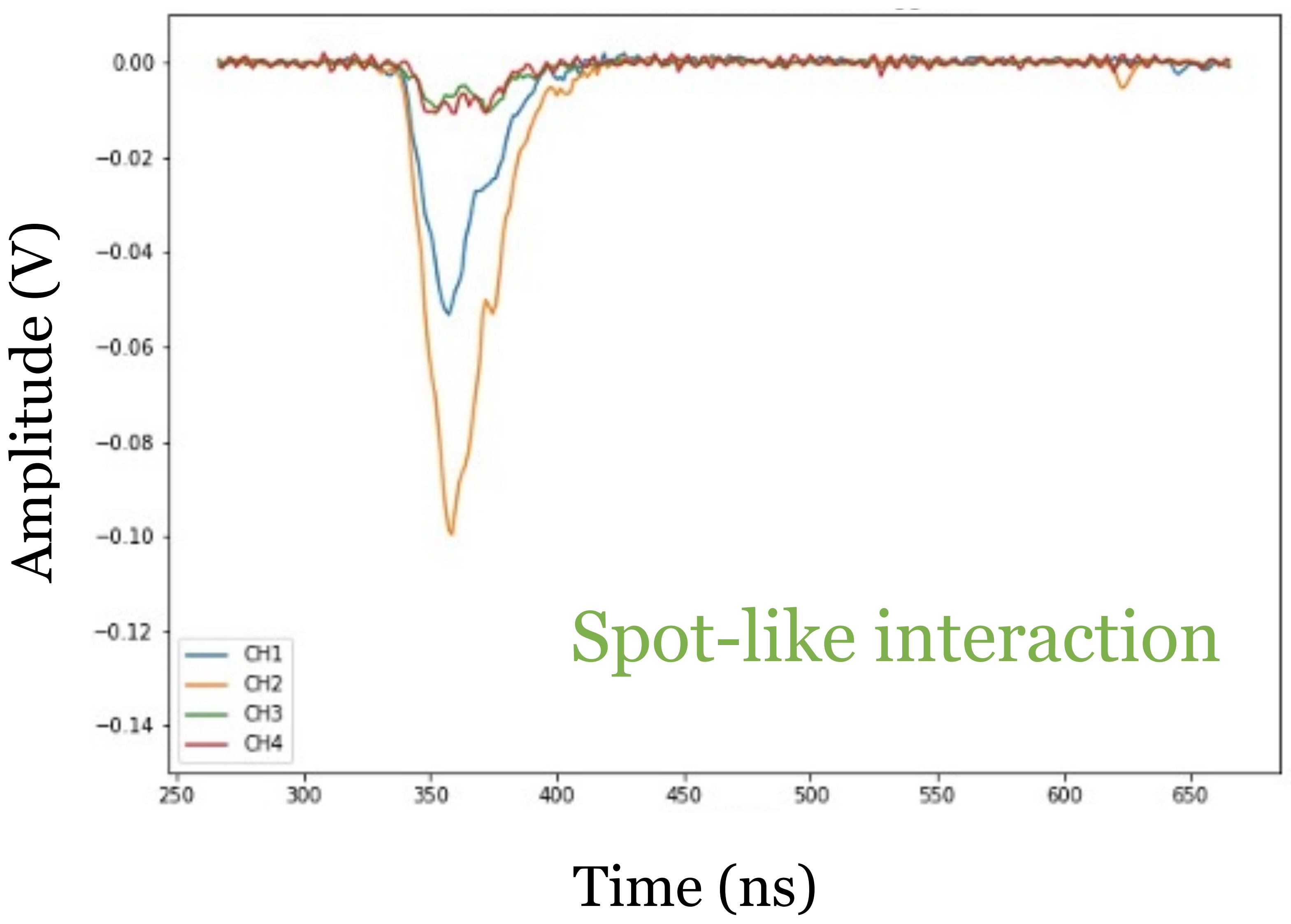
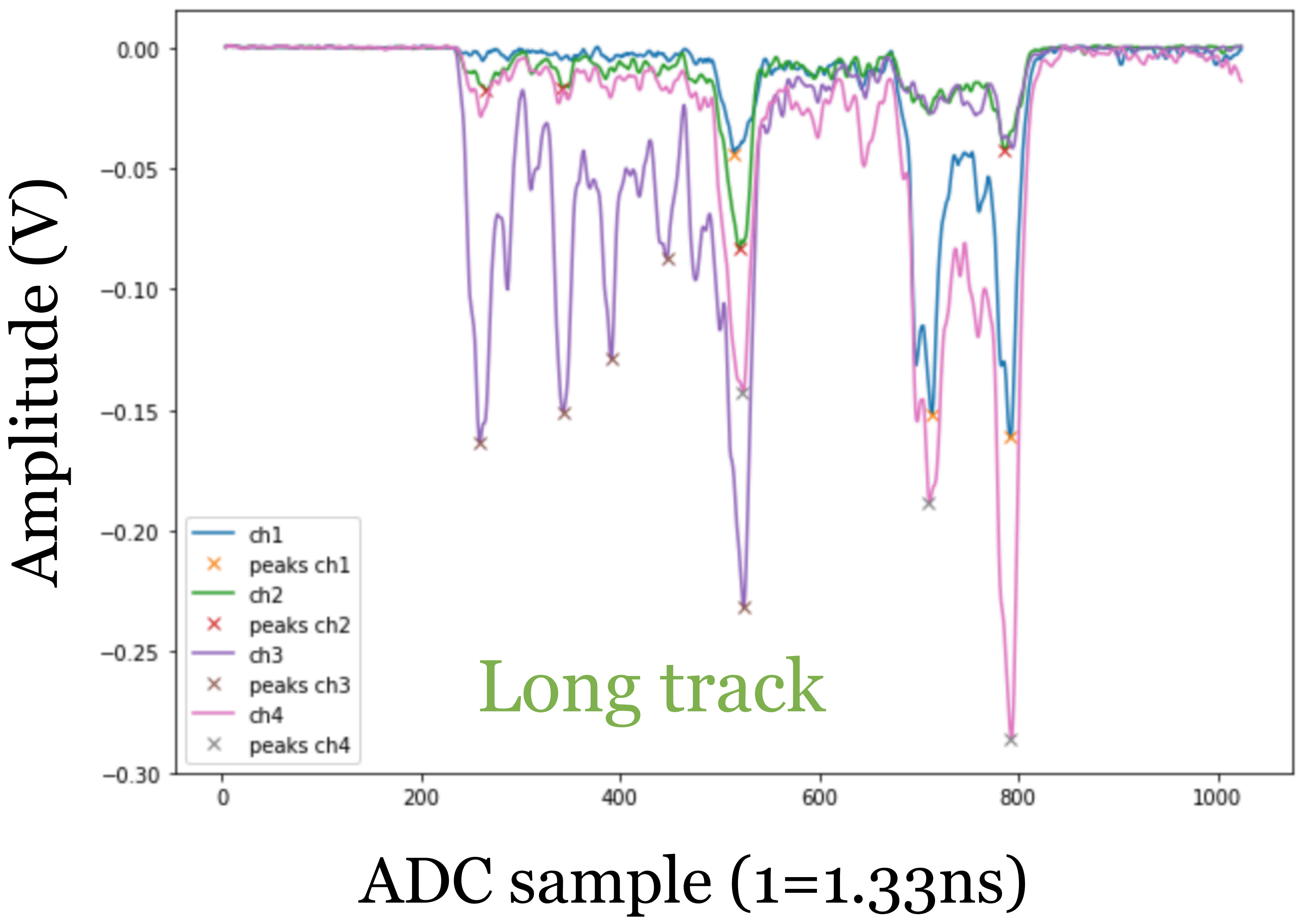
PMTs fit update

Francesco Borra for the PMT working group, **09/19/2024**
francesco.borra@uniroma3.it

PMTs signals

Fast light sensor, needed for the event's **3D reconstruction.**





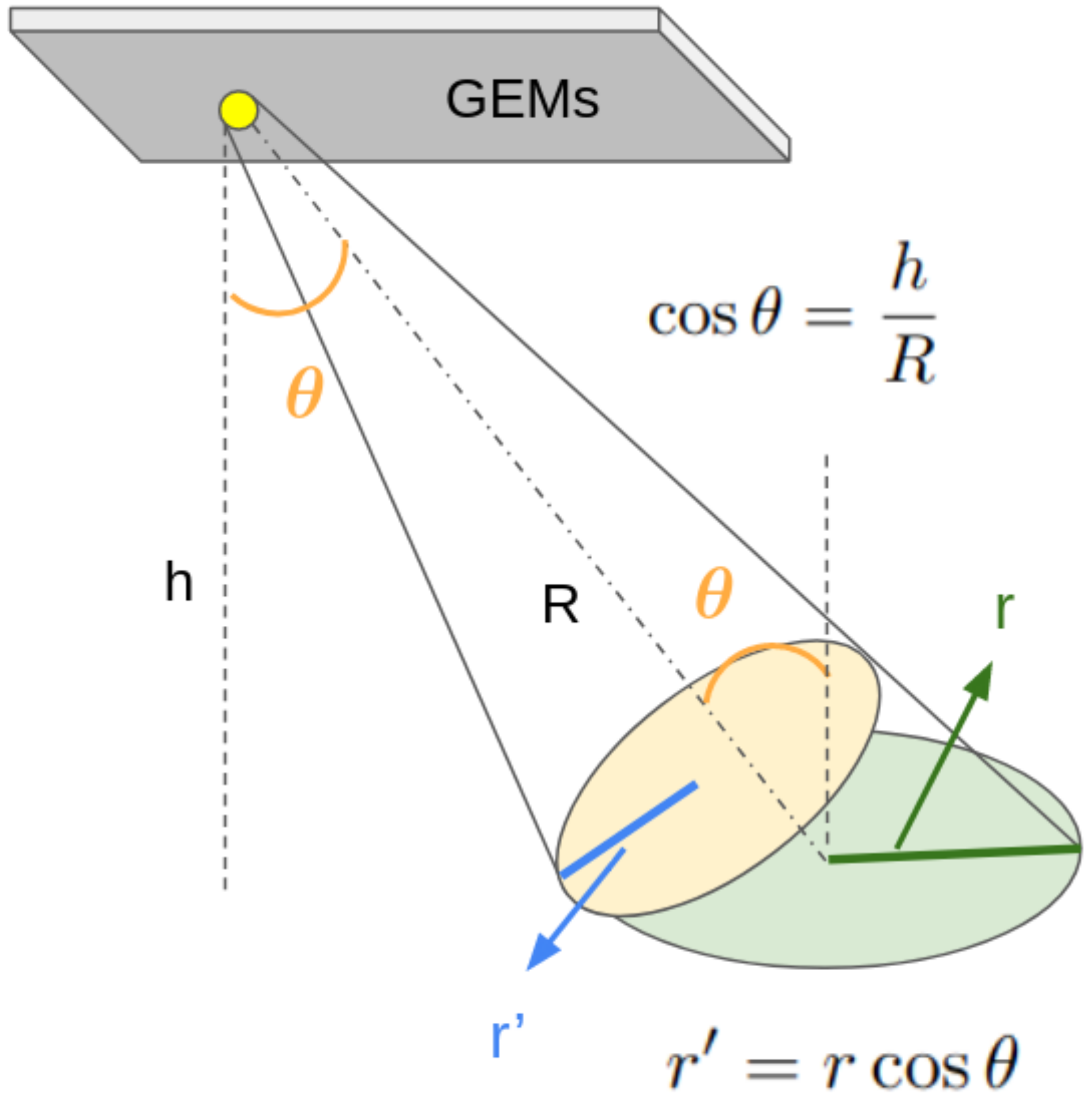
- **Charge:** proportional to the light collected by the PMT.
- **Signal length:** related to the event distance from the GEM plane and track inclination.

