HIPSTER Documentation

Oliver Philcox, Daniel Eisenstein

Dec 03, 2022

Contents:

CHAPTER 1

Overview

HIPSTER is a code to quickly compute small-scale power spectra and isotropic bispectra for galaxy surveys and cosmological simulations, based on the work of Philcox & Eisenstein (2019, MNRAS, arXiv) and Philcox (2020, submitted, arXiv). This computes the Legendre multipoles of the power spectrum, $P_\ell(k)$ or bispectrum $B_\ell(k_1, k_2)$ in *configuration space*, by computing weighted pair counts over the survey or simulation box truncated at some maximum radius. This does not include shot-noise or aliasing, fully accounts for window function effects and is optimized for small-scale power spectrum and bispectrum computation. Note that the bispectrum code algorithm has the same complexity as the power spectrum algorithm, thanks to spherical harmonic decompositions. By combining HIPSTER with conventional FFT-based methods, spectra can be measured efficiency across a vast range of wavenumbers.

The code can be run either in *aperiodic* or *periodic* mode, for galaxy surveys or N-body simulations respectively. The *periodic* mode contains various optimizations relating to the periodic geometry, as detailed in the second paper. For the bispectrum, only *periodic* mode is currently supported, though the alternative is easily added. Generalization to anisotropic bispectra is straightforward (and requires no additional computing time) and can be added on request.

The source code is publicly available on Github, and contains many modules modified from the RascalC covariance matrix code.

To compute a periodic matter power spectrum up to $\ell = L$ from a particles in a simulation box (`data.dat`), with pair counts truncated at radius `R0` with k-space binning file `binning.csv` on 4 CPU-cores, we simply run:

```
./hipster_wrapper_periodic.sh --dat data.dat --l_max L -R0 R0 -k_bin binning.csv --
↪nthreads 4
```

For a galaxy power spectrum with a non-trivial survey geometry, we also need random particle files (`randoms.dat`) and the usage is simply:

```
./hipster_wrapper.sh --dat data.dat --ran_DR randoms.dat --ran_RR randoms.dat -l_max␣
↪L -R0 R0 -k_bin binning.csv --nthreads 4
```

To compute an isotropic bispectrum up to $\ell = L$ from a particles in a simulation box (`data.dat`), with pair counts truncated at radius `R0` with k-space binning file `binning.csv` on 4 CPU-cores, we simply run:

```
./hipster_wrapper_bispectrum.sh --dat data.dat --l_max L -R0 R0 -k_bin binning.csv --
↪nthreads 4
```

This is described in detail in the accompanying pages.

1.1 Package Installation

To install HIPSTER on a Linux machine, simply clone the Github repository:

```
git clone https://github.com/oliverphilcox/HIPSTER.git
```

Assuming the *Dependencies* are satisfied, HIPSTER is now ready to run using the simple bash wrappers `hipster_wrapper.sh`, `hipster_wrapper_periodic.sh` and `hipster_wrapper_bispectrum.sh`, as described in the *Computing Configuration-Space Spectra* section. This requires the inputs specified in *File Inputs*. For advanced usage, the routines in *Advanced Usage* can be used.

For advanced users, the C++ code must also be compiled via:

```
cd HIPSTER
make [Periodic=-DPERIODIC] [Bispectrum=-DBISPECTRUM]
```

HIPSTER can be run for either *periodic* or *aperiodic* data-sets. This is specified by adding compiler flags in the Makefile, or can simply be activated by adding the line `Periodic=-DPERIODIC` to the `make` command. See *Note on Periodicity* for more information. Similarly the bispectrum mode is activated with `Bispectrum=-DBISPECTRUM`. Note that compilation into the correct format is done automatically if the main wrappers are used.

Note for Mac Users: HIPSTER is primarily designed for Linux machines, though running on a Mac is also possible. To do so, you must have a recent version of GCC and gnu-getopt (emulating the `getopt` Linux script). Furthermore the code must be compiled without OpenMP, by removing the `-DOPENMP` and `-lgomp` flags from the Makefile.

