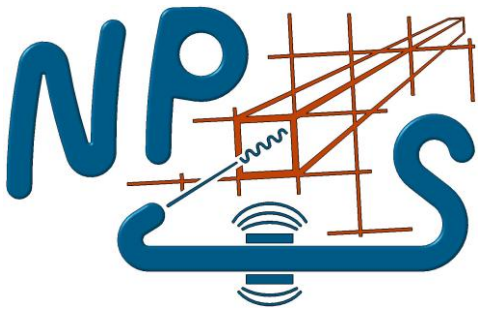
NPS Waveform Analysis

Mitchell Kerver

Old Dominion University

1/27/2026



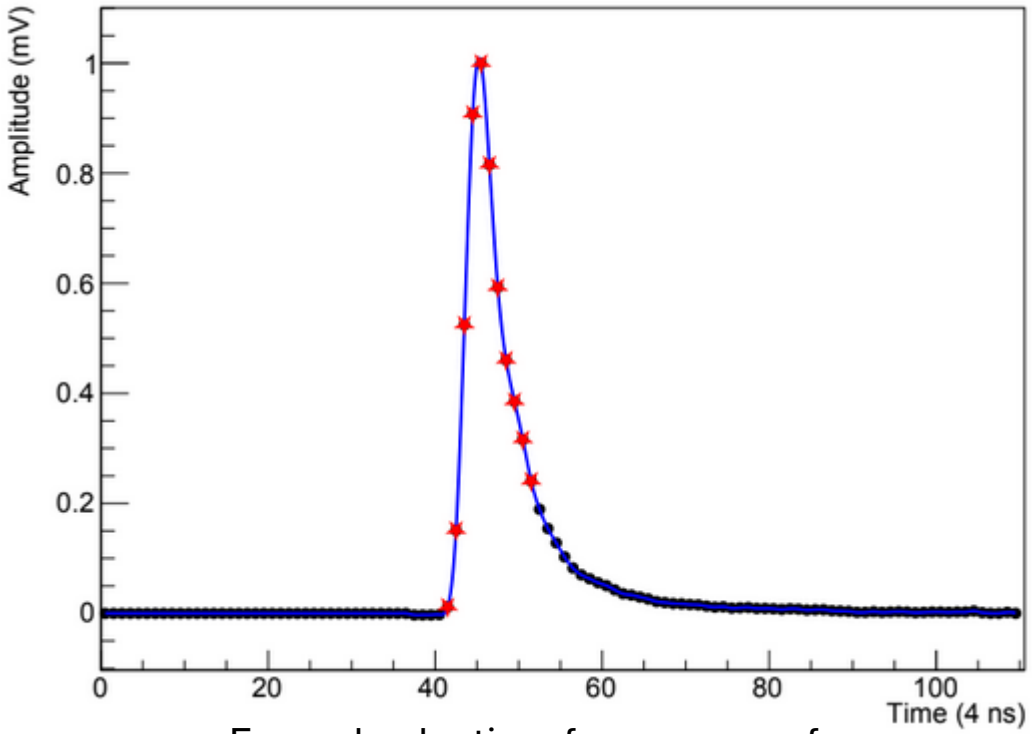
Waveform Analysis: Motivation

- Missing mass technique used to determine the reaction exclusivity.
- Energy resolution of the detected photon(s) is the dominant constraint on the missing mass resolution.
- Waveform analysis allows event-by-event reconstruction of pulses from the digitized waveforms. Providing accurate pulse times/amplitudes.
- More effective than naïve peak finding techniques in high-rate environments with overlapping pulses and noise.

Reference Waveforms

- A unique fit function is constructed for **each** calorimeter channel from a reference waveform.
- The reference waveforms are picked from clean, low noise events from elastic runs.
- The discrete reference waveforms are interpolated into a continuous function using Spline Interpolation:

