User Guide for EMERGE

Benjamin P. Moster
moster@usm.lmu.de
University Observatory, LMU Munich, Germany

EMERGE - Empirical model for the formation of galaxies

Science Paper: http://arxiv.org/abs/1705.05373
Code Website: http://www.usm.lmu.de/emerge
Code Repository: https://github.com/bmoster/emerge
Gitter Community: https://gitter.im/emerge-code

Contents

1 Introduction

Emerge, an Empirical ModEl for the foRmation of GalaxiEs, populates dark matter halo merger trees with galaxies using simple empirical relations between galaxy and halo properties. For each model represented by a set of parameters, it computes a mock universe, which it then compares to observed statistical data to obtain a likelihood. Parameter space can be explored with several advanced stochastic algorithms such as MCMC to find the models that are in agreement with the observations.

The code is written in ANSI C and supports massively parallel computers with distributed memory. Emerge uses an explicit communication model that is implemented with the standardized message passing interface (MPI) and suports hyperthreading via the OpenMP interface. It can be run on essentially all supercomputer systems, including clusters of workstations or individual personal computers.

The main reference for the scientific methods that are used in this code is the paper '*Emerge - An empirical model for the formation of galaxies since $z \sim 10$*' (Moster, Naab & White 2018), and references therein. This document provides additional technical information about the code to allow anyone interested an independent use of the code on parallel machines, and to create mock galaxy catalogues for any dark matter simulation. For further help, visit the code website at usm.lmu.de/emerge or join the Gitter community at gitter.im/emerge-code.

Emerge was written by Benjamin Moster, and is made publicly available under the GNU general public license version 3. This implies that you may freely copy, distribute, or modify the sources, but the copyright for the original code remains with the author. If you find the code useful for your scientific work, we kindly ask you to include a reference to the Emerge paper in all studies that use data obtained with Emerge.

2 Basic Usage

2.1 Obtaining the code

Emerge is hosted on GitHub and can be obtained either by cloning the git repository

```
git clone https://github.com/bmoster/emerge.git
```

or by downloading a compressed tarball of the latest version:

https://github.com/bmoster/emerge/archive/master.zip ($\sim$ **700kB**).

2.2 Compiling

Emerge needs the following non-standard libraries for compilation:

- **MPI** - the Message Passing Interface. There are numerous commercial versions, in addition to excellent open source implementations, e.g.
  MPICH (http://www.mpich.org), or
  Open MPI (http://www.open-mpi.org).
- **GSL** - the GNU scientific library. This open-source package can be obtained at
  http://www.gnu.org/software/gsl, for example. Emerge needs this library for spline interpolations, and for random number generation.

- **HDF5** - the Hierarchical Data Format (version 5.0 or higher, available at https://www.hdfgroup.org). This optional library is only needed when one wants to write snapshot files in HDF5 format. It is possible to compile and use the code without this library.

On most UNIX systems installing these libraries should only take a few steps. Untar the downloaded tarball and change to the newly created directory. Run the configuration script by typing `./configure`, compile the code by typing `make all`, and install the libraries by tyiping `sudo make install`.

After downloading the Emerge code, you will find several files and folders: The `build` folder is initially empty, but will host the object files after compilation. The `data` folder contains all files that store observed data (see Observed data files for more details). The `output` folder is initially empty, but all output data will be written to this folder, see Output files for more details. The `parameterfiles` folder contains an example parameter file (see Parameter file for more details). The `src` folder contains the source code including all c-files and h-files. The `tools` folder contains files that convert Rockstar halo merger trees to the Emerge binary input format, and a sample script file to submit to a computing queue. The `trees` folder is initially empty and can be used to store the input halo merger trees. The files `compile-config.perl` and `config-makefile` are needed for compiation, but do not need to be edited. The file `Template-Config.sh` has to be copied to `Config.sh` and contains all compile-time options. Edit `Config.sh` as needed (see Configuration file for more details). The file `Template-TypeOfSystem.sh` has to be copied to `TypeOfSystem.sh` and contains a number of predefined systems. The appropriate system should be uncommented in `TypeOfSystem.sh`. By following these example systems, further customised system-types can be easily added to the `Makefile`. Slight adjustments will be needed if any of the above libraries is not installed in a standard location on the system. Also, compiler optimisation options may need adjustment, depending on the C-compiler that is used. The executable `emerge` can then be built by typing `make`.

2.3 Running the code

To run Emerge, invoke the executable with a command like

```
mpiexec -np 16 ./emerge emerge.param
```

This will start emerge using 16 computing cores and with parameters as specified in the parameter file `emerge.param` (see Parameter file for more details). The code does not need to be recompiled for a different number of processors, for different parameters, or for different halo marger trees. Emerge can also be run with a single computing core given enough memory for the problem size. To this end, the leading `mpiexec -np 1` can be omitted such that Emerge behaves like a serial code. The MPI library must be installed in any case as it is frequently referenced in the source code. The number of processors can be chosen freely, however it must be a multiple of the number of universes that are computed at the same time. This means that if for example 20 universes are computed in the same step (e.g. for the parallelised MCMC), at least 20 cores are needed, or a multiple thereof.