1.1.1 Dependencies

HIPSTER requires the following (often pre-installed) packages:

- C / C++ compiler: Tested with gcc 5.4.O. Requires g++ above 5.1 (to use C++11 directives).
- Gnu Scientific Library (GSL): Any recent version.
- OpenMP: Any recent version (optional, but required for parallelization)
- Python: 2.7 or later, 3.4 or later (required for pre- and post-processing)

Additionally, for *periodic* surveys, we require Corrfunc (v2.0 or later) to compute geometry correction functions via efficient pair counting. This can be installed using `pip install corrfunc`.

1.1.2 Acknowledgements

Main Authors:

- Oliver H. E. Philcox (Princeton / Harvard)
- Daniel J. Eisenstein (Harvard)

Additional Collaborators:

- David N. Spergel (CCA / Princeton)
- Francisco Villaescusa-Navarro (CCA / Princeton)
- Lehman Garrison (CCA / Harvard)

- Zachary Slepian (Florida)

Please cite the initial theory paper (Philcox & Eisenstein 2019, accepted by MNRAS, arXiv) and the periodic power spectrum and bispectrum paper (Philcox 2020, submitted to MNRAS, arXiv) when using this code in your research.

Note that many of the code modules and convenience functions are based on those of RascalC, developed by Oliver Philcox, Daniel Eisenstein, Ross O'Connell and Alexander Wiegand.

1.2 File Inputs

The HIPSTER code requires two main inputs; (a) files containing galaxy, simulation particle or random particle positions and (b) a file specifying the desired k-space binning strategy. The file types of these are described below. We additionally provide a number of utility functions to assist with the creation of these files. These are located in the `python/` directory.

1.2.1 Data and Random Particle Files

The main inputs to the HIPSTER code are files containing the locations and weights of galaxies or simulation particles (i.e. 'data') and random particles (i.e. 'randoms', only for non-periodic surveys). The random particles are of the same form as those used in real-space correlation function analyses, and we expect their distribution to match that of unclustered galaxies in the survey. These files are usually provided by clustering teams or can be simply created in the case of a periodic box geometry. The file format is a list of particle positions in space-separated (x,y,z,weight) format, with the co-ordinates given in comoving $h^{-1}\mathrm{Mpc}$ units. For periodic simulation box data, weights can represent the masses of different tracer particles, and if not specfied are assumed to be unity (see particle-weights-note.)

We provide a convenience function to convert galaxy/random files in comoving (RA,Dec,z,weight) co-ordinates to the required format (using a simple WCDM co-ordinate converter by Daniel Eisenstein):

```
python python/convert_to_xyz.py {INFILE} {OUTFILE} {OMEGA_M} {OMEGA_K} {W_DARK_ENERGY}
```

where {INFILE} and {OUTFILE} are the filenames for the (RA,Dec,redshift,weight) and the remaining parameters specify the (present-day) cosmology. If these are not specified a cosmology of $\{\Omega_m = 0.31, \Omega_k = 0, w_\Lambda = -1\}$ is assumed by default.

For power spectrum pair-count analysis, we usually require random particle files larger than the data; we suggest using $N_{\rm rand} \sim 50 N_{\rm gal}$ for DR counts and $N_{\rm rand} \sim 10 N_{\rm gal}$ for RR counts to minimize random noise. For this reason we provide a further convenience function to draw a random subset of a given particle file (in (x,y,z,weight) co-ordinates). This is run as follows (where {N_PARTICLES} specifies the size of the output file):

```
python python/take_subset_of_particles.py {INPUT_FILE} {OUTPUT_FILE} {N_PARTICLES}
```

For periodic boxes, we do not need to generate random particle files, but the above script can be useful for subsampling the data, for faster runtimes. (For the bispectrum, random particles are used, but these are generated internally.) Note also that the HIPSTER wrapper has an option for subsampling; if set, this uses a random subset of the data (and random particles), with the fraction specified by the user. Whilst this is useful to get an idea of run-times and for fast evaluation, we recommend applying subsampling separately for full analyses, to ensure that the subsampled data is the same each time the code is run.

