: systematics investigation for precision **neutrino physics with JUNO and** accurate energy calibration

→ requires an extremely **well tuned MC**, but the **traditional MC tuning** process is **computationally prohibitive**.

Summary

Challenge: systematics investigation for precision **neutrino physics with JUNO and** accurate energy calibration

→ requires an extremely **well tuned MC**, but the **traditional MC tuning** process is **computationally prohibitive**.

Solution: we developed a **Simulation-Based Inference framework** using two **neural likelihood estimators** (TEDE and NFDE) that replace the slow MC simulation with a **fast, accurate surrogate** models.

Summary

Challenge: systematics investigation for precision **neutrino physics with JUNO and** accurate energy calibration

→ requires an extremely **well tuned MC**, but the **traditional MC tuning** process is **computationally prohibitive**.

Solution: we developed a **Simulation-Based Inference framework** using two **neural likelihood estimators** (TEDE and NFDE) that replace the slow MC simulation with a **fast, accurate surrogate** models.

Key Achievements:

- our **models successfully learn** the complex, **non-linear energy response of the JUNO detector MC**, including the strong correlations between $\boldsymbol{k_B}$, $\boldsymbol{f_C}$ and $\boldsymbol{Y}$ parameters in the MC.
- when integrated with **Bayesian inference**, our method recovers all parameters with **near-zero systematic bias**.
- parameter **uncertainties are limited purely by statistics** and scale correctly as $1/\sqrt{N}$

  → possible to achieve <1% systematics

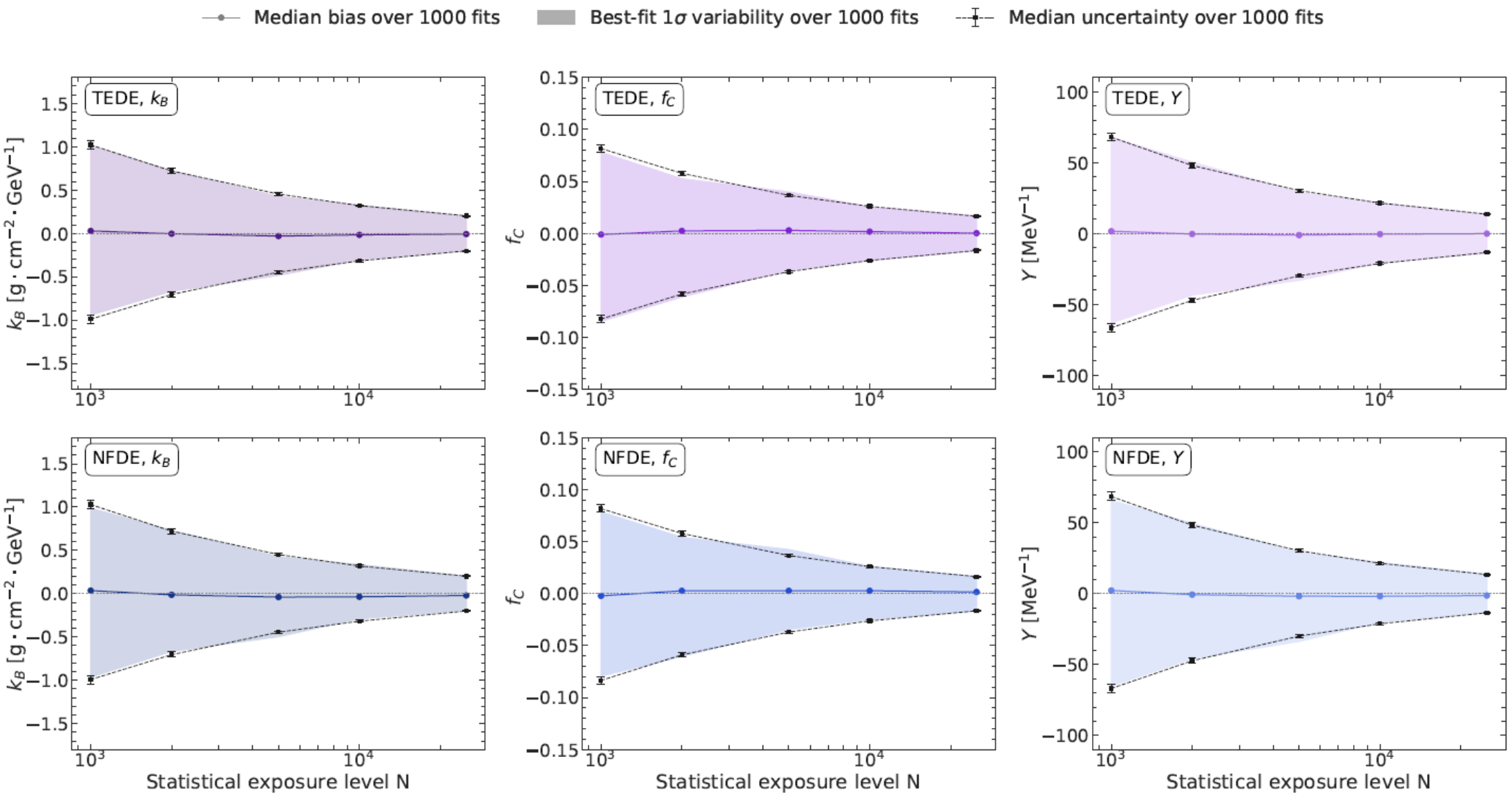
This provides a flexible (binned or unbinned) framework for MC tuning with data! ☺

Thanks!

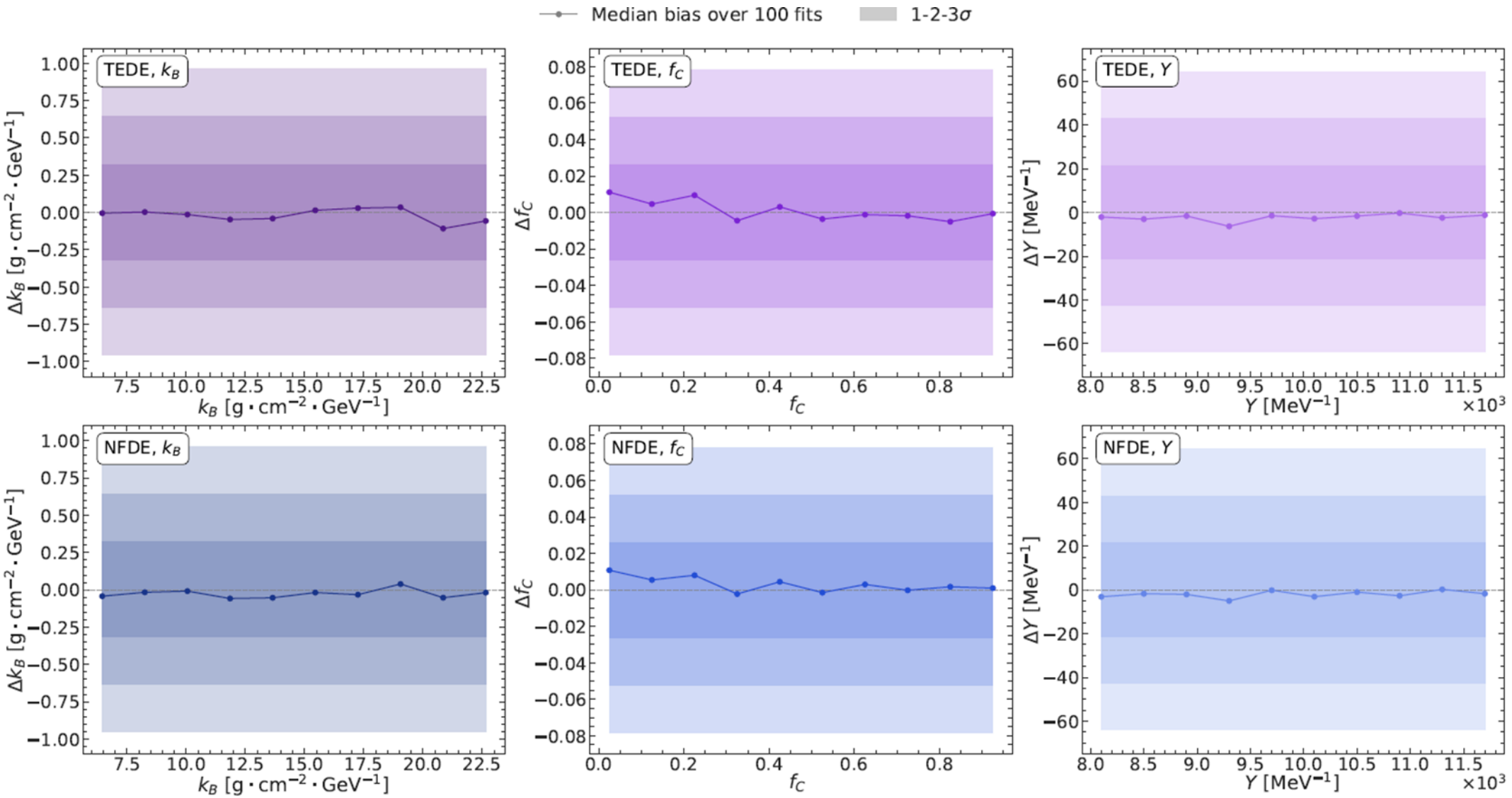
Repo

Pre-print

TEDE vs. NFDE (uncertainty)



TEDE vs. NFDE (bias)



Hyperparameters optimization

Hyperparameters	TEDE	NFDE
Model architecture hyperparameters		
n_{tokens}	$\{k \mid k \in \mathbb{N}, 1 \leq k \leq 5\}$: **1**	———
d_{model}	$\{50k \mid k \in \mathbb{N}, 1 \leq k \leq 10\}$: **100**	———
n_{head}	$\{5, 10, 25\}$: **10**	———
n_{layers}	$\{k \mid k \in \mathbb{N}, 1 \leq k \leq 5\}$: **4**	———
d_{ff}	$\{32k \mid k \in \mathbb{N}, 1 \leq k \leq 15\}$: **384**	———
n_{flows}	———	$\{100 + 5k \mid k \in \mathbb{Z}, 0 \leq k \leq 20\}$: **120**
n_{units}	———	$\{10 + 5k \mid k \in \mathbb{Z}, 0 \leq k \leq 8\}$: **30**
Flow type	———	**`Planar`** (fixed)
Dropout rate	$\{p_{\text{drop}} \mid 0.0 \leq p_{\text{drop}} \leq 0.5\}$: **0.0**	———
Activation function [90–93]	`ReLU`, **`GeLU`**	`ReLU`, **`GeLU`**, `Tanh`, `SiLU`
Softmax temperature	$\{T \mid 0.25 \leq T \leq 3.0\}$: **2.03**	———
General training hyperparameters		
Optimizer [94, 95]	**`AdamW`**, `RMSprop`	**`AdamW`** (fixed)
Learning rate	$\{\eta \mid 10^{-5} \leq \eta \leq 10^{-2}\}$: $\mathbf{1.9 \times 10^{-3}}$	10^{-4} (fixed)
Scheduler type [96, 97]	**`Exponential`**, `ReduceOnPlateau`, `CosineAnnealing`, `None`	`ReduceOnPlateau`, **`CosineAnnealing`**
Weight decay	$\{\lambda \mid 10^{-6} \leq \lambda \leq 10^{-1}\}$: $\mathbf{8 \times 10^{-5}}$	10^{-4} (fixed)
Batch size	$\{2^k \mid k \in \mathbb{N}, 4 \leq k \leq 9\}$: **64**	**50** (fixed, 1 per a CPU unit)
Optimizer-specific hyperparameters		
`AdamW` decay rate β_1	$\{\beta_1 \mid 0.5 \leq \beta_1 \leq 0.95\}$: **0.930**	$\{\beta_1 \mid 0.5 \leq \beta_1 \leq 0.95\}$: **0.87**
`AdamW` decay rate β_2	$\{\beta_2 \mid 0.9 \leq \beta_2 \leq 0.999\}$: **0.929**	$\{\beta_2 \mid 0.9 \leq \beta_2 \leq 0.999\}$: **0.901**
`RMSprop` parameter α	$\{\alpha \mid 0.9 \leq \alpha \leq 0.999\}$: –	———
Scheduler-specific hyperparameters		
Cosine annealing period $T_{\max}$	$\{T_{\max} \mid 5 \leq T_{\max} \leq 75\}$: –	$\{T_{\max} \mid 5 \leq T_{\max} \leq 75\}$: **50**
Exponential rate τ	$\{\tau \mid 0.8 \leq \tau \leq 0.99\}$: **0.92**	———
Reduce on plateau factor γ	$\{\gamma \mid 0.8 \leq \gamma \leq 0.99\}$: –	$\{\gamma \mid 0.8 \leq \gamma \leq 0.99\}$: –
Optimization procedure setting		
Number of trials	**250**	**25**
Accelerator	GPU	50 parallel CPUs