While Emerge is running, it prints out several log messages that inform about the present steps taken by the code. If Emerge is used interactively in a terminal these messages appear on the screen. They can be redirected to a file by adding `> log.out` at the end of the execution command. If Emerge is used to explore parameter space, or if the problem size is too large to fit on a single computing node, it will typically need to be run on a supercomputer. In

this case, the above command has to be put into a script file that can be submitted to the computing queue, and the output should be diverted to a file. The sample script file `submit` in the `tools` folder shows how to do this for the SuperMuc computer at the LRZ, but will need to be adapted for a different supercomputer.

2.4 Different code modes

Invoking the execution command above will run Emerge for the halo merger trees and parameters specified in the parameter file once, and write the resulting galaxy properties into files in the `output` directory. However, Emerge can also be used to explore parameter space with different algorithms. To this end, an optional flag can be added to the end of the execution command. The three currently implemented parameter space exploration methods are:

Flag	Mode
1	Affine-invariant MCMC ensemble sampler by Goodman & Weare (2010)
2	HYBRID optimisation algorithm by Elson et al. (2007)
3	Parallel-tempering as laid out by Sambridge (2014)

The HYBRID optimisation algorithm for example can thus be started by invoking

```
mpiexec -np 16 ./emerge emerge.param 2
```

which will first initialise a number of walkers in a parameter range as specified in the parameter file, and then evolve the walker positions according to the algorithm and several hyperparameters. The number of walkers, i.e. universes, that are computed during one chain step and the time limit of the run can also be specified in the parameter file. Each set of chains requires a seed number to be specified in the parameter file. This number will be included in the name of the output files, such that they can be distinguished, e.g. for the seed number `12345` the MCMC output file name will be `mcmc12345.000.out`, the HYBRID output file name will be `hybrid12345.000.out`, and the parallel-tempering output file name will be `pt12345.000.out`.

Often a single submission to a queueing system will not provide enough time to explore parameter space as thoroughly as needed. However, all algorithms store the current walker positions in dedicated restart files in the `output` folder, e.g. `walkers.mcmc.12345.out`. Emerge can then be restarted from these walker positions and the chains will then be continued. To resume parameter space exploration with a specific algorithm a `1` has to be added to the flag above, e.g. the parallel-tempering algorithm can be restarted from the latest saved walker positions by invoking

```
mpiexec -np 16 ./emerge emerge.param 31
```

The appropriate restart file `walkers.pt.12345.out` has to exist and have the required number of walkers. Once the code restarts, it will write the walker postions to a new output file, e.g. `pt12345.001.out`. Restarting Emerge in this way is transparent to the code, i.e. after the restart Emerge behaves exactly as if had not been interrupted in the first place. When the code is started with a restart flag, the parameter file is parsed, but only some of the parameters are allowed to be changed, while any changes in the others will be ignored. Which parameters these are is explained in the section Parameter file.

3 Configuration file

Many aspects of the Emerge code are controlled with compile-time options in the Config.sh file rather than run-time options in the parameter file. In this way, highly optimised binaries for a specific model can be generated by the compiler. However, the disadvantage of this approach is that different setups may require different binary executables. If several models are run concurrently, there is hence the danger that Emerge is started or resumed with the 'wrong' binary. While Emerge checks the plausibility of some of the most important code options, this is not done for all of them. To minimise the risk of using the wrong code, it is therefore recommended to produce a separate executable for each simulation that is run. This can be easily achieved by creating an alternative Config.sh file for each setup, and then compiling the source code invoking

```
make CONFIG=NewConfig.sh EXEC=emerge_new
```

The Config.sh file contains a list of all available compile-time code options, with most of them commented out by default. To activate a certain option, it should be commented in, and given the desired value, where appropriate. Whenever one of the Config.sh options described below is changed, a full recompilation of the code is necessary, which can be achieved with a simple `make` command. A manual recompilation of the whole code can be enforced with the command `make clean`, which will erase all object files, followed by `make`.

3.1 Code options

`OPENMPTHREADS=2`

Enables OpenMP support and sets the number of threads per task. If the computing system supports hyperthreading the default value of 2 is recommended.

`RANDOM_NUMBER_TABLE=1000000`

If this option is set, random number tables will be created at the start of the run, for a uniform distribution and a gaussian distribution, respectively. The parameter has to be set to the number of random numbers each of these tables will contain. The random numbers created in this way will be only used for the scatter between intrinsic and observed galaxy properties and for the positions of orphan galaxies within their host haloes, *not* for the parameter space exploration algorithms. The advantage of this option is the increased computation speed, as random numbers do not have to be recomputed for each galaxy in each chain step, and the consistent scatter between steps.

