cTreeBalls

Release 1.0.0

Mario A. Rodriguez-Meza

Jan 12, 2026

CONTENTS:

1 Overview **1**
- 1.1 Introduction . . . 1
- 1.2 Compiling and getting started . . . 1
- 1.3 Configuration . . . 2
- 1.4 Parameters . . . 2
- 1.5 Python . . . 2
- 1.6 Plotting utilities . . . 3
- 1.7 License . . . 3
- 1.8 Acknowledgements . . . 3

2 Installation **5**
- 2.1 Makefile . . . 5
- 2.2 Makefile settings . . . 5
- 2.3 Makefile plataform machine . . . 5

3 Parameters **7**
- 3.1 Parameters related to the searching method . . . 7
- 3.2 Parameters to control the I/O file(s) . . . 7
- 3.3 Parameters to control histograms and their output files . . . 8
- 3.4 Set of parameters needed to construct a test model . . . 8
- 3.5 Miscellaneous parameters . . . 8

4 Catalog files **11**
- 4.1 Standard formats . . . 11
- 4.2 Miscellaneous formats . . . 11

5 Pre/Post processing **13**
- 5.1 Pre-processing . . . 13
- 5.2 Post-processing . . . 13

6 AddOn's **15**
- 6.1 Include files . . . 15
- 6.2 Source files . . . 15

7 Python interface **17**
- 7.1 Using command line interface . . . 17
- 7.2 Cython interface . . . 17

8 Hands on **19**

9 2-point correlation functions **21**

10 3-point correlation functions **23**
10.1 Harmonic base 23

11 Indices and tables **25**

CHAPTER

ONE

OVERVIEW

cTreeBalls: Correlation functions computation with Tree/Balls methods

Author: Mario A. Rodriguez-Meza

For download and information, see https://github.com/rodriguezmeza/cTreeBalls

1.1 Introduction

Correlation function computation with Tree/Balls methods (short-name cBalls) is a C code for computing correlation functions using tree and balls methods. So far can compute 2 points correlation function (2pcf) and 3 points correlation function for scalar fields like weak lensing convergence.

1.2 Compiling and getting started

Download the code by cloning it from https://github.com/rodriguezmeza/cTreeBalls

Dependencies: cBalls optionally needs gsl version 2.7.1 and cfitsio version 4.4.1 installed in your system. Go to its web page GSL for details or ask to your system administrator. Make necessary changes in `Makefile_machine` file and look up for `GSL`. Can be switched off: USEGSL = 0 in `Makefile_settings`. I/O cfitsio library is set it OFF. Can be set it ON in `addons/Makefile_settings`.

Go to the `cTreeBalls` directory (`cd cTreeBalls`) and compile (`make clean; make`). If the first compilation attempt fails, you may need to open the Makefile_machine file and adapt the name of the compiler (default: gcc), of the optimization flag (default: `-O4 -ffast-math`) and of the OpenMP flag (default: -fopenmp; this flag is facultative, you are free to compile without OpenMP if you don't want parallel execution; note that you need the version 4.2 or higher of gcc to be able to compile with `-fopenmp`). The code has been tested with gcc version 10 and would be working with version 11, 12. (In particular, for compiling on Mac >= 10.9 despite of the clang incompatibility with OpenMP).

To check that the code runs, type:

```
cd tests
../cballs parameters_explained
```

The parameters_explained file is a reference input file, containing and explaining the use of all possible input parameters.

By default **cBalls** reads/writes catalog of points to analyzed files in 4-column format with x, y, z columns first and then value of the convergence field. It has a two line header:

```
# nbody NDIM Lx Ly Lz
# nbody-value NDIM-value Lx Ly Lz - values
```

Try running:

```
../cballs nbody=6480 o=points_on_sphere testmodel=unit-sphere-random options=stop
```

In the `Output` directory you will have a file: `points_on_sphere.txt`. View its contents to see the two lines header and the 4 columns structure of data.

Note: in the above example `points_on_sphere` was not given an extension. By default cBalls gives to the output files the extension `.txt`.

On top of that, if you wish to modify the code, you will find comments directly in the files in the folder `addons`, and the modules you may add must go in this folder. See one of the most simple in `addons/direct_method`.

