Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2016-0107

Vol. **22**, No. 1, pp. 1-38
July 2017

# PyCFTBoot: A Flexible Interface for the Conformal Bootstrap

Connor Behan*

*Department of Physics and Astronomy, Stony Brook University, Stony Brook, NY 11790, USA.*

Received 15 July 2016; Accepted (in revised version) 7 November 2016

**Abstract.** We introduce PyCFTBoot, a wrapper designed to reduce the barrier to entry in conformal bootstrap calculations that require semidefinite programming. Symengine and SDPB are used for the most intensive symbolic and numerical steps respectively. After reviewing the built-in algorithms for conformal blocks, we explain how to use the code through a number of examples that verify past results. As an application, we show that the multi-correlator bootstrap still appears to single out the Wilson-Fisher fixed points as special theories in dimensions between 3 and 4 despite the recent proof that they violate unitarity.

**AMS subject classifications**: 65Z05, 68W30, 81T80, 90C34

**Key words**: CFT, conformal bootstrap, python, semidefinite programming.

## 1 Introduction

The conformal bootstrap [1, 2] has joined holography [3] as one of the most important tools for understanding strongly coupled conformal field theories (CFTs) in higher dimensions. Much of the progress comes from a numerical procedure initiated in [4], which exploits the constraints of crossing symmetry and unitarity. This has been successfully used to bound scaling dimensions and three point function coefficients in a wide range of conformal [5–16] and superconformal [17–21] theories in dimensions between 2 and 6. The first widely released code designed to perform these calculations was `JuliBoots` [22], a conformal bootstrap package based around a linear program solver. Shortly afterward, the solver `SDPB` [23] was released, giving the community access to the semidefinite programming methods pioneered in [10, 18, 24]†.

*Corresponding author. *Email address:* `connor.behan@gmail.com` (C. Behan)

†Readers interested in conformal blocks for their role in algebraic geometry might appreciate the [25] package.

 

The advantages of the two are largely complementary. Semidefinite programming has superior performance in systems with multiple crossing equations and it is currently the only technique which extracts information from correlators of operators with different scaling dimensions [24]. As such, `SDPB` has become the standard code for most numerical bootstrap studies in the last year [26–34]. Unlike `JuliBoots` however, it does not provide simple methods for specifying important kinematics information. Included in this are the crossing equations which depend on the type of CFT being studied and conformal blocks, special functions that depend on the dimension of space and a number of accuracy parameters. All of the above studies have performed these calculations using customized scripts for `Mathematica`. A new program, aiming to reduce this duplication of effort, is `PyCFTBoot` written in Python. Realizing a hope of [22], it handles the computer algebra that goes into a numerical bootstrap entirely with free software. `PyCFTBoot` may be downloaded from

`https://github.com/cbehan/pycftboot`,

where all future development is expected to take place. Besides `SDPB`, a few other dependencies are required in order to use it.

In mathematical Python software, `numpy` [35] and `sympy` [36] are two widely used packages that come to mind. Both of them are needed by `PyCFTBoot`. However, `sympy` is not fast enough to generate large tables of conformal blocks. It is only used in a few non-critical places that need to call Gegenbauer polynomials or the incomplete gamma function. Instead, the bulk of the symbolic algebra is handled by a fast C++ library called `symengine`. Python bindings have been chosen (over Ruby and Julia) because they are the most mature at the time of writing. These less common packages are downloadable from

`https://github.com/symengine/symengine` (last tested: 5427bbe),
`https://github.com/symengine/symengine.py` (last tested: 9d23ef7).

Surprises are most easily avoided by using `PyCFTBoot` with Python 2.7 on GNU / Linux, but it has also been tested with Python 3.5. Descriptions of the important functions, included in the source code, may be viewed with the Python documentation server. Additionally, readers who are anxious to try the bootstrap may follow the commented tutorial distributed alongside the main file.

In Section 2 of this note, we describe the algorithms that have been chosen to generate derivatives of conformal blocks and report some rough performance figures. Section 3 explains how semidefinite programs are formulated from these tables. In describing the main SDP object, it contains a few parts that read like passages from a user manual. Some examples, worked out in Section 4, demonstrate that most of the known bootstrap results to date can in principle be reproduced with `PyCFTBoot`. Before we conclude, Section 5 extends a previous result in the literature by using `PyCFTBoot` to probe the "islands" of allowed critical exponents in dimensions between 3 and 4 [37].