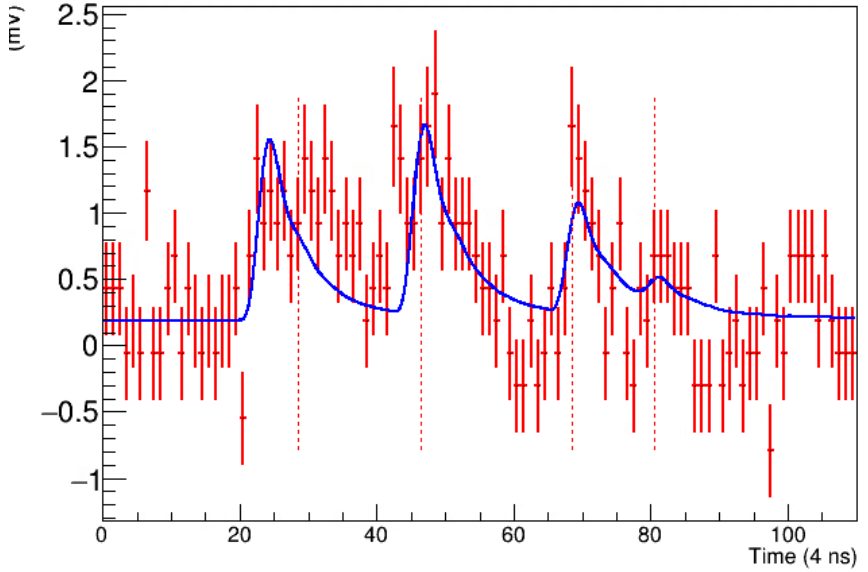
$$S_i(t) = a_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3$$



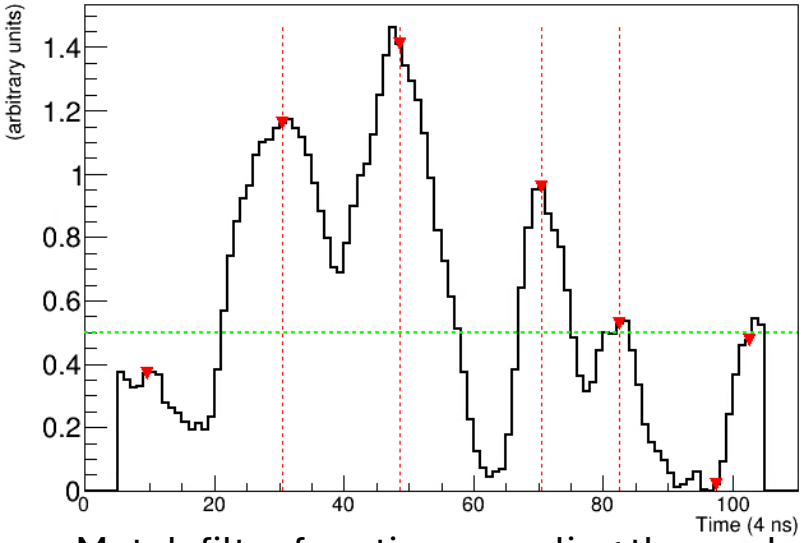
Example elastic reference waveform (red) and cubic spline *S(t)* (blue)

Peak Finding

- *Matched filter* technique maximizes signal-to-noise ratio by correlating each waveform with the reference waveform.
- The matched filter function is constructed by moving a time-reversed reference waveform across every sample in the data waveform and computing the dot product.
- The matched filter function is large in regions where the data is highly correlated to the reference waveform, suppressing noise.



Raw ADC data from a low amplitude, noisy event



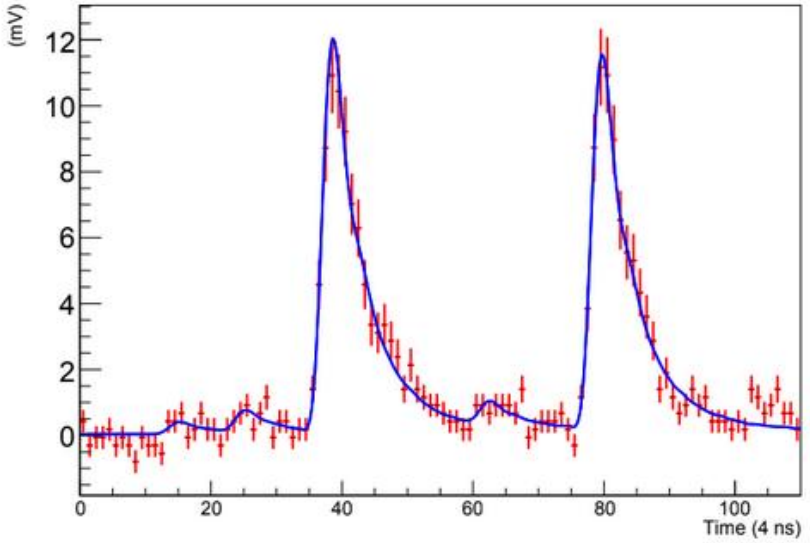
Match filter function revealing the peak positions from noise