1.2.2 Binning Functions

In addition to the sets of data/random positions, we require a file to specify the desired k-space binning of the output power spectra or bispectra. Two Python routines are provided to produce the relevant files in linear or logarithmic (base e) binning and are run as:

```
python python/compute_binning_file_linear.py {N_LOG_BINS} {MIN_K} {MAX_K} {OUTPUT_
↪FILE}
python python/compute_binning_file_log.py {N_LINEAR_BINS} {MIN_K} {MAX_K} {OUTPUT_
↪FILE}
```

with the output file saved to the specified destination.

The binning file can also be manually specified as an ASCII file with each line specifying the upper and lower coordinates of each k-bin (in comoving $h\,\mathrm{Mpc}^{-1}$ units). Note that the bins are required to be contiguous (i.e. the upper limit of the n-th bin should equal the lower limit of the $(n+1)$-th bin.

1.3 Computing Configuration-Space Spectra

1.3.1 Overview

For standard usage, we provide simple bash wrappers for HIPSTER, which compute the power spectrum or bispectrum given (a) a set of galaxy or simulation particle positions, and (b) a k-space binning file (with formats discussed in *File Inputs*). For non-periodic data-sets (e.g. galaxy surveys), a set of random particle positions is also required for the power spectrum. (Though the periodic bispectrum also requires random particles, these are created internally.) These are basic wrappers around the C++ and Python scripts, which, for more advanced usage, can be run separately, as discussed in *Advanced Usage*.

The basic structure of the wrappers is as follows.

Power Spectrum:

1) *(Optional)* Subsample the data (and random) files, such that the analysis uses only a fraction of the full dataset. This can be useful to speed up slow pair counts.
2) *(Non-Periodic Only)* Compute the geometry correction function Φ^{-1} and fit it to a smooth model.
3) *(Non-Periodic Only)* Compute the weighted random-random (RR) pair counts.
4) *(Non-Periodic Only)* Compute the weighted data-random (DR) pair counts.
5) Compute the weighted data-data (DD) pair counts.
6) Combine the pair counts and output the power spectrum estimates.

Bispectrum:

1) *(Optional)* Subsample the data file, as above.
2) Compute the weighted pair counts and construct the bispectrum estimate.

Note that, for the power spectrum, steps (2) and (3) depend only on the survey geometry and random particle files, thus, if multiple mocks are being analyzed, they need only be run once. If the relevant option is specified, the wrapper will look for pre-computed survey-correction functions and RR pair counts and only re-create them if they do not exist. For aperiodic surveys with large truncation radii or many random particles, these steps are slow, thus this provides a significant speed boost. For aperiodic surveys, step (6) is done via a Python script, whilst for periodic simulations and bispectra, it takes place in the main C++ code.

1.3.2 Using the HIPSTER Wrapper

The HIPSTER wrappers can be run simply via `./hipster_wrapper.sh`, `./hipster_wrapper_periodic.sh` or `./hipster_wrapper_bispectrum.sh`. The following arguments are required:

- `--dat`: Data file in (x,y,z,weight) co-ordinates.
- `--l_max`: Maximum Legendre multipole. For the power spectrum, HIPSTER currently only uses even multipoles, but all multipoles are used for the bispectrum.
- `--R0`: Pair count truncation radius (see note below).
- `--f_rand`: *(Bispectrum Only)* Ratio of random particles to data points (see note below).
- `--k_bin`: k-space binning file (created in *File Inputs* or user-defined).
- `--ran_DR`: *(Non-Periodic Power Only)* Random file for DR pair counting.
- `--ran_RR`: *(Non-Periodic Power Only)* Random file for RR pair counting (and survey correction function estimation).

