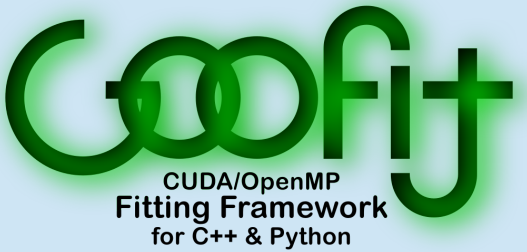
Recent Developments in



Henry Schreiner

February 12, 2018

UNIVERSITY OF Cincinnati

If you would like to follow along on your computer with Himadri's talk, execute the following lines now:

```
pip install scikit-build cmake
PIP_INSTALL_OPTIONS="-- -DGOOFIT_DEVICE=OMP" pip install -v goofit
```

Requirements:

- Linux or macOS
- A recent version of `pip`
- A compiler (gcc 4.8+ or Clang)
- `make` or `ninja` and `git`
- `numpy` and `matplotlib` needed for examples

What is GooFit?

- Package for high-speed fitting
- Resembles RooFit
- High performance on GPU or CPU
- 100-1000x faster than single core

History of GooFit

- Released publicly 2013
- GooFit 2.0 released mid 2017
- GooFit 2.1 released end of 2017

GooFit 2.0

- Easy to build (CMake)
- Dependency simplification (Minuit, etc.)
- Continuous integration online
- Lots of work merged back in
- Supports more platforms (macOS, TBB)
- Support for common tools (XCode, etc.)
- Memory caching improvements
- MPI multi GPU and node
- Docker containers too!

[Modernizing GooFit @ PEARC17]
[GooFit 2.0 @ ACAT 2017]

- Compose existing PDFs
- Details in Himadri's talk!

What it means for you

- Easy to read/write
- Easy to script
- Avoid recompiling

```
from goofit import *
import numpy as np

# Make a dataset
xvar = Variable("xvar", -10, 10)
data = UnbinnedDataSet(xvar)
npdata = np.random.normal(1, 2.5, 100000)
data.from_matrix([npdata], filter=True)

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

gauss.fitTo(data)
```

- Internal reference counting for Variables
- Observables now part of API
- EventNumber also

What it means for you

- Easier model setup
- Fewer segfaults
- Simpler Python bindings

Before:

```
Variable* x = new Variable("x", 0, 3);
```

After:

```
Observable x{"x", 0, 3};
```

Python:

```
x = Observable("x", 0, 3)
```

- Now separate classes
  - 3-body Resonances
  - 4-body LineShapes
- New Resonances/LineShapes

What it means for you

- *Readable* model setup
- Easier to code
- Signature for each constructor
- Future optimizations possible

Before:

```
ResonancePdf("r1", ar, ai, m, w, sp, PAIR_12);
ResonancePdf("r2", ar, ai, m, sp, w, PAIR_12);
ResonancePdf("r3", ar, ai);
```

After:

```
Resonances::RBW("r1", ar, ai, m, w, sp, PAIR_12);
Resonances::LASS("r2", ar, ai, m, w, sp, PAIR_12);
Resonances::NonRes("r3", ar, ai);
```

- Minuit2 will be built if missing
- GooFit coding style enforced
- CCache native support on CMake 3.6+
- CUDA native support on CMake 3.9+
- OpenMP on Apple LLVM compiler!

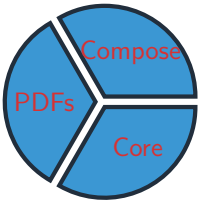
What it means for you

- Less time spent on setup
- More time spent on Physics
- Easier to contribute

- If you have `make`, `CMake`, `g++`, and `git`:

```
git clone https://github.com/GooFit/GooFit.git
cd GooFit
make
```

- Explicit instructions for many platforms available



- Redesigned PDF indexing
- Details saved for Brad's talk

What it means for you

- Easier to author PDFs
- We can optimize the evaluation later

```
__device__ fptype(
device_Exp(fptype *evt,
           ParameterContainer &pc) {

    int id       = pc.getObservable(0);
    fptype alpha = pc.getParameter(0);
    fptype x     = evt[id];

    fptype ret = exp(alpha * x);
    pc.incrementIndex(1, 1, 0, 1, 1);

    return ret;
}
```

Documentation

- See the README
- API documentation being worked on
- Help us with the GooFit 2torial

Analyses

- Usage improvements still being added
- Several ongoing analyses in GooFit
- If you extend the core, please contribute!

