

# A Python upgrade to the GooFit package for parallel fitting

---

Henry Schreiner<sup>1</sup>   Himadri Pandey<sup>1</sup>   Michael Sokoloff<sup>1</sup>  
Hittle Bradley<sup>2</sup>   Karen Tomko<sup>2</sup>   Christoph Hasse<sup>3</sup>

July 9, 2018

<sup>1</sup>University of Cincinnati

<sup>2</sup>Ohio Supercomputer Center

<sup>3</sup>CERN / Technische Universität Dortmund (DE)

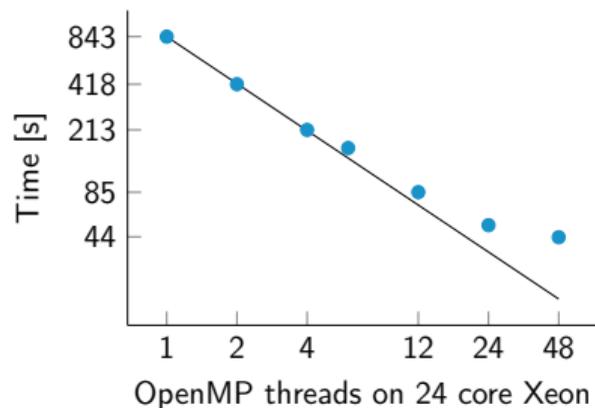


GPU and OpenMP embarrassingly parallel function evaluation engine.

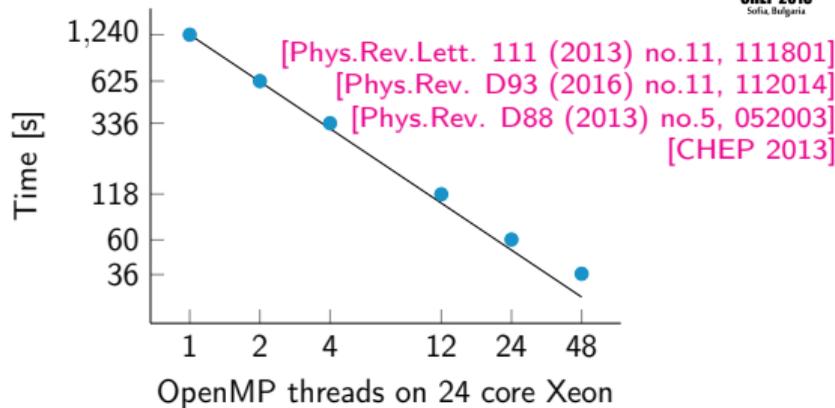
- Designed to look like RooFit, but up to 1000x faster.
- Great for fitting, toy samples, and more.

Python • Indexing • AmpGen  
(simpler functions) (amplitude grammar)

# GooFit Performance [from ACAT 2017]



# GooFit Introduction



## $\pi\pi\pi^0$ , 16 time-dependent amplitudes

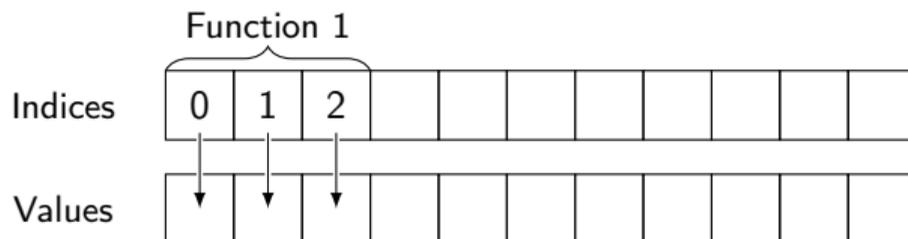
- Original RooFit code: 19,489 s single core

2 Cores	Core 2 Duo	1,159 s
GPU	GeForce GTX 1050 Ti	86.4 s
GPU	Tesla K40	64.0 s
MPI	Tesla K40 ×2	39.3 s
GPU	Tesla P100	20.3 s

