Libstable: Fast, Parallel and High-Precision Computation of α-Stable Distributions in C/C++ and MATLAB

Javier Royuela-del-Val
Image Processing Laboratory
Universidad de Valladolid

November 4, 2013

1 Introduction

α-stable distributions [4] are a family of well-known probability distributions. However, the lack of closed analytical expressions hinders their application. Currently, several tools have been developed to numerically evaluate their density and distribution functions or to estimate their parameters, but available solutions either do not reach sufficient precision on their evaluations or are excessively slow for practical purposes. Moreover, they do not take full advantage of the parallel processing capabilities of current multi-core machines. Other solutions work only on a subset of the α-stable parameter space. We present a C/C++ library and a MATLAB front-end that permits fully parallelized, fast and high precision evaluation of density, distribution and quantile functions (PDF, CDF and CDF^{-1} respectively), random variable generation and parameter estimation of α-stable distributions in their whole parameter space.

2 Usage of libstable

`Libstable` has been developed at the Image Processing Laboratory (LPI) to give support for various research projects based on α-stable distributions, where it is used on a regular basis both in `C/C++` and in `MATLAB` developments. It has been thoroughly tested on specific applications. Its source code and sample programs are publicly available at `http://www.lpi.tel.uva.es/stable` under the GPLv3 [5] license.

2.1 Compiling the library

The developed library can be easily compiled from the source code with the `make` command. `Libstable` depends on several numerical methods provided by the GSL, which must be installed in the system. After compilation, both shared (`libstable.so`) and static (`libstable.a`) versions of the library are produced. Several example programs to test the main functions of the library are also provided and compiled against the static version of the library by default. Further documentation on the library functions can be found within the library distribution.

2.2 Usage in C/C++ developments

The next example program (`example.c`) illustrates how to use `libstable` to evaluate the PDF of an α-stable distribution with given parameters and with 0 parameterization at a single point x:

```
#include <stdio.h>
#include <stable_api.h>

int main (void)
{
  double alpha = 1.25, beta = 0.5, sigma = 1.0, mu = 0.0;
  int param = 0;
  double x = 10;
  StableDist *dist = stable_create(alpha,beta,sigma,mu,param);
  double pdf = stable_pdf_point(dist,x,NULL);
  printf("PDF(%g;%1.2f,%1.2f,%1.2f,%1.2f) = %1.15e\n",
                               x,alpha,beta,sigma,mu,pdf);
  stable_free(dist);
  return 0;
}
```

The output of one execution of the example is:

```
PDF(10;1.25,0.50,1.00,0.00) = 3.225009046591384e-03
```

2.2.1 Compiling and linking

If the `libstable` header files and compiled library are not located on the standard search path of the compiler and linker respectively, their location must be provided as command line flag to compile and link the previous program. The program must also be linked to the GSL and system `math` libraries. Typical command for compilation and static linking of a source file `example.c` with the GNU `C` compiler `gcc` is

```
$ gcc -O3 -I/path/to/headers -c example.c
$ gcc example.o /path/to/libstable/libstable.a -lgsl -lgslcblas -lm
```

The `-O3` option activates several optimization procedures of the `gcc` compiler. Other options can also be considered. When linking with the shared version of the library, the path to `libstable.so` must be provided to the system's dynamic linker, typically by defining the shell variable `LD_LIBRARY_PATH`. The path to the shared library must also be provided when linking the program:

```
$ gcc -L/path/to/libstable example.o -lgsl -lgslcblas -lm -lstable
```

2.2.2 Setting general parameters

Some general parameters can be adjusted on the library, such as precision required or available number of threads. These parameters are stored as global variables that can be read and modified with the functions described below.

In multi-core systems, the functions

```
unsigned int stable_get_THREADS()
void stable_set_THREADS(unsigned int threads)
```

return and set up, respectively, the number of threads of execution used by the library. When setting the number of threads to 0, the library will use as many threads as processing cores available in the system.