A number of additional arguments are possible:

- `--string` (Optional): Identification string for output file names. Default: hipster.
- `--nthreads` (Optional): Number of CPU threads on which to run. Default: 10.
- `--subsample` (Optional): Factor by which to sub-sample the data. Default: 1 (no subsampling)
- `--load_RR`: *(Non-Periodic Power Only)* If set, load previously computed RR pair counts and survey correction functions for a large speed boost. If these are not found, they will be recomputed.
- `-h`: Display the command line options.

Note that, for the power spectrum of non-periodic surveys, two different random catalogs can be provided; one to compute the DR counts and one to compute the RR pair counts. It is usually preferable to use a larger random catalog for the DR pair counts to reduce noise. We recommend around 50x randoms for DR counts and :math:‘sim 10‘x randoms for the RR counts.

As an example, consider computing the isotropic ($\ell = 0$) power spectrum cut at $R_0 = 50h^{-1}\mathrm{Mpc}$ from a single set of galaxies (`galaxies.dat`) and randoms (`randoms.dat`), given some k-binning file `binning.csv`:

```
./hipster_wrapper.sh --dat galaxies.dat --ran_DR randoms.dat --ran_RR randoms.dat -l_
↪max 0 -R0 50 -k_bin binning.csv --nthread 4
```

We've specified that the code should run on 4 cores here.

Similarly, for a simulation with periodic boundary conditions with data-file `data.dat` containing particle positions:

```
./hipster_wrapper_periodic.sh --dat data.dat -l_max 0 -R0 50 -k_bin binning.csv --
↪subsample 2
```

Here, we've set the subsampling to 2, meaning that we'll use (a randomly selected) half of the available data, to get faster computation.

The output of the wrapper is saved in the HIPSTER directory as `output/{STRING}_power_spectrum_n{K_BINS}_l{MAX_L}_R0{R0}.txt` where {STRING} is the identification string described above, {MAX_L} is the maximum Legendre multipole, {K_BINS} is the number of k bins in the binning file and {R0} is the truncation radius. The output file contains power spectrum estimates for each k-bin on a separate line, with the column indicating the (even) Legendre multipole.

For the bispectrum, we can run the following to get the spectrum up to $\ell = 4$ (assuming that the input file `data.dat` has periodic boundary conditions):

```
./hipster_wrapper_bispectrum.sh --dat data.dat -l_max 4 -R0 50 -k_bin binning.csv --
↪subsample 2 --f_rand 3
```

We've chosen to use three times as many randoms as data points, which is usually sufficient. This outputs a bispectrum in the HIPSTER directory as `output/{STRING}_bispectrum_n{K_BINS}_l{MAX_L}_R0{R0}.txt` where {STRING} is the identification string described above, {MAX_L} is the maximum Legendre multipole, {K_BINS} is the number of k bins in the binning file and {R0} is the truncation radius. The output file contains bispectrum estimates for each Legendre multipole in a separate column with the row indicating the k_1, k_2 bins (in $h\,\mathrm{Mpc}^{-1}$ units), using the indexing $n_{\rm collapsed} = in_{\rm bin} + j$ for the i-th k_1 and j-th k_2 bin, with $n_{\rm bin}$ total bins.

Note on Periodicity

For the power spectrum, HIPSTER can be run in either *periodic* or *aperiodic* mode. In the former, we assume the simulation takes the form of a cubic box and measure the angle μ from the Z-axis, as appropriate for most simulations. In the aperiodic case, we measure μ relative to the local line of sight, as appropriate for (non-uniform and non-cubic) surveys. The periodic wrapper runs many times faster than the non-periodic one; this is as a result of many simplifications in the underlying equations. Currently the bispectrum is only supported in periodic mode.

To specify periodicity when using the C++ code alone (without the bash wrapper), we can pass the `-perbox` argument to the C++ code, which must be compiled with the `-DPERIODIC` flag (that can be manually added to the Makefile). The C++ code will crash if this is not specified.

