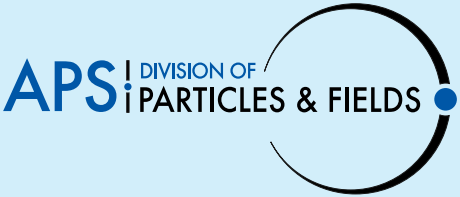
GPUs in LHCb for Analysis

Henry F. Schreiner[1] on behalf of the LHCb collaboration

August 3, 2017

[1]University of Cincinnati



NVIDIA GPUs

- Programing language: CUDA
- Massively parallel identical operations
- Separate memory model (coprocessor)

	Name	Stream processors	Clock	TFLOPS	Cost
Gamer	GTX 1050 Ti	768	1290 Mhz	1.98	$150
	GTX 1080 Ti	3,584	1596 Mhz	11.3	$850
Server	Tesla K40	2,880	745 Mhz	4.29	$3,000
	Tesla P100	3,584	1329 Mhz	9.3	$10,000

GooFit

CPU and GPU fitting package

Hydra

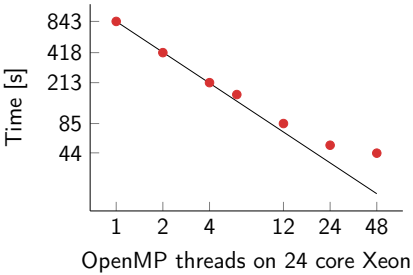
CPU and GPU system for HEP computation

Manet

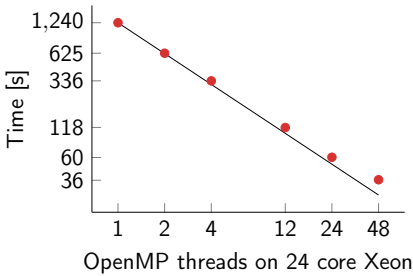
Energy test GPU code

/GooFit/GooFit LGPLv3

- Designed for speed; resembles the popular RooFit package in ROOT
- Built for CUDA or OpenMP using the Thrust library
- Binned and unbinned fits; 3- and 4-body time integrated and dependent analyses
- Composed in C++ 2.1 (Python coming soon)



[CHEP 2013]



$\pi\pi\pi^0$ 3-body 16 amplitudes

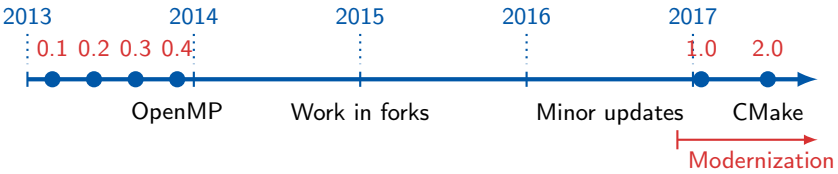
- Original RooFit code: 19,489 s

CPU	Core 2 Duo	1,159 s
GPU	GeForce GTX 1050 Ti	86.4 s
GPU	Tesla K40	64.0 s
MPI	Tesla K40 $\times 2$	39.3 s
GPU	Tesla P100	20.3 s

ZachFit: $D^{*+} - D$ BaBar measurement

- 142,576 events in unbinned fit

CPU	Core 2 Duo	738 s
GPU	GeForce GTX 1050 Ti	60.3 s
GPU	Tesla K40	96.6 s
MPI	Tesla K40 $\times 2$	54.3 s
GPU	Tesla P100	23.5 s



CMake: New build features

- IDEs, macOS, multiple backends
- Datafiles auto-download
- Auto-library download and discovery
- Unit tests, Docker, CI builds
- /CLIUtils/cmake
- /GooFit/Minuit2

New design features

- C++11, code cleanup
- Colorful logging
- /CLIUtils/CLI11
- Optimization warnings
- MPI support
- Optimizations for newer NVIDIA cards