Charge collected: integral of the waveform divided by the termination resistance.

$$I = \frac{V}{R} = \frac{Q}{t} \longrightarrow Q \cdot R = \int_{t_0}^{t_1} V(t)\, dt$$

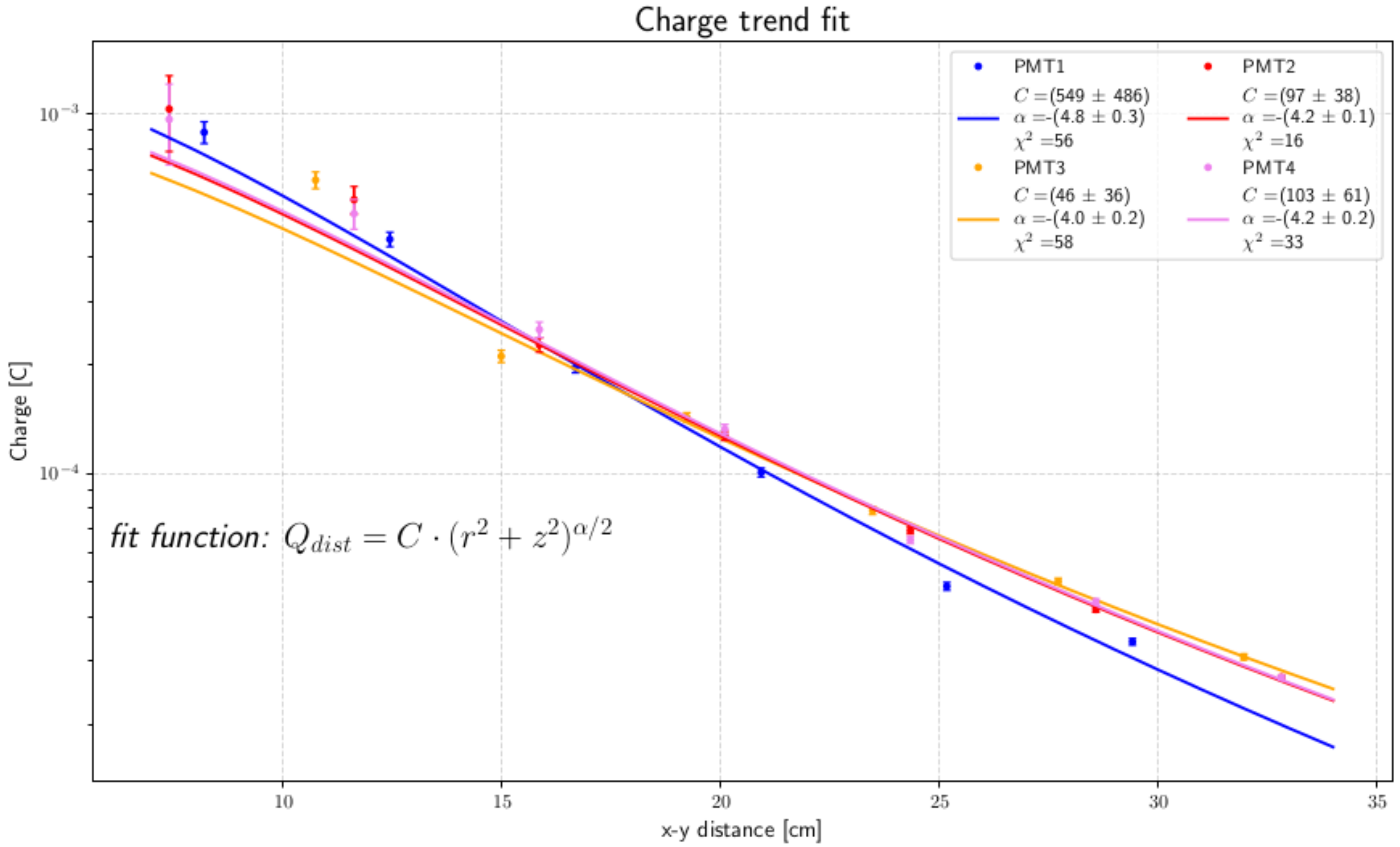
$$\longrightarrow Q = \frac{\int_{t_0}^{t_1} V(t)\, dt}{R}$$

Geometrical dependence



$$L_{PMT} = \frac{L_{spot}}{4\pi R_i^2}\frac{\pi r^2 h}{R_i}$$

From a geometrical analysis we expect a $\propto R^{-3}$ dependence.



A simple study revealed a $\propto R^{-4}$ dependence.

The idea

Use the dependence of the charge collected by the 4 PMTs on the distance of the source to retrieve the emission position and light produced at the GEM plane.

Going with a $\propto R^{-4}$ dependence

First step: PMTs calibration

Not all PMTs have the same response due to:

- Improper calibration
- Misalignment
- ...

$$Q_i \propto c_i \frac{L}{R_i^4}$$

A new 'calibration' is needed:

c_i parameters

Energy and position from PMTs data

- Measure: Q_{1-4}
- Infer: x, y, L

$$p(\theta \mid \{x_i\}) = \frac{p(\{x_i\} \mid \theta) \cdot \pi(\theta)}{p(\{x_i\})}$$

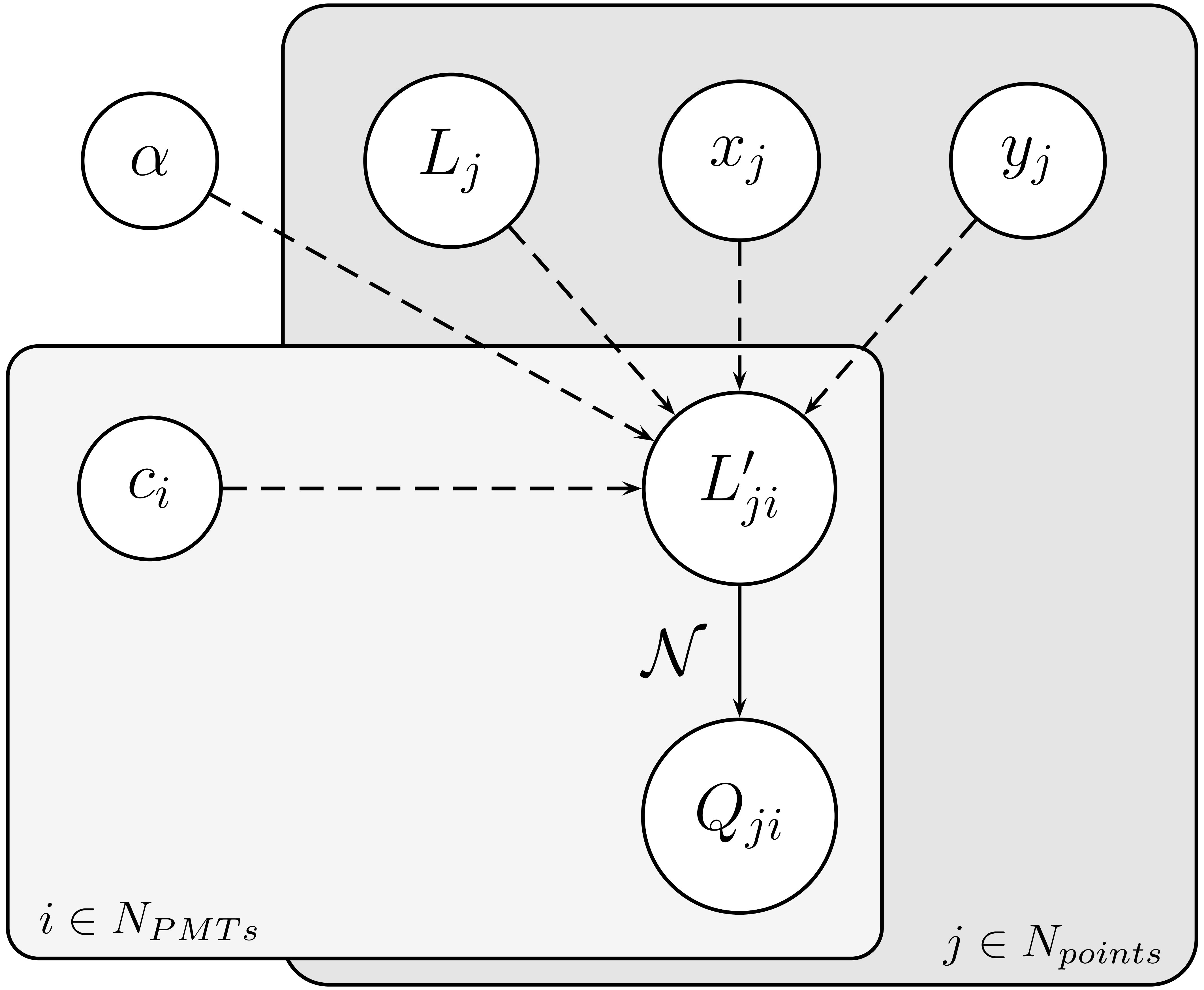
Likelihood

Prior

Posterior

Normalization factor

Likelihood



$$p(\{x_{ij}\} \,|\, \theta) = \prod_{j=1}^{N_{points}} \prod_{i=1}^{4} \mathcal{N}(\{x_{ij}\} \,|\, L'_{ij}(\theta))$$

With:

- $L'_{ji} = c_i \dfrac{L_j}{R^{\alpha}_{ij}}$
- $R_{ji} = \sqrt{x^2_{ji} + y^2_{ji} + z^2}$

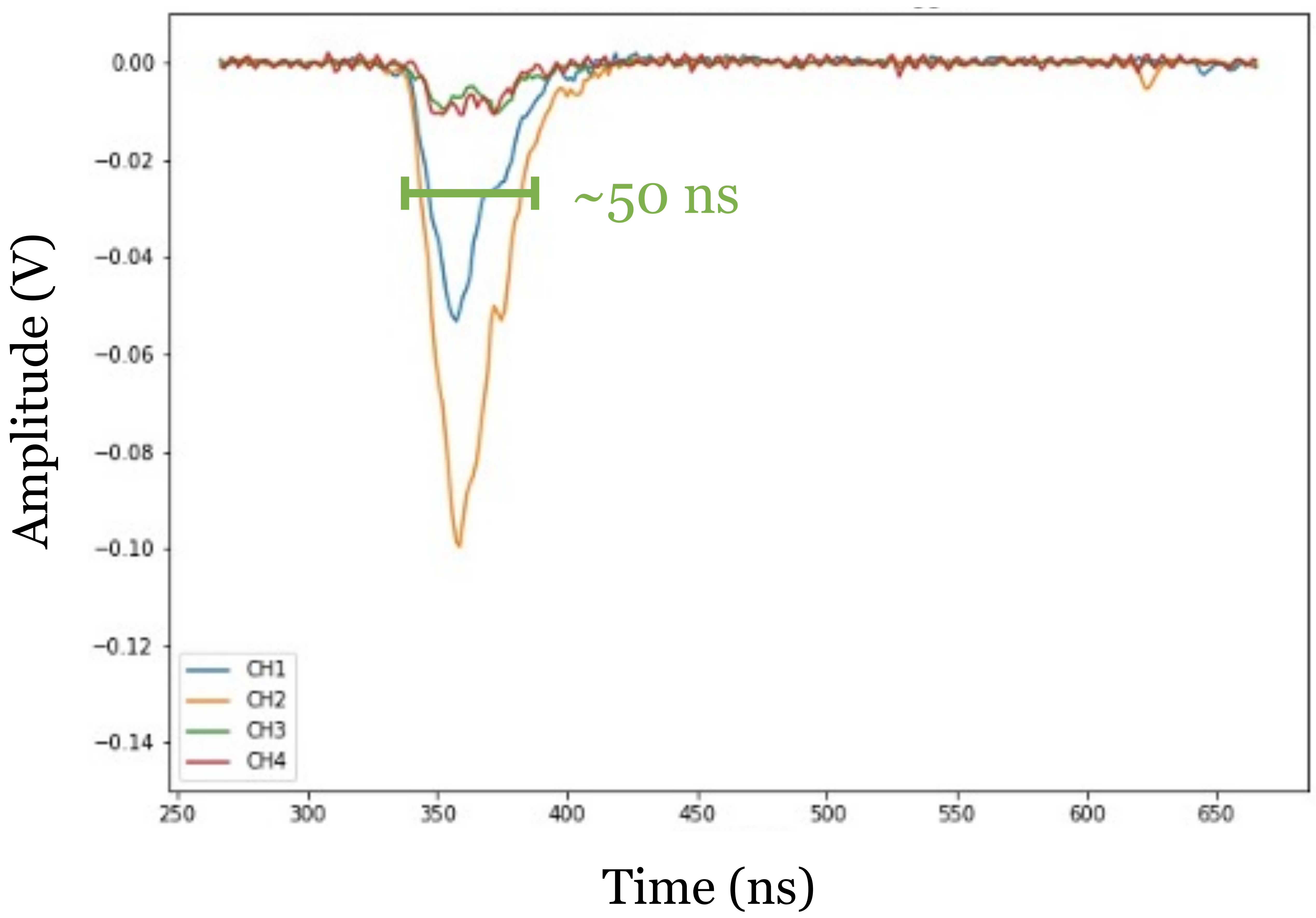
Solid lines represent probabilistic links between the variables, while **dashed lines** indicate deterministic links.