## ZachFit: $M(D^{*+}) - M(D^0)$

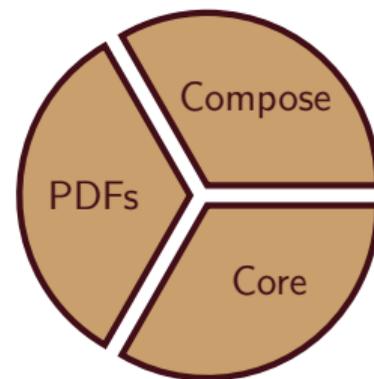
- 142,576 events in unbinned fit

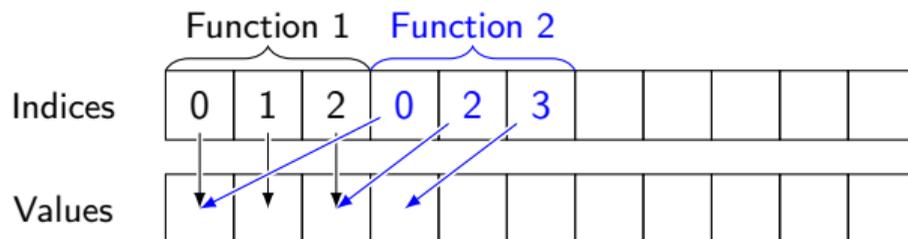
2 Cores	Core 2 Duo	738 s
GPU	GeForce GTX 1050 Ti	60.3 s
GPU	Tesla K40	96.6 s
MPI	Tesla K40 ×2	54.3 s
GPU	Tesla P100	23.5 s



## How GooFit Works

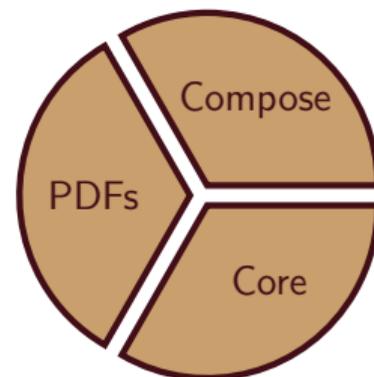
- CPU classes: Variable, Observable, DataSet, GooPdf
- Functions in CUDA with pointers held by GooPdf
- Function and variable arrays populated by GooFit
- Evaluation runs through CUDA functions through pointers (one kernel)
- Launching is handled by Thrust

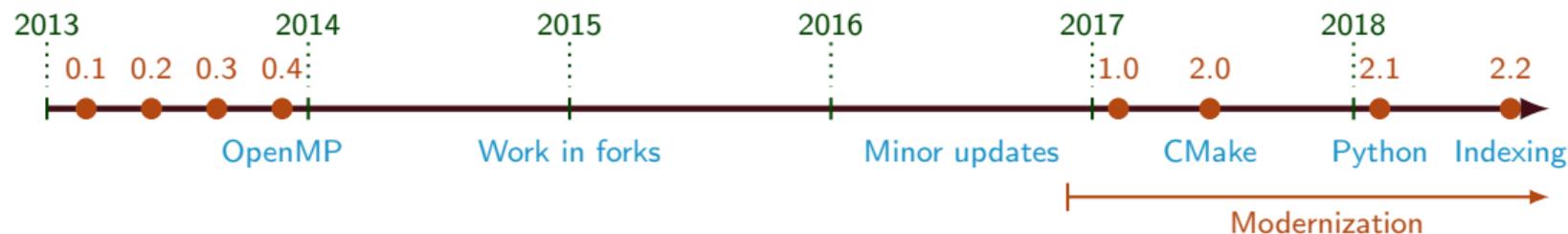




## How GooFit Works

- CPU classes: Variable, Observable, DataSet, GooPdf
- Functions in CUDA with pointers held by GooPdf
- Function and variable arrays populated by GooFit
- Evaluation runs through CUDA functions through pointers (one kernel)
- Launching is handled by Thrust





### Recent History

- 2.0: New build system, C++11, and 4-body time dependent analyses support
- 2.1: Python bindings using Pybind11
- 2.2: New indexing (and lots of Python improvements)



## Pip install

- Pip always builds from source
- Uses CUDA if found, otherwise OpenMP
- It is possible to pass in CMake arguments

