This is support for black-box optimization with CMA-ES from within ROOT (http://root.cern.ch/drupal/).

**This is a work in progress, support for ROOT is under continuous improvement, see https://github.com/beniz/libcmaes/issues/13 **

libcmaes can be used from CERN's ROOT6 as a replacement or addition to Minuit2 optimizer. It is designed to be used from ROOT6 **exactly** as Minuit2 is used, so code using Minuit2 should be easily run against CMA-ES.

Note: at this early stage, not all features of Minuit2, such as subroutines Contour and Scan, are ported to CMA-ES. This is a work in progress.

Below are instructions for testing it out.

Beware: at the moment support is alpha, this is NO production code

Building ROOT6 and libcmaes

As for now, the only way to use libcmaes is from ROOT6, using the following special repository, and compiling it from sources (1): https://github.com/beniz/root/tree/cmaes4root_master

Proceed with the following steps:

- get and install libcmaes into your home repository (or globally on your system, remove the --prefix option to the configure script below):

```
git clone https://github.com/beniz/libcmaes.git
git branch minos
./configure --prefix=/home/yourusername
make
make install
```

- get ROOT6 from https://github.com/beniz/root/tree/cmaes4root_master, configure & compile it (this will take a while) (2):

```
git clone https://github.com/beniz/root/tree/cmaes4root_master
cd root
./configure --enable-minuit2 --build=debug --with-cmaes-incdir=/home/yourusername/include/libcmaes --with-cmaes-libdir=/h
make
```

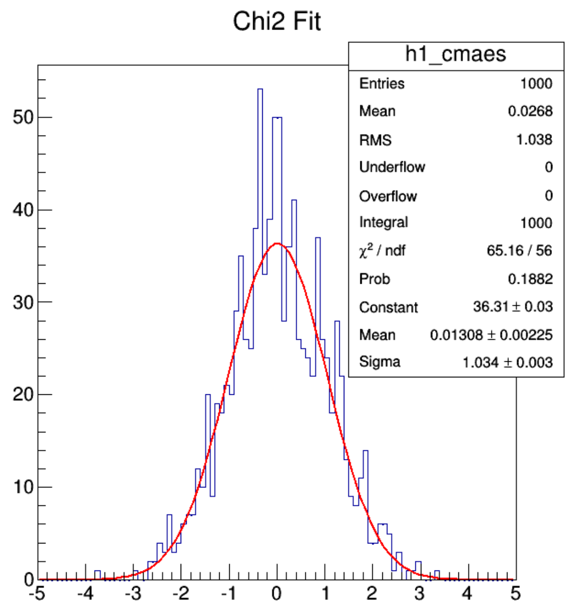
use make -jx where x is the number of cores on your system in order to minimize the building time.

Running an example with CMA-ES

To run the basic fitting of a Gaussian, originally taken from Minuit2's tutorial files, do:

```
root
.L tutorials/fit/cmaesGausFit.C++g
cmaesGausFit()
```