For the moment you may consult man page:

```
man ../docs/cballs.m
```

or open with a browser the html file: docs/man/cballs.html

See more details about parameters needed by cBalls below (*Parameters*).

1.3 Configuration

cBalls can be configured by switching on/off several options. Configuration file is `Makefile_setting`. Search method `balls-omp` in folder addons/balls_omp: `Makefile_settings_balls_omp` has other options that can be switched on/off.

1.4 Parameters

The list of available command line parameters can be consulted using the `--help` flag:

```
../cballs --help
```

or its short version:

```
../cballs -h
```

1.5 Python

cBalls python module (cballys) will be installed if you use:

```
make clean; make all
```

Note: if problems, you can go to python directory and execute:

```
python setup.py build
python setup.py install --user
```

To test it go to directory tests and 'run':

```
python test_cython_balls.py
```

Note: this interface in Cython was tested in a python environment with `python3.7`.

1.6 Plotting utilities

Several Jupyter notebooks, written by Abraham Arvizu and Eladio Moreno, are available to process cBalls results. They are in the github repository:

https://github.com/joar-cafe/CBalls_plots/tree/main/benchmarks

or you can find more scripts in `tests` folder.

1.7 License

cBalls is written by Mario A. Rodriguez-Meza, and is distributed under the MIT license. If you use this program in research work that results in publications, please cite the following paper:

Abraham Arvizu et al., arXiv:2048.16847

1.8 Acknowledgements

cBalls use/is based on the following codes or projects:

- Zeno
- Gadget-2
- CUTE
- Numerical recipies
- GSL
- CLASS
- CFITSIO
- HEALPix

Also author acknowledges for helpful discussion and testing to the following people:

- Abraham Arvizu
- Alejandro Aviles
- Juan Carlos Hidalgo
- Eladio Moreno
- Gustavo Niz
- Axel Romero Tisnado
- Sofia Samario

CHAPTER

TWO

INSTALLATION

This section describes the installation settings of cBalls:

2.1 Makefile

2.2 Makefile settings

2.3 Makefile plataform machine

CHAPTER

THREE

PARAMETERS

This section describes the various parameters cBalls needs for controlling what the searching process do:

3.1 Parameters related to the searching method

searchMethod
(str or list) [alias: search] The searching method to use. Default is `tree-omp-sincos`. Fastest method so far is `octree-ggg-omp`.

Use it as:

```
searchMethod = octree-ggg-omp
```

In command line version do not use spaces before and after = or it won't be parsed correctly. In a parameter file you have more liberty.

mChebyshev
(positive int) [alias: mcheb] The number of multipoles to compute for a 3pcf computation. Number of them are: `mChebyshev + 1`, because it includes the monopole.

Use it as:

```
mChebyshev = 7
```

Default value is 7. As multipoles comes from a harmonic expansion, we may be interested in computing the whole 3pcf. This process involves a FFT, therefore give `mChebyshev + 1` as a power of 2.

3.2 Parameters to control the I/O file(s)

infile
(str, default="") [alias: in] File names with points to analyse.

infileformat
(str, default=columns-ascii) [alias: infmt] Data input files format (columns-ascii, binary or takahasi)

The input columns for `columns-ascii` format:

- `x` = x position of a point in the catalog.
- `y` = y position of a point in the catalog.
- `z` = z position of a point in the catalog.
- `kappa` = The kappa value of the point.

3.3 Parameters to control histograms and their output files

useLogHist
(bool, default=true) Which type of binning should be used.

logHistBinsPD
(float) The minimum separation to include in the histograms.

rangeN
(float) The maximum separation to include in the histograms.

rminHist
(float) The minimum separation to include in the histograms.

sizeHistN
(int) The number of output bins to use.

3.4 Set of parameters needed to construct a test model

seed
(int, default=123) Random number seed to test run or useful to change a random region in Takahasi simulations.

testmodel
(str, default=simple-cubic-random) [alias: tstmodel] Test model name to analyse.

nbody
(int, default=16348) Number of points to test.

lengthBox
(float, default=10000) [alias: lbox] Length of the box to test.