## Pip 9

- `pip install skbuild cmake`
- `pip install -v goofit`

## Pip 10

- `pip install -v goofit`
- Note: not a formal endorsement of Pip 10

## CMake and normal directory

- Use `GOOFIT_PYTHON=ON` (Auto)
- Build directory should be in path

## Also available in repository

- 12 of 13 C++ examples converted
- Interactive notebook examples
- Can be built ROOTless



```
#include <goofit/...>
using namespace GooFit;

Observable x{"x", 0, 10};
Variable mu{"mu", 1};
Variable sigma{"sigma", 1, 0, 10};
GaussianPdf gauss{"gauss", &x, &mu, &sigma};
UnbinnedDataSet ds{x};

std::mt19937 gen;
std::normal_distribution<double> d{1, 2.5};
for(size_t i=0; i<100000; i++)
    ds.addEvent(d(gen));

gauss.fitTo(&ds);

std::cout << mu << std::endl;
```

```
from goofit import *
import numpy as np

x = Observable("x", 0, 10)
mu = Variable("mu", 1)
sigma = Variable("sigma", 1, 0, 10)
gauss = GaussianPdf("gauss", x, mu, sigma)
ds = UnbinnedDataSet(x)

data = np.random.normal(1, 2.5, (100000,1))
ds.from_matrix(data, filter=True)

gauss.fitTo(ds)

print(mu)
```

```
mu.value = 2  
print(mu)
```

## Variables

- Variables provide property access
- Variables can be printed
- Getters and Setters supported too

## Memory

- GooFit object memory handled transparently



```
ds.from_matrix(numpydata, filter=True)
```

## DataSet: from/to Python

- DataSets can be read in/out to 2D buffers
- Option to filter invalid values

```
ds = BinnedDataSet(x, y, z)  
ds.addEvent(xval, yval, zval)
```

## Arguments

- Automatic conversion for lists
- Variable length arguments supported



```
grid, pts = gauss.evaluatePdf(x)
gauss.setData(grid)
```

## PDF evaluation

- Evaluate on a grid
- Can be rerun interactively

```
gauss.fillMCDataSimple(1000000)
```

## 1D MC generation

- Simple way to produce MC
- Initial MC on CPU, evaluation on GPU

```
dplt = DalitzPlotter(prod, dp)
arr = dplt.make2D()
```

## Amp3Body (TD)

- Can produce simple 3-body Toy MC
- DalitzPlotter functionality planned for merge into Amp3Body

```
aa.setGenerationOffset(0);
aa.GenerateSig(1000000);
```

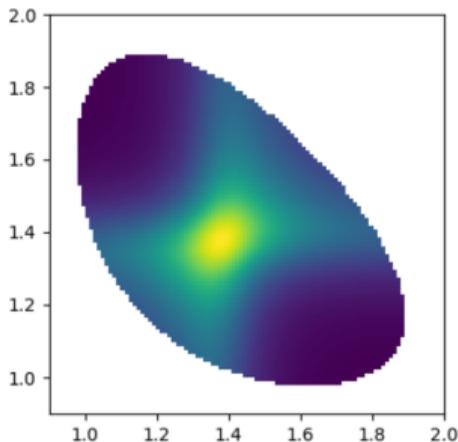
## Amp4Body (TD)

- Full GPU Toy MC using **MCBooster**



```
In [13]: fig, ax = plt.subplots(figsize=(6, 4))
v = ax.imshow(arr, extent=ext, origin='lower')
```