Waveform Fitting

- The fit function is constructed from the cubic spline of the reference waveform:

$$F(t) = B + \sum_{i=1}^{Npulses} A_i \; f(t - t_i)$$

Baseline — **Amplitude** — **Time of pulse #i (4*ns) relatively to the ref wf time**



- Initial parameters A_i and t_i are assigned simply from the values of the peaks found in the matched filter.
- χ^2 is minimized with respect to A_i and t_i using the *Migrad* algorithm from the Minuit2 library:

$$\chi^2(A_i, t_i) = \sum_{k=1}^{N} \frac{\left[x_k - F(t_k; A_i, t_i)^2\right]^2}{\sigma^2}$$

Value of data at time t_k

Value of fit function at time t_k given parameters A_i and t_i

$$\frac{\partial x^2}{\partial A_i} = 0 \qquad \frac{\partial x^2}{\partial t_i} = 0$$

Multi-Threading

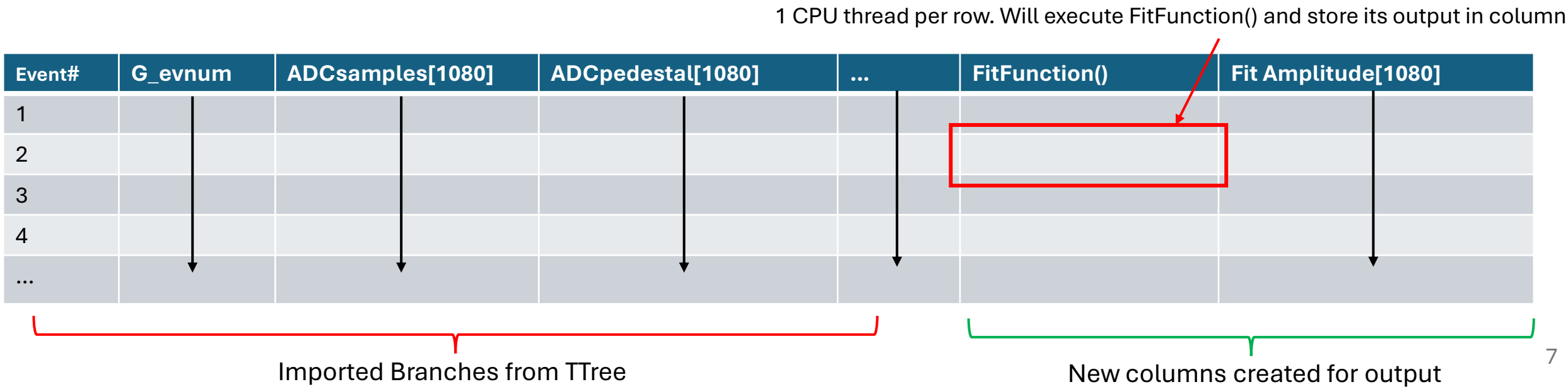
- Runs are broken up into multiple segments with up to ~500k events per segment.
- Original waveform fitting code estimated to take up to 25 days for the longest segments!
- Max wall time on the farm CPUs = 2 days -> jobs will fail
- Solutions:
  - ✘ Further subdivide each segment, run wf analysis, then recombine.
  - ✓ Run wf analysis on full segments across multiple CPU threads. Give each job $\#\text{of cores} \approx \frac{Days\ expected}{2\ Day\ walltime}$

Kinematics	Run	Beam	Target	Prescale	Total Entries	Time/Event	Total Fits	Time/Fit	Fits/Entry	Actual CPU	Est. Time
kinC_x25_4	4986	20 uA	LD2	PS6=0	260854	2.73	1360516	12.28	5.2156	0 days, 4 hours	8 days, 6 hours
kinC_x25_1	5919	20 uA	LD2	PS6=0	111547	6.61	3687865	11.28	33.0611	0 days, 11 hours	8 days, 12 hours
kinC_x50_0	5437	20 uA	LD2	PS6=0	295105	2.57	1374502	11.54	4.6577	0 days, 4 hours	8 days, 18 hours
kinC_x36_6	4714	20 uA	LD2	PS6=0	136582	7.31	2773186	16.47	20.3042	0 days, 12 hours	11 days, 13 hours
kinC_x50_0a	5249	20 uA	LD2	PS6=0	130320	9.55	2923283	20.12	22.4316	0 days, 16 hours	14 days, 9 hours

Table of some kinematics and estimated compute time for WFA

Multi-Threading Existing Code: RDataframes

- ROOT's multi-thread friendly interface for data analysis built around a column/row structure.
- Rows represent the event number, and Columns represent a TTree branch, variable, or even a function!
- Fill the first columns with branches from the input ROOT file
- Make new columns containing logic/variables needed for WF fitting.
- No Looping over events anymore. ROOT handles filling all columns for each row on different threads.
- When all entries are filled…snapshot the Rdataframe into a normal Root Ttree for output!