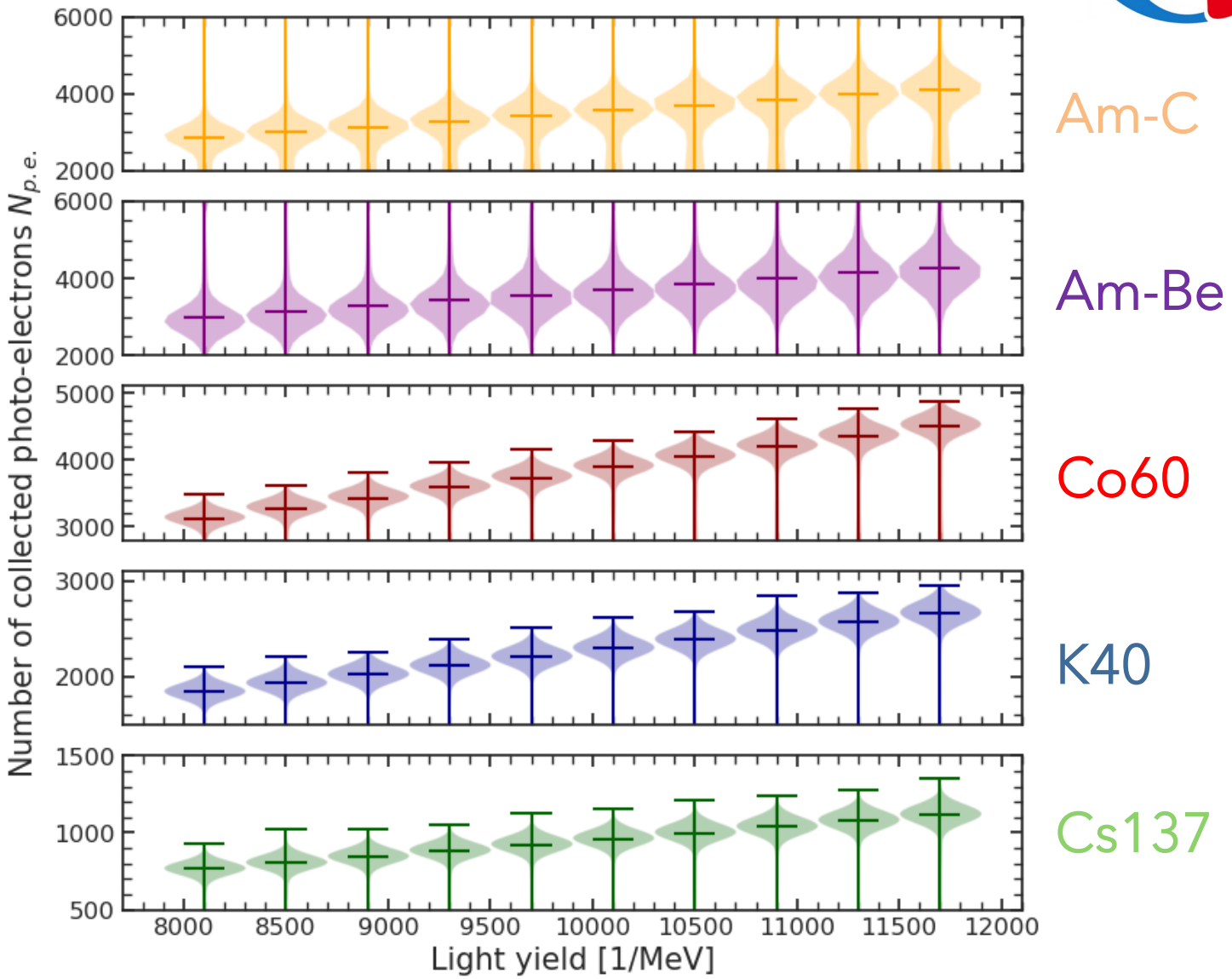
How LS parameters impact the calibration data

LY effect

- **kB and fC are fixed:**
  - kB = 15.45 [g/cm^2/GeV]
  - fC = 0.525
- **LY is varying**

- Light yield is the most influential parameter
- All sources are highly affected

Only main peaks are shown



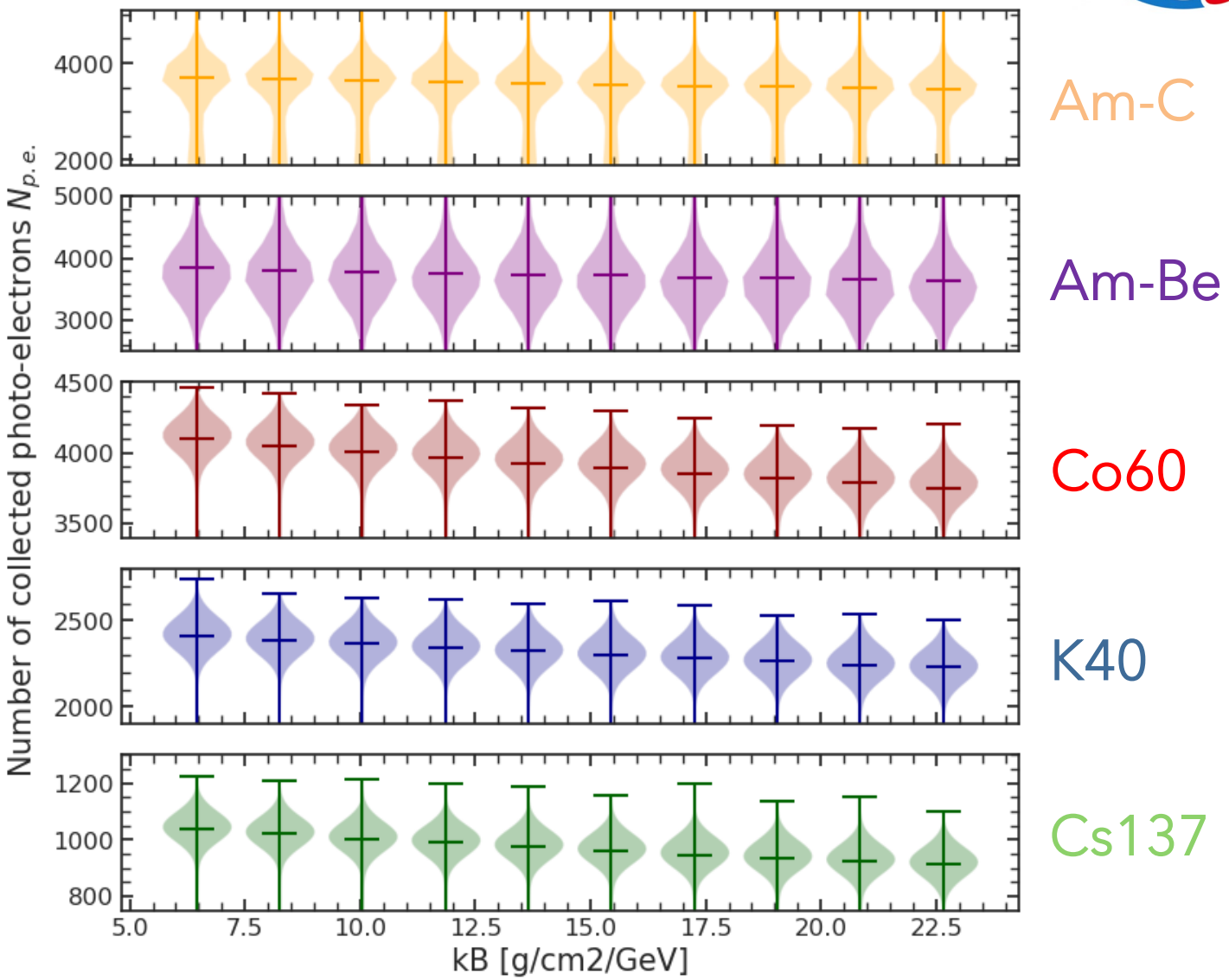
How LS parameters impact the calibration data

kB effect

- **LY and fC are fixed:**
  - LY = 10100 [1/MeV]
  - fC = 0.525
- **kB is varying**

- kB effect is smaller than LY and **anticorrelated with the photo peak**
- All sources are affected

Only main peaks are shown



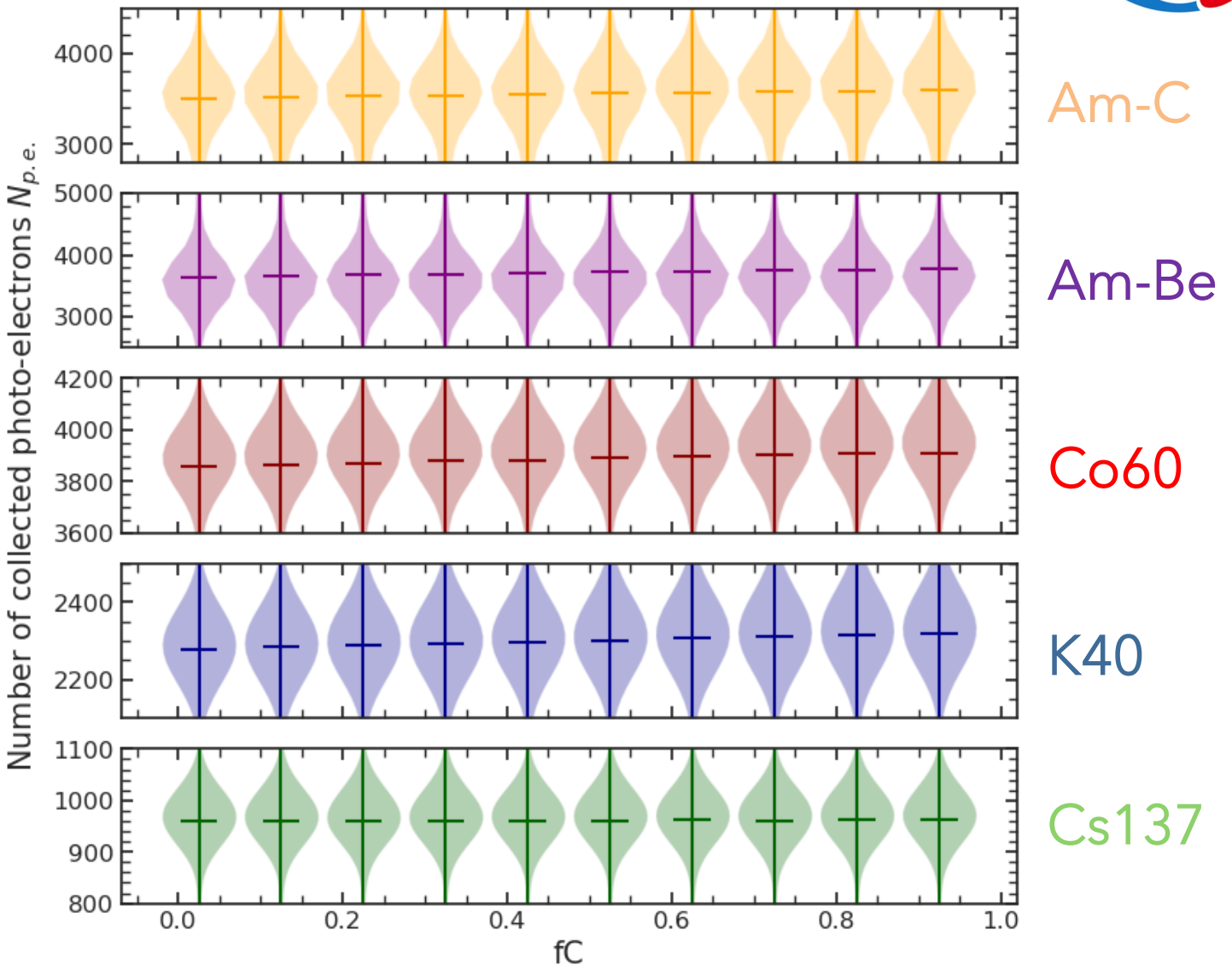
How LS parameters impact the calibration data

fC effect

- **kB and LY are fixed:**
  - kB = 15.45 [g/cm^2/GeV]
  - LY = 10100 [1/MeV]
- **fC is varying**