Figure 1



```
In [16]: @interact
def f(a=r_pi, b=r_pi, m1=r_mass, w1=r_width, m2=r_mass, w2=r_width):
    ar1.value, ai1.value = np.cos(a), np.sin(a)
    ar2.value, ai2.value = np.cos(b), np.sin(b)
    mass1.value, width1.value = m1, w1
    mass2.value, width2.value = m2, w2
    dplt = gf.DalitzPlotter(prod, dp)
    arr = dplt.make2D()
    arr = np.ma.array(arr, mask=arr==0)
    ax.clear()
    v = ax.imshow(arr, extent=ext, origin='lower')
```

a  1.57

b  1.57

m1  1.18

w1  0.11

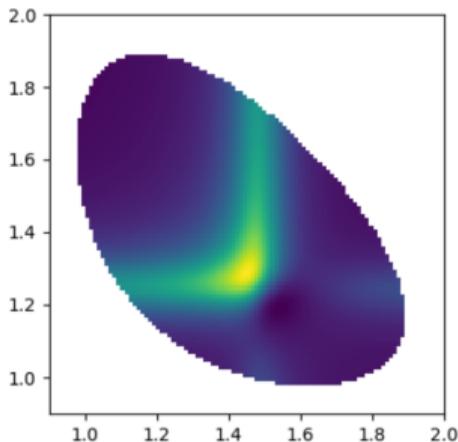
m2  1.18

w2  0.11



```
In [13]: fig, ax = plt.subplots(figsize=(6, 4))
v = ax.imshow(arr, extent=ext, origin='lower')
```

Figure 1



```
In [16]: @interact
def f(a=r_pi, b=r_pi, m1=r_mass, w1=r_width, m2=r_mass, w2=r_width):
    ar1.value, ai1.value = np.cos(a), np.sin(a)
    ar2.value, ai2.value = np.cos(b), np.sin(b)
    mass1.value, width1.value = m1, w1
    mass2.value, width2.value = m2, w2
    dplt = gf.DalitzPlotter(prod, dp)
    arr = dplt.make2D()
    arr = np.ma.array(arr, mask=arr==0)
    ax.clear()
    v = ax.imshow(arr, extent=ext, origin='lower')
```

a

b

m1

w1

m2

w2



## Documentation

- Documentation exported to Jupyter

## Implementation details

- Generated by CMake from Doxygen style comments
  - ▶ Conversion to Jupyter style markdown for math
- Attached to class in PyBind11

```
In [2]: import goofit
```

```
In [3]: goofit.ExpGausPdf
```

Out[3]: An exponential decay convolved with a Gaussian resolution:

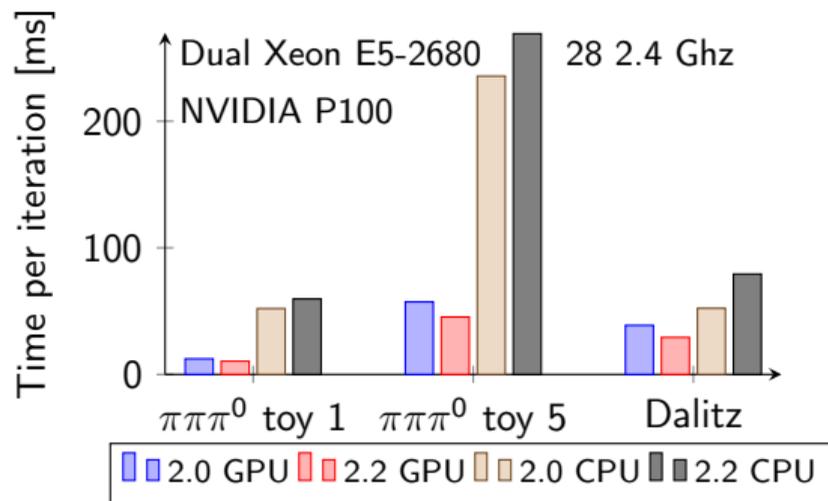
$$P(t; m, \sigma, \tau) = e^{-t/\tau} \otimes e^{-\frac{(t-m)^2}{2\sigma^2}} \\ = (\tau/2) e^{(\tau/2)(2m+\tau\sigma^2-2t)} \operatorname{erfc}\left(\frac{m + \tau\sigma^2 - t}{\sigma\sqrt{2}}\right)$$

where  $\operatorname{erfc}$  is the complementary error function. The constructor takes the observed time  $t$ , mean  $m$  and width  $\sigma$  of the resolution, and lifetime  $\tau$ . Note that the original decay function is zero for  $t < 0$ .