Multi-Threading Existing Code: RDataframes

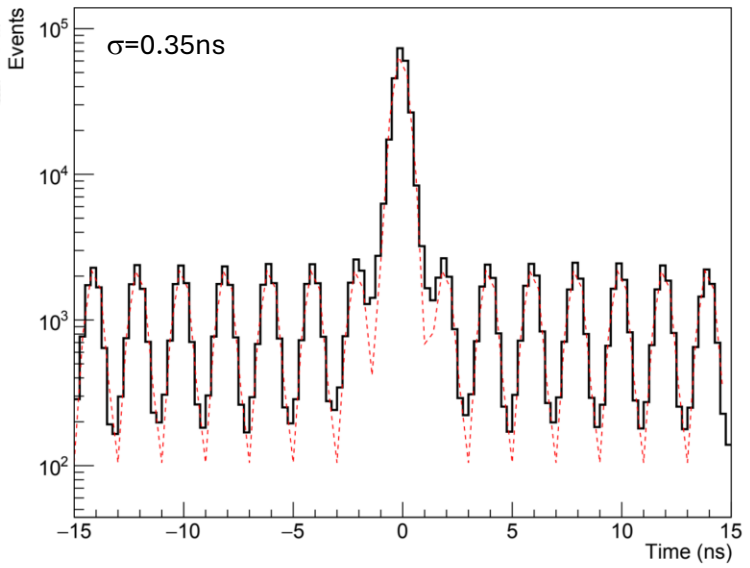
- Create a RDataframe with a new column that is a function which contains ALL the logic within the event loop of the original code.
- ROOT sends buckets of events(rows) to different cpu threads to be evaluated asynchronously.
- Whenever a thread evaluates a row of the RDataframe it will call the function with a unique set of all it's local variables (histograms, interpolator, fit function, etc)
- The output of that function is stored in that column for the corresponding row.

- Benefits:
  - Maintain exact same logic and data structures as original code
  - Same fitting routine -> Outputs should be identical (with very small numerical differences)
  - Avoid having to deal with Mutex, I/O, and threads sharing variables

Resolution Results

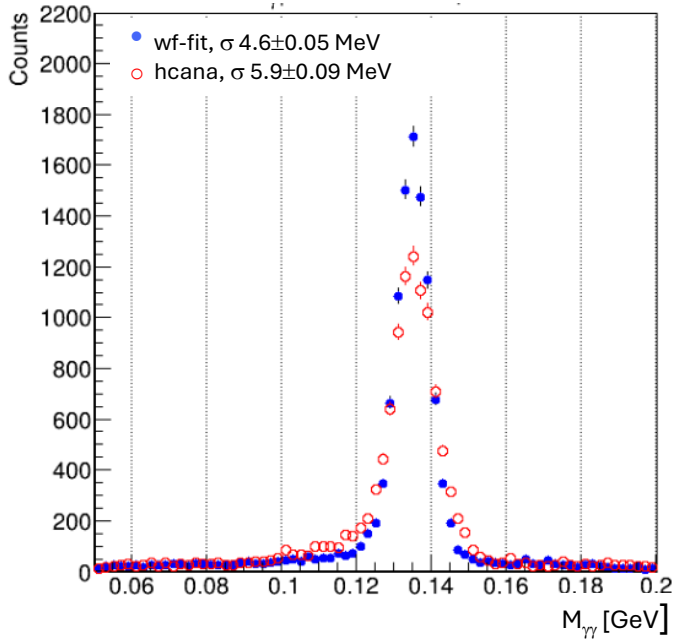
- Using the pulse amplitudes from the waveform fit shows improvements in the missing mass and invariant mass resolution.
- Timing resolution mostly unchanged, but good. (see Wassim's presentation)

t_e(HMS)-t_γ (NPS-Calo)