The *relative tolerance* indicates the precision required when calculating the PDF and CDF. The following functions return the current relative tolerance and set the desired one, respectively:

```
double stable_get_relTOL()
void   stable_set_relTOL(double reltol)
```

2.2.3 Managing distributions

The library defines the structure `StableDist` which contains some values associated to an α-stable distribution, such as its parameters, the parameterization employed and a random number generator. An α-stable distribution with desired parameters α, β, σ and μ in parameterization `param` can be created by

```
StableDist * stable_create(double alpha, double beta,
                           double sigma, double mu, int param)
```

The function returns a pointer to a `StableDist` structure. This pointer is passed as an argument to other functions. Once the StableDist structure is created, the parameters of a distribution can be easily changed:

```
int stable_setparams(StableDist * dist, double alpha, double beta,
                     double sigma, double mu, double param)
```

The returned value can be one of the following predefined constants:

INVALID:	Invalid or out of range parameters introduced
STABLE:	General α-stable case
ALPHA_1:	$\alpha = 1$ case
GAUSS:	Gaussian distribution ($\alpha = 2$)
CAUCHY:	Cauchy distribution ($\alpha = 1$ and $\beta = 0$)
LEVY:	Lévy distribution ($\alpha = 0.5$ and $\beta = 1$)

A copy of an existing distribution can be obtained by

```
StableDist * stable_copy(StableDist * src_dist);
```

To delete a distribution and free its associated memory resources call

```
void stable_free(StableDist * dist)
```

2.2.4 Calculating PDF, CDF and CDF^{-1}

This section describes the functions provided to calculate the PDF, CDF and CDF^{-1} of α-stable distributions. The calculation is based the equations provided by Nolan [3]. Two possibilities are provided for each: a single point function and an array function. The prototypes of the single point functions are:

```
double stable_cdf_point(StableDist * dist, const double x, double * err)
double stable_pdf_point(StableDist * dist, const double x, double * err)
double stable_inv_point(StableDist * dist, const double q, double * err)
```

Each function returns the value of the stable function being evaluated and stores in `err` an estimation of the absolute error committed. If this estimation is not required, a `NULL` pointer can be passed as argument instead. For the array functions:

```
void stable_cdf(StableDist * dist, const double * x, const int Nx,
                double * pdf, double * err)
void stable_pdf(StableDist * dist, const double * x, const int Nx,
                double * cdf, double * err)
void stable_inv(StableDist * dist, const double * q, const int Nq,
                double * inv, double * err)
```

the number of evaluation points (`Nx`, `Nq`, respectively) must be provided. An estimation of the absolute error committed at each point of evaluation is stored in an array `err`. If this estimation is not required, a `NULL` pointer can be passed as argument instead.

2.2.5 Random sample generation

In order to generate an α-stable random sample with desired parameters and population size, a distribution must be created with those parameters as exposed above. Once the parameters have been set, the function

```
double stable_rnd_point(StableDist * dist)
```

returns a single realization of an α-stable random variable. To obtain an array of realizations the function

```
void stable_rnd(StableDist * dist, double * rnd, const unsigned int N)
```

stores in the `rnd` array `N` independent realizations of an α-stable random variable.

When generating random samples, the function

```
void stable_rnd_seed(StableDist * dist, unsigned long int s)
```

initializes the internal random generator to a desired seed. This allows to reproduce results across different executions. Nevertheless, it will be usually desirable to obtain different results in different realizations, which involves to initialize the random generator to different seeds. This can be easily achieved by initializing to a time dependent seed such as

```
stable_rnd_seed(dist,time(NULL))
```

2.2.6 Parameter estimation

Several methods are available in `Libstable` to estimate the parameters of α-stable distributions from a data sample. Given its speed and simplicity, [2] method is always used as initial approximation to the final estimation. It can be invoked by the function

```
void stable_fit_init(StableDist * dist, const double * data,
                     const unsigned int N,  double * nu_c,double * nu_z)
```

This function sets the parameters of the distribution structure `dist` to the estimation realized from the sample in `data`, of length `N`. `stable_fit_init` is employed in other iterative estimation methods which make use of the values stored in `nu_c` and `nu_z`. If these values are not required, a `NULL` pointer can be passed as an argument.

