Approaches to learning of the neutrino detector response function **probe**

Chuanhui Hao, Xinyang Wen, Boyuan Sun, Zhuoran Wang, Chuang Xu, Benda Xu

Tsinghua University

haoch23@mails.tsinghua.edu.cn

Apr 11, 2025

Contents

Contents

Definition of the probe

For vertex $\{E, \boldsymbol{r}, t_0\}$, the PE counts received on the J_{th} PMT follows an inhomogeneous Poisson process from a time-dependent function R_j.

- Probe function: $R_j(t; E, \boldsymbol{r}, t_0)$
- Expected PE counts: $\lambda_j = \int_0^T R_j(t; E, \boldsymbol{r}, t_0)\mathrm{d}t$. (marginal distribution)

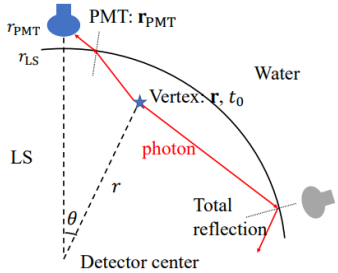


Fig 1: Geometric model for the probe

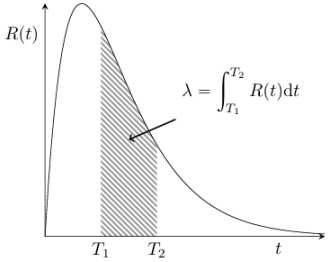
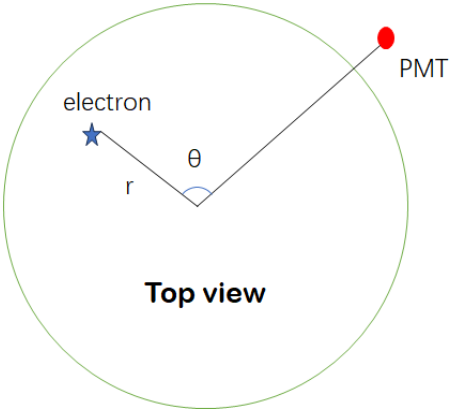
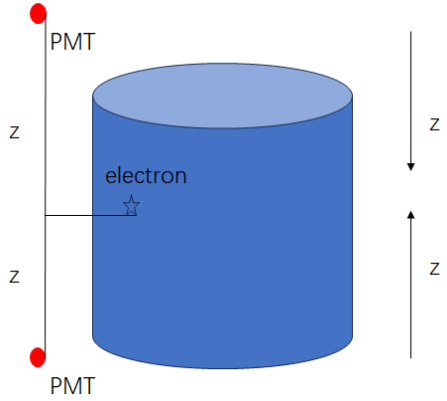


Fig 2: Schematic curve of $R(t)$

Relative coordinates for OSIRIS

- **Three coordinates**
  - r: the radius of the electron in the x-y plane
  - θ: the angle between the PMT and the electron in the x-y plane
  - z: the distance between the PMT and the electron in the z direction

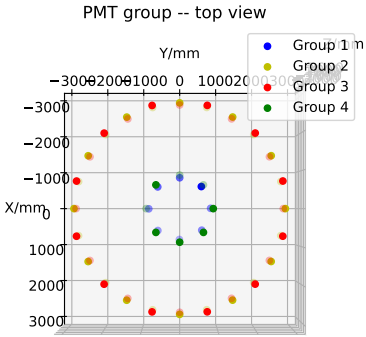


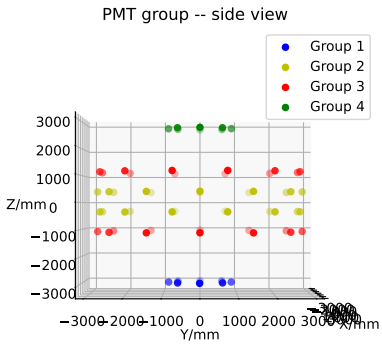


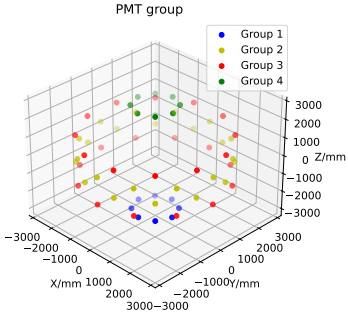
PMT groups

- **Four groups**

1. Bottom
2. Lateral, near the center
3. Lateral, away from the center
4. Top







OSIRIS probe

- Detector symmetry
- Linearity of visible energy
- Event time t_0 set to 0
- Divide the 64 PMTs into 4 groups, p is the PMT group.