Primordial nodes must be **fixed** (grey) or have a **prior** (white).

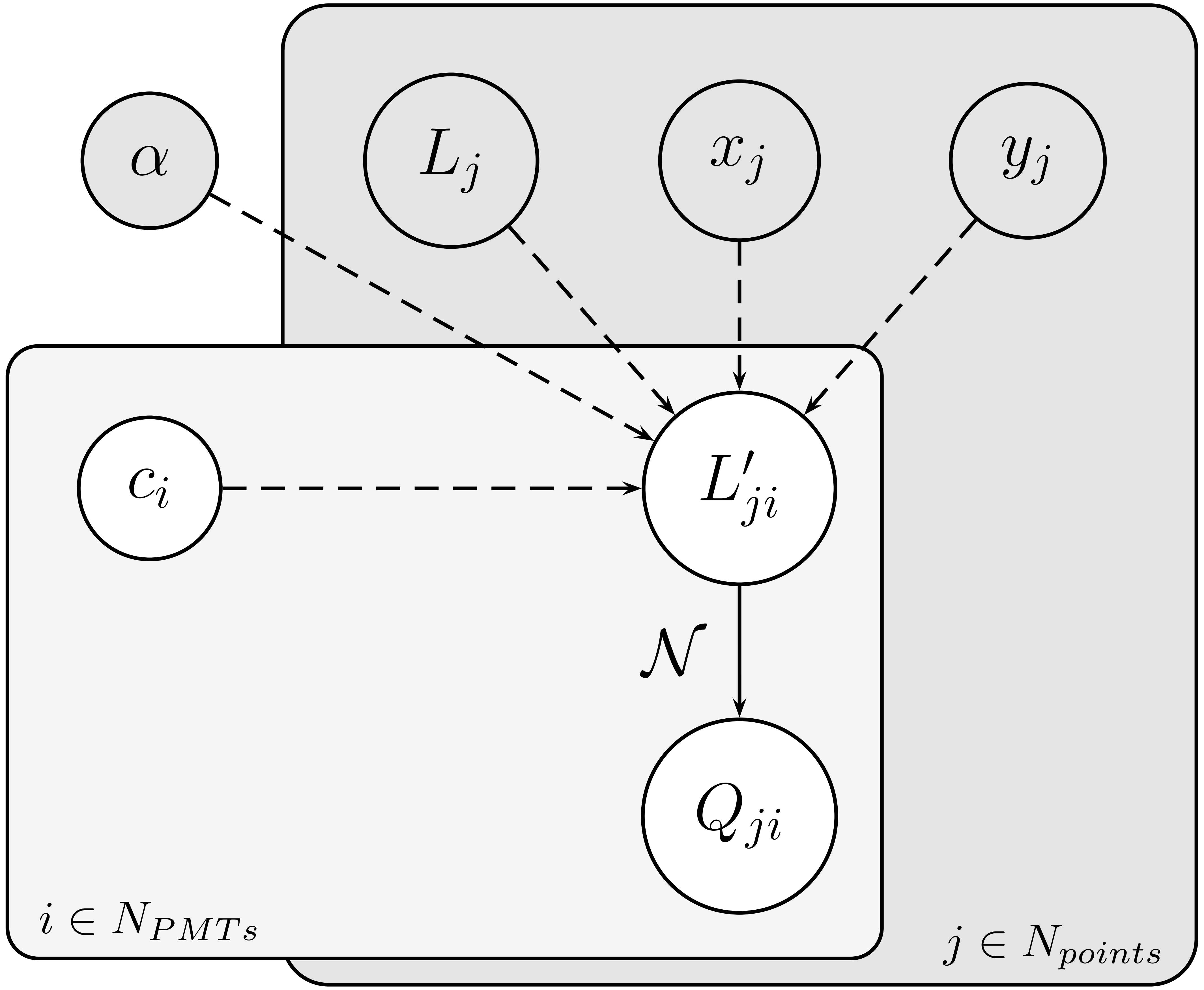
Spot-like interactions

- Find "majority 2 peak"
- Integrate 50 samples (Q_i)
  (~0.4cm resolution in z)
- Perform the Bayesian fit over the 4 PMTs' charges

$$\mu_i = c_i \frac{L}{R_i^4}$$



Likelihood PMTs calibration



Solid lines represent probabilistic links between the variables, while **dashed lines** indicate deterministic links.

Primordial nodes must be **fixed** (grey) or have a **prior** (white).

Taken from camera variables

Fixed x_j, y_j, L_j **!**

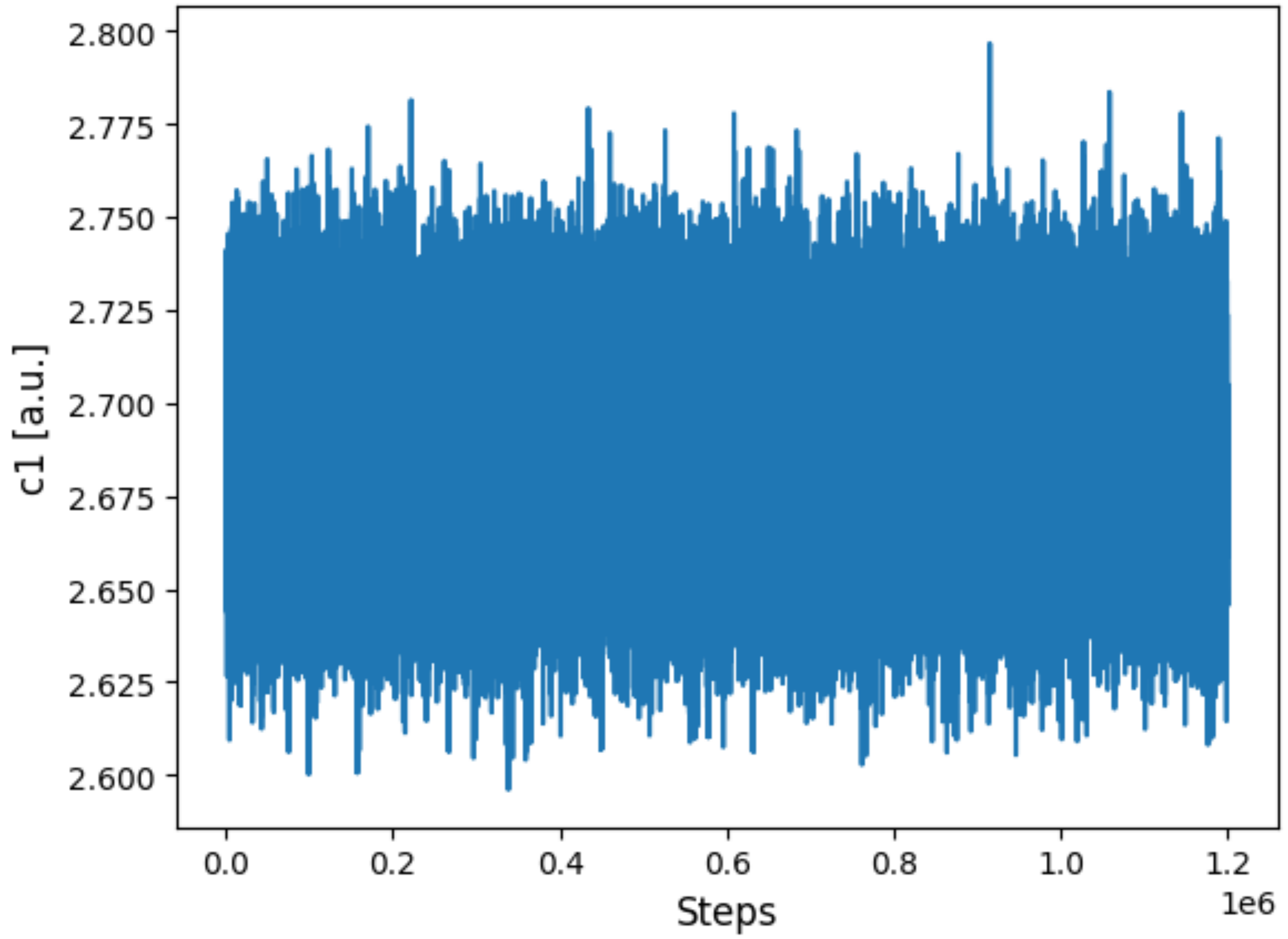
$$p(\{x_{ij}\}\,|\,\theta) = \prod_{j=1}^{N_{points}} \prod_{i=1}^{4} \mathcal{N}(\{x_{ij}\}\,|\,L'_{ij}(\theta))$$

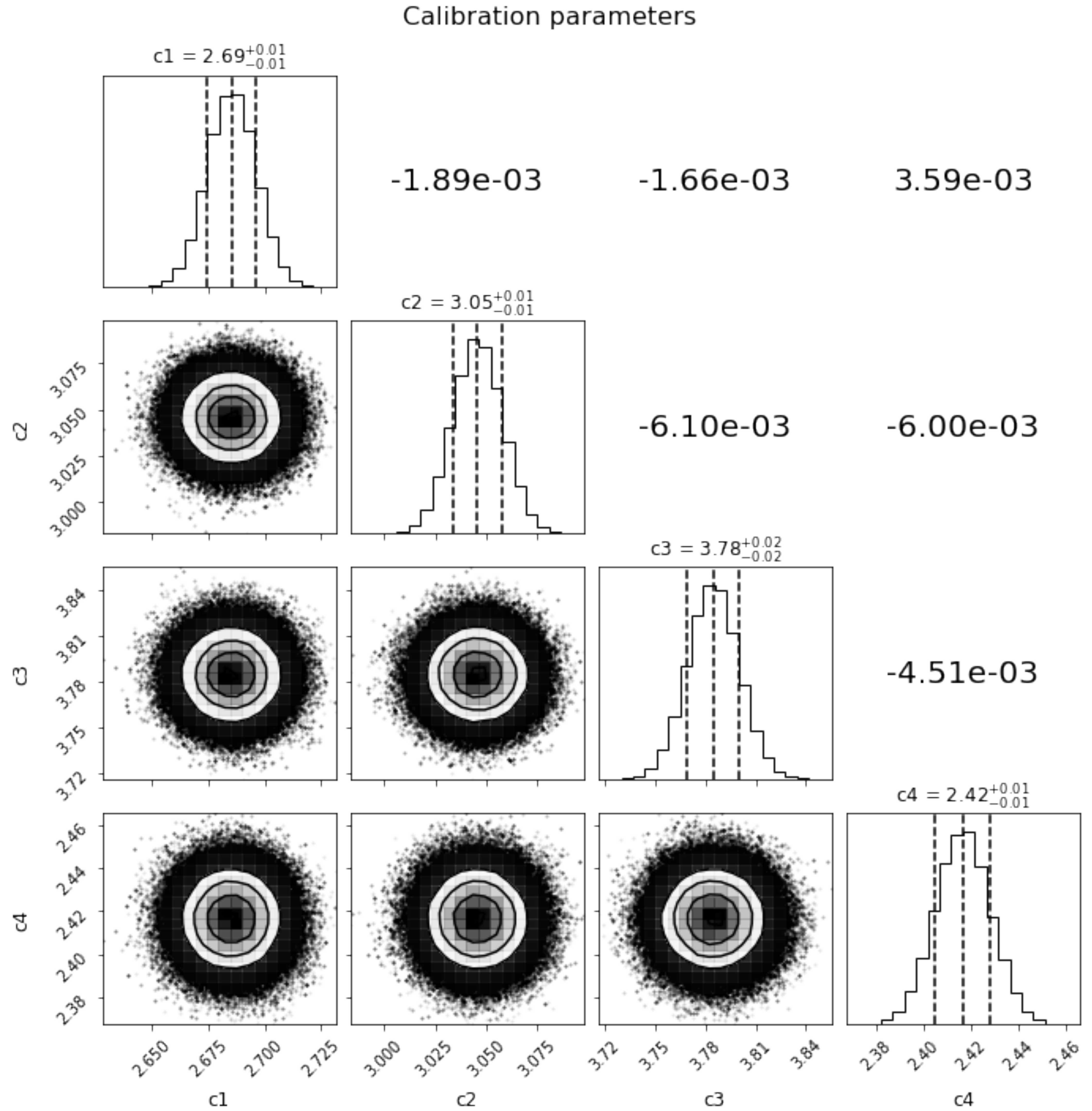
With:

- $L'_{ji} = c_i \dfrac{L_j}{R_{ij}^{\alpha}}$
- $R_{ji} = \sqrt{x_{ji}^2 + y_{ji}^2 + z^2}$
- $\alpha = 4$

PMTs calibration chains and posteriors

- 12 parallel chains
- 100k steps each



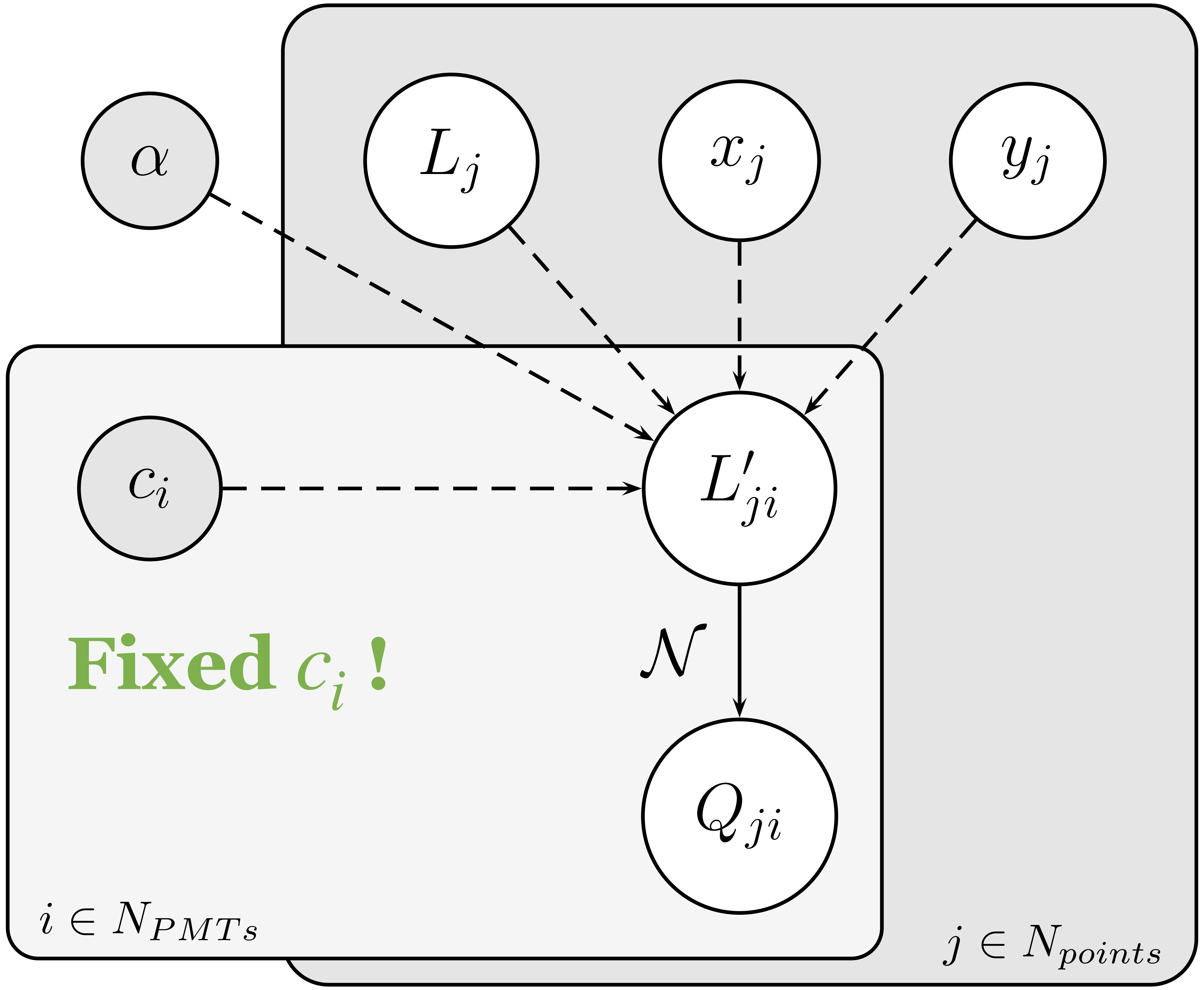


We are only interested in the ratio between coefficients!

The obtained coefficients are then used in the fit

Likelihood for “association”

$N_{points} = 1$



$$p(\{x_{ij}\}\,|\,\theta) = \prod_{j=1}^{N_{points}} \prod_{i=1}^{4} \mathcal{N}(\{x_{ij}\}\,|\,L'_{ij}(\theta))$$

With:

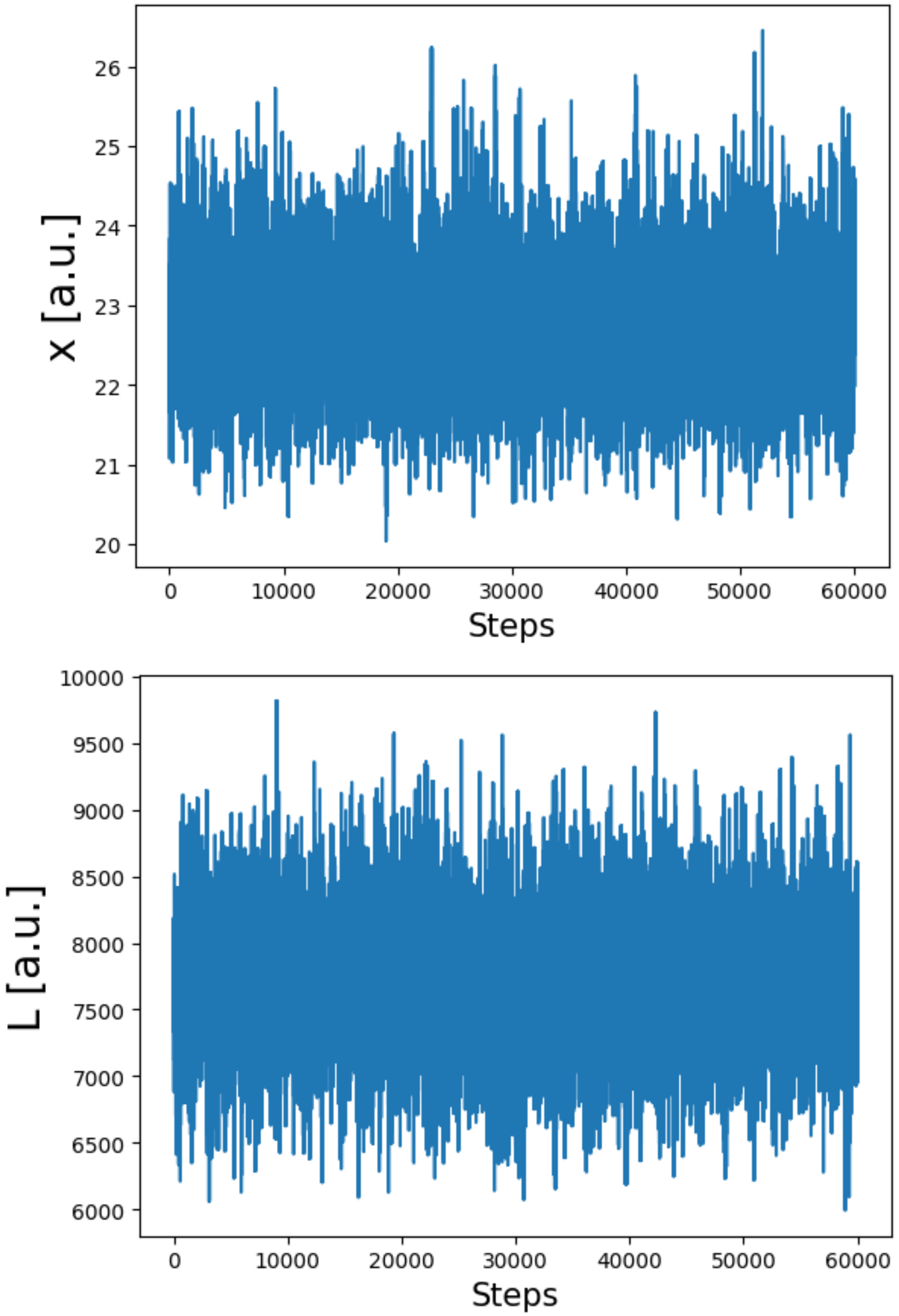
- $L'_{ji} = c_i \dfrac{L_j}{R_{ij}^{\alpha}}$
- $R_{ji} = \sqrt{x_{ji}^2 + y_{ji}^2 + z^2}$
- $\alpha = 4$