As previously exposed, McCulloch estimator has low accuracy. `Libstable` provides other estimators when higher accuracy is required. [1] iterative estimation is provided by the function

```
int stable_fit_koutrouvelis(StableDist * dist, const double * data,
                            const unsigned int N)
```

In this case, the parameters stored in the distribution `dist` when calling the function are considered as initial guesses of the final estimation. Therefore, `stable_fit_init` could be used in first place to obtain such approximation. The parameters in `dist` are then actualized by the estimator procedure. If the iterative method finishes correctly, the function returns 0. In other case, a different value indicates that an error has occurred, such as no convergence of the iterative method or estimated parameter values out of range.

The high performance achieved in the calculation of the PFC allows to perform ML estimation, although its recommended to apply it only with a reduced size of data samples. Besides, the library should be set to require a lower relative precision than the maximum achievable (a value of 10^-8 is recommended). Both indications intend to reduce computational costs associated to the evaluation of the likelihood function. Under these conditions, ML estimation of α-stable distributions can be realized by

```
int stable_fit_mle(StableDist *dist, const double *data,
                   const unsigned int N)
```

In order to overcome the limitations related to the high computational cost of ML estimation a modified ML method is provided. It performs an optimization procedure only the $\alpha - \beta$ 2D space, what simplifies the procedure and reduces number of evaluations of the likelihood function required to find a solution. This method is implemented by the function

```
int_stable_fit_mle2d(StableDist *dist, const double *data,
                     const unsigned int N)
```

As in Koutrouvelis estimator, ML and modified ML use the parameters in `dist` when calling the corresponding function are used as initial guesses then actualized by the method. A return value different from 0 indicates that some error has occurred during the execution of the algorithm.

An example of how to use the library to calculate the PDF, CDF and CDF^{-1} with desired parameters, generate a random sample and, given the sample, estimate the parameters of an α-stable distribution that better fits the generated data follows:

```
#include <stable_api.h>
[...]

int main(void) {
  double x[100],q[100],pdf[100],
         cdf[100],inv[100],rnd[100];
  double alpha=1.5, beta=0.5, sigma=2.0, mu=4.0;

  for (i=0;i<100;i++) {
    x[i] = -5+i*0.1;
    q[i] = 0.01*(i+0.5);
  }

  dist=stable_create(alpha,beta,sigma,mu,0));
  stable_pdf(dist,x,100,pdf,NULL);
  stable_cdf(dist,x,100,cdf,NULL);
  stable_inv(dist,q,100,inv,NULL);
  stable_rnd_seed(dist,time(NULL));
  stable_rnd(dist,rnd,100);

  stable_fit_init(dist,rnd,100,NULL,NULL);
  stable_fit_koutrouvelis(dist,rnd,100);

  printf("Estimated parameters: %f %f %f %f\n",
     dist->alpha,dist->beta,dist->sigma,dist->mu_0);

  stable_free(dist);
  return 0;
}
```

2.3 Usage in MATLAB environment

`MATLAB` environment supports loading shared `C` libraries by calling the `loadlibrary` function. The shared version of the proposed library (`libstable.so`) and the header file `stable_api.h` are required. In order to start using `Libstable` execute the following command in `MATLAB` environment:

```
loadlibrary('libstable','stable_api.h')
```

Paths to `libstable.so` and `stable_api.h` must be in current folder or included in `MATLAB` search path.

When the library is no longer needed, it can be unloaded by executing

```
unloadlibrary('libstable')
```

Several `MATLAB` functions in the form of `.m` files are provided to access the capabilities of `Libstable` library. These files can be easily modified by the user to adjust library parameters as needed. The managing of α-stable distributions described in section 2.2 is performed by the provided functions, so it is not necessary to create or to delete the distributions.

