Manual HBSPS

1 Overview

HBSPS (Hierarchical Bayesian Stellar Population Synthesis) is a stellar population synthesis code that was designed for inferring the IMF from the spectra of unresolved stellar populations. It can be used to infer other properties of these populations as well. The model and several updates to the model are described in Dries et al. (2016), Dries et al. (2017) and Dries et al. (in prep.). Within the current version of the model the following features are included: multiple single stellar population (SSP) fits, flexible IMFs with single and double power law IMF parameterizations of the IMF prior, a multiplicative polynomial for absorbing the continuum mismatch between the data and the model, SSP-based response functions for Mg, Ca, Si, Ti and Na, a Gaussian kernel for modelling the velocity dispersion of the stellar population, an additional global covariance term to take into account systematic uncertainties, and local covariance structures for increasing the error in uncertain spectral regions.

There are two versions of the code: the parameterized version and the full version. In the parameterized version of the code the model spectrum is compared directly to the data on the basis of a χ^2-value and in the full version of the code the IMF is inferred through a linear inversion that allows deviations from the IMF prior if this is demanded by the data. We use the parameterized version of the code to infer the velocity dispersion and the ages and metallicities of the SSPs that we fit to the data. Then we run the full version of the code with fixed SSPs and velocity dispersion to infer the IMF. The reason for using this sampling procedure is that it is much faster than running the full version of the code with free SSP ages and metallicities. For more details of the model we refer the reader to the aforementioned papers. In the remainder of this document we describe how to use HBSPS.

2 Installation

HBSPS is written in Python (version 2.7) and is developed as a module of the cosmological parameter estimation code CosmoSIS (Zuntz et al., 2015). The advantage of combining HBSPS with CosmoSIS is that it allows the used to combine HBSPS with a wide variety of sampling techniques and that is provides a clear interface to the sampler. We mostly use Multinest (Feroz and Hobson, 2008; Feroz et al., 2009, 2013) since we often want to determine the Bayesian evidence but changing to a different sampler is straightforward with CosmoSIS.

Before using HBSPS, make sure to install CosmoSIS and its dependencies. We do not describe the installation of CosmoSIS here, it is extensively described at the CosmoSIS wiki: https://bitbucket.org/joezuntz/cosmosis/wiki/Home. We recommend to read the CosmoSIS wiki and try some of the examples provided with the code before using HBSPS to get a basic understanding of how to use CosmoSIS. Within HBSPS we use the HDF5 binary data format for large data files (i.e. the stellar templates and the response functions). The Python interface to this data format is provided by the package h5py (http://www.h5py.org) which is required before using HBSPS.

After installing CosmoSIS and h5py and downloading the files of HBSPS to a directory of your choice, you are ready to start using HBSPS.

3 Input file

The file containing the input spectrum should be formatted according to the input file expected by `HBSPS`. Currently, `HBSPS` only supports plain text files where the data is ordered into three columns. The first column should represent the wavelength grid of the data, the second column the input spectrum in the same units as the stellar templates and the third column the error spectrum in the same units as the input spectrum. The wavelength grid of the data and the wavelength grid of the stellar templates are expected to be the same so this might require to rebin the stellar templates to the wavelength grid of the data or vice versa. If necessary, `HBSPS` can be changed quite easily to deal with other file formats as well.

4 Stellar templates

`HBSPS` can only be used in combination with a set of stellar templates. These stellar templates may be created by combining a set of isochrones with a stellar library and a spectral interpolator (see e.g. Dries et al., 2016). Within `HBSPS` the stellar templates are assumed to be in a separate directory where there is one HDF5-file for each SSP in the log age-metallicity grid.

Currently we adopt a log age-metallicity grid with $\log t = 9.175 \longrightarrow 10.125$ (i.e. $1.5 \longrightarrow 13.3$ Gyr) and $\Delta \log t = 0.025$ and with $[\mathrm{Fe/H}] = -1.4 \longrightarrow 0.4$ and $\Delta[\mathrm{Fe/H}] = 0.05$. This log age-metallicity grid is hardcoded in `HBSPS` so you need to change the code if you want to use a different age-metallicity grid. The HDF5-files corresponding to the individual SSPs in the age-metallicity grid should at least contain the following datasets:

- `mass`: masses of the stellar templates, make sure that this array is strictly increasing and remove templates that have the same mass.
- `massLow`: lower-mass boundaries of the stellar templates for converting IMF to weights and vice versa.
- `massUp`: upper-mass boundaries of the stellar templates for converting IMF to weights and vice versa.
- `spectra`: spectra of the stellar templates in a two-dimensional array where each of the columns represents a stellar spectrum.
- `luminosity`: luminosity of the stellar templates for calculating mass-to-light ratios.