- fC has **a minor effect** to the spectra
- Cs137 **is not affected at all**
- **Slight effect** for Co60 and K40

Only main peaks are shown

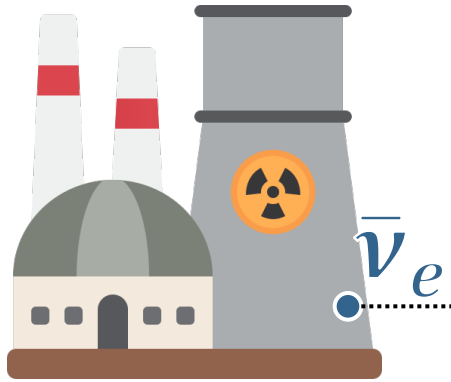


The JUNO detection process

JUNO will measure the **antineutrinos** ($\bar{\nu}_e$) generated in the fissions occurring in 8 nuclear cores at 52.5 km

The **detection** is based on a charged current interaction named Inverse Beta Decay (**IBD**) on protons (p)

→ **sensitive only to electron $\bar{\nu}_e$**

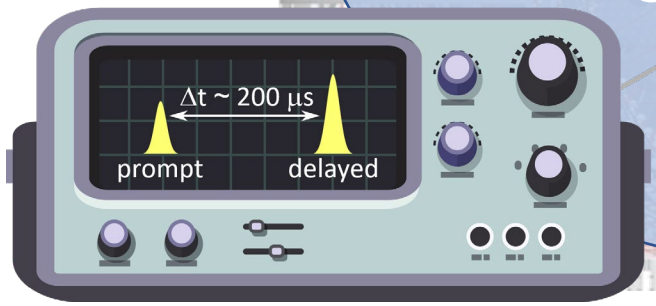


$E_{e^+} \sim E_{\bar{\nu}} - 0.78\ MeV$

Detection relies on a **double coincidence**:

- **prompt** signal: positron (e^+) annihilation
- **delayed** signal: neutron (n) capture

→ **strong handle against most backgrounds**

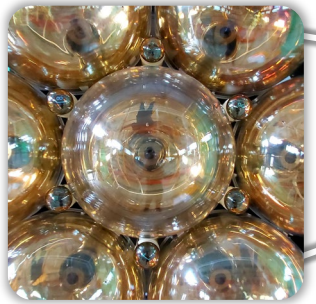


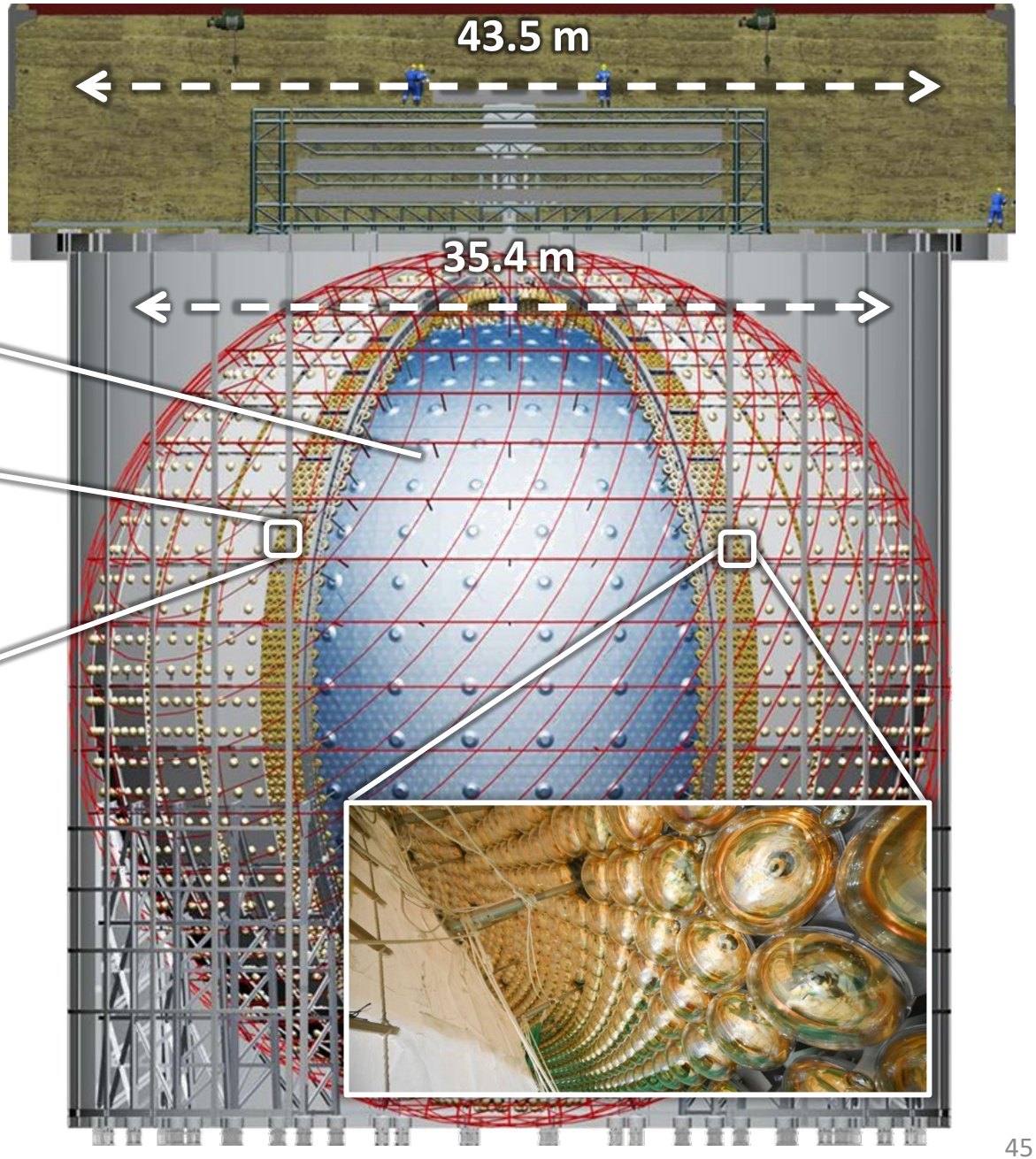
The JUNO detector

[Prog. Part. Nucl. Phys. 2021.103927]

Main requirements:

- **high statistics**
  → 20 kton of liquid scintillator acrylic sphere
- **<3% energy resolution @ 1 MeV**
  → photocoverage ~78%
- **energy-scale systematics below 1%**
  → 17612 20" Large-PMT
  → 25600 3" Small-PMT





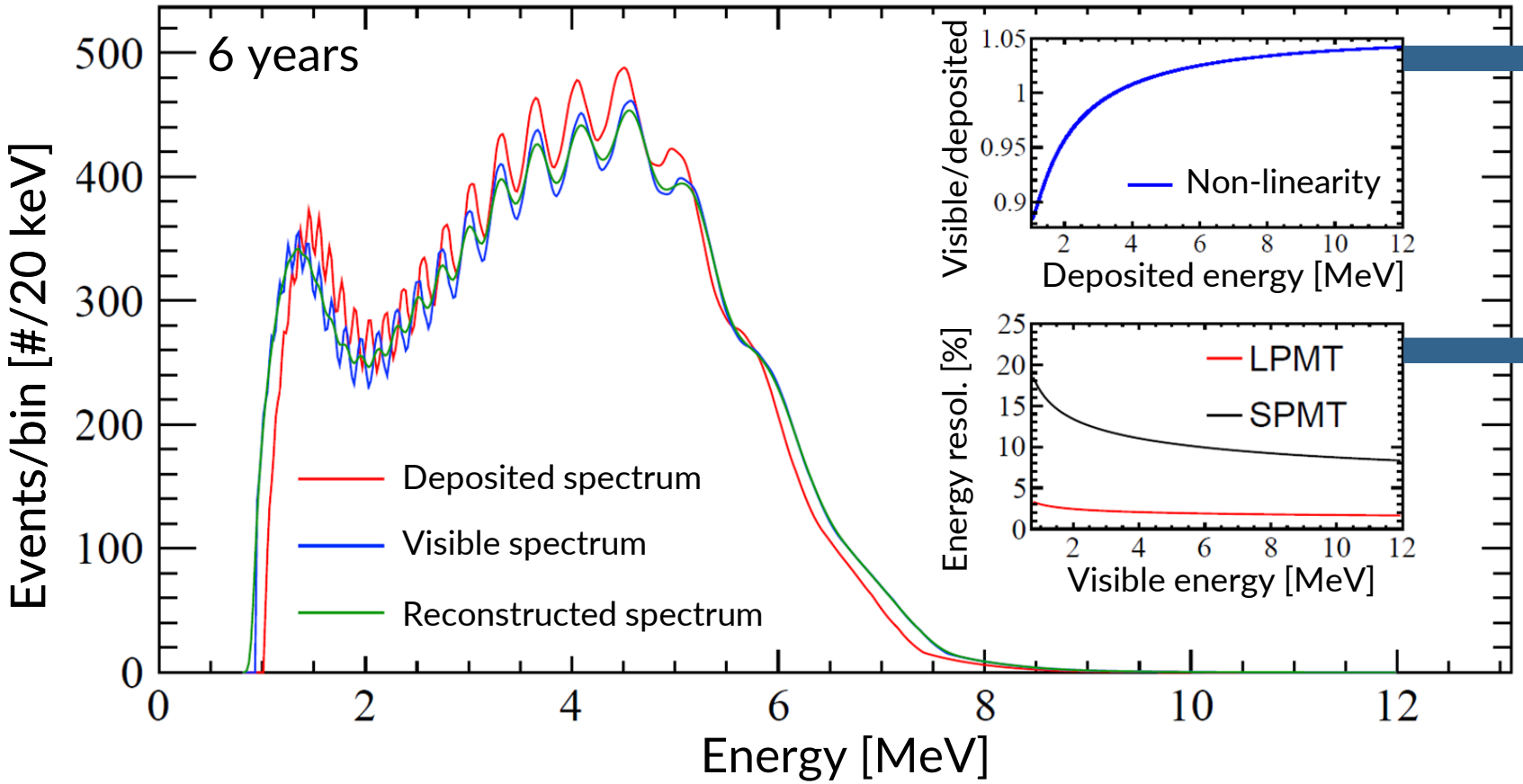
	Target mass [kton]	Energy resolution	Light yield [PE/MeV]
Daya Bay	0.02	8%/√E	160
Borexino	0.3	5%/√E	500
KamLAND	1	6%/√E	250
JUNO	**20**	**3%/√E**	**~1600**