Note on choice of Truncation Radius and Bin Widths

A key hyperparameter of the code is the power spectrum estimation is the *truncation radius* R_0. This is the maximum radius up to which particle counts are computed and sets the computation time of the algorithm (which scales as R_0^3). As discussed in the introductory paper, the effect of R_0 is to convolve the true power spectrum with a window function of characteristic scale $3/R_0$, giving a small bias which is important at low-k, but negligible on small-scales. Considering moments up to $\ell = 4$, we find $R_0 = 50h^{-1}\mathrm{Mpc}$ to be sufficient for measuring $k \gtrsim 0.5h\,\mathrm{Mpc}^{-1}$ and $R_0 = 100h^{-1}\mathrm{Mpc}$ to be sufficient for $k \gtrsim 0.25h\,\mathrm{Mpc}^{-1}$. For fixed truncation error, R_0 scales inversely with the minimum k-bin of interest.

The choice of R_0 also sets the k-binning scale via $\Delta k \gtrsim 3/R_0$ (assuming linear binning). Using narrow k-bins will not give additional information, but lead to the k-bins becoming more correlated.

Note on Random Particles in the Bispectrum

Whilst it is possible to compute the periodic-box bispectrum without any use of random particles (by performing all random particle integrals analytically), this turns out to be very computationally intensive for the bispectrum. As detailed in the second HIPSTER paper, there is one particular term (labelled $\widetilde{DDR}^{II}$) that is difficult to compute, thus we elect to compute it via pair counts with a random catalog which is created internally via HIPSTER. The HIPSTER parameter $f_{\rm rand}$ is the ratio of random particles to galaxies (after subsampling, if applied), and controls this effect. Generally a ratio of order a few gives little sampling noise, but this can be easily experimented with. The runtime of the code scales in proportion to $(1 + f_{\rm rand})$.

Note on Particle Weights

In the input data file to HIPSTER, particle weights can optionally be specified. For aperiodic computations, these can take any form, for example FKP weights for the power spectrum. For periodic surveys, weights are also supported, which are conventionally used to compute polyspectra from multiple-tracer simulations, for example with dark matter and neutrino particles. In this case, each tracer particle carries a weight proportional to its mass. These weights have the additional constraint that, if the particles are unclustered, the weights must not be varying across the survey, though can vary between sets of particles. In other words, HIPSTER can use a collection of different particle sets at once, each set of which has a (different) uniform weight. If unspecified, weights are set to unity.

1.4 Advanced Usage

Here we describe the individual code modules used to compute the power spectrum multipoles, $P_\ell(k)$ or bispectrum multipoles $B_\ell(k_1, k_2)$. For basic usage, the HIPSTER wrappers (*Computing Configuration-Space Spectra*) can be used, which simply run all the relevant packages in sequence connecting the relevant inputs and outputs. If the user requires more flexibility (e.g. to use multiple sets of random particle files or to run particular sections of the code on an HPC cluster) the modules can be run separately, as described below.

1.4.1 Survey Correction Function

Only Relevant for Non-Periodic Power Spectra

An important ingredient in the weighted power spectrum pair counts is the 'survey correction function' Φ, defined as the ratio of ideal and true (unweighted) RR pair counts. For periodic data, this is simply unity, and does not need to be computed. In this package, we compute Φ using the Corrfunc pair counting routines (for aperiodic data) and store the results as quadratic fits to the first few multipoles of Φ^{-1}. Note that a normalization is also performed for later convenience.