$$R_p(t; E, \boldsymbol{r}, t_0) = E R_p(t; r, \theta, z)$$

$$\lambda_p = \int_0^T R_p(t; E, \boldsymbol{r}, t_0) \mathrm{d}t = E \lambda_p(r, \theta, z)$$

Probe score

- Likelihood function of inhomogeneous Poisson process is

$$P_p(\vec{t}|r,\theta,z,t) = e^{-\int_0^T R_p(r,\theta,z,t)\mathrm{d}t} \prod_{j=1}^{k} R_p(r,\theta,z,t_j)$$

- For N samples,

$$\mathcal{L} = \prod_{i=1}^{N} P_p(\vec{t_i}|r_i,\theta_i,z_i) = \prod_{i=1}^{N} e^{-\int_0^T R_p(r_i,\theta_i,z_i,t)\mathrm{d}t} \prod_{j=1}^{k_i} R_p(r_i,\theta_i,z_i,t_{ij})$$

- Score is

$$\log\mathcal{L} = \sum_{i=1}^{N} \left(-\int_0^T R_p(r_i,\theta_i,z_i,t)\mathrm{d}t + \sum_{j=1}^{k_i} \log R_p(r_i,\theta_i,z_i,t_{ij}) \right)$$

Contents

Motivation

Wei Dou suggests that ideally a probe from histogramming the simulation is the best.

Limitations of **histogram probe**

- it is discrete and non-differentiable.
- It relies on simulations with large statistics.

1. We are looking for an elegant way to get continuous and differentiable **probe** with good scores.
2. GAM, GBM, MLP and so on maybe good methods to do this.

Generalized Additive Model(GAM)

$$g(E(Y)) = \alpha + f_1(x_1) + \cdots + f_p(x_p) + \varepsilon$$

- We can use such a linear form to regress the target variable.
- g is link function, f_p is smooth function.
- For GAM model, smooth functions do not necessarily correspond to only one variable, but can be tensor product functions between different input variables.
- Minimize the penalized residual sum of squares:

$$\mathrm{PRSS}(\alpha, f_1, f_2, \ldots, f_p) = \sum_{i=1}^{N}(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}))^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)^2 dt_j$$

- For the base of smooth function we select the B-spline.

Generalized Additive Model(GAM): $\lambda(x, y, z)$

Get histogram like this:

x	y	z	nEV	nPE
...	...	...	...	...

- x: mean value of x of events in bins(from r,θ).
- y: mean value of y of events in bins(from r,θ).
- z: mean value of z of events in bins.
- nEV: Event counts in bins.
- nPE: PE counts in bins.

Simulation data:

- 1 MeV e^-
- uniformly in LS.

regression formulation

$$\mathrm{nPE} \sim \mathrm{Poisson}(\lambda), \log \lambda = \mathrm{te}(x(r,\theta), y(r,\theta), z) + \log(\mathrm{nEV})$$

te means the tensor product between different variables.

Generalized Additive Model(GAM): $R(x, y, z, t)$

Get histogram like this:

x	y	z	t	nEV	nPE
...	...	...	...	...	...

- x: mean value of x of events in bins(from r,θ).
- y: mean value of y of events in bins(from r,θ).
- z: mean value of z of events in bins.
- t: center value of t bins.
- nEV: Event counts in bins.
- nPE: PE counts in bins and t bins.

regression formulation

$$\mathrm{nPE} \sim \mathrm{Poisson}(\lambda_{\mathrm{tbin}})$$

$$\log R = \mathrm{te}(x(r,\theta), y(r,\theta), z, t) + \log(\mathrm{nEV}) + \log(\mathrm{tbinwidth})$$

te means the tensor product between different variables.

