A Python upgrade to the GooFit package for parallel fitting

Henry Schreiner\textsuperscript{1} Himadri Pandey\textsuperscript{1} Michael Sokoloff\textsuperscript{1}

Hittle Bradley\textsuperscript{2} Karen Tomko\textsuperscript{2} Christoph Hasse\textsuperscript{3}

July 9, 2018

\textsuperscript{1}University of Cincinnati

\textsuperscript{2}Ohio Supercomputer Center

\textsuperscript{3}CERN / Technische Universität Dortmund (DE)
What is GooFit?

GooFit Introduction

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]

\[ \pi \pi \pi^0, \text{16 time-dependent amplitudes} \]

- Original RooFit code: 19,489 s single core

<table>
<thead>
<tr>
<th>2 Cores</th>
<th>Core 2 Duo</th>
<th>1,159 s</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>GeForce GTX 1050 Ti</td>
<td>86.4 s</td>
</tr>
<tr>
<td>GPU</td>
<td>Tesla K40</td>
<td>64.0 s</td>
</tr>
<tr>
<td>MPI</td>
<td>Tesla K40 ×2</td>
<td>39.3 s</td>
</tr>
<tr>
<td>GPU</td>
<td>Tesla P100</td>
<td>20.3 s</td>
</tr>
</tbody>
</table>

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

- 142,576 events in unbinned fit

<table>
<thead>
<tr>
<th>2 Cores</th>
<th>Core 2 Duo</th>
<th>738 s</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>GeForce GTX 1050 Ti</td>
<td>60.3 s</td>
</tr>
<tr>
<td>GPU</td>
<td>Tesla K40</td>
<td>96.6 s</td>
</tr>
<tr>
<td>MPI</td>
<td>Tesla K40 ×2</td>
<td>54.3 s</td>
</tr>
<tr>
<td>GPU</td>
<td>Tesla P100</td>
<td>23.5 s</td>
</tr>
</tbody>
</table>

A Python upgrade to the GooFit package for parallel fitting

Henry Schreiner
July 9, 2018
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

GooFit Introduction
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)
Installation (Python)

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

A Python upgrade to the GooFit package for parallel fitting
Henry Schreiner
July 9, 2018
5/16
Comparison

GooFit and Python

```cpp
#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;
```

```python
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)
```
Pythonisms

```python
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

GooFit and Python

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

DataSet: from/to Python

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

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

Arguments

- Automatic conversion for lists
- Variable length arguments supported
 Simulation and Evaluation

```
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

---

GooFit and Python

```
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
Evaluation Example

A Python upgrade to the GooFit package for parallel fitting

Henry Schreiner
July 9, 2018
A Python upgrade to the GooFit package for parallel fitting

Henry Schreiner

July 9, 2018
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

```python
In [2]: import goofit

In [3]: goofit.ExpGausPdf

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

\[
P(t; m, \sigma, \tau) = e^{-\frac{t}{\tau}} \otimes e^{-\frac{(t-m)^2}{2\sigma^2}}
\]

\[
= (\frac{\tau}{2}) e^{(\frac{\tau^2}{2})(2m + \tau^2 - 2t)} \text{erfc}\left(\frac{m + \tau^2 - t}{\sigma\sqrt{2}}\right)
\]

where \text{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 \( t < 0 \).
### New Indexing System

<table>
<thead>
<tr>
<th>Observable ID</th>
<th>(1)</th>
<th>(0)</th>
<th>(1)</th>
<th>(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>(2)</td>
<td>3.0</td>
<td>2.0</td>
<td>(1)</td>
</tr>
<tr>
<td>Constants</td>
<td>(0)</td>
<td>(1)</td>
<td>3.14</td>
<td></td>
</tr>
<tr>
<td>Normalization factors</td>
<td>(1)</td>
<td>1.0</td>
<td>(1)</td>
<td>1.0</td>
</tr>
</tbody>
</table>

#### 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

---

**Indexing**

![Performance Chart]

<table>
<thead>
<tr>
<th>Time per iteration [ms]</th>
<th>Dual Xeon E5-2680</th>
<th>NVIDIA P100</th>
<th>Dalitz</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\pi\pi\pi^0) toy 1</td>
<td>(\pi\pi\pi^0) toy 5</td>
<td>(\pi\pi\pi^0) toy 5</td>
<td>Dalitz</td>
</tr>
<tr>
<td>GPU</td>
<td>2.0</td>
<td>2.2</td>
<td>2.2</td>
</tr>
<tr>
<td>CPU</td>
<td>2.0</td>
<td>2.2</td>
<td>2.2</td>
</tr>
</tbody>
</table>

Henry Schreiner | July 9, 2018
__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-}}

A Python upgrade to the GooFit package for parallel fitting

Henry Schreiner  
July 9, 2018
DecayLanguage

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

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

Summary

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
GooFit 2.2 release delayed by critical bug in 4-body Amplitude calculation.
New Features Since 2.1

- 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)