HYDRA

Multithreaded Data Analysis Framework

Hydra

- Templated C++11 framework
- Made for highly parallel scientific calculations
- CUDA, OpenMP, TBB, or single threaded
- Eventual integration/interaction planned
- See /MultithreadCorner/Hydra

Summary

- GooFit is a fast tool for fitting
- GooFit 2.0 is easier to build and use
- GooFit 2.1 adds Python and more
- GooFit 2.2 (coming soon) makes PDFs better
- We are available to help you with GooFit

If you have been building GooFit, it should be ready for Himadri's talk! Follow along with her.

Questions?

Thanks to the work and support of:

- A. Augusto Alves Jr.
- Rolf Andreassen
- Christoph Hasse
- Bradley Hittle
- Zachary Huard
- Brian Maddock
- Himadri Pandey
- Henry Schreiner
- Michael Sokoloff
- Liang Sun
- Karen Tomko

Continued development on GooFit is supported by the U.S. National Science Foundation under grant number 1414736 and was developed under grant number 1005530. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the developers and do not necessarily reflect the views of the National Science Foundation. In addition, we thank the nVidia GPU Grant Program for donating hardware used in developing this framework.

Code for Himadri's example (addition.py)

```
# Imports
from goofit import *
import numpy as np
import matplotlib.pyplot as plt

# Print GooFit and computer info
print_goofit_info()

# GooFit DataSet
xvar = Observable("xvar", -5, 5)
data = UnbinnedDataSet(xvar)

# Make a dataset from 90% gauss + 10% flat
dat = np.random.normal(0.2,1.1,100000)
dat[:10000] = np.random.uniform(-5,5,10000)
data.from_matrix([dat], filter=True)
```

Code for Himadri's example (addition.py)

```
# GooFit PDFs and fitting variables
xmean = Variable("xmean", 0, 1, -10, 10)
xsigm = Variable("xsigm", 1, 0.5, 1.5)
signal = GaussianPdf("signal", xvar, xmean, xsigm)

constant = Variable("constant", 1.0)
backgr = PolynomialPdf("backgr", xvar, [constant])

sigfrac = Variable("sigFrac", 0.9, 0.75, 1.00)
total = AddPdf("total", [sigfrac], [signal, backgr])

# Do the fit
total.fitTo(data)
```

Code for Himadri's example (addition.py)

```
# Plot data
plt.hist(dat, bins='auto', label='data', normed=True)

# Make grid and evaluate on it
grid = total.makeGrid()
total.setData(grid)
main, gauss, flat = total.getCompProbsAtDataPoints()
xvals = grid.to_matrix().flatten()

# Plotting components
plt.plot(xvals, main, label='total')
plt.plot(xvals, np.array(gauss)*sigfrac.value, label='signal')
plt.plot(xvals, np.array(flat)*(1-sigfrac.value), label='background')

# Show the plot
plt.legend()
plt.show()
```

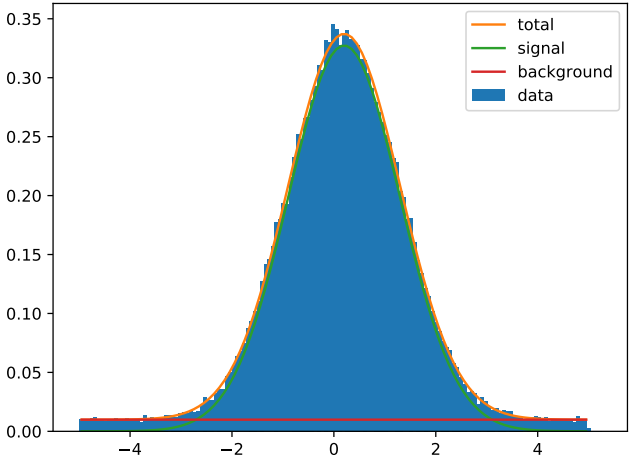


Figure 1: Simulated data and fit