Gradient Boosting Machine(GBM)

- **GBM Principle:** Iterative ensemble method combining weak learners (decision trees) through gradient descent optimization.
- **Loss Function:** Mean Squared Error (MSE) for regression tasks:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Minimized via additive model training.

Input Features:

- Spatial coordinates: (r, θ, z)
- Vertex event parameters

Output:

- PE time sequence: $\{t_{PE}^1, t_{PE}^2, \dots\}$
- Amplitude sequence: $\{A_{PE}^1, A_{PE}^2, \dots\}$

- Convolve output sequences with scintillator distribution $P_{scint}(t)$:

$$R(t) = \left(\sum_k A_{PE}^k \delta(t - t_{PE}^k) \right) \otimes P_{scint}(t)$$

GBM Training

GBM training process

- **Tuning:** Bayesian Optimization (BO) identifies optimal hyperparameters for LightGBM with only 1/100 dataset.
- **Parameter:** Optimal parameters from BO, using full dataset to train.

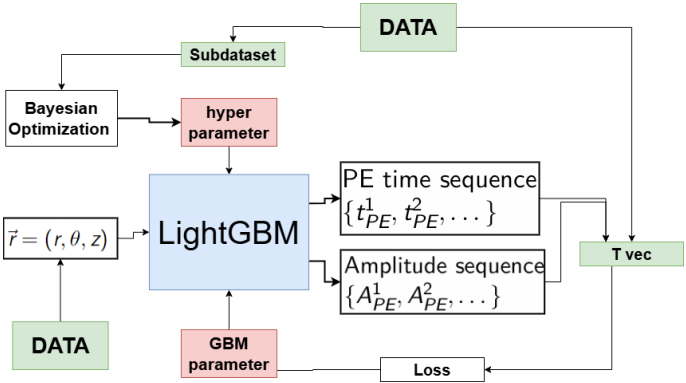


Fig 3: GBM training

Multilayer Perceptron(MLP)

We also try to use the Neural Network for Time-Vector Regression.

MLP Training process

1. Convert Time-Vector to Step Function.
   - **Input:** Time vector $\vec{t_j} = \{t_{PE}^1, t_{PE}^2, \dots\}$ containing PE event times.
   - **Step Function:** Discrete-to-continuous conversion via:

$$N(t) = \sum_i H(t - t_{PE}^i), \quad \text{where } H(t) = \begin{cases} 1 & t > 0 \\ 0 & \text{otherwise} \end{cases}$$

   - $N(t)$: Cumulative PE counts over time.
2. Poisson Loss:

$$\mathcal{L}_{\text{Poisson}} = \sum_i \left(\lambda(t_i) - N(t_i) + N(t_i) \log \frac{N(t_i)}{\lambda(t_i)} \right)$$

3. Objective: Train NN to approximate $\lambda(t) \to N(t)$

Contents

Score results

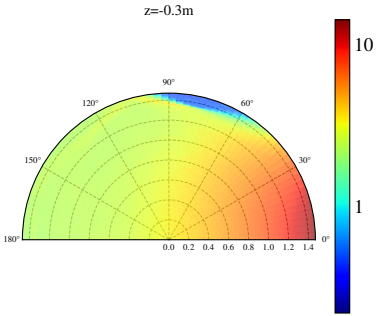
$$\log \mathcal{L} = \sum_{i=1}^{N} \left(- \int_0^T R_p(r_i, \theta_i, z_i, t) \mathrm{d}t + \sum_{j=1}^{k_i} \log R_p(r_i, \theta_i, z_i, t_{ij}) \right)$$

Table 1: Scores for different models

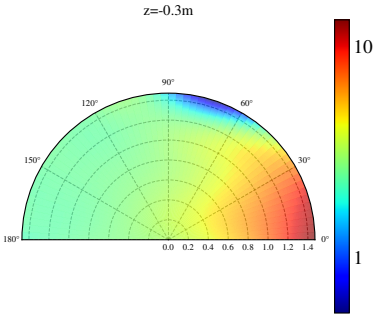
model	score1	score2	score3	score4
hist[865,60,72,100]	-1851714168.4	-6415594302.6	-5563946722.8	-2205449463.9
GAM[20,10,20,10]	-2007537861.7	-7222560337.7	-6231036604.4	-2413635858.5
GAM[20,10,20,20]	-1821995412.5	-6192649788.0	-5391145670.4	-2150804122.3
GBM	-1991698572.6	-6602019202.7	-5788576582.2	-2296800533.0
MLP	-2747704064.0	-9381692416.0	-8270123008.0	-3248942592.0

- GAM is better than histogram, and get the best scores among all the approaches.
- GBM is another approach with great potential, the scores of MLP is not good.