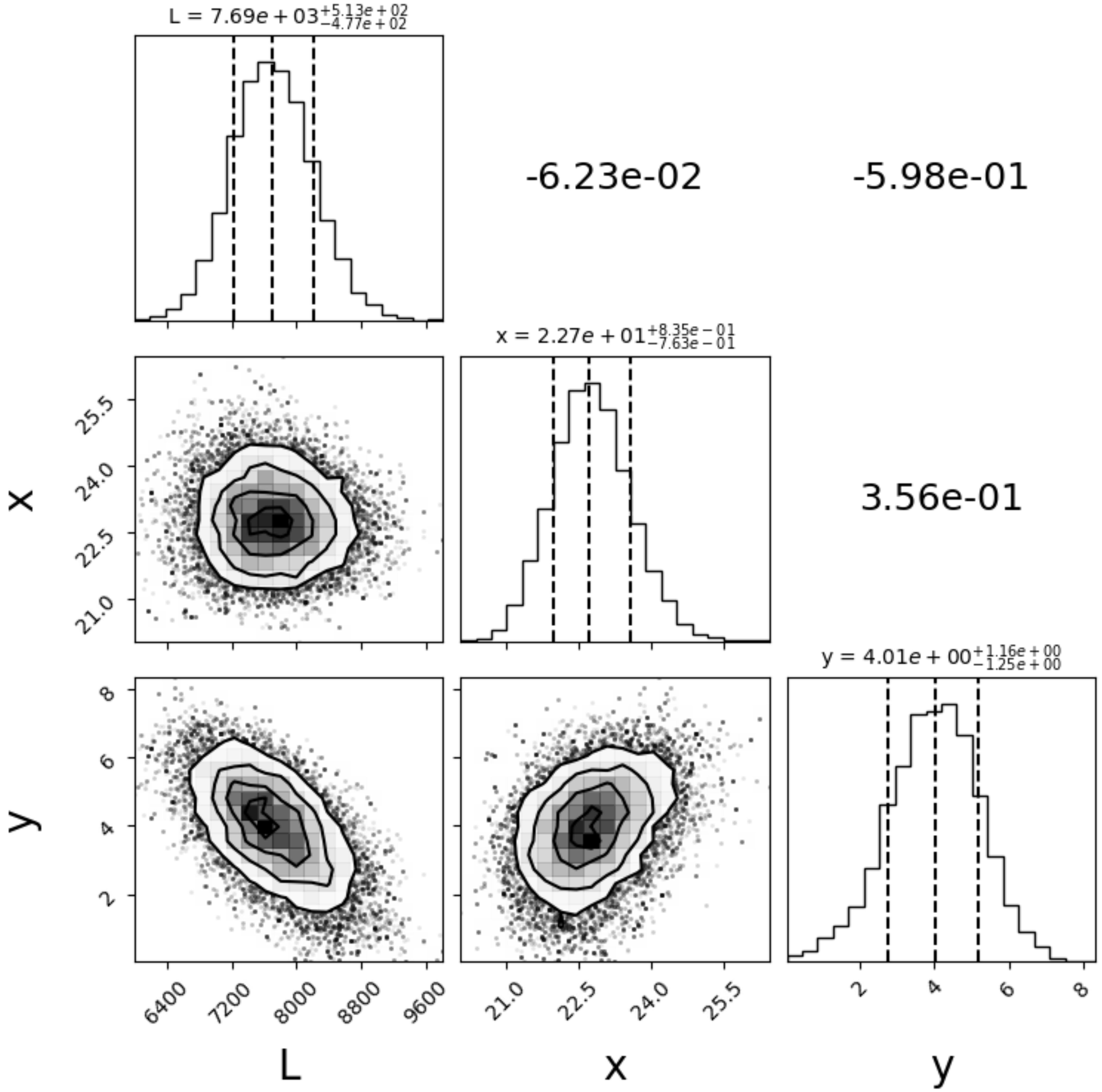
Solid lines represent probabilistic links between the variables, while **dashed lines** indicate deterministic links.

Primordial nodes must be **fixed** (grey) or have a **prior** (white).

"Association" chains and posteriors

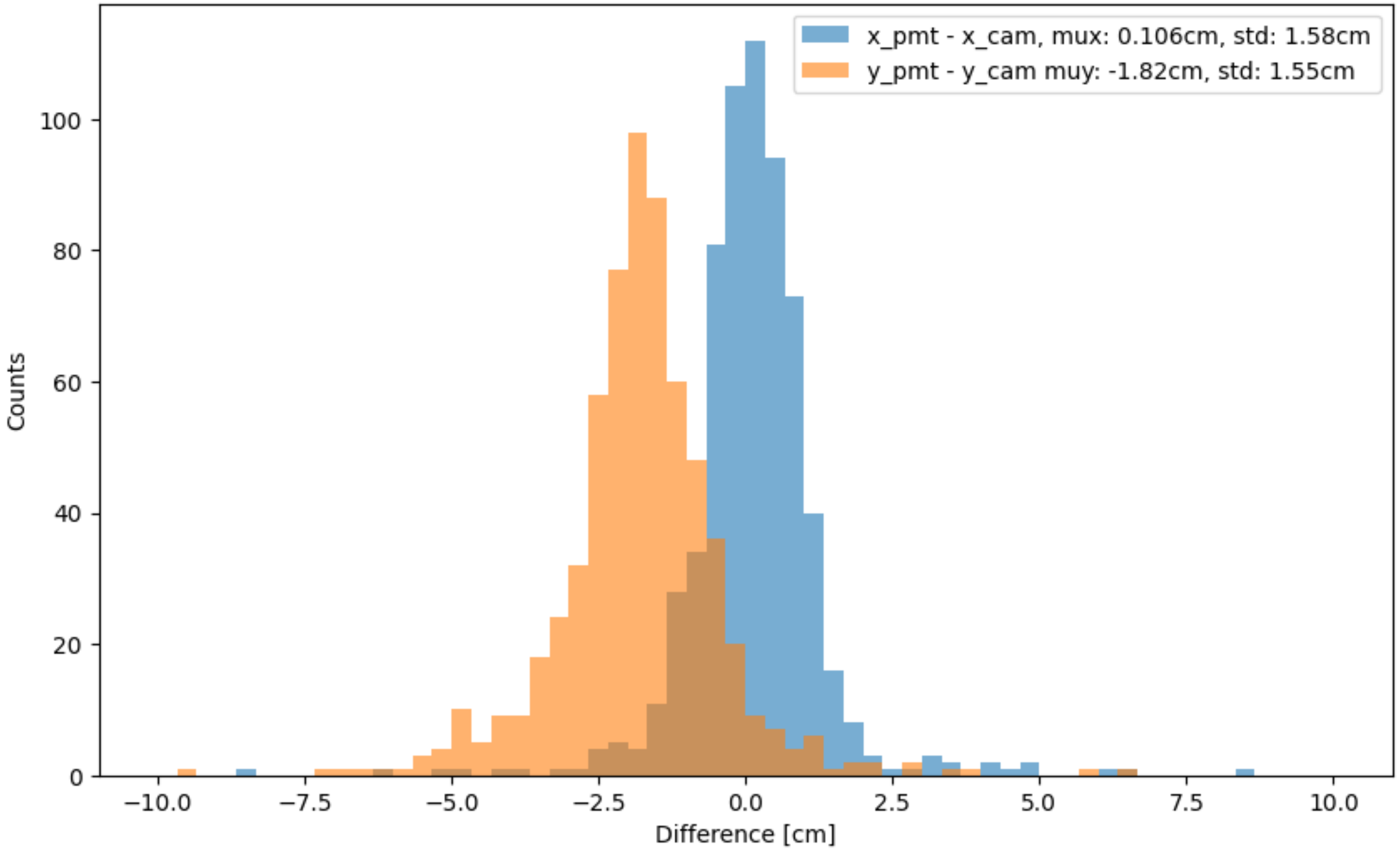
- 6 parallel chains
- 10k steps each





Performance on golden dataset

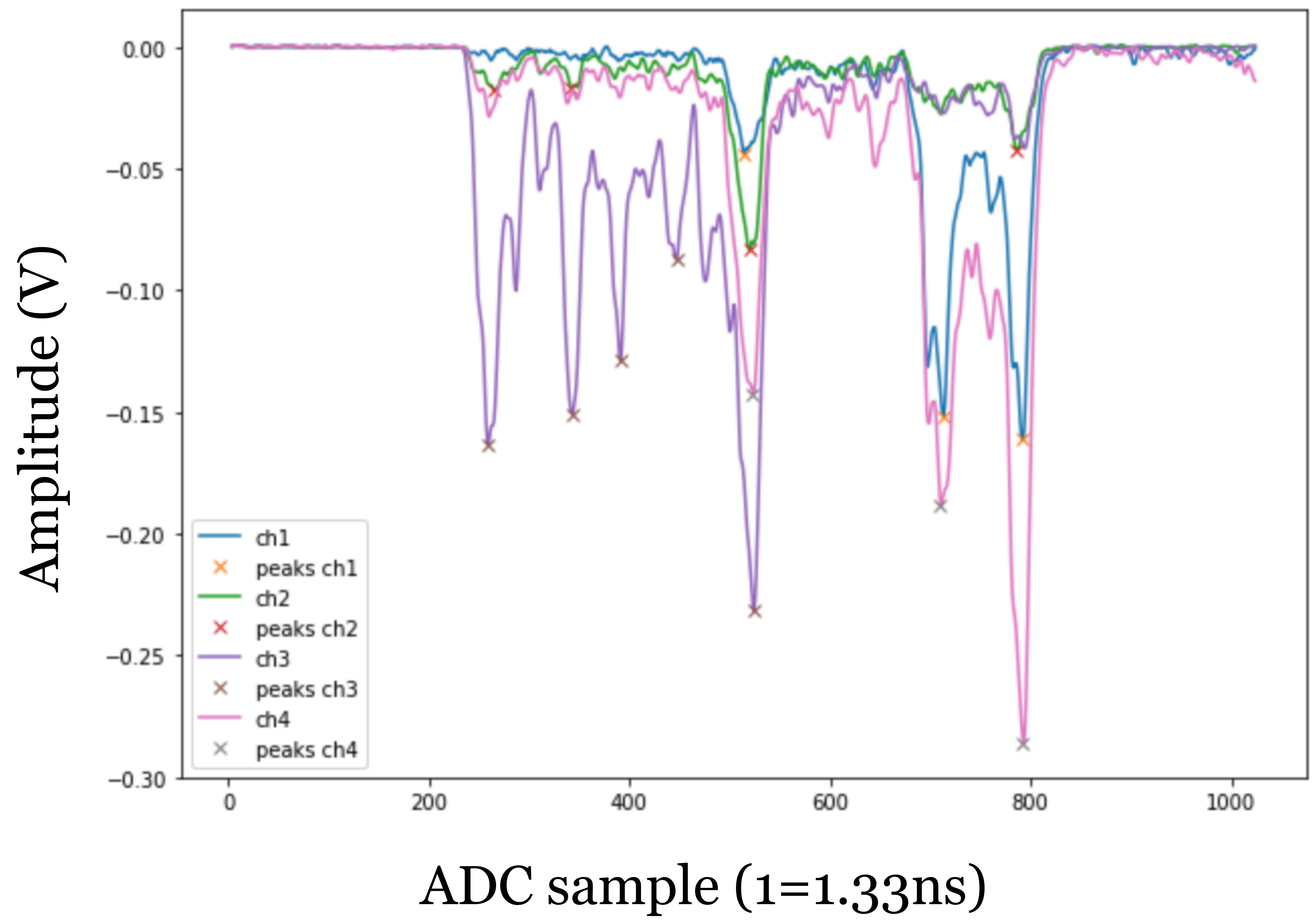
1 to 1 dataset (1 spot and 1 waveform)