Make sure to use the same names for the datasets as above or update the file 'SPSbasics.py' to use your own naming convention. The naming convention for the HDF5-files is as follows:

'templates-logT' + $\log t$ + '-FeH' + $[\mathrm{Fe/H}]$ + '.hdf5'

For example, the templates with $\log t = 9.775$ and $[\mathrm{Fe/H}] = 0.1$ are stored in the file 'templates-logT9.775-FeH0.1.hdf5'. To use your own naming convention, simply update the file 'SPSbasics.py'.

5 Setup files

To run `CosmoSIS` in combination with the `HBSPS`-modules 'HBSPSPar.py' (parameterized version of the code) and 'HBSPS.py' (full version of the code) we need to create a `CosmoSIS` parameter file and a `CosmoSIS` values file. We discuss the parameters in these files on the basis of some example files. We first discuss the parameter file of the parameterized version of the code, then the parameter file for the full version of the code and finally we discuss the value files.

5.1 Parameterized version of HBSPS

In Listings 1 we include the file 'examplePar.ini' which give an example of a parameter file for the parameterized version of the code. The Python-module that is required for running this code is 'HBSPSPar.py'.

```
[runtime]
sampler = multinest

[multinest]
max_iterations = 50000
live_points = 600
feedback = True
tolerance = 1.0
update_interval = 200
log_zero = -1e14
multinest_outfile_root = /data/users/dries/research/model-v2/public_code/HBSPS/output/

[output]
filename = /data/users/dries/research/model-v2/public_code/HBSPS/output/testOutputPar
format = text

[pipeline]
modules = HBSPSPar
values = /data/users/dries/research/model-v2/public_code/HBSPS/valuesPar.ini
likelihoods = HBSPSPar
quiet = T
timing = F
debug = F

[HBSPSPar]
file = /data/users/dries/research/model-v2/public_code/HBSPS/HBSPSPar.py
inputSpectrum = testSSP.txt
templatesDir = /data/users/dries/research/model-v2/templates/MIX/stitched/
resFunHDF5 = /data/users/dries/specInt/resFunctions/intResFunctions.hdf5
nSSPs = 2
nSlopes = 1
logbcov = -2.0
polOrder = 10
sampleMg = True
sampleCa = False
sampleSi = False
sampleTi = False
sampleNa = False
```

Listing 1: Example of parameter file for parameterized version code.

The parameter file consists of different sections that may be summarized as follows

- The section [**runtime**] defines some basic facts for running the code. This section should at least specify which sampler should be used.
- For whatever sampler you choose to use in the [**runtime**] section, there should be a section with the name of the specified sampler (in this case we use `Multinest` so we include the section

Table 1: Free parameters in `CosmoSIS` parameter file for running parameterized version of the code.

parameter	description
`file`	Path to the `Python` file 'HBSPSPar.py' that will be used for likelihood evaluation.
`inputSpectrum`	Path to the input spectrum. Note that the input spectrum should be in the format described in Section 3.
`templatesDir`	Path to the directory with the stellar templates. See Section 4 for more information on the format of the HDF5-files with the stellar templates.
`resFunHDF5`	Path to the HDF5-file containing the response functions. If sampleMg, sampleCa, sampleSi, sampleTi and sampleNa are all set to `False` then this parameter does not need to be specified.
`nSSPs`	Number of SSPs included in the fit.
`nSlopes`	Number of IMF slopes in the parameterization of the IMF: 1 = single power law IMF, 2 = double power law IMF.
`logbcov`	Logarithm of the additional global covariance that will be used to inflate the covariance matrix to deal with systematic uncertainties. Note that `logbcov` is parameterized in terms of the median of the original covariance matrix $\mathbf{C}_{\rm D}$ such that the new (diagonal) covariance matrix becomes $\mathbf{C}_{\rm D,new} = \mathbf{C}_{\rm D}\left(1+10^{\rm logbcov}\right)$.
`polOrder`	Order of the multiplicative polynomial.
`sampleMg`	Boolean to indicate if abundance variations in Mg are modelled or not.
`sampleCa`	Boolean to indicate if abundance variations in Ca are modelled or not.
`sampleSi`	Boolean to indicate if abundance variations in Si are modelled or not.
`sampleTi`	Boolean to indicate if abundance variations in Ti are modelled or not.
`sampleNa`	Boolean to indicate if abundance variations in Na are modelled or not.