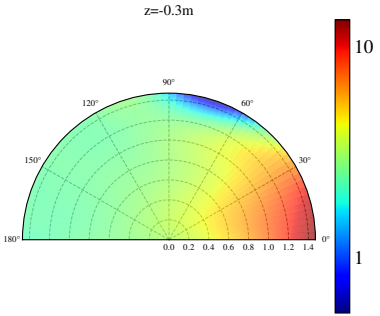
r-θ Plots of λ for group2(lateral PMTs)



hist[865,60,72,100]



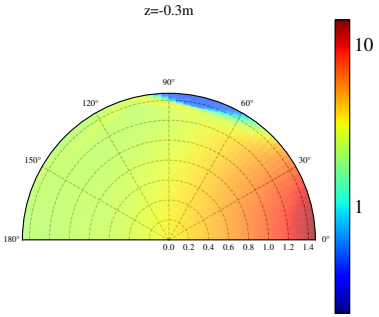
gam[20,10,20,10]



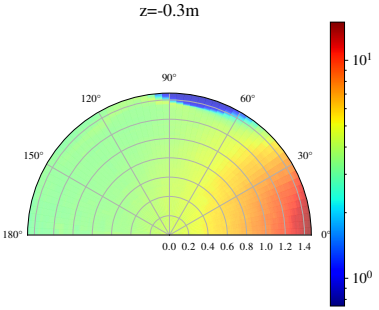
gam[20,10,20,20]

- GAM probe can describe all regions including total reflection region accurately.
- The expected PE counts can be accurately predicted by GAM.

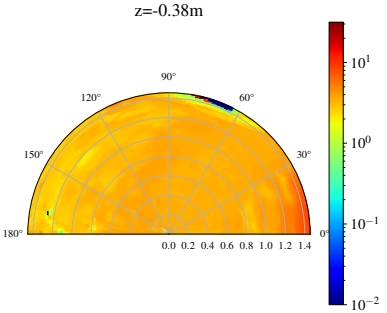
r-θ Plots of λ for group2(lateral PMTs)



hist[865,60,72,100]



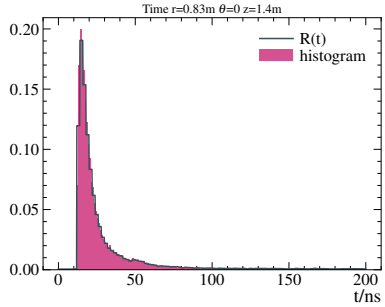
GBM



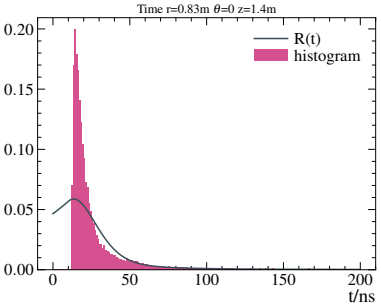
MLP

- GBM shows great potential.
- MLP has notable optimization opportunities.