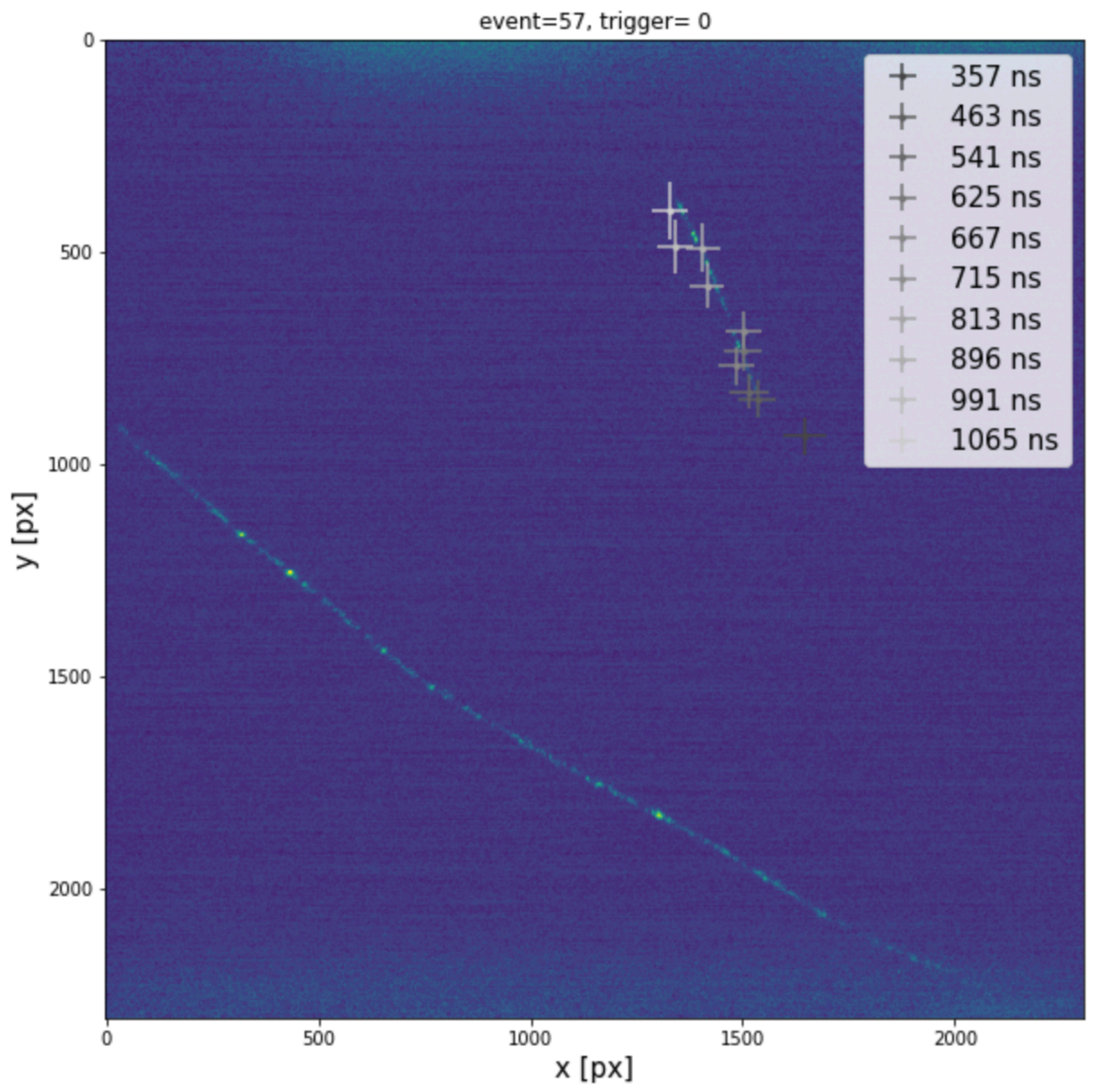


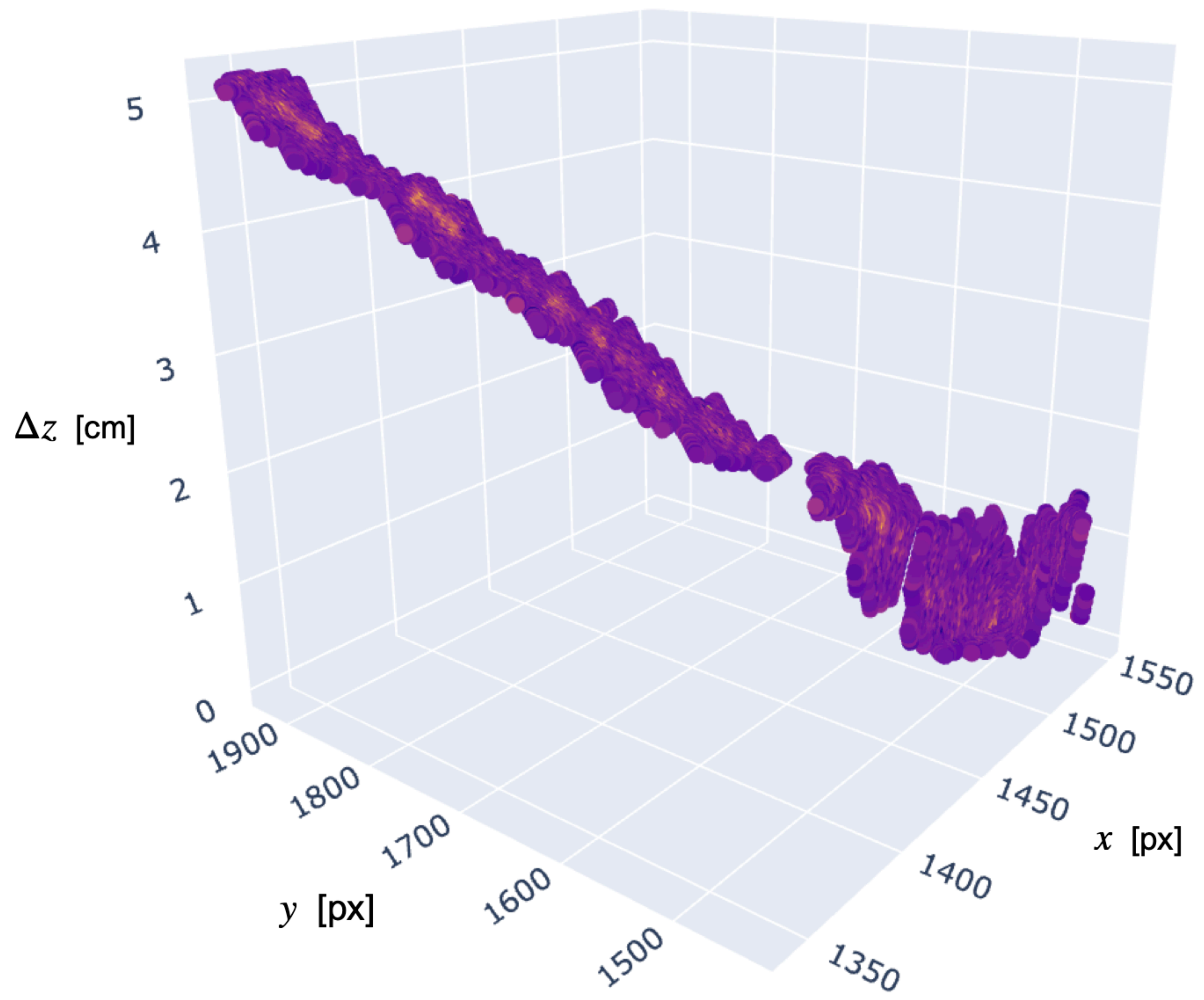
- **Performance:**
  - $x_{std} = 1.58$ cm
  - $y_{std} = 1.55$ cm
- **Issues unrelated to the fit:**
  - PMT-to-camera coordinate transformation
  - Effects of lens distortion (need spots in a wider GEM space)

Association for longer waveforms

- **Find peaks** of the waveform
- Take **majority 2 peaks**
- **Open a window** around these peaks of 50 samples
- Fit the slice of the waveform **as a spot-like interaction**