This can be computed using the `compute_correction_function.py` script:

```
python python/compute_correction_function.py {RANDOM_PARTICLE_FILE} {OUTFILE} {R_MAX}
↪{N_R_BINS} {N_MU_BINS} {NTHREADS}
```

with inputs:

- {RANDOM_PARTICLE_FILE}: File containing random particles in the survey geometry. Since the correction function is being fit to a smooth function, we can use a relatively small random catalog here.
- {OUTFILE}: Location of output ASCII file. This is automatically read in by the C++ code
- {PERIODIC}: Whether to assume periodic boundary conditions
- {R_MAX}: Radius (in $h^{-1}\mathrm{Mpc}$) up to which to count pairs and fit the correction function. This should be at least as large as the truncation radius (R_0) used for the power spectrum computation. Note that the computation time scales as R_{max}^3.
- {N_R_BINS}: Number of radial bins in the pair counting. We recommend using $\sim 1\,h^{-1}\mathrm{Mpc}$ radial binning here.
- {N_MU_BINS}: Number of angular bins used in the pair counting. We recommend 100 μ bins.
- {N_THREADS}: Number of CPU threads on which to perform pair counts.

This may take some time to compute, but only needs to be computed once for a given survey. Whilst the user can specify the number of radial and angular bins used by the pair counts, this does not have a significant affect on the output function, assuming a moderately fine binning is used.

1.4.2 Computing Weighted Pair Counts

The main functionality of HIPSTER is to compute weighted pair counts across a survey, and combine these into a power spectrum or bispectrum. This is done via the `grid_power.cpp` C++ code, which must be compiled before use.

Compilation

To compile the C++ code, simply run the following in the installation directory:

```
bash clean
make [Periodic=-DPERIODIC] [Bispectrum=-DBISPECTRUM]
```

This creates the `./power` executable in the working directory. If the `Periodic=-DPERIODIC` statement is included, the code is compiled assuming periodic boundary conditions (measuring the angle μ from the z-axis), as appropriate for N-body simulations. If the `Bispectrum=-DBISPECTRUM` statement is added, the code will compute the bispectrum rather than power spectrum.

The Makefile may need to be modified depending on the specific computational system. In particular, Mac users may need to remove the OpenMP references (`-DOPENMP` and `-lgomp`) to ensure compilation occurs. Note that this will force the code to run single threaded.

Running the Code

Configuration space power spectra are computed by estimating DD (and, for non-periodic surveys RR and DR) pair counts (analogous to 2PCF computation) weighted by a k-bin-dependent kernel. For the bispectra, we estimate DDD and DDR counts, via spherical harmonic decomposition and pair counting, with the remaining counts performed semi-analytically.

To compute power spectrum pair counts for two input fields we simply run the `./power` executable, specifying the input parameters on the command line. (Parameters can also be altered in the `power_mod/power_parameters.h` file, but this requires the code to be re-compiled after each modification.). As an example let's compute a set of data-random (DR) counts from input files `galaxy_positions.txt` and `random_positions.txt`:

```
./power -in galaxy_positions.txt -in2 random_positions.txt -binfile binfile.csv -
↪output output/ -out_string DR -max_l 2 -R0 100 -inv_phi_file inv_phi_coefficients.
↪txt -nthread 10
```

This runs in a few seconds to hours (depending on the catalog size and computational resources available).

Analogously, for the bispectrum we can run the `./power` executable, which, in this case, will compute all required pair counts and save the bispectrum:

```
./power -in galaxy_positions.txt -binfile binfile.csv -output output/ -out_string DR -
↪max_l 2 -R0 100 -inv_phi_file inv_phi_coefficients.txt -nthread 10 -f_rand 3
```

The code uses the following main arguments:

- `-in`: First input ASCII file containing space separated (x,y,z,weight) positions of particles in comoving $h^{-1}\mathrm{Mpc}$ units. Note that the weight column is optional, and will be set to unity if not included (see particle-weights-note).
- `-in2`: (*Aperiodic power spectrum only*) Second input ASCII file, with format as above.
- `-binfile`: k-space ASCII binning file, as described above.
- `-output`: Directory in which to house output products. This will be created if not already in existence.
- `-out_string`: String to include in output filename for identification (e.g. RR, DR, DD or a simulation name)
- `-max_l`: Maximum Legendre multipole required (must be even for the power spectrum). Currently, only multipoles up to the tetrahexacontapole ($\ell = 6$) have been implemented for the power spectrum or $\ell = 10$ for the bispectrum, but more can be added if required. For power spectra (but not bispectra), all multipoles are even; if there is a need for odd multipoles, please contact the author and these can be easily added in.
- `-R0`: Truncation radius in Mpc/h units (default: $100\,h^{-1}\mathrm{Mpc}$). See truncation-radius-note.
- `-inv_phi_file`: (*Non-Periodic Power spectrum only*) Location of survey geometry correction file, as produced above.