Detector response: what JUNO actually sees

interaction → light emission → light detection

E_ν → E_{dep} → E_{vis} → E_{rec}

Antineutrino energy | **Deposited energy** | **Visible energy** | **Reconstructed energy**



Calibration campaigns

- **automated** multiple-**position** and multi-**source calibration** (link)
- **periodic calibration** campaigns
- **dual-calorimetry** system (link)

Energy resolution

$$\frac{\sigma}{E} = \sqrt{\left(\frac{a}{\sqrt{E}}\right)^2 + b^2 + \left(\frac{c}{E}\right)^2}$$

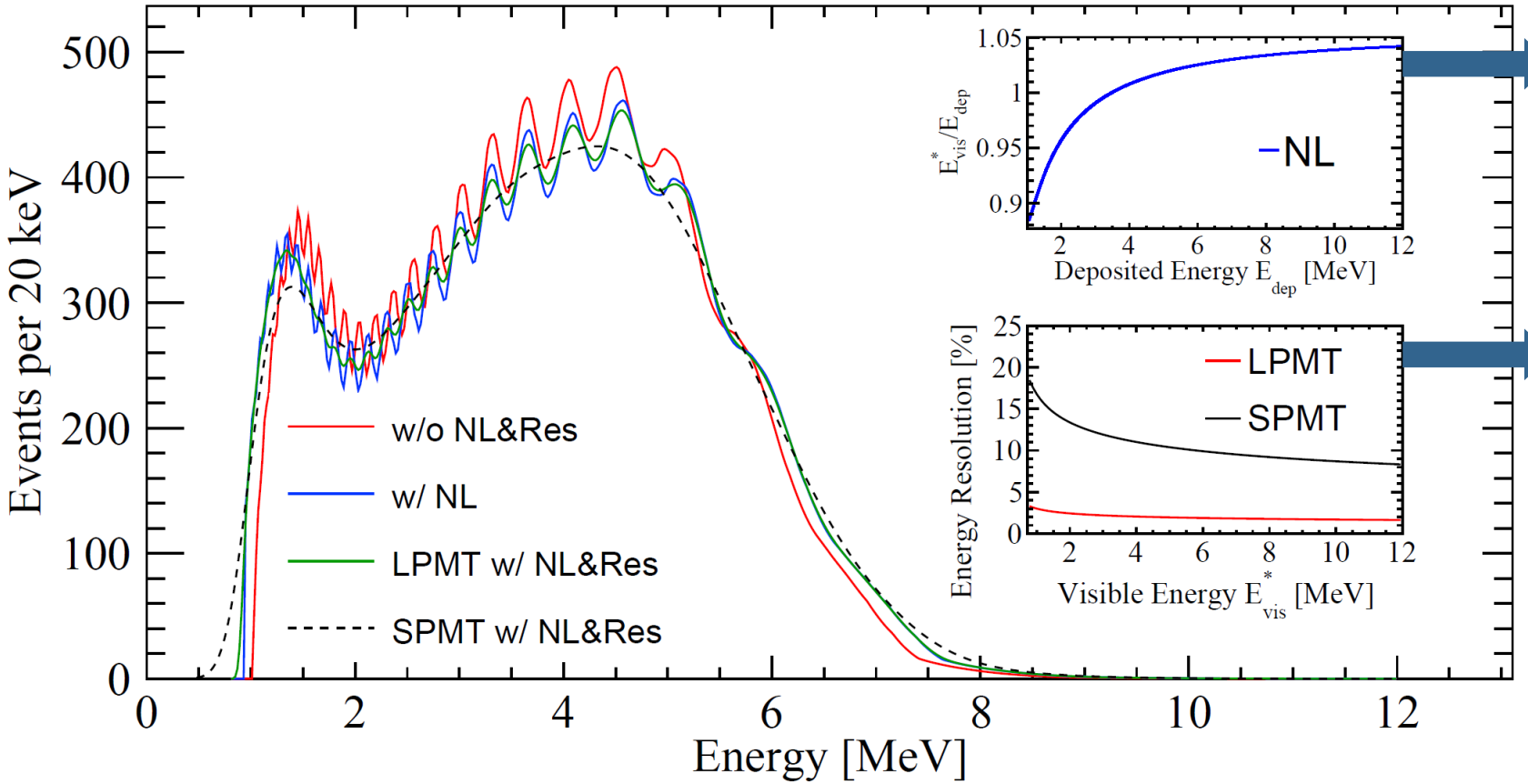
- a **Stochastic term**: light yield (from source calibration)
- b Dominated by **non-uniformity** (from multi-source calibration)
- c PMT **dark noise**

Detector response: what JUNO actually sees

interaction → light emission → light detection

E_ν → E_{dep} → E_{vis} → E_{rec}

Antineutrino energy — **Deposited energy** — **Visible energy** — **Reconstructed energy**



Calibration campaigns

- **automated** multiple-**position** and multi-**source calibration** (link)
- **periodic calibration** campaigns
- **dual-calorimetry** system (link)

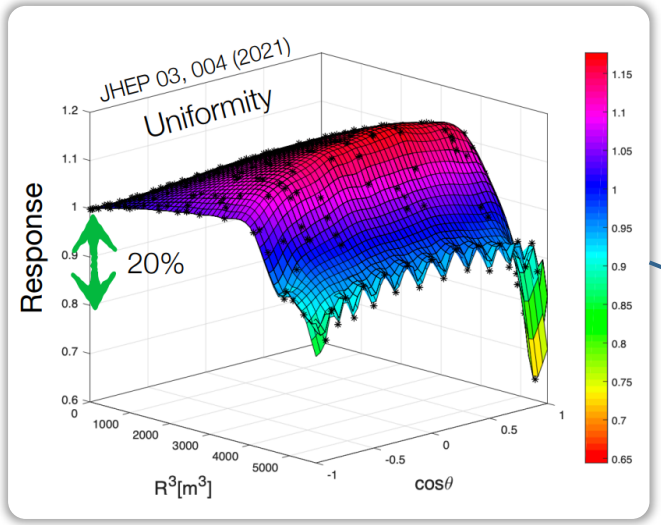
Energy resolution

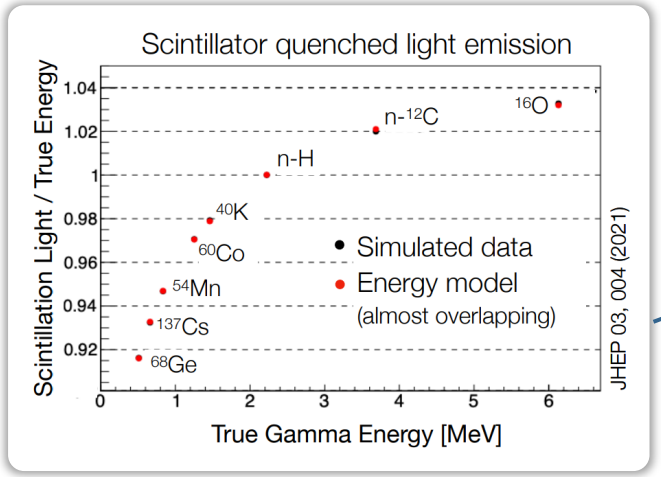
$$\frac{\sigma}{E} = \sqrt{\left(\frac{a}{\sqrt{E}}\right)^2 + b^2 + \left(\frac{c}{E}\right)^2}$$

- a **Stochastic term**: light yield (from source calibration)
- b Dominated by **non-uniformity** (from multi-source calibration)
- c PMT **dark noise**

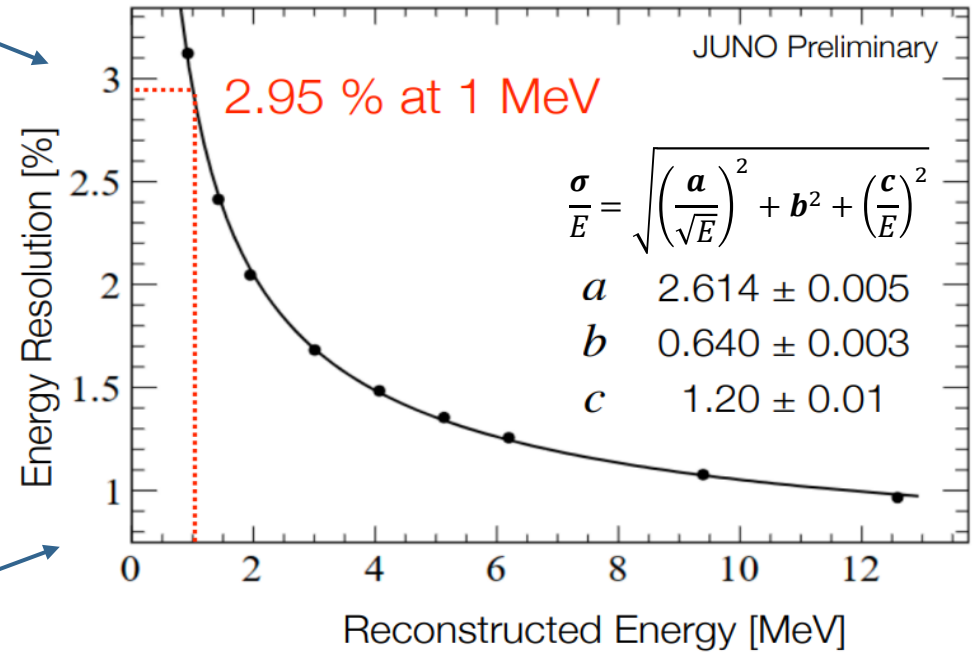
JUNO detector response: state of the art

