

A study on fiTQun performance

V. V. Gligorov, M. Guigue, G. Díaz López, LPNHE, 25.02.2025







Introduction

The idea of this study is to

- 1. Profile fiTQun performance
- while controlling the impact on physics performance

All results have been obtained running the prefit branch on a Macbook equipped with an M3 Max processor and 64 GB of ram

Profiling is done using the CLion IDE's built-in tools

2. Understand which parts of the reconstruction take the most time 3. Begin developing some tools to allow them to be made faster





Input datasets

Used simulated samples of 300 events each of

- 1. Electrons
- 2. Non-decaying muons
- 3. Decaying muons

Generated with WCSim v1.12.19

Particles generated with 500, 750, and 1000 MeV kinetic energy

Work ongoing to also look at proton decay and pi0 events



Flamegraph of fiTQun

Itali Itali Vitali Vitali Vitali alanuy Itali V.alanuy Itali	libfiTOup		libfiTOurlib 6.2.4 dylib `TSoot]
libiti QunLib.6.3.4.dylib Tit Qun.:GetwiRnglog libiti Qu	libMipuit		libit Quillib.6.3.4.dylib 15cati
	libMinuit.	libfiTOur	libfiTQunLib.6.3.4.dylib fiTQun
libMinuit.so`TMinuit::mnsimn		libMinuit	libMinuit so`TMinuit: Eval
libMinuit.so`TMinuit::mneycm	libMinuit (lib	libMinuit	libMinuit.so`TMinuit
libMinuit.so `TFitter::ExecuteCommand	libfiTQunl libfiTQu libfi	libMinuit	libMinuit.so`TMinuit::mnexcm
libfiTQunl ib 6.3.4 dvlib`fiTQun: MultiRingFit(int_int*_	libfiTQunl ib 6.3.4 dvlib	libMinuit	libMinuit.so`TFitter::ExecuteCo
libfiTQunLib.6.3.4.dvlib`fiTQun::OneMoreRing	. nonrigunzio.o. nayno	libfiTQur	libfiTQunl ib.6.3.4.dvlib`fiTQun
libfiTQunLib.6.3.4.dvlib`fiTQun::DoMRFit			libfiTQunLib.6.3.4.dvlib`fiTQun
runfiTQun`main			
dyld`start			

This is obtained running the prefit branch "out of the box" on some decaying muon events. Notice that the MR + PiO fits take > 50% of the total time (but contribute nothing on these events). Will come back to this later.







Flamegraph of fitQun

		1	81			
libfiTQunLib.e libfiTQunLib libfiTQ libfi libf libfi						
libfiTQunLib.6.3.4.dylib`fiTQun::GetMRnglog libfiTQu	libfiTQun			libfiTQunLib	6.3.4.dylib`	TScatT
libfiTQunLib.6.3.4.dylib`fiTQun::MultiRingWrapper	libMinuit.			libfiTQunLib	6.3.4.dylib`	fiTQun
libMinuit.so`TMinuit::Eval	libMinuit.		libfiTQur	libfiTQunLib	6.3.4.dylib`	fiTQun
libMinuit.so`TMinuit::mnsimp	libMinuit.		libMinuit	libMinuit.so`	TMinuit::Eva	al
libMinuit.so`TMinuit::mnexcm	libMinuit.: lib		libMinuit	libMinuit.so`	TMinuit::mn	isimp
libMinuit.so`TFitter::ExecuteCommand	libfiTQunl libfiT	Qu libfi	libMinuit	libMinuit.so`	TMinuit::mn	excm
libfiTQunLib.6.3.4.dylib`fiTQun::MultiRingFit(int, int*, o	libfiTQunLib.6.3	.4.dylib`	i libMinuit	libMinuit.so`	TFitter::Exe	cuteCo
libfiTQunLib.6.3.4.dylib`fiTQun::OneMoreRing libfiTQ			libfiTQur	libfiTQunLib	6.3.4.dylib`	fiTQun
libfiTQunLib.6.3.4.dylib`fiTQun::DoMRFit libfiTQunLib.6.3.4.dylib`fiT					fiTQun	
runfiTQun`main						
dyld`start						

The 6D interpolation used in the scattering table is about 35% of all the function calls in fiTQun





Impact of 6D interpolation

In order to better understand the impact of the 6D interpolation, decided to replace it with a "nearest neighbour" approach

belongs to, however when going float (data point) to integer (bin changed this to calculate and then return the closest one.

- To do this, modified the existing GetInterpVal function. The function
- was already computing which bin in the scattering table our event
- index) it was rounding down instead of calculating the closest bin. I