- `-nthread`: Number of CPU threads to use for the computation.
- `-perbox`: This flag must be set if we require the code to be run with *periodic* boundary conditions, measuring μ from the z-axis. The code must also be compiled with the -DPERIODIC flag.
- `f_rand`: (*Bispectrum only*) Ratio of random particles to galaxies used for DDR counts. See bispectrum-randoms-note.

Note that a full list of command line options to the executable can be shown by running `./power` without any arguments.

For the power spectrum, the code creates the output file `{OUT_STRING}_power_counts_n{N_BINS}_l{MAX_L}_R0{R0}.txt`, specifying the `out_string` parameter, the number of radial bins and the maximum Legendre multipole. Each line of the output file has the (weighted) pair count with the column specifying the Legendre multipole. If the code has been run in periodic mode, it additionally outputs `{OUT_STRING}_analyt_RR_power_counts_n{N_BINS}_l{MAX_L}_R0{R0}.txt` containing the RR counts (computed from a 1-dimensional Hankel transform) and `{OUT_STRING}_power_spectrum_n{N_BINS}_l{MAX_L}_R0{R0}.txt` containing the full power spectrum estimate. This is the main output of the code.

To compute the full power spectra for non-periodic surveys, the data-data (DD), data-random (DR) and random-random (RR) pair counts must be computed. (For periodic surveys, we require only the data-data counts). We do *not* have to use the same sized random catalogs for the DR and RR counts. It is usually preferable to use a larger random catalog for the DR pair counts to reduce noise. We recommend $\sim 50\times$ randoms for DR counts and $\sim 10\times$ for the RR counts. Note that the RR counts are the most computationally intensive procedure, but they only need be computed for each survey once (i.e. when analyzing mock data, the RR pair counts are the same for each mock).

For the bispectrum, the code instead outputs the files `{OUT_STRING}_{TYPE}_n{N_BINS}_l{MAX_L}_{R0}R0.txt` where {TYPE} is `bispectrum`, `DDD_counts`, `DDR_I_counts`, `DDR_II_counts` and `analyt_RRR_counts`, giving the full bispectrum and various components. Each line of the output file has the (weighted) pair count in a combination of k_1, k_2 bins with the column specifying the Legendre multipole. The i-th k_1 and j-th k_2 bin is indexed as $in_{\rm bins} + j$.

1.4.3 Reconstructing the Power Spectrum

Only Relevant for Non-Periodic Surveys

Once the pair counts have been computed, it is straightforward to reconstruct the power spectrum. This can be done via a simple Python script:

```
python python/reconstruct_power.py {DD_FILE} {DR_FILE} {RR_FILE} {GAL_FILE} {N_RAND_
↪RR} {N_RAND_DR} {OUTFILE}
```

where {DD_FILE}, {DR_FILE} and {RR_FILE} give the locations of the DD, DR and RR weighted pair counts, {GAL_FILE} gives the input galaxy file (needed for normalization), {N_RAND_RR} and {N_RAND_DR} give the number of random particles used for RR and DR counts. {PERIODIC} is unity if the code is computed with periodic boundary conditions and zero else. The output power spectrum is given in ASCII format in the specified {OUTFILE}, with the power spectrum estimates for each k-bin on a separate line, with the column indicating the (even) Legendre multipole.

For periodic simulations, the full power spectrum is created inside the C++ code, as described above.

For any queries regarding the code please contact Oliver Philcox.