Realistic MC simulation

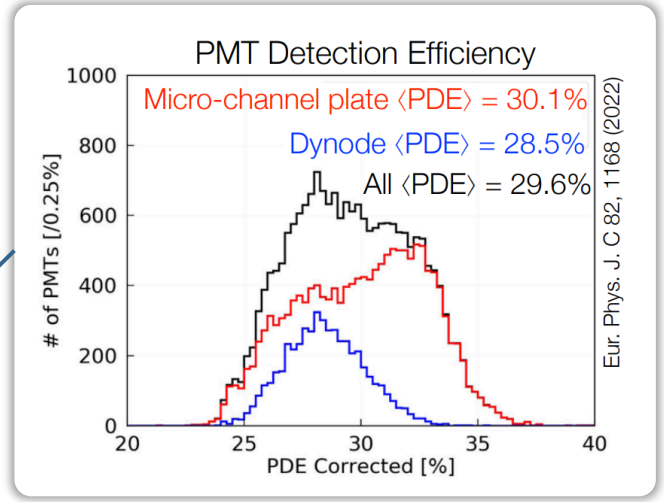


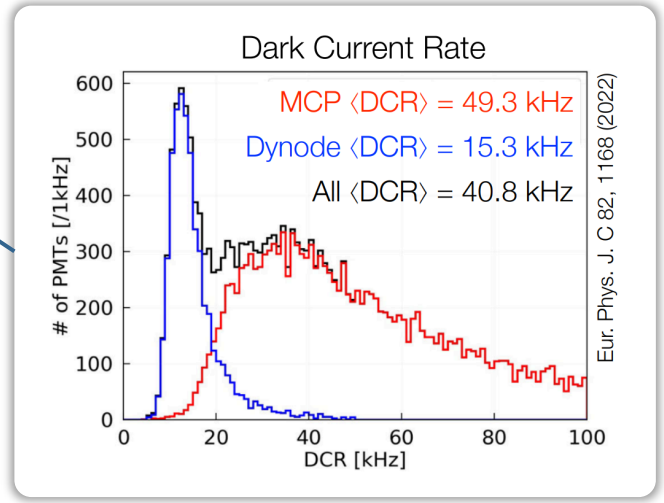


Updated values based on **commissioning data** and **realistic MC simulation**



Measured data

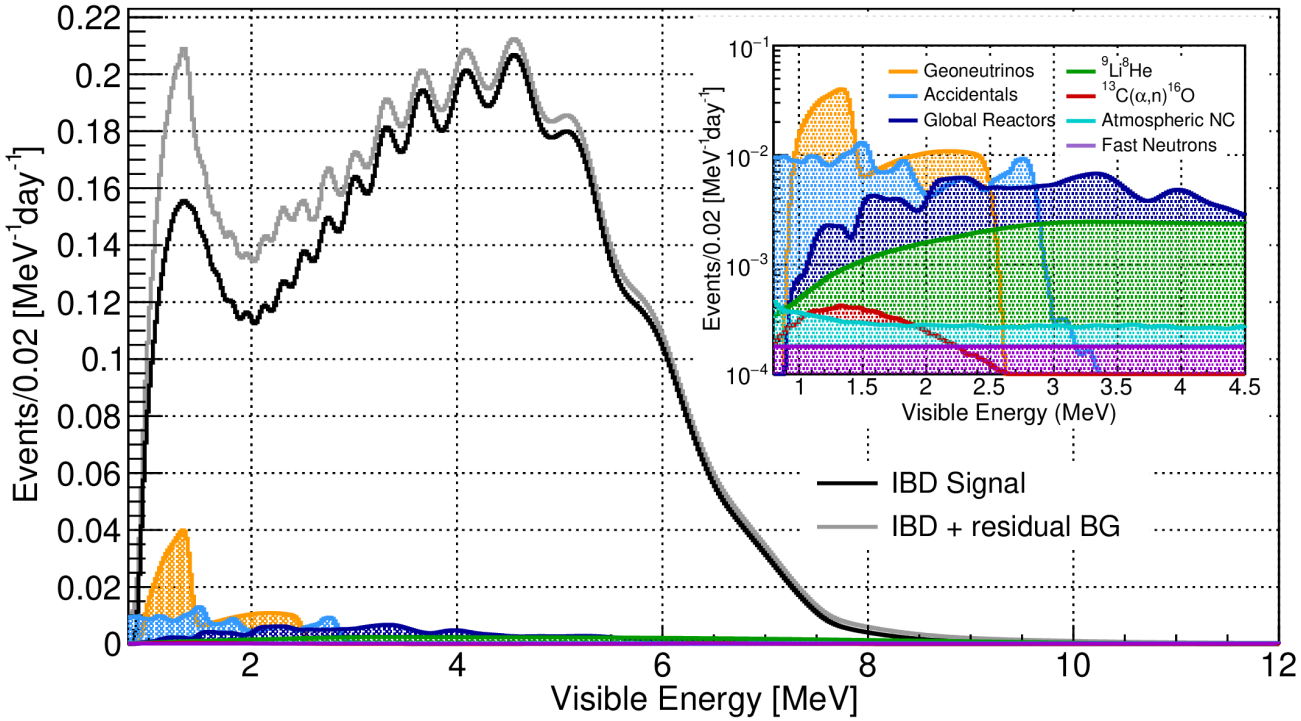




IBD backgrounds in JUNO

JUNO employs various **selection cuts** to retain **high efficiency** and assure **high purity** in the IBD signal:

- **Cosmogenic** backgrounds → **muon veto**
- **Accidental coincidences** → **fiducial volume + IBD cuts**
- **Irreducible** backgrounds → **negligible** (~1/20 of signal)



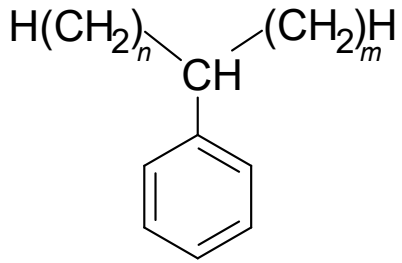
IBD selection cuts	Efficiency [%]	IBD rate [day^{-1}]
All IBDs	**100.0**	**57.4**
Fiducial volume	91.5	52.5
IBD selection	98.1	51.1
Energy range	99.8	-
Time correlation ($\Delta T_{p\text{-}d}$)	99.0	-
Spatial correlation ($\Delta R_{p\text{-}d}$)	99.2	-
Muon veto (Time+spatial)	91.6	47.1
Combined selection	**82.2**	**47.1**

Residual backgrounds	Rate [day^{-1}]	Rate unc. [%]	Shape unc. [%]
Geoneutrinos	1.2	30	5
World reactors	1.0	2	5
Accidentals	0.8	1	negligible
^{9}Li/^{8}He	0.8	20	10
Atmospheric neutrinos	0.16	50	50
Fast neutrons	0.1	100	20
^{13}C(α,n)^{16}O	0.05	50	50
Total background	**4.11**	-	-

Detection channels in JUNO

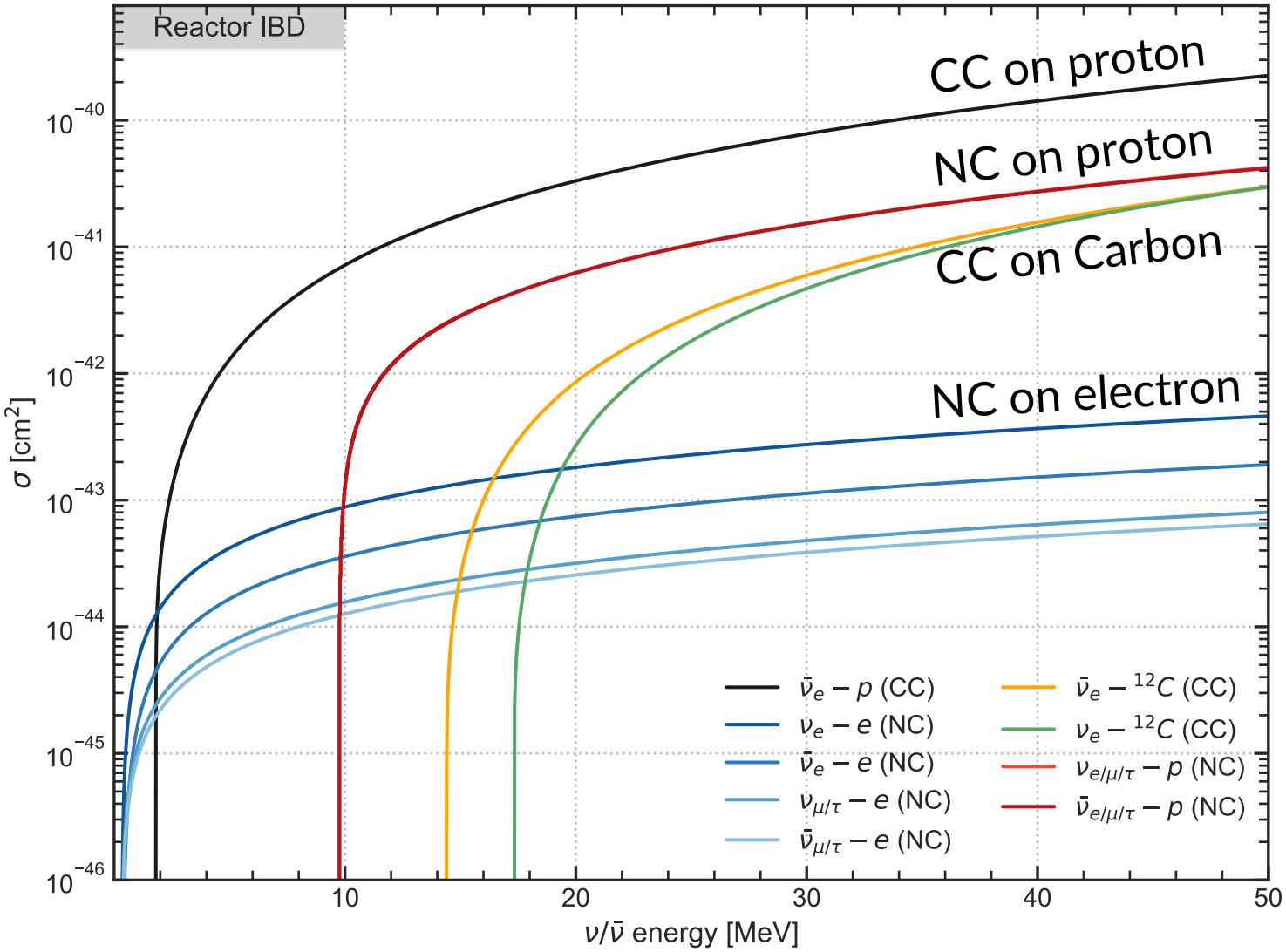
Liquid scintillator:
Linear alkylbenzenes

C – 88%
H – 12%



- $\bar{\nu}_e + p \rightarrow e^+ + n$
- $\nu + p \rightarrow \nu + p$
- $\nu + e \rightarrow \nu + e$
- $\nu_e + {}^{12}C \rightarrow e^- + {}^{12}N$
- $\bar{\nu}_e + {}^{12}C \rightarrow e^+ + {}^{12}B$

NC recoil threshold: 200 keV



Photomultiplier Tubes

5000 x 20” Hamamatsu R12860

High QE: 28.5%
Fine TTS: 1.3 ns

15012 x 20” NNVT MCP-PMTs

Highest QE: 30.1%
Good TTS: 7.0 ns

High efficiency of photon detection

25600 x 3” HZC XP72B22

- Calibration of 20” PMTs’ non-linearities
- Extension of dynamic range

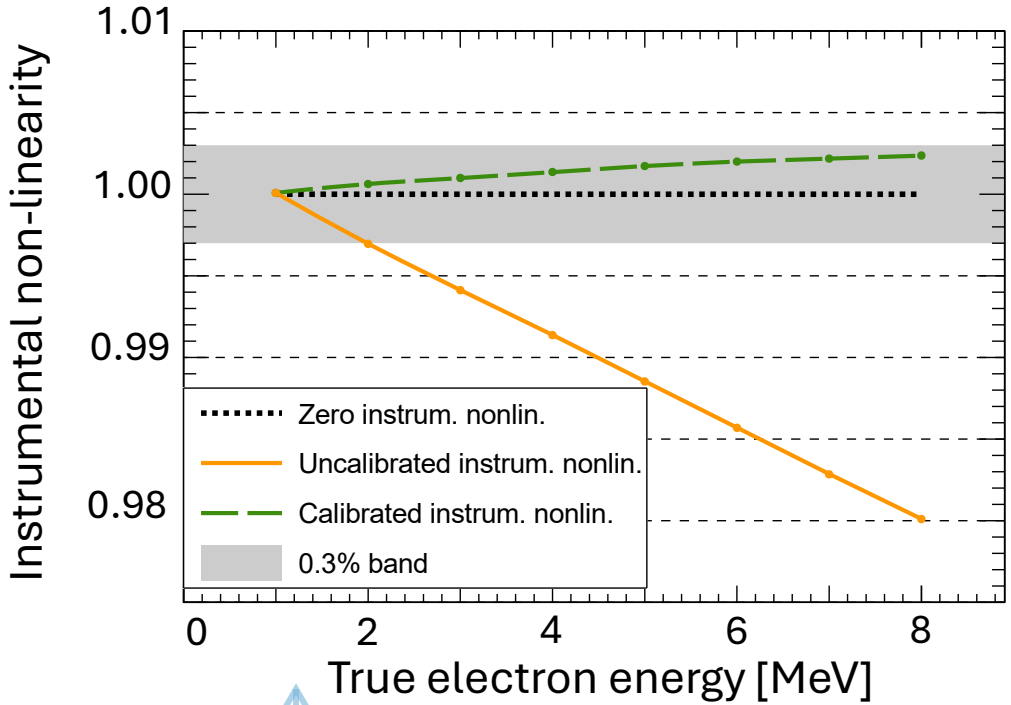
Dual calorimetry system

JUNO employs the dual calorimetry system to correct for electronics non-linearity

Sources of non-linear response:

- Non-uniformity (NU)
- Liquid scintillator non-linearity (LSNL)
- Charge non-linearity (QNL) of L-PMTs

