Welcome to the libcmaes documentation

libcmaes is a multithreaded C++11 library for high performance blackbox stochastic optimization using the CMA-ES algorithm for Covariance Matrix Adaptation Evolution Strategy.

It is especially useful and efficient for finding minimums and maximums of possibly complex, non-separable, non-derivable functions.

The library exposes a set of simple structures to the user. Below, the function to be optimized is refered to as the objective function, whose parameters define the state-space into which the search for solution takes place. The hyper-parameters are the set of parameters that govern the CMA-ES algorithm: some have their value determined automatically, and others can be overridden by the user.

Available algorithms: CMA-ES, active CMA-ES, separable CMA-ES

Available restart strategies: none, IPOP-CMA-ES, BIPOP-CMA-ES

Algorithms and restart strategies can be all combined, thus yielding nine flavors of CMA-ES to call on. See [[Practical hints]] section on how to best choose among these possibilities.

1. Building and installing libcmaes

Supported platforms at this time:

- Linux, all flavors
- Max OSX with gcc and macports

Requirements and dependencies:

- gcc 4.7 or higher (mandatory)
- eigen for all matrix operations (mandatory);
- glog for logging events and debug (optional);
- gflags for command line parsing (optional);
- gtest for unit testing (optional).
- doxygen for generating the API documentation (optional).

```
sudo apt-get install libgoogle-glog-dev libgflags-dev libeigen3-dev
```

For compiling with basic options enabled:

```
git clone https://github.com/beniz/libcmaes.git
./autogen.sh
./configure
make
```

IMPORTANT: clang has a bug that prevents it from compiling libcmaes, see #19

To specify the location of your Eigen header repository, do:

```
./configure --with-eigen3-include=/path/to/eigen3
make
```

In order to install in a repository of your choice, for instance your home, do:

```
./configure --prefix=/home/username
```

```
make
make install
```

The installed files will be found in /home/username/include/libcmaes and /home/username/lib

All available options to configure can be listed with:

```
./configure --help
```

To check that everything is fine, if gtest is available:

```
make check
```

Otherwise, a quick check can be run with:

```
./tests/test_functions --all
```

2. Quick start

Below is a simple typical example, available from examples/sample-code.cc

```
#include "cmaes.h"
#include <iostream>

using namespace libcmaes;

FitFunc fsphere = [](const double *x, const int N)
{
  double val = 0.0;
  for (int i=0;i<N;i++)
    val += x[i]*x[i];
  return val;
};

int main(int argc, char *argv[])
{
  int dim = 10; // problem dimensions.
  //int lambda = 100; // offsprings at each generation.
  //CMAParameters cmaparams(dim,lambda);
  CMAParameters<> cmaparams(dim);
  //cmaparams._algo = BIPOP_CMAES;
  CMASolutions cmasols = cmaes<>(fsphere,cmaparams);
  std::cout << "best solution: " << cmasols << std::endl;
  std::cout << "optimization took " << cmasols._elapsed_time / 1000.0 << " seconds\n";
  return cmasols._run_status;
}
```

The code snippet above compiles with

```
g++ -fopenmp -std=gnu++11 -I/path/to/eigen3/ -I/path/to/include/libcmaes -L/path/to/lib -o sample_code sample-code.cc -lc
```

where /path/to refers to the possibly different path to eigen3 headers, libcmaes headers and lib. If you've installed in your home:

```
g++ -fopenmp -std=gnu++11 -I/path/to/eigen3/ -I/home/username/include/libcmaes -L/home/username/lib -o sample_code sample
```

3. Library structures and API

The library uses object-oriented policy-based design and makes extensive use of templates. However, no knowledge of templates is required to use the library.

The main structures are:

- the FitFunc object that captures the objective function to be optimized
- the CMAParameter object that captures hyper-parameters and inputs to the optimizer, i.e. offprings per iteration, initial step-size, initial search point in parameter space
- the ESOptimizer object from which the optimizer algorithm runs; this object is abstracted away for simple usage and thus remains optional and unseen in most applications
- the CMASolution object that captures all outputs of optimization, from running status and error code, to actual best solution in parameter space, best objective function value, ...

The library comes with an integrated documentation of the API at code level. The documentation can be generated from the main repository as follows:

```
doxygen doc/Doxyfile
```

The html documentation will then be found in doc/html/index.html

This API documentation is also available online: http://beniz.github.io/libcmaes/doc/html/index.html

4. Using libcmaes

1. [[Optimizing a function]]
2. [[Visualizing optimization results and convergence]]
3. [[Defining and using bounds on parameters]]
4. [[Using parameter space transforms known as genotype/phenotype transforms]]
5. [[Using a custom progress function]]
6. [[Using a user-defined gradient function]]
7. [[Defining a custom algorithm based on libcmaes primitives]]
8. [[Compiling and running BBOB'2013 benchmark]]
9. [[Practical hints]]
10. [[Multithreading]]

5. Applications

1. [[using CMA-ES in CERN's ROOT]]
2. [[experiments with CMA-ES and neural networks]]