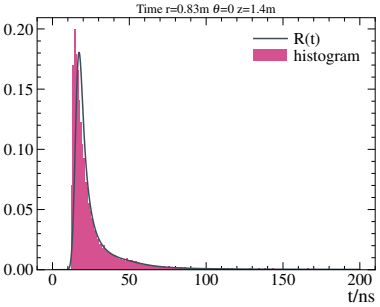
Conditional distribution of PE time for group2(lateral PMTs)



hist[865,60,72,100]



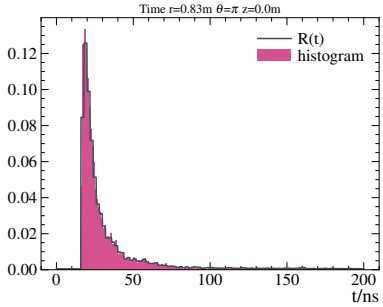
gam[20,10,20,10]



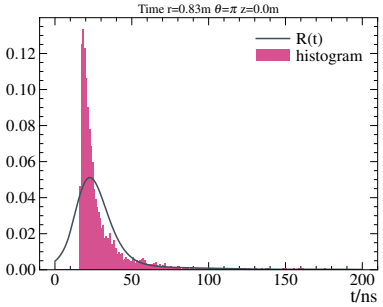
gam[20,10,20,20]

- The red histogram is the MC truth of PEtime.
- The $R(t)$ is the result predicted by GAM.
- GAM with more t-bins has a significantly better performance.

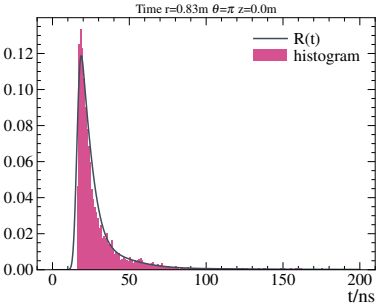
Conditional distribution of PE time for group1(bottom PMTs)



hist[865,60,72,100]



gam[20,10,20,10]



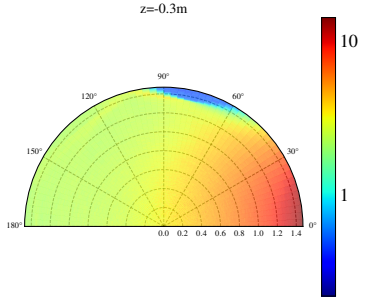
gam[20,10,20,20]

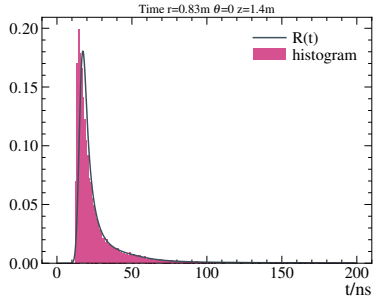
- The red histogram is the MC truth of PEtime.
- The $R(t)$ is the result predicted by GAM.
- GAM with more t-bins has a significantly better performance.

Contents

Summary

1. We build the response model **probe** based on the geometry and symmetry and give the evaluation of **probe**.
2. We try several approaches to get the continuous and differentiable result.
3. Up to now, GAM is the approach with the best performance, With smaller number of bins than histogram, GAM performs even better.





Goal of the workshop

- We use the bam function from the mgcv package in R to perform generalized additive model (GAM) fitting.
- The limitation of the current fitting is that λ and R require separate fitting procedures. But due to their mathematical relationship, a single fitting process should suffice.
- So we want to use a monotonic spline in the t-direction to fit the integral of R, then take the derivative to get R. This way we only need one fitting, and theoretically the fitted function will be smoother.

$$\lambda(\vec{r},t) = \int_0^t R(\vec{r},\tau)\mathrm{d}\tau, \lambda(\vec{r},T) = \lambda, \frac{\mathrm{d}\lambda(\vec{r},t)}{\mathrm{d}t} = R(\vec{r},t)$$

- The mgcv package doesn't support this kind of monotonic spline fitting for tensor product functions. We hope to achieve this requirement by modifying the source code of the mgcv package.

Thank you for listening!
Q & A

Backup: B-spline

- $n+1$ data points $P_0, P_1 ... P_n$, $m+1$ knots $t_0, t_1 ... t_m (t_0 \leq t_1 \leq ... t_m)$.
- Recursive definition of B-spline base function $B_{i,p}(t)$:

$$B_{i,0} = \begin{cases} 1 & (t_i \leq t \leq t_{i+1}), \\ 0 & \text{otherwise.} \end{cases}$$

$$B_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} B_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(t)$$

- k-order B-spline curve $B(t) = \Sigma_{i=0}^{m-k-1} P_i B_{i,k}(t)$.