Three body time-dependent amplitude analyses

- Mixing in $D^0 \to \pi^+\pi^-\pi^0$ time-dependent amplitude analysis (BaBar)
  [Phys.Rev. D93 (2016) no.11, 112014]
- Mixing and CP violation search in $D^0 \to K^0_S\pi^-\pi^-$
  [CERN-THESIS-2015-348] (paper in preparation)

Four body time-integrated and time-dependent amplitude analyses

- Mixing parameters in $D^0 \to K^+\pi^-\pi^+\pi^-$
  [CHEP 2016]

Toy Monte Carlo generation using /MultithreadCorner/MCBooster

- MIPWA in GooFit, such as $D^+ \to h^+h^+h^+$
  [CHEP 2016]

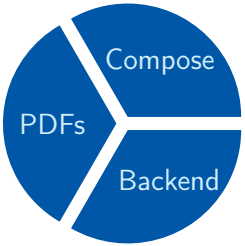
```
docker run -it alpine
apk add --no-cache make cmake g++ git
git clone --branch=stable https://github.com/GooFit/GooFit.git
cd GooFit
make
```

Simple installation

- More systems available on 
- Or use Docker images: `goofit/goofit-omp` and `goofit/goofit-cuda`

Python Install 2.1

```
pip install scikit-build cmake
pip install -v goofit
```



Python bindings

- Interface to composition
- Working prototype in GooFit 2.0
- All PDFs added for 2.1
- Pythonization of objects ongoing
- Converting/adding examples

PDF rework

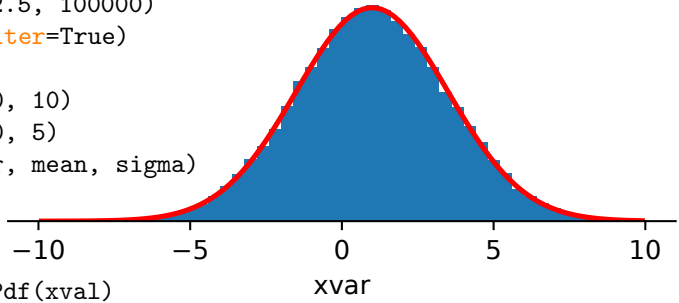
- Work by Bradley Hittle at Ohio Supercomputer Center
- Simpler PDF authoring
- Easier to alter backend

Future work

- Add Hydra (optional at first)

```
from goofit import *
import numpy as np

xvar = Variable("xvar", -10, 10)
xdata = UnbinnedDataSet(xvar)
npdata = np.random.normal(1, 2.5, 100000)
xdata.from_numpy([npdata], filter=True)

mean = Variable("mean", 0, -10, 10)
sigma = Variable("sigma", 1, 0, 5)
gauss = GaussPdf("gauss", xvar, mean, sigma)

exppdf.fitTo(data)

grid, values = gauss.evaluatePdf(xval)
```

Data for red line PDF plot



/MultithreadCorner/Hydra GPLv3

- Header only templated C++11 library
- For parallel HEP data analysis on GPUs and CPUs
- Uses variadic version of Thrust and CUDA 8
- Supports all Thrust backends: CUDA, OpenMP, TBB, CPP 2.0 (runtime selection)
- Developed by A. Augusto Alves Jr., replaces /MultithreadCorner/MCBooster

Speed up: 15x to 250x depending on algorithm, problem size, and device

Features

- Phase-space Monte Carlo generation
- Multidimensional PDF sampling
- Function evaluation over multiple dimensions
- Interface to Minuit2 minimization
- Numerical integration 2.0 (advanced)

Design

- Designed using static polymorphism
- Clean and concise
- No explicit backend coding needed
- Interfaces hard to use incorrectly
- Single source for multiple backends
- Structure of arrays (SOA) helper 2.0

User formulas as functors

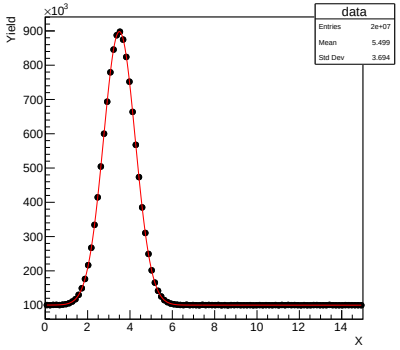
- Functors are created by the user
- C++11 lambda functions wrapped
- Supports caching
- Arithmetic and composition overloaded
- No limit to number of functors
- Named parameters 2.0