[**multinest**]). The options that you can specify in this section depend on the sampler. See the wiki of `CosmoSIS` for more information on the different samplers and their options.

- The [**output**] section specifies where to store the output results after the sampling procedure has finished. See `CosmoSIS` wiki for more details.
- The [**pipeline**] section describes the external module(s) that `CosmoSIS` should use for the likelihood evaluation. For every module specified here there should be a separate section in the parameter file. In this case we use the external module HBSPSPar for likelihood evaluations so in the parameter file there should also be a section [**HBSPSPar**]. In this section also the reference to the values file is specified. For more details see `CosmoSIS` wiki.
- The [**HBSPSPar**] section specifies the options for running the parameterized version of the code. The different options that are available for running the parameterized version of the code are specified in Table 1.

5.2 Full version of HBSPS

In Listings 2 we include an example of a parameter file for the full version of the code. This parameter file is very similar to the parameter file of the parameterized version of the code. The module that is used for likelihood evaluations is HBSPS and therefore the parameter file now has a section [**HBSPS**] instead of the section [**HBSPSPar**]. Within this section the options for running the full version of the code are specified. The different options that are available for running the full version of the code are specified in Table 2.

```
[runtime]
sampler = multinest

[multinest]
max_iterations = 50000
live_points = 300
feedback = True
update_interval = 200
log_zero = -1e14
multinest_outfile_root = /data/users/dries/research/model-v2/public_code/HBSPS/output/

[output]
filename = /data/users/dries/research/model-v2/public_code/HBSPS/output/testOutput
format = text

[pipeline]
modules = HBSPS
values = /data/users/dries/research/model-v2/public_code/HBSPS/values.ini
likelihoods = HBSPS
quiet = T
timing = F
debug = F

[HBSPS]
file = /data/users/dries/research/model-v2/public_code/HBSPS/HBSPS.py
inputSpectrum = testSSP.txt
templatesDir = /data/users/dries/research/model-v2/templates/MIX/stitched/
nSSPs = 1
nSlopes = 1
sigma = 150.0
ageIndices = 38
FeHIndices = 28
polOrder = 10
sampleMg = True
sampleCa = False
sampleSi = False
sampleTi = False
sampleNa = False
```

Listing 2: Example of parameter file for full version of the code.

Table 2: Free parameters in CosmoSIS parameter file for running full version of the code.

parameter	description
file	Path to the Python file 'HBSPS.py' that will be used for likelihood evaluation.
inputSpectrum	Path to the input spectrum. Note that the input spectrum should be in the format described in Section 3.
templatesDir	Path to the directory with the stellar templates. See Section 4 for more information on the format of the HDF5-files with the stellar templates.
resFunHDF5	Path to the HDF5-file containing the response functions. If sampleMg, sampleCa, sampleSi, sampleTi and sampleNa are all set to False then this parameter does not need to be specified.
nSSPs	Number of SSPs included in the fit.
nSlopes	Number of IMF slopes in the parameterization of the IMF: 1 = single power law IMF, 2 = double power law IMF.
sigma	Velocity dispersion of the stellar population (from parameterized version of the code).
ageIndices	Age-indices of SSPs in age-metallicity grid to use in the fit (from parameterized version code).
FeHIndices	FeH-indices of SSPs in age-metallicity grid to use in the fit (from parameterized version code).
polOrder	Order of the multiplicative polynomial.
sampleMg	Boolean to indicate if abundance variations in Mg are modelled or not.
sampleCa	Boolean to indicate if abundance variations in Ca are modelled or not.
sampleSi	Boolean to indicate if abundance variations in Si are modelled or not.
sampleTi	Boolean to indicate if abundance variations in Ti are modelled or not.
sampleNa	Boolean to indicate if abundance variations in Na are modelled or not.