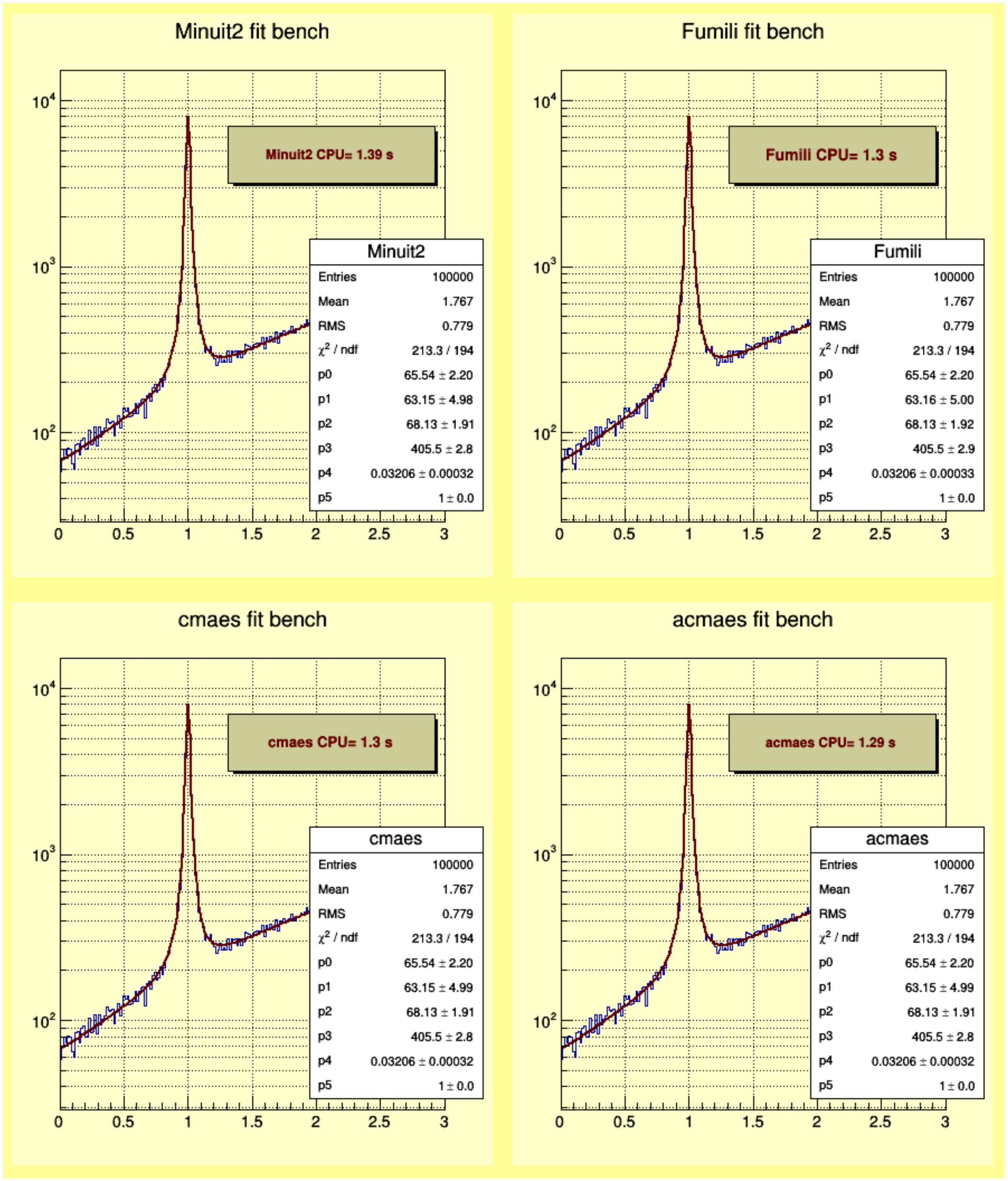
You should see a plot similar to



To quick test competitiveness against Minuit2:

```
root
.L tutorials/fit/cmaesFitBench.C
 cmaesFitBench()
```

You should witness a plot similar to



Minuit2	
Entries	100000
Mean	1.767
RMS	0.779
χ^2 / ndf	213.3 / 194
p0	65.54 ± 2.20
p1	63.15 ± 4.98
p2	68.13 ± 1.91
p3	405.5 ± 2.8
p4	0.03206 ± 0.00032
p5	1 ± 0.0

Fumili	
Entries	100000
Mean	1.767
RMS	0.779
χ^2 / ndf	213.3 / 194
p0	65.54 ± 2.20
p1	63.16 ± 5.00
p2	68.13 ± 1.92
p3	405.5 ± 2.9
p4	0.03206 ± 0.00033
p5	1 ± 0.0

cmaes	
Entries	100000
Mean	1.767
RMS	0.779
χ^2 / ndf	213.3 / 194
p0	65.54 ± 2.20
p1	63.15 ± 4.99
p2	68.13 ± 1.91
p3	405.5 ± 2.8
p4	0.03206 ± 0.00032
p5	1 ± 0.0

acmaes	
Entries	100000
Mean	1.767
RMS	0.779
χ^2 / ndf	213.3 / 194
p0	65.54 ± 2.20
p1	63.15 ± 4.99
p2	68.13 ± 1.91
p3	405.5 ± 2.8
p4	0.03206 ± 0.00032
p5	1 ± 0.0

Running a benchmark comparison of CMA-ES and Minuit2

To run the current benchmark and visualize results, take the following steps:

```
root
.L tutorials/fit/cmaesFullBench.C
run_experiments(10)
python math/cmaes/test/cmaesFullBench.py
```

This should show a series of histograms comparing results from both optimizers on a selection of problems.

Options to the CMA-ES minimizers within ROOT

There's built-in control for several hyper-parameters and options of CMA-ES:

- several flavors of the algorithm are available, and can be choosen at creation of the Minimizer object:

```
TVirtualFitter::SetDefaultFitter(``acmaes'');
```

or

```
ROOT::Fit::Fitter fitter;
fitter.Config().SetMinimizer(``cmaes'',''acmaes'');
```

The available algorithms are: `cmaes, ipop, bipop, acmaes, aipop, abipop, sepcmaes, sepipop, sepbipop`.

'acmaes' should be the most appropriate in most cases, and 'sepacmaes' when the number of dimensions nears a thousand.

The options below are not required, but can be used by filling up a MinimizerOptions object beforehand:

```
const char *fitter = "acmaes"
TVirtualFitter::SetDefaultFitter(fitter);
ROOT::Math::IOptions &opts = ROOT::Math::MinimizerOptions::Default(fitter);
opts.SetIntValue("lambda",100);
```

Options below are not activated by default:

- 'sigma': initial step-size
- 'lambda': number of offsprings at each generation
- 'noisy': flag that updates some hyper-parameters if the objective function is noisy
- 'restarts': maximum number of restarts, only applies to ipop, bipop, aipop, abipop, sepipop and sepbipop
- 'ftarget': the objective function target that stops optimization when reached, useful when the final value is known, e.g. 0
- 'fplot': output file in libcmaes format for later plotting of eigenvalues and state convergence, mostly for debug purposes
- 'lscaling': automatic linear scaling of parameters with auto-selection of step-size sigma, usually recommended if results are not satisfactory.

(1) more convenient ways will be provided. (2) we recommend building support for both Minuit2 (i.e. for comparison to CMA-ES) and debug.