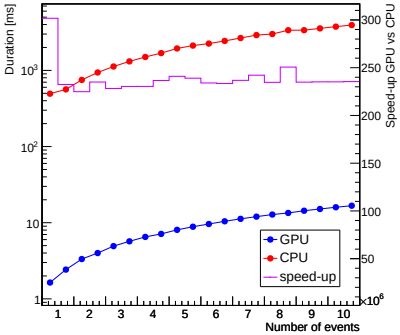
Data

- Organized in memory to support coalesced access and vectorization

Integrators

- Flat Monte Carlo sampling
- Vegas-like self-adaptive importance
- Gauss-Kronrod quadrature 2.0
- Genz-Malik quadrature 2.0





20M maximum likelihood unbinned fit

- Tesla K40: 4.865 seconds
- Xeon 2.5 Ghz 1 thread: 299.9 seconds
- 63 times faster

3-body phase space

- Tesla K40
- Xeon 2.5 Ghz 1 thread
- Well over 200 times faster

```
// Creating a parameter: named arguments
std::string Mean("Mean");
auto mean = Parameter::Create().Name(Mean).Value(3).Limits(1, 4);

// Registering parameters with Hydra
UserParameters upar;
upar.AddParameter(&mean);

// Making a PDF and FCN
Gauss gaussian(mean, sigma, 0, kFalse);
auto modelFCN = make_loglikehood_fcn(gaussian, data_d.begin(), data_d.end());

// Minuit2 minimization
MnMinimize minimize(modelFCN, upar.GetState(), strategy);
FunctionMinimum fmin(minimize(iterations, tolerance/1000));
```

Manet

Manchester **E**nergy **T**est

$D^0 \to \pi^-\pi^+\pi^0$ | $D^0 \to \pi^+\pi^-\pi^+\pi^-$

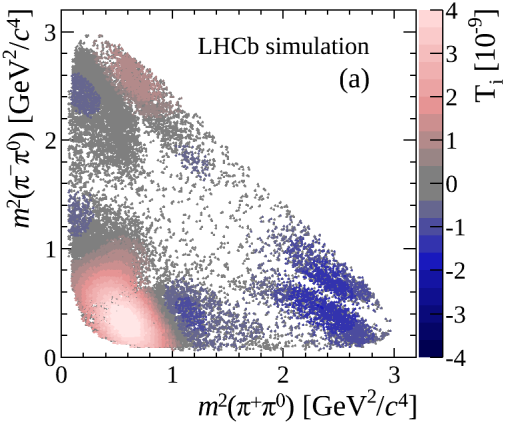
Energy Test

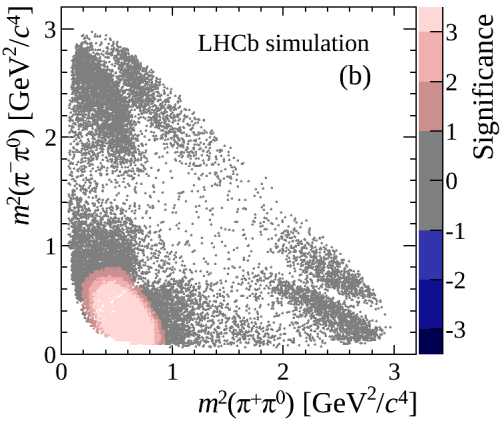
- An unbinned model-independent statistical method
- Searches for time-integrated CP violation in multi-body decays
- Made possible in reasonable computation time using GPUs
- Two analyses published using Manet

$$T \approx \overbrace{\frac{1}{n(n-1)} \sum_{i,j>i}^{n} \psi_{ij}}^{\text{Matter decay}} + \underbrace{\frac{1}{\bar{n}(\bar{n}-1)} \sum_{i,j>i}^{\bar{n}} \psi_{ij}}_{\text{Antimatter decay}} - \overbrace{\frac{1}{n\bar{n}} \sum_{i,j}^{n,\bar{n}} \psi_{ij}}^{\text{Between events}}$$