Simulations with extreme channel-level non-linearity of 50% over 100 PE show that L-PMT response (R) non-linearity can be corrected using S-PMT as calibration reference:



$$R^{L-PMT} = R_{LSNL} \cdot R_{NU}^{L-PMT} \cdot R_{QNL}^{L-PMT}$$

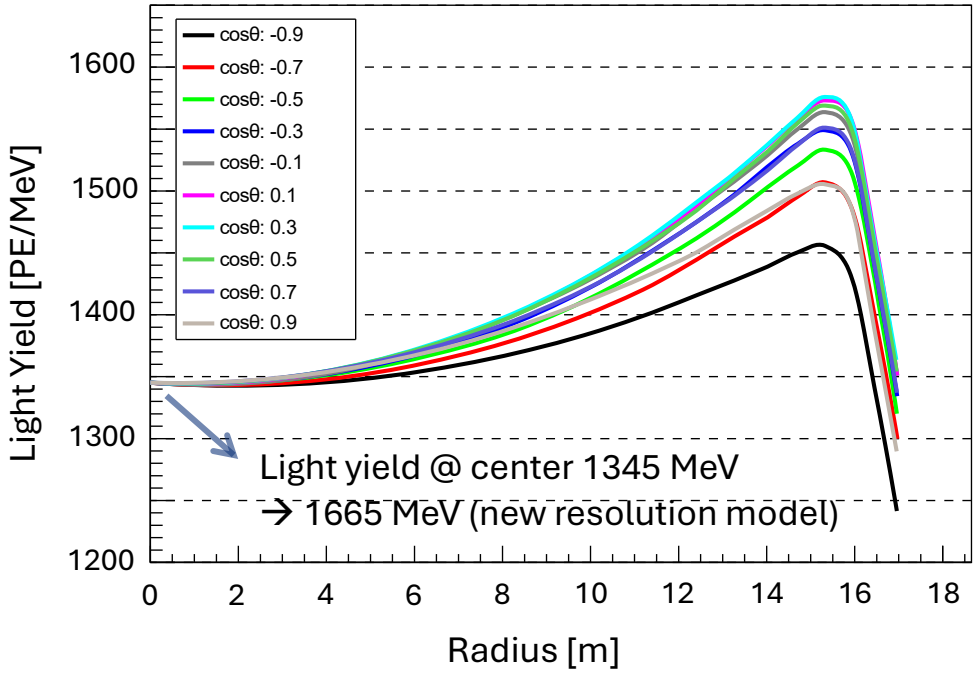
$$R^{S-PMT} = R_{LSNL} \cdot R_{NU}^{S-PMT} \cdot R_{QNL}^{S-PMT}$$

$$\frac{R^{L-PMT}}{R^{S-PMT}} = \frac{R_{QNL}^{L-PMT}}{R_{QNL}^{S-PMT}}$$

Other non-linearities

Detector non-uniformity

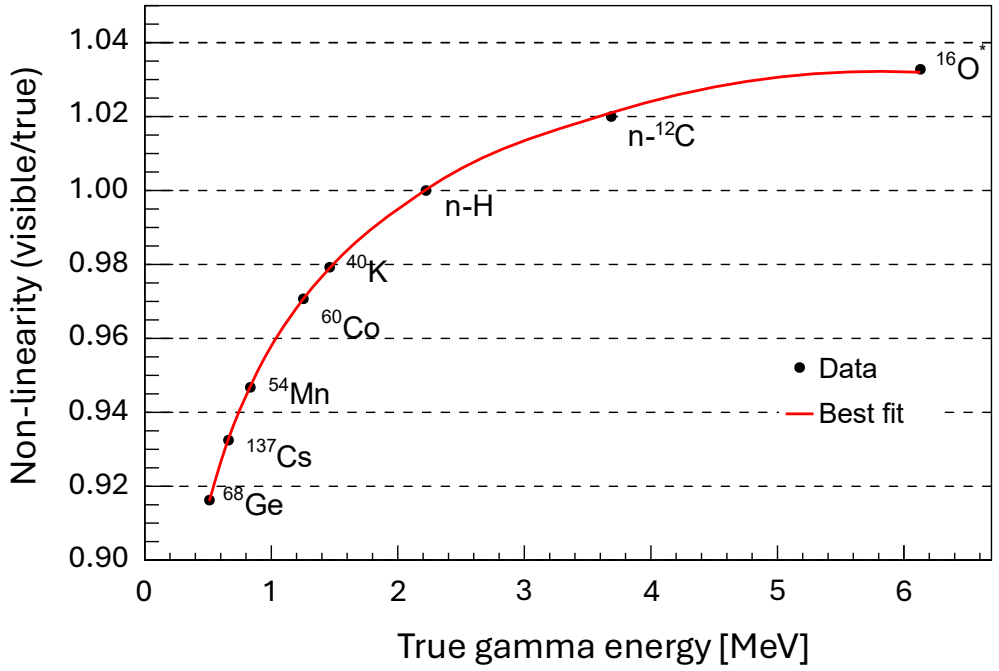
The detector response to the same charge deposition depends on the position at which the event occurs and needs to be properly characterized.



Liquid scintillator non-linearity

Light emission has an intrinsic non-linearity because of:

- Birks' quenching effect in scintillation photon yield;
- Velocity-dependent Cherenkov emission.



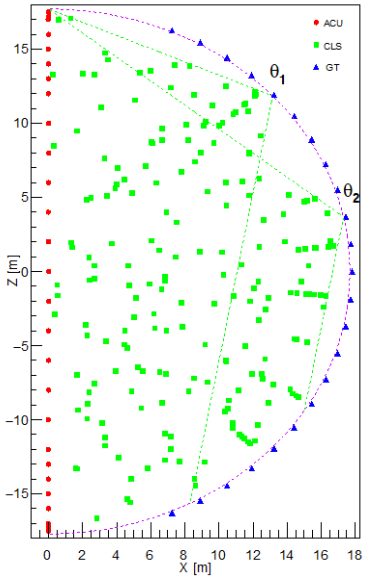
Calibration of the JUNO detector

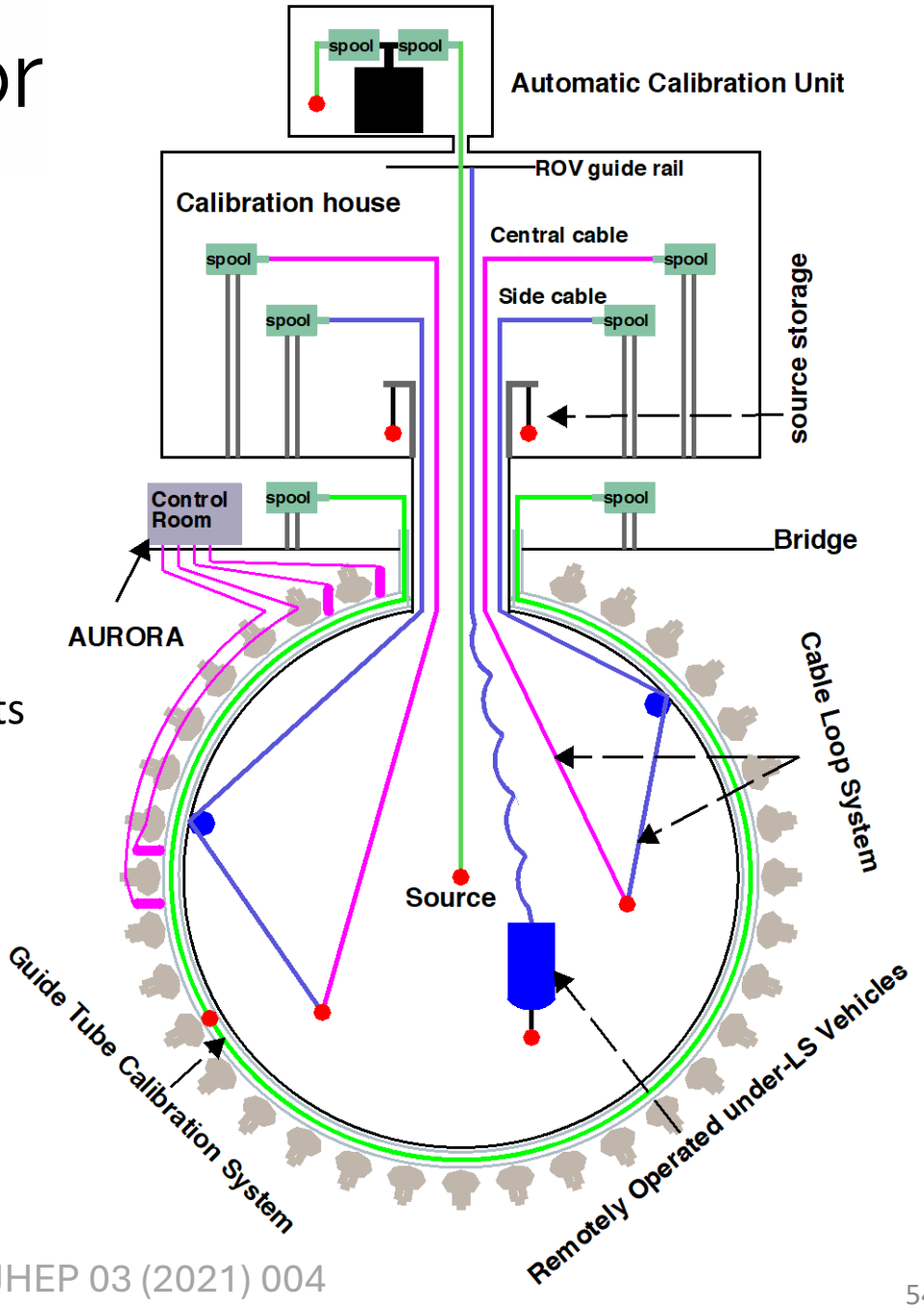
Radioactive sources (100-200 Hz) + Laser sources

- 1D: Automatic Calibration Unit (ACU)
- 2D: Cable Loop System (CLS)
- 3D: Remotely Operated under-LS Vehicles (ROV)
- Boundary: Guide Tube Calibration System (GTCS)

Sources/Processes	Type	Radiation
^{137}Cs	γ	0.662 MeV
^{54}Mn	γ	0.835 MeV
^{60}Co	γ	1.173 + 1.333 MeV
^{40}K	γ	1.461 MeV
^{68}Ge	e^+	annihilation 0.511 + 0.511 MeV
^{241}Am-Be	n, γ	neutron + 4.43 MeV (^{12}C*)
^{241}Am-^{13}C	n, γ	neutron + 6.13 MeV (^{16}O*)
(n,γ)p	γ	2.22 MeV
(n,γ)^{12}C	γ	4.94 MeV or 3.68 + 1.26 MeV

250 calibration points





JHEP 03 (2021) 004

Calibration strategy

Comprehensive calibration (250 points, ~48h)

→ basic understanding of the CD performance

Monthly calibrations (~100 points, ~11h)

→ monitor non-uniformity

Weekly calibrations (~15 points, ~2.4h)