3.5 Miscellaneous parameters

script
(str, default="") Scripts in shell or python that can be run in pre-processing or post-processing.

stepState
(int, default=10000) number of steps to save a state-run info (pivot number completed in the log file).

verbose
(int, default=1) [alias: verb] How verbose the code should be during processing.

- 0 = no output unless there is an error
- 1 = output warnings
- 2 = output progress information
- 3 = output extra debugging lines

verbose_log
(int, default=1) [alias: verblog] To print messages to a log file `cballs.log` in directory `tmp` under output directory given by the parameter: ``rootDir`.

Amount of message information is controlled by the int given.

numberThreads
(int, default=4) [alias: nthreads] How many OpenMP threads should be used.

It is needed to switch on OpenMP: `OPENMPMACHINE = 1` in `Makefile_settings` and recompile cBalls again.

options

(str, default="") [alias: opt] You may give here various code behavior options.

Use it as:

```
options = str1,str2,str3,...
```

where str# is one of the:

- stop = stop execution before searching process
- compute-HistN = compute NN encounters and save histogram in a file
- and-CF = if you use `compute-HistN` then you may compute and save the correlation funcion of NN encounters (the equivalent to the radial distribution function in liquids).
- no-one-ball = during the searching process does not use balls criterion to speed up the code

Note

- It is not necessary to specify all the parameters. You need to give only the ones apropriate to the run. The rest of parameters will use their default values if they are OK with you.
- When you specify the root output directory using: `rootDir`, and this is a single directory that will be located in the pwd dir, then do not use `./` at the begining of the name or `/` at its end.

CHAPTER

FOUR

CATALOG FILES

This section describes the various formats of catalog points you need to compute correlation functions on them.

They are given with the option: `infileformat`.

4.1 Standard formats

columns-ascii
The file(s) with the data to be correlated.

binary
The file with data to be correlated in binary format.

takahashi
The file with data to be correlated using Takahashi etal simulations format (can be transformed to Healpix format).

4.2 Miscellaneous formats

gadget
Gadget files can be read using this format. Catalog can be only one file or composed of several files.

multi-columns-ascii
This format is to be able to read any ascci file with values arrage in columns. Positions can be in any order and with/without header. For example, files with halo catalogs from Rockstar can be read.

fits-healpix
to read maps in Healpix format using cfitsio library. This format permits to apply a mask in the same format.

CHAPTER

FIVE

PRE/POST PROCESSING

This section describes the various pre/pos processing for controlling what the *cballs* before/after main processing:

5.1 Pre-processing

5.2 Post-processing

CHAPTER

SIX

ADDON'S

This section describes the mechanism to add more functionally to cBalls:

6.1 Include files

6.2 Source files

CHAPTER

SEVEN

PYTHON INTERFACE

This section describes the mechanism to add more functionally to cBalls:

7.1 Using command line interface

7.2 Cython interface

CHAPTER

EIGHT

HANDS ON

This section describes a practical guide to use **cBalls**:

Let us test one of the Takahashi realizations. We download the realization using unix `wget` command in the terminal:

```
wget http://cosmo.phys.hirosaki-u.ac.jp/takahasi/allsky_raytracing/sub1/nres12/allskymap_
↪nres12r000.zs9.mag.dat
```

Then we will have in our working directory the file `allskymap_nres12r000.zs9.mag.dat`, 3 Gb in size with ~200 million points distributed on the surface of a unit sphere.

CHAPTER

NINE

2-POINT CORRELATION FUNCTIONS

cBalls can compute 2-point correlations (2pcf) when counts (N) or a scalar field are involved (like convergence K):

NN
: It is the normal 2-point correlation function of number counts (typically galaxy counts).

KK
: So far we have consider that the scalar field is the convergence in weak lensing κ. Then this gives the 2-point kappa-kappa correlation function.

N represent simple counting and K represent a real scalar field, like convergence in weak lensing.

CHAPTER

TEN

3-POINT CORRELATION FUNCTIONS

cBalls can also compute 3-point correlations (3pcf) when counts (N) or a scalar field are involved (like convergence κ). In particular to compute the correlation KKK the numerical code use the harmonic base.

10.1 Harmonic base

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search