5.3 Values file

In addition to a parameter file, both versions of the code require a values file. The values file specifies the free parameters of the model (i.e. those parameters that should be sampled by the sampler) and the (uniform) priors for these parameters. For every free parameter in the model the values file provides three numbers: the first number is the lower boundary for the uniform prior of the parameter, the second number is a starting value for the parameter and the third number is the upper boundary for the uniform prior of the parameter. Obviously the starting value should lie somewhere between the lower and upper boundary of the prior. In Listings 3 we include an example of a values file for the parameterized version of the code with two SSPs included in the fit. The values file for the full version of the code has the same structure (but different parameters). Within CosmoSIS it is also possible to use non-uniform priors for the model parameters. If you want to use these, the CosmoSIS wiki explains how to include them.

5.3.1 IMF normalization parameterized version code

In the parameterized version of the model the normalization of the IMF that determines the contribution of the different SSPs to the total integrated flux is parameterized in the form of the parameter lumFrX. This parameter represent the relative contribution of SSP X to the total

integrated luminosity of the input spectrum. We assume that

$$\sum_{i=1}^{n_{\mathrm{SSPs}}} \mathrm{lumFr}_i = 1.0, \tag{1}$$

which implies that the value of `lumFr` for the last SSP can be derived from this equation. Therefore, there is always one `lumFr`-parameter less than the number of SSPs in the values file of the parameterized version of the code which increases the sampling speed significantly. For example, in Listings 3 two SSPs are included in the fit but only the parameter `lumFr1` is sampled and `lumFr2` is derived from Equation 1. In principle, for more than two SSPs the sum of the relative contributions might become more than one but we include a prior to prevent this (otherwise there would be a degeneracy with the multiplicative polynomial).

```
[parameters]
alpha = 0.0 2.3 4.0
sigma = 140 260 350
FeH1 = -1.4 0.0 0.4
age1 = 1.5 8.0 13.5
lumFr1 = 0.001 0.9 1.0
FeH2 = -1.4 0.0 0.4
age2 = 1.5 8.0 13.5
```

Listing 3: Example of values file for parameterized version of the code.

6 Running the code

Once you have a parameter file and a values file for either the parameterized version or the full version of the code, running it with `CosmoSIS` is straightforward. Suppose the parameter file is 'example.ini' use

```
cosmosis example.ini
```

to run the code. If the sampler that you use allows for parallel sampling, you can run the sampler in parallel with e.g. 8 cores with

```
mpirun -np 8 cosmosis --mpi example.ini
```

Note that this requires a working MPI environment and the `Python` package `mpi4py`.

7 Output files

The minimum output of a sampling procedure is the file specified under `filename` in the section [**output**] of the parameter file. To generate some summary statistics and plots you can use the `postprocess` command that comes with the installation of `CosmoSIS`. Suppose that the output file is 'output.txt' simply use

```
postprocess output.txt
```

to generate the summary statistics and plots. Generating the plots may be time-consuming, if you only want to see the statistics use the option `--noplots`.

Note that if you use `Multinest`, the output file is not a posterior sample but a set of weighted samples. If the parameter `multinest_outfile_root` is set in the [**multinest**] section of the

parameter file, this will generate the `Multinest` output files. Among these files is the file `post_equal_weights.dat` which contains a posterior sample of the sampled parameters.

8 Running the pipeline

In principle, the parameterized and the full version of the code may be used independently[1]. However, the idea is to use the parameterized and the full version of the code together. First the parameterized version of the code is used to determine the velocity dispersion and the ages and metallicities of the SSPs. Then the full version of the code is used to sample the IMF-related parameters and the global covariance parameter `logbcov`.

To automate this process we have developed a pipeline for running `HBSPS`. The only thing that you need to do before running this pipeline is to set the configuration options in the file 'HBSPS.ini'Then run the pipeline with the command

`python HBSPSpipeline.py` *inputSpectrum*

with *inputSpectrum* the name of the file with the input spectrum. The pipeline will then first run the parameterized version of the code, determine the best-fitting velocity dispersion and SSP ages and metallicities from this run and then run the full version of the code with these parameters. Parameter files and values files are created automatically by the pipeline.