→ track variations in LY of LS, PMT gains, and electronics

JHEP 03 (2021) 004

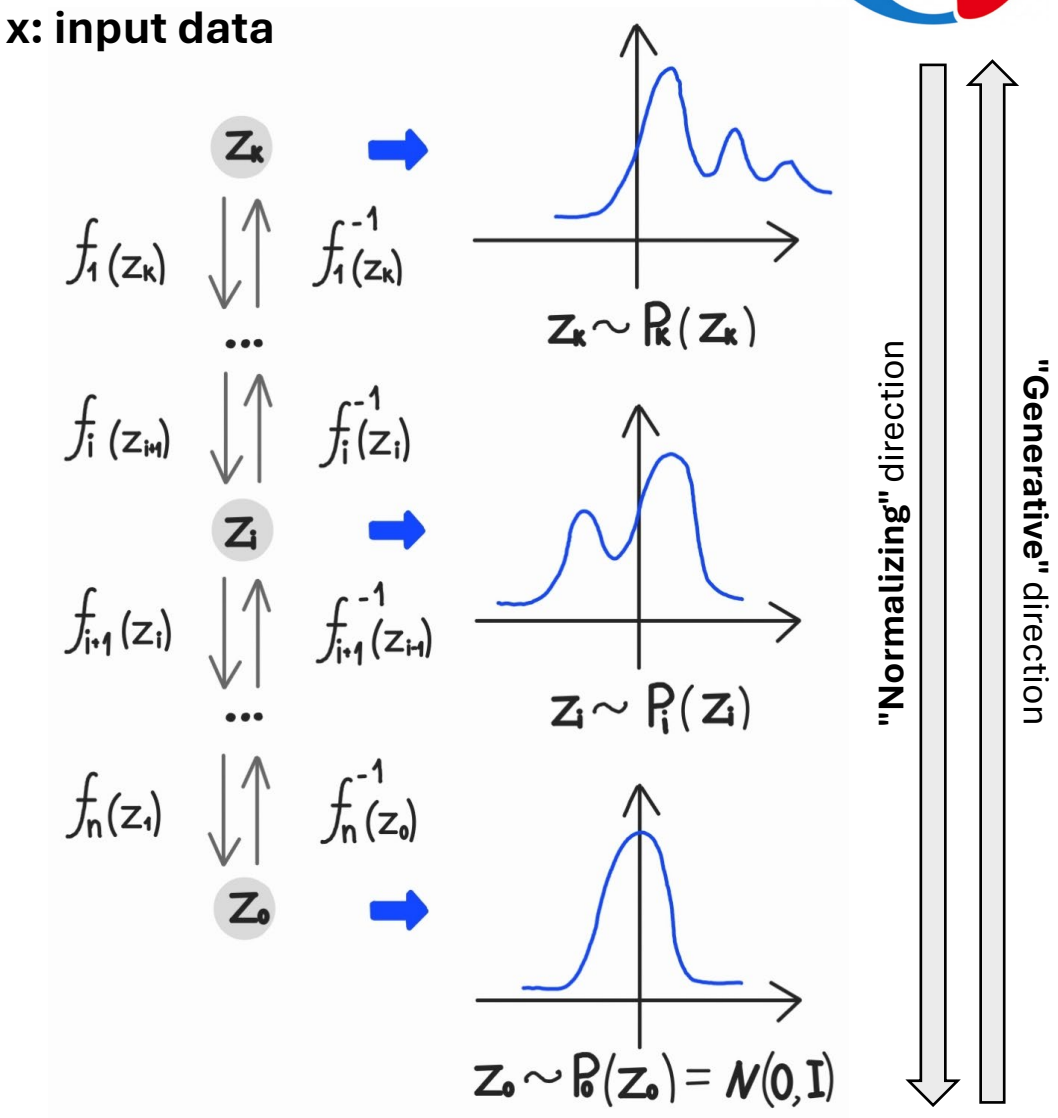
Source	Energy [MeV]	Points
Neutron (Am-C)	2.22	250
Neutron (Am-Be)	4.4	1
Laser	/	10
^{68}Ge	0.511×2	1
^{137}Cs	0.662	1
^{54}Mn	0.835	1
^{60}Co	1.17+1.33	1
^{40}K	1.461	1
Total	/	/

System	Source	Points
ACU	Neutron (Am-C)	27
ACU	Laser	27
CLS	Neutron (Am-C)	40
GT	Neutron (Am-C)	23
Total	/	/

Source	Energy [MeV]	Points
Neutron (Am-C)	2.22	5
Laser	/	10
Total	/	/

Normalizing flows-based density estimation

- **Normalizing flows [1]:**
  - **Generative** machine learning **model**
  - Aims to **explicitly model a density estimation**
  - Can be generalized for modeling **a conditional density estimation** as well
  - To obtain a value from the target distribution:
    - sample a value from the base distribution and passing it through all the flows

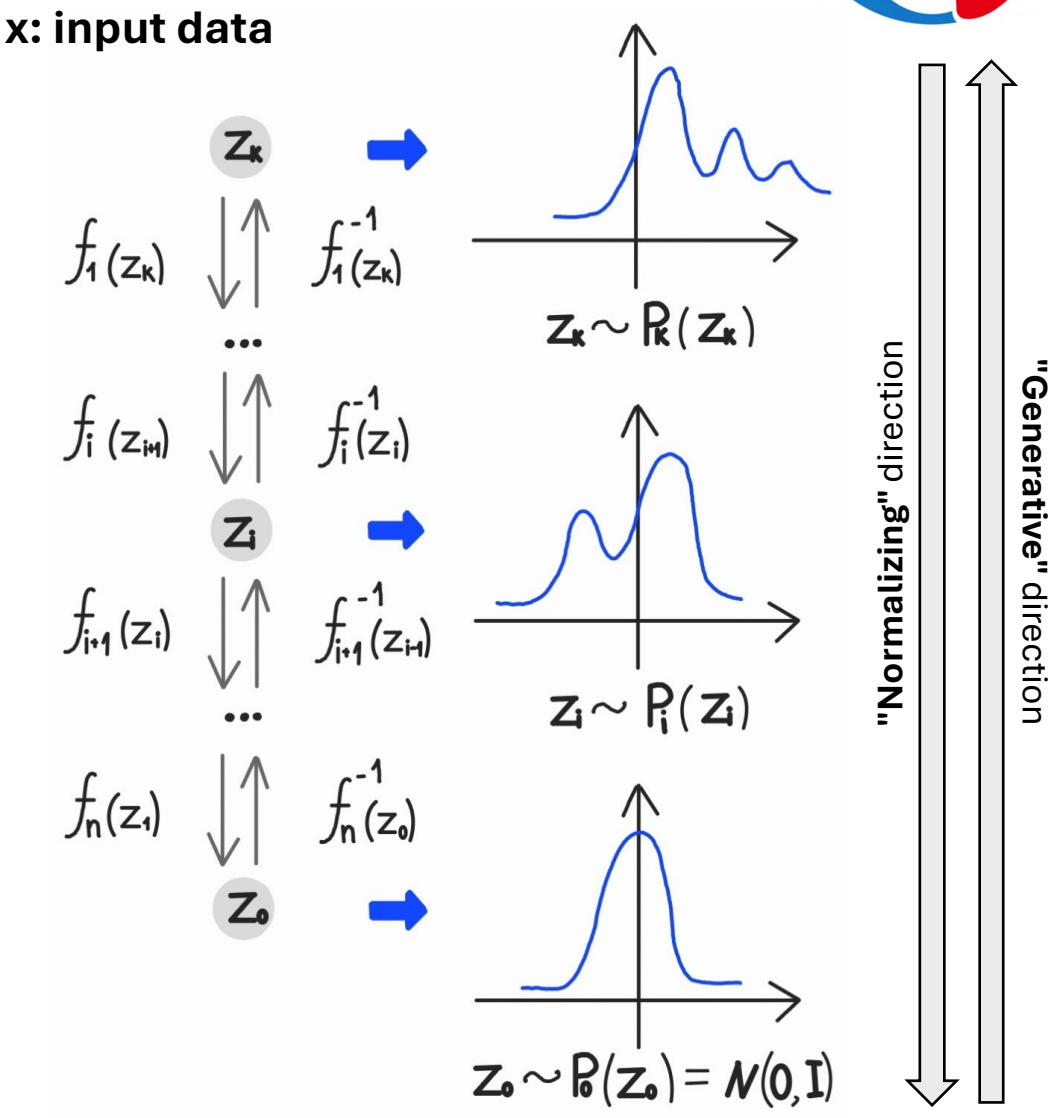


[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

- **Normalizing flows [1]:**
  - **Generative** machine learning **model**
  - Aims to **explicitly model a density estimation**
  - Can be generalized for modeling **a conditional density estimation** as well
  - To obtain a value from the target distribution:
    - sample a value from the base distribution and passing it through all the flows

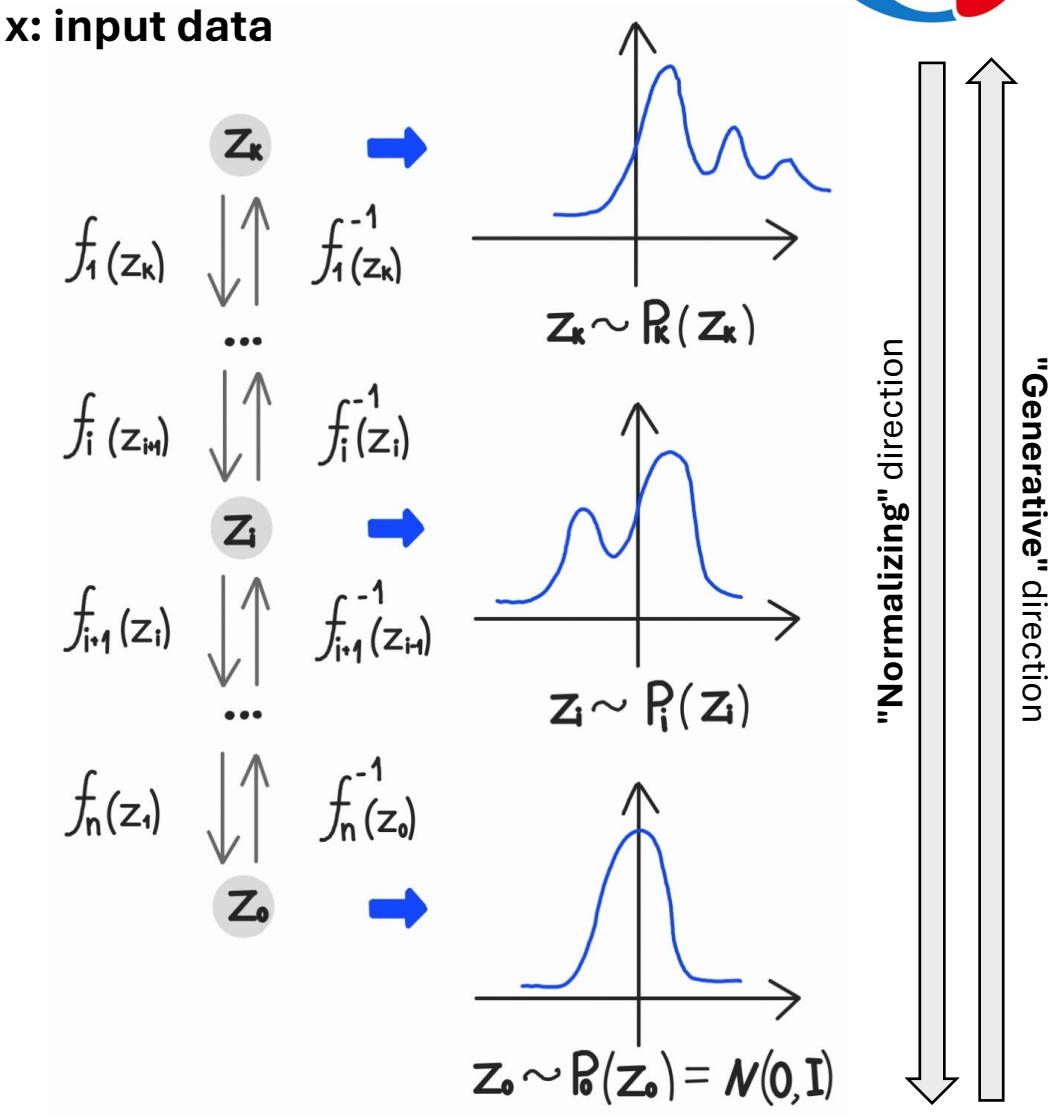
- Normalizing flows are based on an **idea of transforming a simple known distribution** (*e.g. standard Gaussian*) **to the complex, desired distribution** by applying multiple learnable (*using data*) transformations
- **Each transformation** is called Flow and **must be invertible**