Test Statistic

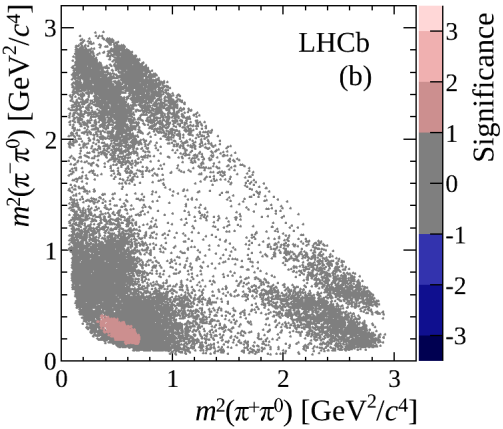
- $\psi_{ij} \equiv e^{-d_{ij}^2/2\sigma^2}$ is Gaussian with tunable width
- d_{ij} is distance between two events in 3-body phase space
- Sum of weighted distances among events
- ψ goes down as distance increases, so T is large for CP asymmetry





Simulation: $D^0 \to \pi^-\pi^+\pi^0$ [Phys. Lett. B 740 (2015) 158]

- 2% CP violation in amplitude, T (left) and significance (right)



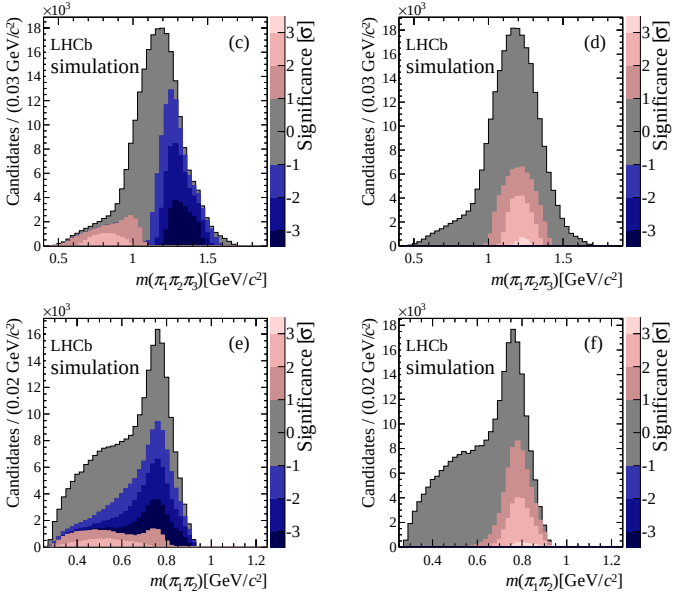
[Phys. Lett. B 740 (2015) 158]

Results

- CP symmetry: $p = (2.6 \pm 0.5)\%$
- Best sensitivity in single experiment

Manet [J.Phys. G44 (2017) no.8, 085001]