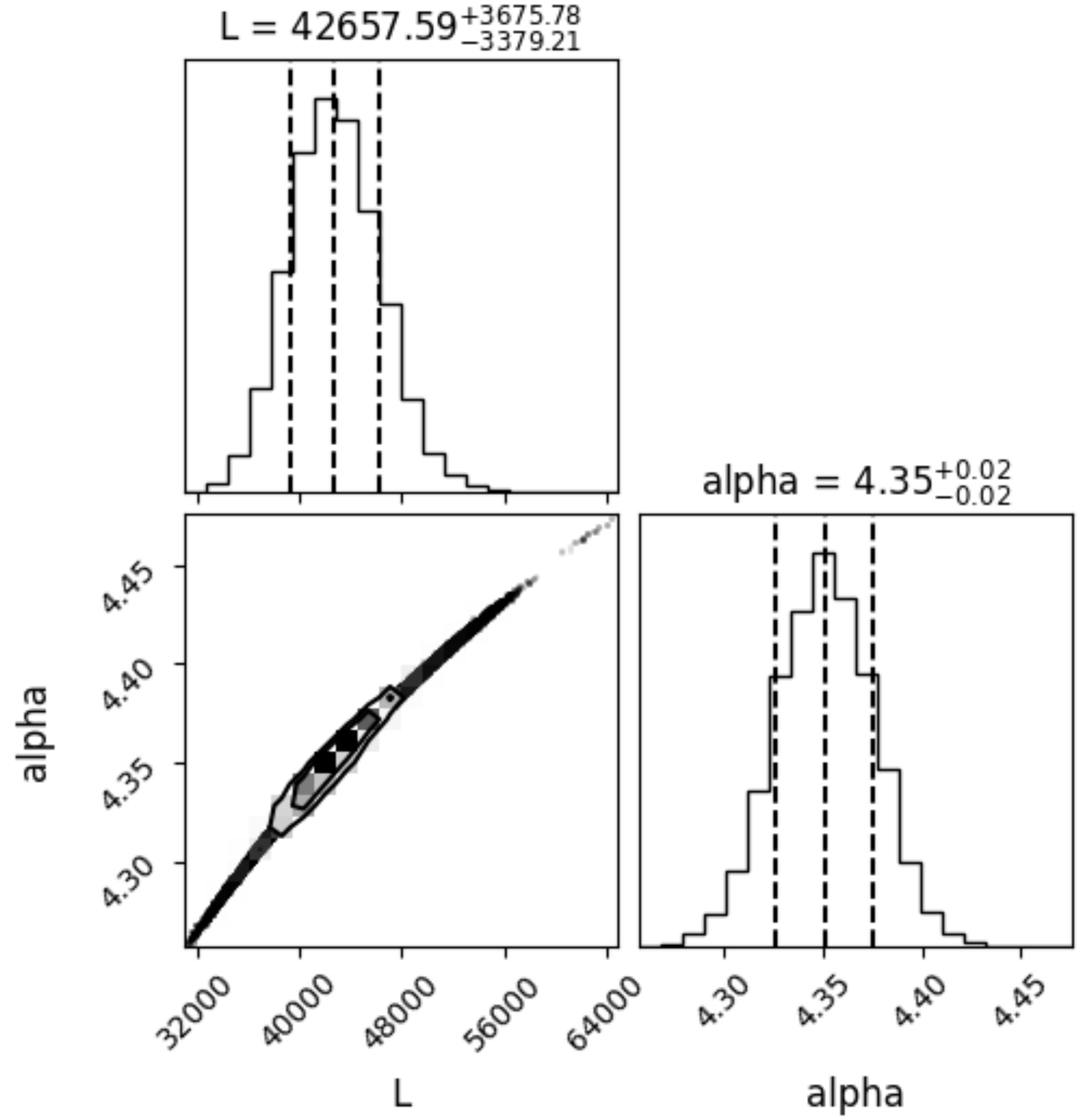


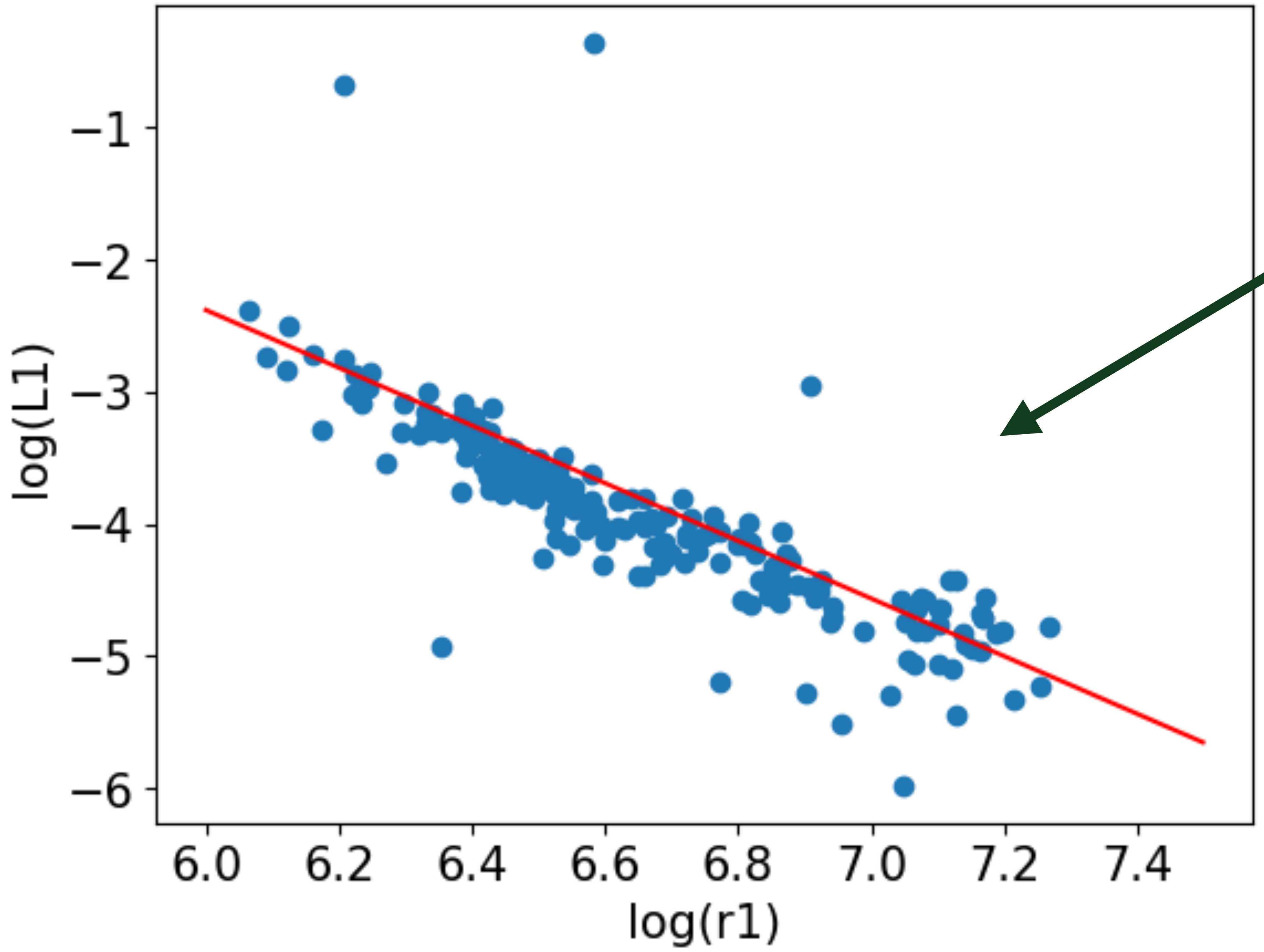


"FindAlpha" By Matteo F.

- **We can think it as a linear fit:**

$$L_i = \frac{L}{r_i^{\alpha}} \rightarrow \log L_i = \log L - \alpha \cdot \log r_i$$





Points don't seem to be properly aligned!

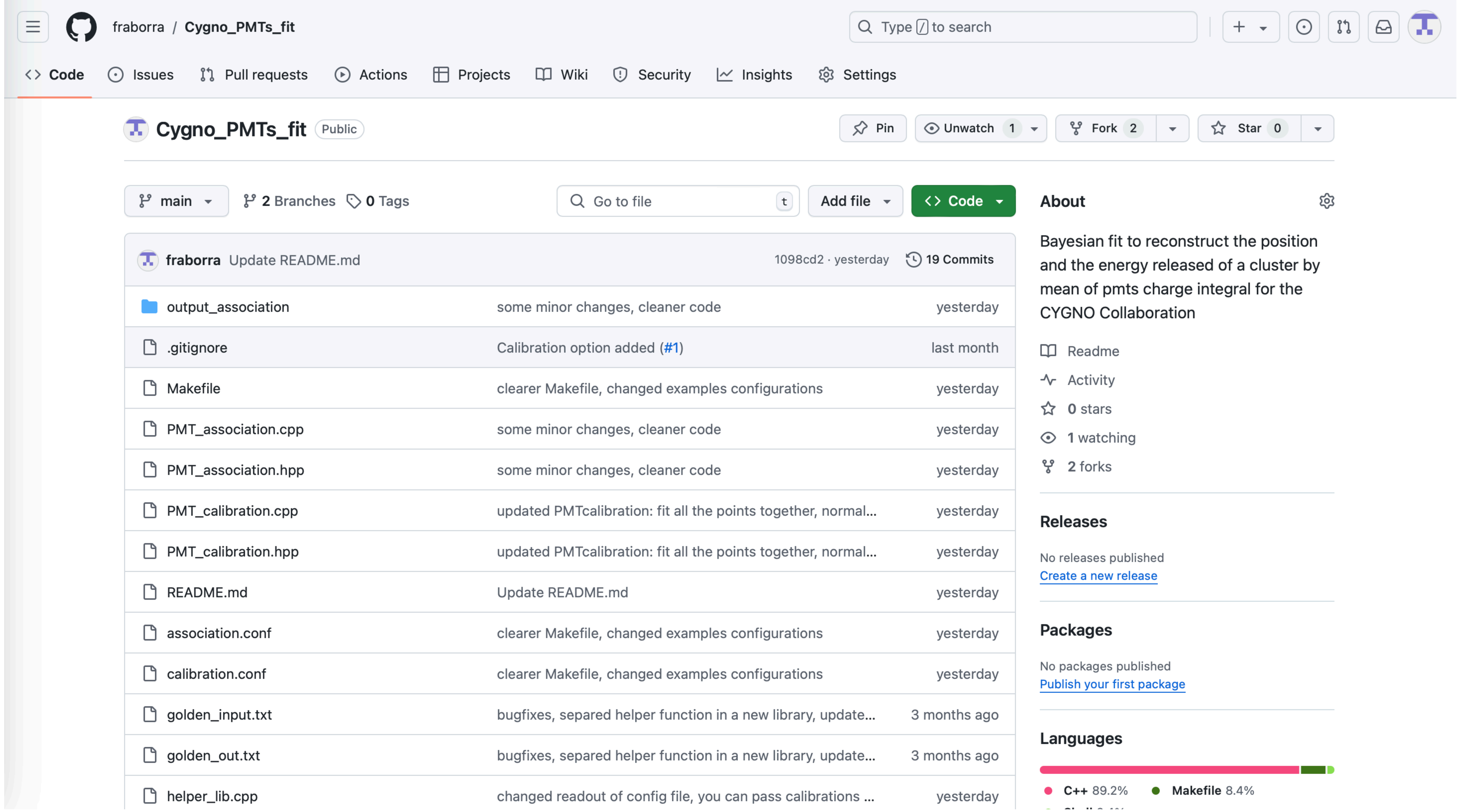
PMTs not positioned as we expect?

Another indication of an **unclear PMT-to-camera coordinate transformation?**

Code Status

The fit is implemented using the software **BAT**.

The package is already on GitHub ([here]) ready for everyone to be used.



How to use it

1. Clone the repository
2. Compile the code
3. Run it!

} README.md

```
./runfit configuration.conf
```

```
[borrfran@gap01 Cygno_PMTs_fit]$ ./runfit association.conf
Initialization of 'association reconstruction'...
mkdir: cannot create directory './output_association': File exists
Starting fit for 'association reconstruction'
 +-------------------------------------------------------+
 |                                                       |
 | BAT version 1.0.0                                     |
 | Copyright (C) 2007-2018, the BAT core developer team |
 | All rights reserved.                                  |
 |                                                       |
 | For the licensing terms see doc/COPYING               |
 | For documentation see http://mpp.mpg.de/bat          |
 | Please cite: DOI 10.1016/j.cpc.2009.06.026            |
 |              http://arxiv.org/abs/0808.2552           |
 |                                                       |
 +-------------------------------------------------------+

Summary : Marginalize using Metropolis
Summary : Pre-run Metropolis MCMC for model "association" ...
Summary :  --> Perform MCMC pre-run with 6 chains, each with maximum 100000 iterations
Summary :  --> Set of 6 Markov chains converged within 3500 iterations, and all scales are adjusted.
Summary :  --> 6 updates to multivariate proposal function's covariances were made.
Summary : Run Metropolis MCMC for model "association" ...
Summary :  --> Perform MCMC run with 6 chains, each with 10000 iterations.
Summary :  --> Markov chains ran for 10000 iterations.
```

Configuration file

```
mode=association
input_file=golden_input.txt
start_ind=0
end_ind=-1
output_file=out_golden_calibrated.txt
plot=false
write_chains=false
write_log=false
print_summary=false
nPoints=1
c1=2.687
c2=2.910
c3=3.615
c4=3.760
```

association.conf

```
mode=PMTcalibration
input_file=cal_test_v3.txt
start_ind=0
end_ind=-1
plot=false
```

calibration.conf

Input and output for association

	run	event	trigger	index	L1	L2	L3	L4
0	11278	95	1	0	0.005768	0.012255	0.054120	0.022083
1	11278	103	1	0	0.036428	0.026936	0.012346	0.013160
2	11278	170	1	0	0.015781	0.018762	0.025400	0.018652
3	11278	204	2	0	0.014438	0.021052	0.028779	0.017780
4	11278	267	3	0	0.023758	0.030755	0.011108	0.006798
...	...	...	...	...	...	...	...	...
200	11177	342	0	0	0.016921	0.019090	0.016594	0.018878
201	11176	41	1	0	0.013646	0.025133	0.025018	0.013525
202	11176	114	0	0	0.024645	0.026661	0.012077	0.013246
203	11176	219	1	0	0.017910	0.027555	0.026793	0.021970
204	11176	339	0	0	0.010736	0.009577	0.043512	0.029371

Input data:

- *index* = peak index in the waveform (needed for non spot-like tracks!)
- $L_{1\text{-}4}$ must be in **nC**!
- **Input file** must have each line with these fields separated by a tab.

	run	event	trigger	index	L	L_std	x	x_std	y	y_std
0	11278	95	1	0	13235.1	809.763	21.7532	0.777428	4.64162	1.130360
1	11278	103	1	0	13232.8	676.696	15.1976	0.673908	21.66050	0.732076
2	11278	170	1	0	12467.9	641.158	18.1150	0.690379	13.86610	0.711467
3	11278	204	2	0	12925.0	667.176	19.3543	0.702933	13.72070	0.684188
4	11278	267	3	0	10356.8	543.171	18.7740	0.693979	23.39400	0.801156
...	...	...	...	...	...	...	...	...	...	...
200	11177	342	0	0	11276.8	588.717	16.4418	0.723091	15.56490	0.704940
201	11176	41	1	0	12032.6	604.847	20.6567	0.712246	15.57900	0.666436
202	11176	114	0	0	11607.8	589.118	16.5667	0.693212	20.29030	0.721522
203	11176	219	1	0	14764.8	756.024	18.5750	0.733153	14.95250	0.724314
204	11176	339	0	0	13981.6	826.190	17.8052	0.682390	5.86767	0.973841

Output data:

- x & y are given in **cm**!
- **Output file** has each line with these fields separated by a tab.