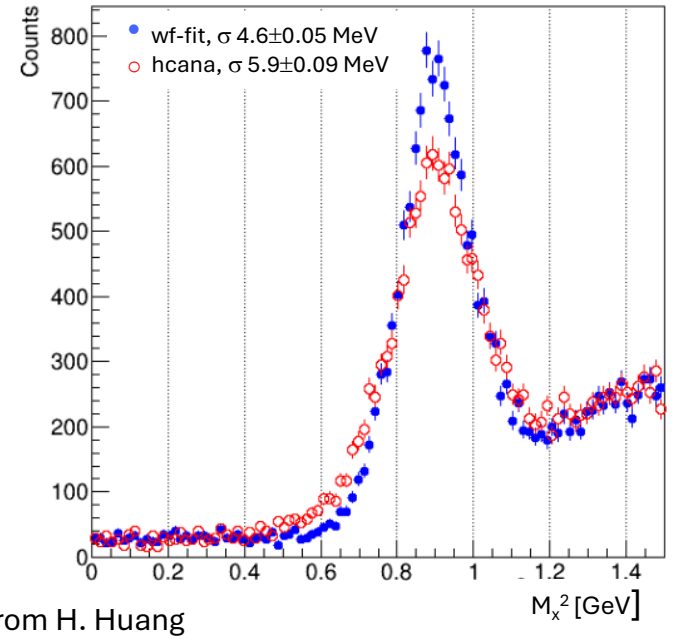


Plot from W. Hamdi

π^0 Invariant Mass from 2 Photons



H(e,e'γγ)X : Missing Mass Squared for π^0

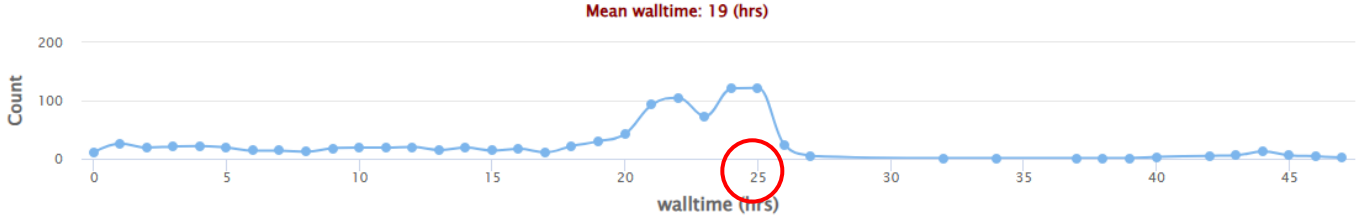


Plots from H. Huang

Replay Status

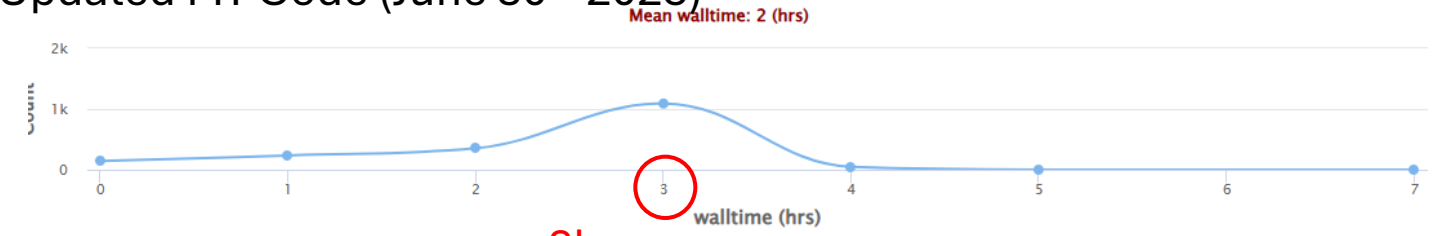
- 7 Kinematics completed in 2025. ~4,000 segments.
- All runs that previously failed due to exceeding max walltime, now complete with MT.
- Code optimizations beyond the MT have additionally improved compute time. Time improvements have ranged from 3.9x – 13x faster than previous code (depending on # of cores used (2-8))

Old Method (May 19th 2025)



25hr

Updated MT Code (June 30th 2025)



3hr

Kinx60_4_b and Kinx60_2 Walltime

Thank You!

Special Thanks to Wassim Hamdi, Malek Mazouz, and Casey Morean for contributions to the code. To Hao Huang for help testing the results. And the rest of the NPS Collaboration members for guidance.