Flamegraph of fiTQun after change

	W I I	
		1
libfiTQunLib.6.3.5.dyl libfiTQunLib.6.3.5. libfiTQunL libfiTQ libfiT libfiTQ	libfi and a lib	
libfiTQunLib.6.3.5.dylib`fiTQun::GetMRnglogL(int, int*, double (*) [libfiTQunLib	libfiTQunLib.	
libfiTQunLib.6.3.5.dylib`fiTQun::MultiRingWrapper	libMinuit.so`1	
libMinuit.so`TMinuit::Eval	libMinuit.so`T	libfi1
libMinuit.so`TMinuit::mnsimp	libMinuit.so`T	libM
libMinuit.so`TMinuit::mnexcm	libMinuit.so`T libfi	libM
libMinuit.so`TFitter::ExecuteCommand	libfiTQunLib.6 libfiTQunLib libfiT	libM
libfiTQunLib.6.3.5.dylib`fiTQun::MultiRingFit(int, int*, double (*) [8], int&, bool, boo	libfiTQunLib.6.3.5.dylib`fiTQun::Sea	libM
libfiTQunLib.6.3.5.dylib`fiTQun::OneMoreRing		libfiT
libfiTQunLib.6.3.5.dylib`fiTQun::DoMRFit		
runfiTQun`main		
dvld`start		

all

Impact of removing the 6D interpolation clear

Benchmarking indicates a ~20% speedup in the time taken per event by using the closest point rather than interpolating





Impact on physics performance

Studied the impact on physics performance using a (very) simple configurable python script which takes the ntuples produced by fiTQun as input

Ran on each simulated sample with the 6D interpolation and the nearest neighbour approach and compared

- 1. PID assigned to each sub-event
- 2. Reconstructed particle momentum
- 3. Reconstructed particle direction





Ran over 300 muon events generated at 500 MeV in which the muon decays One pi+ (6D) identified as an electron (closest) Differences in recoed momentum and directions are fairly small, tails to be understood

- The 6D Interpolation and the closest bin method agree on the best PID for 546/547 sub-events







Ran over 300 muon events generated at 500 MeV in which the muon does not decay

bigger and there is a bigger offset from 0 in the difference of reconstructed momenta

- The 6D Interpolation and the closest bin method agree on the best PID for all sub-events
- Differences in recoed momentum and directions are still relatively small, but the tails become



Physics performance differences (3)



Ran over 300 electron events generated at 500 MeV The 6D Interpolation and the closest bin method agree on the best PID for all sub-events

Differences in recoed momentum and directions are bigger than for muons



How can we make things faster? Tried the following things in addition to the nearest-neighbour approach for scattered light

- 1. Turn off the final fit in the One Ring fit
- 2. Turn off the Multi Ring fit
- 3. Turn off the Pi0 fit

All together these gain 4-5x in speed compared to the "out of the box" fiTQun. While (1) has an impact on the physics, (2) and (3) have no impact on the reconstructed best PID hypothesis, momentum, or direction in these specific MC samples (expected but good!).



Removing final 1R fit: electrons



All results shown for the combination of using the nearest-neighbour approach for scattered light and removing the final One Ring fit.



Removing final 1R fit: muons



All results shown for the combination of using the nearest-neighbour approach for scattered light and removing the final One Ring fit.



Removing final 1R fit: non-decaying muons



All results shown for the combination of using the nearest-neighbour approach for scattered light and removing the final One Ring fit.



Flamegraph of fiTQun for 1R fit

	libfiTQunLib.6.3.5.
	libfiTQunLib.6.3.5.dylib`fiTQun::Sr libsystem_m.dy libfiTQu li
	libfiTQunLib.6.3.5.dylib`fiTQun::OneRingWrapper
	libMinuit.so TMinuit::Eval
libfiTQunLib.6.3.5.dvlib`fiT libfiTQunLib.6.3 libfiTQ libfiTQ libfi	libMinuit.so`TMinuit::mnexcm
libfiTQunLib.6.3.5.dylib`fiTQun::DirGridSrch	libMinuit.so`TFitter::ExecuteCommand
libfiTQunLib.6.3.5.dylib`fiTQun::SetOneRingSeed	libfiTQunLib.6.3.5.dylib`fiTQun::OneRingFit
libfiTQunLib.6.3.5.dylib`fiTQun::Do1RFit	
runfiTQun`main	
dyld start	

30% of time is taken by reading shared parameters?



Conclusions and next steps Carried an initial look into fiTQun performance. switching to a nearest-neighbour approach (since we do not seem to be memory-bound we could make the lookup table more granular in some regions?) could gain roughly 4x in computational time. Final one ring fit seems to make things worse for electrons? reconstruction, worth committing for validation purposes? after cuts as people show in HK meetings? what is the most useful place for me to contribute.

- For processing MC already turning off the multi-ring and PiO fits and
- Have written a simple script to compare two output ntuples from the

 - How can I do more fine-grained comparisons of PID performance
- Very happy + motivated to keep working on this and get suggestions









BACKUPS

Memory usage and scaling

this is allocated at the start of the processing

with the number of physical but also virtual cores set process priorities and copy input files to avoid I/O limits

To be verified on cc-in2p3 servers as a follow-up

- An instance of fiTQun uses about 1 GB of memory, and basically all of
- This means it should be possible to run multiple jobs in parallel on more or less any modern x86 server and get a good scaling not only Not the case out of the box with Apple silicon even if I use nice to





Reconstructed momenta for muons



Resolution is noticably worse at 500 MeV for the "closest" setting, 15.9 vs. 10.8 MeV Mean is also a little bit more biased in the "closest" setting than in the 6D

Similar conclusions hold for the non-decaying muons





Reconstructed momenta for electrons



Resolution is a bit better at 500 MeV for the "closest" setting, ~16.3 vs. ~17.3 MeV Mean is also a little bit *less* biased in the "closest" setting than in the 6D