- Tesla K40: 30 minutes for 1 M events
- `manet.hepforge.org`

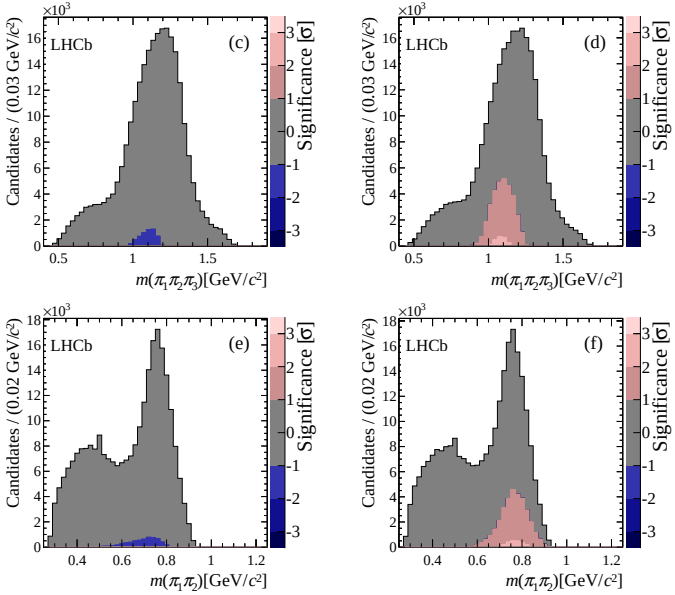


Simulation

- 3° phase *CP* violation (both)
- *P*-even in *S*-wave $a_1(1260)^+$ (left)
- *P*-odd in *P*-wave $\rho^0(770)\rho^0(770)$ (right)

[Phys.Lett. B769 (2017) 345-356]

See CP violation and mixing in charm at LHCb by Riccardo Cenci: Quark and Lepton Flavor 14:30



Final results

- $3.0\,\mathrm{fb}^{-1}$ Run 1
- p-value: $(4.6 \pm 0.5)\%$ P-even
- p-value: $(0.6 \pm 0.2)\%$ P-odd
- CP non-conservation: 2.7σ
- First test for P-odd

[Phys.Lett. B769 (2017) 345-356]

See CP violation and mixing in charm at LHCb by Riccardo Cenci: Quark and Lepton Flavor 14:30

Summary

Manet

Manchester **E**nergy **T**est

GooFit

- Now easier to use
- Many examples & PDFs
- Active development
- Python bindings soon

Hydra

- New lower-level library
- Templated header only
- Multiple backends
- Versatile toolkit

Manet

- Energy test method
- High sensitivity for *CP*
- Used in 3- and 4-body
- Possible using GPUs

Questions?

lPanema-β

- A Python CUDA package for fits
- A collection of examples and helpers
- `https://arxiv.org/abs/1706.01420`

General notes

- You can pick cards with the prefix: `CUDA_VISIBLE_DEVICES=0,1`

$\pi\pi\pi^0$

- `time ./pipipi0DPFit canonical dataFiles/cocktail_pp_0.txt --blindSeed=0`
- `time mpiexec -np 2 ./pipipi0DPFit canonical dataFiles/cocktail_pp_0.txt --blindSeed=0`

ZachFit

- `time ./zachFit 0 1`
- `time mpiexec -np 2 ./zachFit 0 1`

Build features

- Travis CI build
- Coverage, docs
- Unit tests
- Docker support

CMake features

- IDE support (Xcode, etc.)
- Library configuration
- Multiple compiler support
- Debug/tidy/format...
- Datafiles from releases

Git submodules

- Libraries are submodules
- Automatic checkout by CMake build
- Separate CMake folder (/`CLIUtils/cmake`)

C++11

- Limited to CUDA 7.0+
- Simpler code
- Used Clang-Tidy to convert (CMake 3.6+ integration)

Standalone: /GooFit/Minuit2

- Newly forked from ROOT 6.08
- CMake build, no other changes
- Already being used outside GooFit

Cleanup

- Readability: Clang-Format
- Moved all code to namespace
- Compile-time logging choice /fmtlib/fmt
- Smart color output /agauniyal/rang
- Removed custom classes and iterators (complex, etc)

/CLIUtils/CLI11

- No dependencies
- Compiles to single header file
- Nested subcommands
- Configuration files
- 100% test coverage
- CI tests on macOS/Linux/Windows
- + GooFit's features

```
./MyAnalysis generate_toy
    --params=file.ini
    --release_K892_mass
    --A12=0.3
    --plot
```

GooFit::Application

- Auto logging
- Optimization warnings
- GPU switches
- MPI support
- Completely optional

Expanded physics tools

- Three body time-dependent amplitude analyses
- Four body time-integrated and time-dependent amplitude analyses
- Toy Monte Carlo generation using MCBooster

Caching: /bryancatanzaro/generics

- Support for LDG caching
- LDG generalized form
- Performance boost for mid-age cards

MPI

- Available for `Application`
- Supports multiple GPUs

/MultithreadCorner/MCBooster is deprecated in favor of /MultithreadCorner/Hydra