The library can output internal values during optimization and store them into a file for later visual inspection by a series of graphs.

While this is mostly useful for debug purposes and deep objective function and algorithms analysis, it is show below how to generate and interpret the data, as well as how to incorporate this functionality into external code.

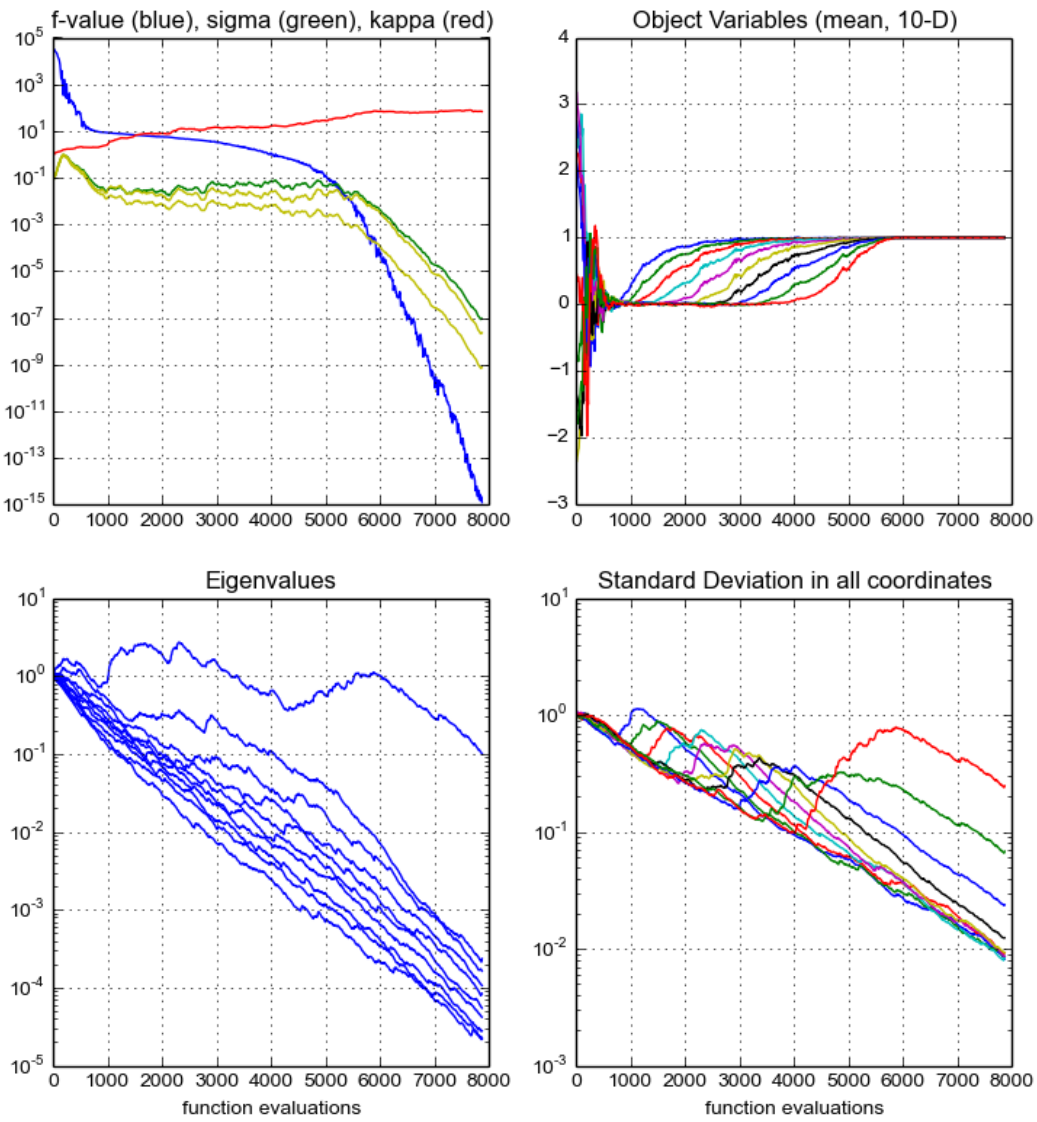
For using the built-in visualization script you will need both Python and Matplotlib on your system.

Visualization of optimization and convergence

As an example, we study the Rosenbrock function in 10-D using one of the library built-in testing tool:

```
cd tests
./test_functions --fname rosenbrock --dim 10 --fplot rosen10D.dat
python cma_multiplt.py rosen10D.dat
```

You should get a picture very similar to the one below:



Explanations:

- top left: fmin and sigma in internal parameter space;
- top right: the function parameter values, converging to their final values;
- bottom left: convergence of eigenvalues;
- bottom right: convergence of error covariance matrix.

Generating data from external code

From external code using libcmaes, it is easy to generate the data for a similar visualization:

```
int dim = 10; // problem dimensions.
double sigma = 0.1; // initial step-size, i.e. estimated initial parameter error
std::vector<double> x0(dim,10.0); // initialize x0 as 10.0 in all 10 dimensions
CMAParameters<> cmaparams(dim,&x0.front(),sigma); // &x0.front() allows to turn the vector into a double*
cmaparams._fplot = "youroutput.dat"
```

This will generate the youroutput.dat file and fill it out in the repository your program is launched from. Then call:

```
python cma_multiplt.py youroutput.dat
```

to generate the visuals.