GooFit: A library for massively parallelising maximum-likelihood fits

[1] University of Cincinnati, 2600 Clifton Avenue, Cincinnati OH 45220, USA
[2] Ohio Supercomputer Center, 1224 Kinnear Road, Columbus OH 43212, USA

E-mail: rolfa@slac.stanford.edu

Abstract. Fitting complicated models to large datasets is a bottleneck of many analyses. We present GooFit, a library and tool for constructing arbitrarily-complex probability density functions (PDFs) to be evaluated on nVidia GPUs. The massive parallelisation of dividing up event calculations between hundreds of processors can achieve speedups of factors 200-300 in real-world problems.

1. Introduction

Parameter estimation is a crucial part of many physics analyses.

2. Real-world example: Time-dependent Dalitz-plot fit

3. User-level code

The purpose of GooFit is to give users access to the parallelising power of CUDA without requiring them to write CUDA code. At the most basic level, GooFit objects representing PDFs, `GooPdf`s, can be created and combined in plain C++. Only if a user needs to represent a function outside the existing GooFit classes does he need to do any CUDA coding; Section 4 shows how to create new PDF classes. We intend, however, that this should be a rarity, and that the existing PDF classes should cover all the most common cases.

A GooFit program has four main components:

- The PDF that models the physical process, represented by a `GooPdf` object.
- The fit parameters with respect to which the likelihood is maximised, represented by `Variable`s contained in the `GooPdf`.
- The data, gathered into a `DataSet` object containing one or more `Variable`s.
- A `FitManager` object which forms the interface between MINUIT (or, in principle, a different maximising algorithm) and the `GooPdf`.

Listing 1 shows a simple fit of an exponential function.

Listing 1. *Fit for unknown parameter α in $e^{\alpha x}$. GooFit classes are shown in red, important operations in blue.*

```
int main (int argc, char** argv) {
  // Independent variable.
  Variable* xvar = new Variable("xvar", 0, log(RAND_MAX));

  // Create data set
  UnbinnedDataSet data(xvar);
  for (int i = 0; i < 100000; ++i) {
    // Generate toy event
    xvar->value = xvar->upperlimit - log(1+rand());
    if (xvar->value < 0) continue;
    // ...and add to data set.
    data.addEvent();
  }

  // Create fit parameter
  Variable* alpha = new Variable("alpha", -2, 0.1, -10, 10);
  // Create GooPdf object
  ExpPdf* exppdf = new ExpPdf("exppdf", xvar, alpha);
  // Move data to GPU
  exppdf->setData(&data);

  FitManager fitter(exppdf);
  fitter.fit();

  return 0;
}
```

4. Adding new PDFs

5. Program flow

6. Examples

7. Summary

8. Acknowledgements
NSF funding, code from such-and-such, valuable suggestions and help from Cristoph Deil, feedback from Stefanie and Ollie.