2.3.1 Calculating PDF, CDF and CDF^{-1}

The functions

```
pdf = stable_pdfC(p,x,param)
cdf = stable_cdfC(p,x,param)
inv = stable_invC(p,q,param)
```

returns a vector containing the evaluation of the PDF, CDF and CDF^{-1}, respectively, at the points in `x` (`q` for the CDF^{-1}). The parameters of the distribution are indicated in `p = [alpha, beta, sigma, mu]`, and `param` is the parameterization employed. The returned vector will have the same size as `x`.

By default, previous functions set the library to use the maximum number of available threads of execution and establish the relative precision of the library to a fixed value of 10^{-12}. This values can be easily changed by modifying the corresponding `.m` files.

The letter "C" on the functions names is included to indicate that a C shared library is being invoked when calling the function.

2.3.2 Random variable generation

The generation of α-stable random variables is provided by the function

```
rnd = stable_rndC(p,N,param)
```

A column vector containing `N` independent realizations of an α-stable random variable with parameters `p = [alpha, beta, sigma, mu]` in parameterization `param` is returned.

By default, the random generator seed is established to system time each time `stable_rndC` is called. These behavior can be modified in the function file.

2.3.3 Parameter estimation

A MATLAB function is provided for each of the estimation methods described in section 2.2:

```
p = stable_fit_initC(data)
p = stable_fit_koutrouvelisC(data)
p = stable_fit_mleC(data)
p = stable_fit_mle2dC(data)
```

These functions perform McCulloch, Koutrouvelis, ML and modified ML estimation, respectively, in the sample data `data`. The estimated parameters are returned in `p` vector. By default, McCullock estimator is used by the rest of methods as initial estimation of the parameters. A user defined initial estimation can be used by passing an additional argument `p_ini = [alpha_ini, beta_ini, sigma_ini, mu_ini]`, e.g.:

```
p = stable_fit_mle2dC(data, p_ini)
```

Following lines serve as an example of a MATLAB session in which Libstable is used to calculate the PDF, CDF and CDF^{-1} of an α-stable distribution with desired parameters, generate a random sample and, given the sample, estimate the parameters of the α-stable distribution.

In first place, the library must be loaded:

```
>> loadlibrary('libstable','stable_api.h')
```

Vectors of points of evaluation and parameters are initialized. parameterization employed is also indicated:

```
>> x = -5:.1:5;
>> q = 0.005:0.01:0.995;
>> p = [1.5, 0.5, 0.5, -1.0];
>> param = 0;
```

PDF, CDF, CDF^{-1} are evaluated and the results stored in corresponding vectors:

```
>> pdf = stable_pdfC(p, x, param);
>> cdf = stable_cdfC(p, x, param);
>> inv = stable_invC(p, q, param);
```

A sample of `N=500` realizations of the α-stable random variable is generated:

```
>> N = 500;
>> rnd = stable_rndC(p,N,param);
```

From the generated sample, the original parameters are estimated by Koutrouvelis method:

```
>> p_est = stable_koutrouvelisC(rnd)

p_est =

    1.6819   -0.6550    0.5816   -0.8226
```

Once the session has finished, the library can be unloaded:

```
>> unloadlibrary('libstable');
```

References

[1] Ioannis A. Koutrouvelis. An iterative procedure for the estimation of the parameters of stable laws. *Communications in Statistics - Simulation and Computation*, 10(1):17–28, 1981.

[2] John H. McCulloch. Simple consistent estimators of stable distribution parameters. *Communications in Statistics – Simulation and Computation*, 15(4):1109–1136, 1986.

[3] John P. Nolan. Numerical calculation of stable densities and distribution functions. *Stochastic Models*, 13(4):759–774, 1997.

[4] Gennady Samorodnitsky and Murad S. Taqqu. *Stable non-Gaussian Random Processes: Sstochastic Models with Infinite Variance*. Chapman and Hall/CRC, Boca Raton, CA, USA, 1994.

[5] The Free Software Foundation. Gnu general public license, version 3, June 2007. Last retrieved 10-30-2013.