[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

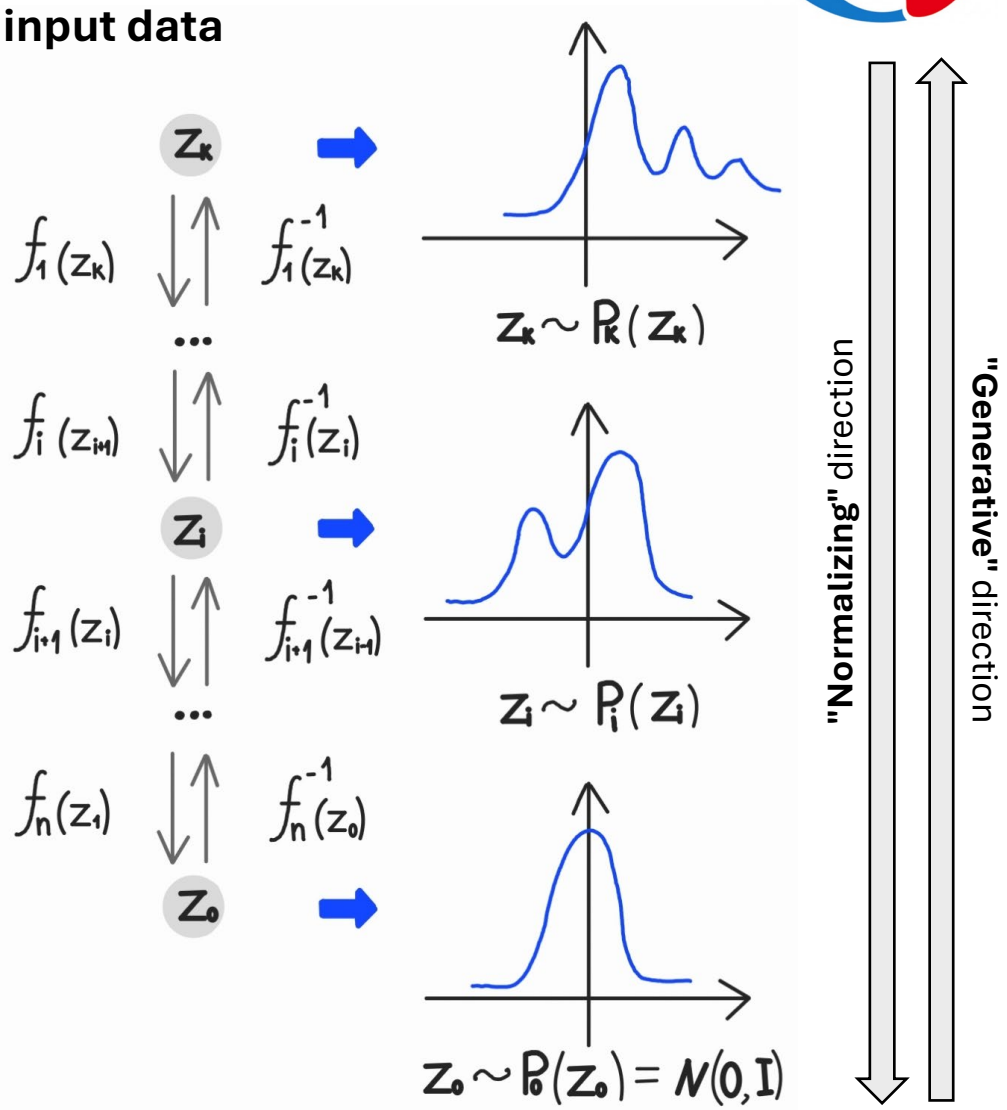
- **Normalizing flows [1]:**
  - Transformations can be very diverse as long as they follow **the invertibility condition**
  - We use **Planar flow**:
    - $f_\theta(x) = x + u_\theta \tanh(xw_\theta + b_\theta)$



[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

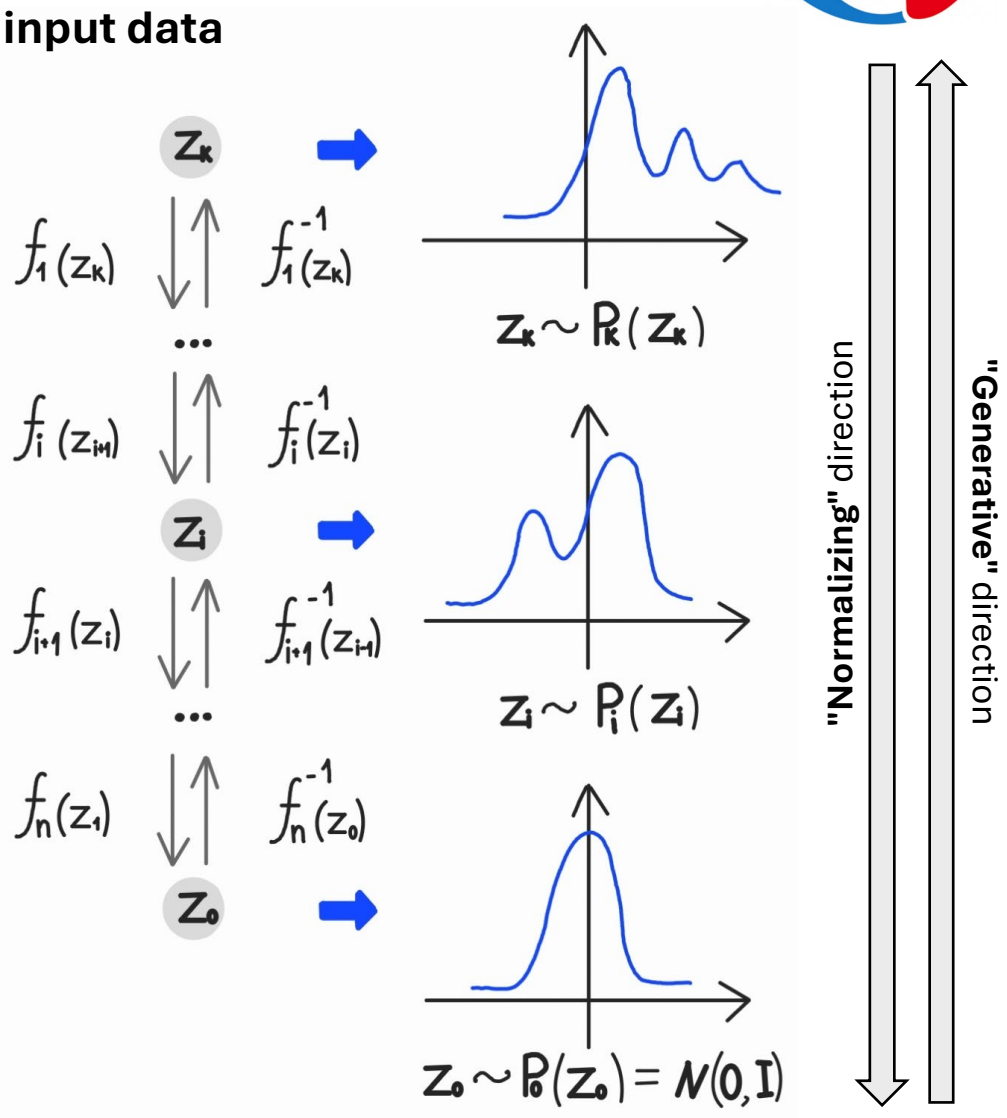
- **Normalizing flows [1]:**
  - Transformations can be very diverse as long as they follow **the invertibility condition**
  - We use **Planar flow**:
    - $f_\theta(x) = x + u_\theta \tanh\left(x w_\theta + b_\theta\right)$
    - where parameters $u_\theta, w_\theta, b_\theta$ are parametrized by neural networks to include the conditions (MC parameters + source type): **conditional probability**



[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

- **Normalizing flows [1]:**
  - Transformations can be very diverse as long as they follow **the invertibility condition**
  - We use **Planar flow**:
    - $f_\theta(x) = x + u_\theta \tanh(x w_\theta + b_\theta)$
    - where parameters $u_\theta, w_\theta, b_\theta$ are parametrized by neural networks to include the conditions (MC parameters + source type): **conditional probability**
- How to perform the inference with the model?



[1] R. J. Rezende et al, arxiv: 1505.05770

Normalizing flows-based density estimation

- **Normalizing flows [1]:**
  - Transformations can be very diverse as long as they follow **the invertibility condition**
  - We use **Planar flow**:
    - $f_\theta(x) = x + u_\theta \tanh(xw_\theta + b_\theta)$
    - where parameters $u_\theta, w_\theta, b_\theta$ are parametrized by neural networks to include the conditions (MC parameters + source type): **conditional probability**
- How to perform the inference with the model?

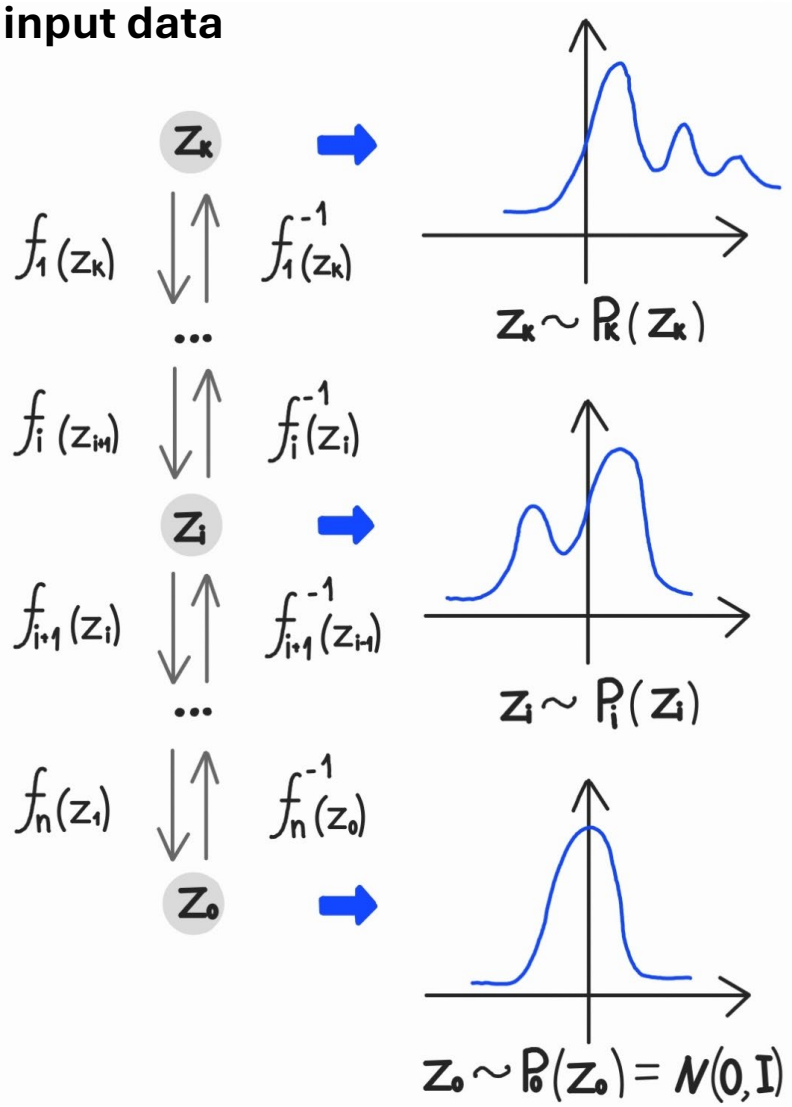
$$\log p(x|y) = \log p(z_0) + \sum_{k=1}^{K-1} \log \frac{df_{k,\theta_k}(z_k)}{dz_k} + \log \frac{df_{K,\theta_K}(x)}{dx}$$

Data

MC parameters + source type

well-known (e.g. standard Gaussian)

Computed based on the learned transformations



[1] R. J. Rezende et al, arxiv: 1505.05770