In Listings 4 we include an example of the configuration file 'HBSPS.ini'The different options in this file are summarized in Table 3.

```
[templates]
templatesDir = /data/users/dries/specInt/XSL/stitched-data/hdf5FilesSDSS/
nSSPs = 3

[IMFprior]
nSlopes = 1
regScheme = 1 ;regularization scheme, 1 = identity matrix and 2 = 1 / w0**2

[responseFunctions]
sampleMg = True
sampleCa = False
sampleSi = False
sampleTi = False
sampleNa = False
hdf5File = /net/thales/data/users/dries/research/model-v2/data/intResFunctions.hdf5

[polynomial]
order = 10

[covariance]
logbcov = 1.587 ; log value of additional global covariance bcov for parameterized version
    ↪ model (bcov is sampled in full version of the model)

[sampling]
outputDir = /net/thales/data/users/dries/research/model-v2/public_code/HBSPS/output/
nCores = 64
```

[1]The full version of the code only supports fixed ages and metallicities of the SSPs and a fixed velocity dispersion but it is relatively easy to change this in the code.

```
livepoints = 300

[priors]
alpha = 0.0 2.3 4.0
alpha1 = 0.0 2.3 4.0
alpha2 = 0.0 2.3 4.0
norm = -4.0 0.0 2.0
sigma = 140 260 350
age = 1.5 8.0 13.5
FeH = -1.4 0.0 0.4
logbcov = -2.0 0.0 4.0
dex = -0.3 0.0 0.4
```

Listing 4: Example of the configuration file 'HBSPS.ini' that is required for running the `HBSPS` pipeline.

References

M. Dries, S. C. Trager, and L. V. E. Koopmans. A hierarchical Bayesian approach for reconstructing the initial mass function of single stellar populations. *MNRAS*, 463:886–912, November 2016. doi: 10.1093/mnras/stw2049.

F. Feroz and M. P. Hobson. Multimodal nested sampling: an efficient and robust alternative to Markov Chain Monte Carlo methods for astronomical data analyses. *MNRAS*, 384:449–463, February 2008. doi: 10.1111/j.1365-2966.2007.12353.x.

F. Feroz, M. P. Hobson, and M. Bridges. MULTINEST: an efficient and robust Bayesian inference tool for cosmology and particle physics. *MNRAS*, 398:1601–1614, October 2009. doi: 10.1111/j.1365-2966.2009.14548.x.

F. Feroz, M. P. Hobson, E. Cameron, and A. N. Pettitt. Importance Nested Sampling and the MultiNest Algorithm. *ArXiv e-prints*, June 2013.

J. Zuntz, M. Paterno, E. Jennings, D. Rudd, A. Manzotti, S. Dodelson, S. Bridle, S. Sehrish, and J. Kowalkowski. CosmoSIS: Modular cosmological parameter estimation. *Astronomy and Computing*, 12:45–59, September 2015. doi: 10.1016/j.ascom.2015.05.005.

Table 3: Free parameters in 'HBSPS.ini' the configuration file for running the `HBSPS` pipeline.

parameter	description
	[templates]
`templatesDir`	Path to directory with stellar templates.
`nSSPs`	Number of SSPs to use in the fit.
	[IMF prior]
`nSlopes`	Number of slopes in IMF (prior) parameterization, 1 = single power law IMF and 2 = double power law IMF.
`regScheme`	Which regularization scheme should be used in full version code. 1 = identity matrix and 2 = $1/w_0^2$.
	[Response functions]
`sampleMg`, `sampleCa` `sampleSi`, `sampleTi` `sampleNa`	Booleans to indicate if response functions of these elements should be sampled.
`hdf5File`	Path to the HDF5-file with the response functions. If no response functions are used this parameter is not necessary.
	[polynomial]
`order`	Order of the multiplicative polynomial.
	[covariance]
`logbcov`	Logarithm of the additional covariance used in the parameterized version of the code.
	[multinest]
`outputDir`	Directory where the output files will be saved.
`nCores`	Number of CPU's to use in the sampling procedure.
`livepoints`	Number of livepoints used by `Multinest` in the sampling procedure.
	[priors]
`alpha...dex`	Priors and starting values for sampled parameters that are used for creating the values files (see Section 5.3).