MPCOTool (the Multi-Purposes Calibration and Optimization Tool): an open source software to supply empirical parameters required in simulation models

Software authors: J. Burguete and B. Latorre
Manual authors: J. Burguete, P. García-Navarro and S. Ambroj

December 30, 2022

Contents

Chapter 1

Building from the source code

The source code in MPCOTool is written in C language. This software has been built and tested in the following operative systems:

- Debian Hurd, kFreeBSD and Linux 9;
- Devuan Linux 2;
- DragonFly BSD 5.2;
- Dyson Illumos;
- Fedora Linux 29;
- FreeBSD 11.2;
- Linux Mint DE 3;
- Manjaro Linux;
- Microsoft Windows 7[1], and 10[1];
- NetBSD 7.0;
- OpenBSD 6.4;
- OpenIndiana Hipster;
- OpenSUSE Linux Leap;
- Ubuntu Mate Linux 18.04;
- and Xubuntu Linux 18.10.

Probably, this software can be built and it works in other operative systems, software distributions or versions but it has not been tested.

In order to build the executable file from the source code, a C compiler (GCC [2018] or Clang [2018]), the configuration systems Autoconf [2018], Automake [2018] and Pkg-config [2018], the executable creation control program GNU-Make [2018] and the following open source external libraries are required:

[1]Windows 7 and Windows 10 are trademarks of Microsoft Corporation.

- Libxml [2018]: Library required to read the main input file in XML format.
- GSL [2018]: Scientific library required to generate the pseudo-random numbers used by the genetic and the Monte-Carlo algorithms.
- GLib [2018]: Library required to parse the input file templates and to implement some data types and the routines used to parallelize the usage of the computer's processors.
- JSON-GLib [2018]: Library used to read the main input file in JSON format.
- GTK+3 [2018]: Optional library to build the interactive GUI application.
- OpenMPI [2018] or MPICH [2018]: Optional libraries. When installed, one of them is used to allow parallelization in multiple computers.

The indications provided in Install-UNIX [2018] can be followed in order to install all these utilities.

On OpenBSD 6.4, prior to build the code, you have to select adequate version of Autoconf and Automake doing on a terminal:

```
> export AUTOCONF_VERSION=2.69 AUTOMAKE_VERSION=1.16
```

On Window systems, you have to install MSYS2 (http://sourceforge.net/projects/msys2) and the required libraries and utilities. You can follow detailed instructions in https://github.com/jburguete/install-unix.

On Fedora Linux 29, in order to use OpenMPI compilation, do in a terminal (in 64 bits version):

```
> export PATH=$PATH:/usr/lib64/openmpi/bin
```

On FreeBSD 11.2, due to a wrong error in default gcc version, do in a terminal:

```
> export CC=gcc5 (or CC=clang)
```

Once all the tools installed, the Genetic source code must be downloaded and it must be compiled following on a terminal:

```
> git clone https://github.com/jburguete/genetic.git
> cd genetic/2.2.2
> ./build
```

The following step is to download the source code MPCOTool, to link it with Genetic and compile together by means of:

```
> git clone https://github.com/jburguete/mpcotool.git
> cd mpcotool/4.0.8
> ln -s ../../genetic/2.2.2 genetic
> ln -s genetic/libgenetic.so (or .dll on Windows systems)
> ./build
```

On servers or clusters, where no-GUI with MPI parallelization is desirable, replace the *build* script by:

```
> ./build_without_gui
```

Optionally, to compile the tests with the standard analytical optimization functions, you have to do (the executable files of test2, test3 and test4 use also the *Genetic* library):

```
> cd ../tests/test2
> ln -s ../../../genetic/2.2.2 genetic
> ln -s genetic/libgenetic.so (.dll on Windows systems)
> cd ../test3
> ln -s ../../../genetic/2.2.2 genetic
> ln -s genetic/libgenetic.so (.dll on Windows systems)
> cd ../test4
> ln -s ../../../genetic/2.2.2 genetic
> ln -s genetic/libgenetic.so (.dll on Windows systems)
> cd ../../4.0.8
> make tests
```

Finally, the next optional step build the PDF manuals:

```
> make manuals
```

Chapter 2

Interface

WARNING! Real numbers are represented according to the international standard, overriding locale settings, separating the integer and decimal parts by ".".

2.1 Command line format

In this section optional arguments are typed in square brackets.

- Command line in sequential mode (where X is the number of threads to execute and S is a seed for the pseudo-random numbers generator):

```
> ./mpcotoolbin [-nthreads X] [-seed S] input_file.xml
[result_file] [variables_file]
```

- Command line in parallelized mode (where X is the number of threads to open for every node and S is a seed for the pseudo-random numbers generator):

```
> mpirun [MPI options] ./mpcotoolbin [-nthreads X] [-seed S]
input_file.xml [result_file] [variables_file]
```

- The syntax of the simulator program has to be:

```
> ./simulator_name input_file_1 [input_file_2] [...] output_file
```

  There are two options for the output file. It can begin with a number indicating the objective function value or it can be a results file that has to be evaluated by an external program (the evaluator) comparing with an experimental data file.

- In the last option of the former point, the syntax of the program to evaluate the objective function has to be (where the results file has to begin with the objective function value):

```
> ./evaluator_name simulated_file experimental_file results_file
```

- On UNIX type systems the GUI application can be open doing on a terminal:

```
> ./mpcotool
```

2.2 Using MPCOTool as an external library

MPCOTool can also be used as an external library by doing:

1. Copy the dynamic library ("libmpcotool.so" on Unix systems or "libmpcotool.dll" on Windows systems) to your program directory.
2. Include the function header in your source code:
   ```
   > extern int mpcotool (int argn, char **argc);
   ```
3. Build the executable file with the linker flags:
   ```
   > $ gcc ... -L. -Wl,-rpath=. -lmpcotool ...
   ```
4. Calling to this function is equivalent to command line order (see previous section):
   - argn: number of arguments including the program name.
   - argc[0]: "mpcotool" (program name).
   - argc[1]: first command line argument.

     ...
   - argc[$argn - 1$]: last command line argument.

2.3 Interactive graphical user interface application

An alternative form to execute the software is to perform the interactive graphical user interface application, called *MPCOTool*. In this application the parallelization in multiple computers with OpenMPI or MPICH is deactivated, it can be only used in command line execution. In the figure 2.1 a plot of the main window of this tool is represented. The main windows enable us to access to every variable, coefficient, algorithm and simulation softwares.

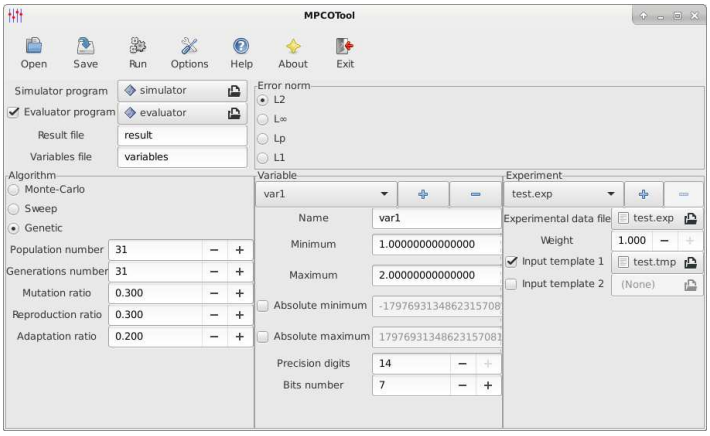

Figure 2.1: Main window of MPCOTool graphical user interface application.

Final optime results are presented in a dialog as the shown in the figure 2.2.