Input and output for PMTcalibration

	run	event	trg	indx	L1	L2	L3	L4	x	y	sc_integral
0	25487	177	0	0	0.018113	0.035682	0.032313	0.019504	18.891071	18.134634	9036.582942
1	25487	217	0	0	0.015878	0.021040	0.030809	0.026613	16.345759	15.147998	9414.066941
2	25487	226	0	0	0.042458	0.023214	0.015807	0.027526	12.860946	21.270352	8838.243888
3	25487	230	0	0	0.017853	0.075018	0.012359	0.006827	21.663326	27.124687	7234.006896
4	25487	378	0	0	0.037493	0.020464	0.016531	0.025102	12.152177	20.344308	8879.340221
...	...	...	...	...	...	...	...	...	...	...	...
689	25720	379	0	0	0.008187	0.010167	0.063639	0.039814	17.739679	8.399647	9566.899737
690	25722	30	0	0	0.036401	0.021208	0.022167	0.032891	12.307099	19.178925	10497.141015
691	25722	35	0	0	0.015806	0.026035	0.032038	0.016908	19.412857	17.096711	7794.342909
692	25722	157	0	0	0.016540	0.016953	0.034628	0.035401	15.716448	12.469627	10205.052995
693	25722	377	0	0	0.045819	0.028803	0.012266	0.018965	13.443374	23.589987	9461.155418

694 rows × 11 columns

Input data:

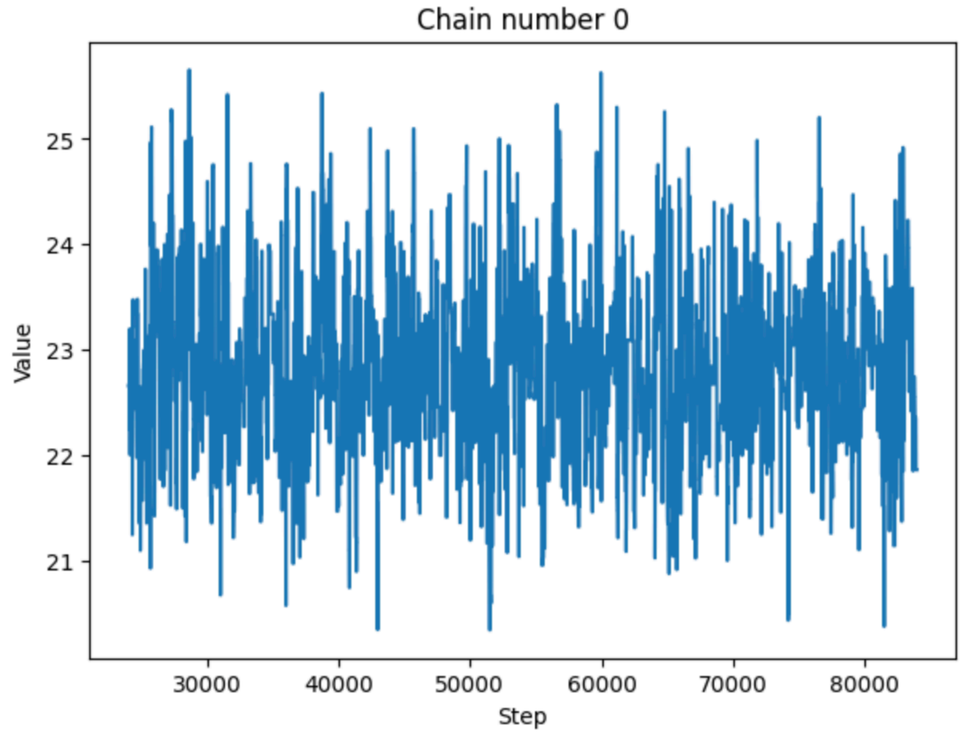
- $L_{1\text{-}4}$ must be in **nC**!
- *x & y* must be in PMTs coordinate and in **cm**
- *sc_integral* must be in camera ADC, needed to normalise the LY
- **File** as before

Fit output:

- **Only chains** of the parameters!

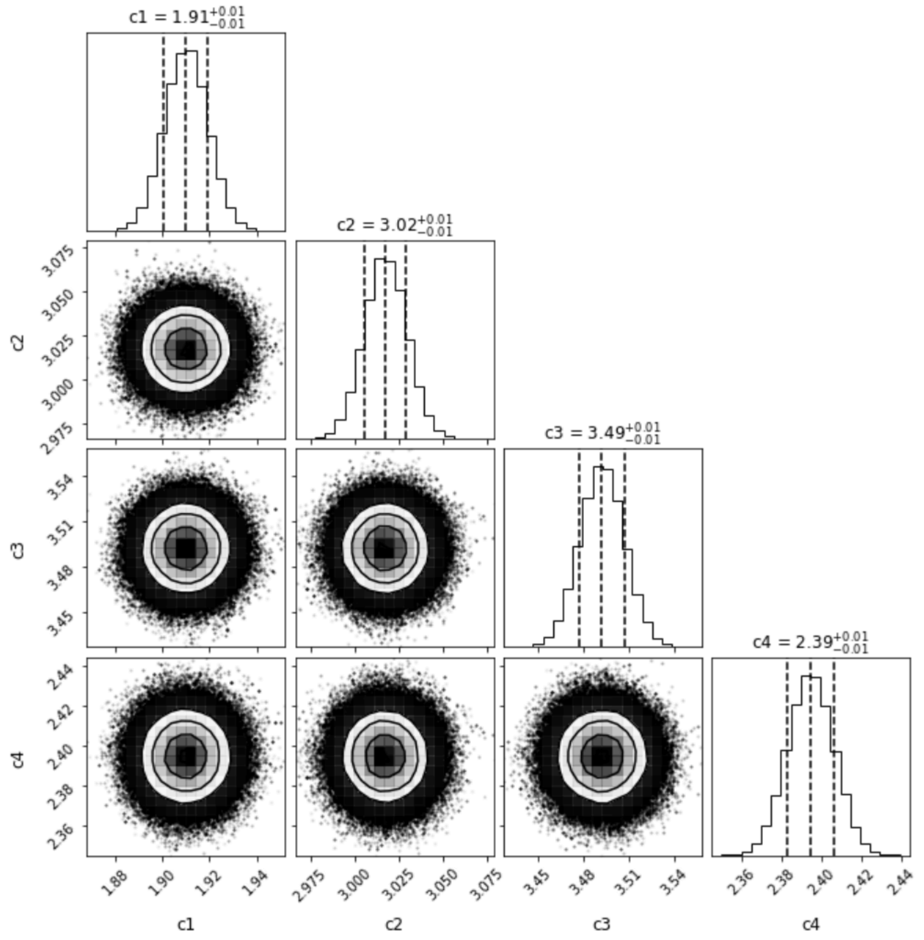
Plot the chains

In [19]:
```
variable = 'x'
for ch in df['Chain'].unique():
File display figure(figsize=(7,5))
    plt.plot(df[set_chain(df,ch)][variable])
    plt.xlabel('Step')
    plt.ylabel('Value')
    plt.title('Chain number %d'%ch)
    plt.show()
```



In [34]:
```
figure = corner.corner(samples,
                       labels=variables,
                       show_titles=True,quantiles=[0.16, 0.5, 0.84],
                       title_kwargs={"fontsize": 12},label_kwargs={"fontsize": 12})
plt.rcParams['xtick.labelsize'] = 10
plt.rcParams['ytick.labelsize'] = 10
plt.suptitle("Corner Plot", fontsize=16, y=1.02)
# plt.savefig('./corner_plot.pdf',format='pdf')
plt.show()
```

Corner Plot



How to read BAT chains output

Handy notebook made by Matteo F. as an example of how to read BAT chains output. Available in the repository.

main · Cygno_PMTs_fit / How_read_BAT_output.ipynb · Go to file · t

Matteo Folcarelli First version of the how-to-use notebook · 8fdd313 · 14 hours ago · History

Preview · Code · Blame · 433 lines (433 loc) · 553 KB · Code 55% faster with GitHub Copilot · Raw

In [1]:

```python
import numpy as np
import matplotlib.pyplot as plt
import corner
import pandas as pd
import uproot
```

In [2]:

```python
def set_chain(df,i):
    return ((df['Chain']==i)
            & (df['Phase']==1)
            )

def set_run(df):
    return ((df['Phase']==1))
```

Read the chains

BAT is set of C++ libraries implementing the Metropolis-Hastings MCMC algorithm, required for the sampling of the parameter space of the fit.

It's main output are MCMC chains and here there are some lines of code to a starting handling of such chains.

In [7]:

```python
output_filename = './output_associationassociation_golden_input.txt_0_mcmc.root'
mcmc = uproot.open(output_filename)
variables = ['L','x','y']
default = ['Chain','Iteration','Phase']
df = mcmc[mcmc.keys()[1]].arrays(default + variables,library='pd')
```

Notes

- **Be sure to check the input file!** Most of the times when the fit does not converge is when there are some problems with the inputs (wrong unit measures, wrong columns order, negative charges…)

- We can modify it to **run it in the reconstruction software**!

Conclusions

- Though **still in its preliminary phase**, this method has already proven to be a **valuable tool for matching the two readout systems**.

- **New calibration technique implemented!** Now uses all the points you give to the fit (i.e. 700) and fit the calibration parameters. You can use any dataset of 1 waveform to 1 spot and **the fit normalise the LY** using the camera variables.

- Will work on α!

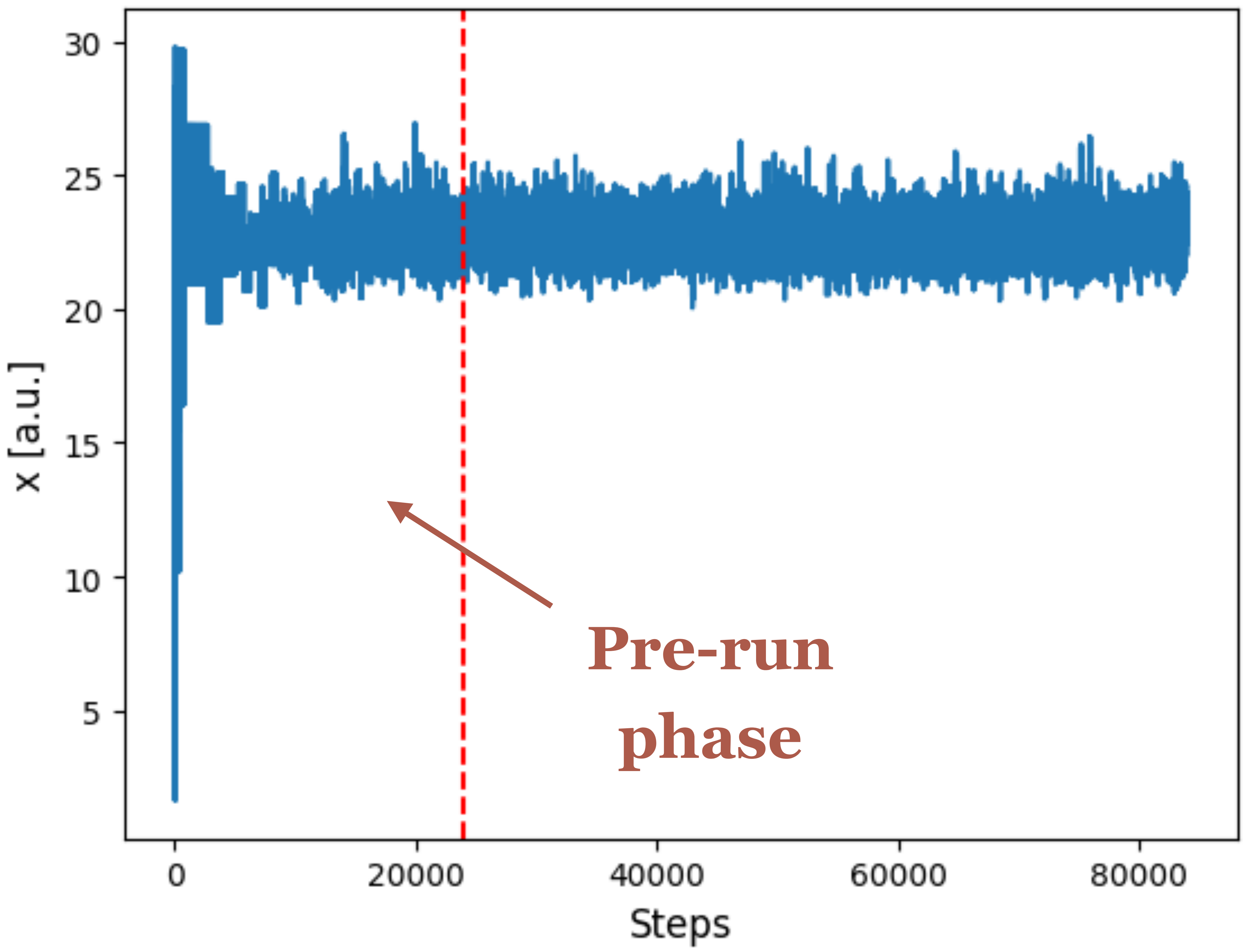
Thank you for the attention

Backup

Code implementation: BAT

The posterior of all the parameters is sampled using the **Metropolis-Hastings MCMC algorithm**, as implemented in **BAT.**

Convergence of the Markov chains is ensured by **BAT's pre-run phase**, which **tunes** the Metropolis-Hastings MCMC **parameters** in order all the parallel chains converge to the same region with an optimal Metropolis-Hastings MC rejection rate.



Method (5): longer waveforms - "energy" focused

- Slice the waveform in 50 samples slices
- Fit **all the slices** of the waveform **as a spot-like interaction**
- Roughly 6 times slower for background runs