Observable ID	1	0	1	1	
Parameters	2	3.0	2.0	1	3.0
Constants	0	1	3.14		
Normalization factors	1	1.0	1	1.0	

## ParameterContainer

- Nice API for PDFs
- Handles initialization/updates
- Can work past unknown components



## Performance: 2.0 to 2.2

- Faster on CUDA
- A bit slower on OpenMP
- Further optimizations possible

```
__device__ fptype f_device(fptype* evt, ParameterContainer& pc) {  
    int id = pc.getObservable(0);  
    fptype x = evt[id];  
    fptype v = pc.getParameter(0);  
    pc.incrementIndex(1, 1, 0, 1, 1);  
    return x * v;  
}  
  
__device__ device_function_ptr ptr_to_Gaussian = device_Gaussian;
```

## Device functions

- Simpler, easier than before

```
__host__ MyPdf::MyPdf(std::string name, Observable x, Variable v,)  
    : GooPdf("MyPdf", name, x, v) {  
    registerFunction("ptr_to_f", ptr_to_f);  
    initialize();  
}
```

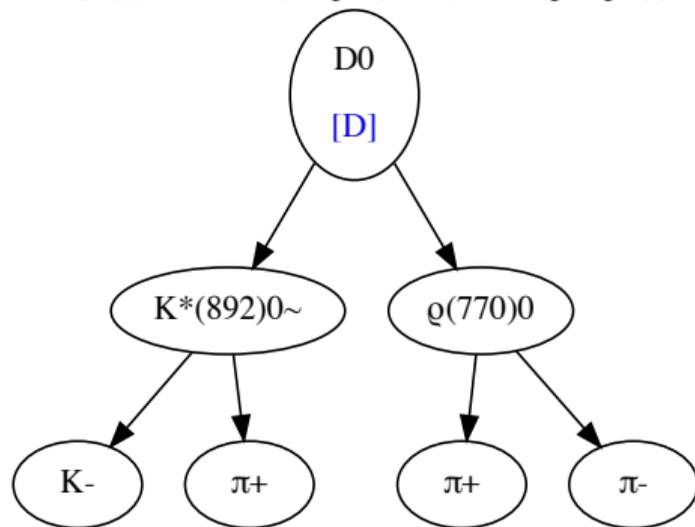
## Registration

- Simply register the function (with debug name)
- Observables, Variables can be registered in the constructor

## AmpGen by Tim Evans

- Successor to MINT3: JIT compiler for amplitudes
- Includes easy to use and read grammar
- Inside LHCb framework, but can build standalone
- Includes pure Python ctypes interface

$D0[D]\{K(892)^0\{K^-, \pi^+\}, \rho(770)^0\{\pi^+, \pi^-\}\}$



## DecayLanguage: BETA

- Python package implementing AmpGen's syntax
- Powerful PDG particle class included
- Reads in and understands decay chains
- Still in beta: future potential
- Expands lines, produces pictures
- Can output GooFit code!
- On [PyPI](#) and [ReadTheDocs](#)

$D^0 \rightarrow K^\mp \pi^\pm \pi^\pm \pi^\mp$  model  
[[Eur.Phys.J. C78 \(2018\)](#)]

- AmpGen model: 222 lines
- GooFit model: 1314 lines



## GooFit Python

- Easy to compose model
- Easy to manipulate model
- Pythonic interface

## GooFit Indexing

- Simpler to add PDFs
- Faster on GPU
- More flexibility for developers

## AmpGen/DecayLanguage

- Syntax for amplitudes
- Beta Python package
- Future potential

Need help? Use GitHub issues, Gitter, or [hschrein@cern.ch](mailto:hschrein@cern.ch)



GooFit 2.2 release delayed by critical bug in 4-body Amplitude calculation.

- CLI11 1.6 (2.2)
- Better MPI testing (2.2)
- Many new tests, in Catch2 (2.2)
- Dropped max limits for indexes (2.2)
- Large internal rename and new inheritance tree (2.2)
- 3-body fit fractions (2.1.3)
- Pip 10 support (2.1.3)
- Live notebook examples and support (2.1.2)
- DalitzPlotter (2.1.1)