`LONGIDS`

If this option is set, the code assumes that IDs for haloes, descendants, hosts, and forests, are stored as 64-bit long integers. This is only really needed if there are more than ~ 2 billion haloes in the merger trees.

`DISABLE_MEMORY_MANAGER`

By default, Emerge is using a memory manager that handles the dynamic memory allocation. The overhead for the allocation and dellocation of memory during the run is avoided by allocating a large part of the total available memory during initialisation. This section is then filled dynamically by individual memory blocks using

a stack structure and a 64 bit boundary. If this option is set, the memory manager is disabled, and all memory blocks are allocated on the fly, which may reduce the performance of the code.

3.2 Model options

`SFE_MPEAK_ZEVOLV`

By default, the instantaneous conversion efficiency $\epsilon(M) = 2\epsilon_{\rm N}[(M/M_1)^{-\beta}+(M/M_1)^{\gamma}]^{-1}$ does not depend on redshift. Setting this option makes the characteristic halo mass M_1 depend linearly on the scale factor a:

$$M_1(z) = M_0 + M_{\rm z}(1-a) = M_0 + M_{\rm z}\cdot z/(z+1)\ .$$

`SFE_NORM_ZEVOLV`

Setting this option makes the normalisation of the instantaneous conversion efficiency $\epsilon_{\rm N}$ depend linearly on the scale factor a:

$$\epsilon_{\rm N}(z) = \epsilon_0 + \epsilon_{\rm z}(1-a) = \epsilon_0 + \epsilon_{\rm z}\cdot z/(z+1)\ .$$

`SFE_BETA_ZEVOLV`

Setting this option makes the low-mass slope of the instantaneous conversion efficiency β depend linearly on the scale factor a:

$$\beta(z) = \beta_0 + \beta_{\rm z}(1-a) = \beta_0 + \beta_{\rm z}\cdot z/(z+1)\ .$$

`SFE_GAMMA_ZEVOLV`

Setting this option makes the high-mass slope of the instantaneous conversion efficiency γ depend linearly on the scale factor a:

$$\gamma(z) = \gamma_0 + \gamma_{\rm z}(1-a) = \gamma_0 + \gamma_{\rm z}\cdot z/(z+1)\ .$$

`SFE_SAME_SLOPE`

If this option is set, the low and high-mass slopes of the instantaneous conversion efficiency are required to have the same absolute value at any redshift, i.e. $\beta(z) = \gamma(z)$.

`SAT_QUENCH_MASS_DEPENDENT`

By default, satellite galaxies are quenched after a fixed time τ once their halo stops growing, given by $\tau = \tau_0 \cdot t_{\rm dyn}$, where $t_{\rm dyn} = R_{\rm vir}/V_{\rm vir}$ is the dynamical time of the halo. If this option is enabled, the quenching time τ depends on the stellar mass of the galaxy, with low-mass galaxies having a larger quenching time, given by

$$\tau = \tau_0 \cdot t_{\rm dyn} \cdot \max[(m_*/10^{10}{\rm M}_\odot)^{-\tau_{\rm s}}, 1]\ ,$$

where the quenching slope $\tau_{\rm s}$ is a free parameter.

`SAT_STRIP_USE_MSTAR`

By default, a satellite galaxy gets stripped and its stellar mass added to the ICM of the main halo, if the mass of its subhalo drops below a fraction of the peak mass during its history: $M < f_\mathrm{s} \cdot M_\mathrm{peak}$. If this option is set, satellite galaxies are instead stripped if the mass of their subhalo drops below a fraction of their own stellar mass: $M < f_\mathrm{s} \cdot m_*$.

`SAT_SFR_EXP_DECAY`

If this option is selected, the star formation rate of a galaxy declines exponentially on a timescale τ_d after its halo has reached its peak mass:

$$\dot{m}_* = \dot{m}_{*,\mathrm{peak}} \cdot \exp[-\tau_\mathrm{d}\,(t - t_\mathrm{peak})\,/\,t_\mathrm{dyn}]\ .$$

`COMPUTE_ICM`

This option enables the computation of the ICM, and requires the halo merger trees to be organised in forests. A forest contains all merger trees that are in contact through their history, i.e. if a halo is located within the virial radius of another halo at any point in time, their merger trees are part of the same forest. To be able to handle forests, Emerge requires a forest file with the same base file name as the merger tree files and a `.forests` file extension. This file has to store the tree ID for each merger tree (first column), and the corresponding forest ID (second column). Emerge then distributes the merger trees to computing cores such that each forest is loaded on a single core. These additional sorting processes require somewhat more time at the start of Emerge.

`DF_USE_BK08`

If this option is set, Emerge uses the dynamical friction formula by Boylan-Kolchin et al. (2008). Otherwise, the standard formula by Binney & Tremaine (1987) is used.

`ORPHAN_MASSLOSS`

This option only affects orphan galaxies, i.e. galaxies for which the corresponding subhalo has dropped below the resolution limit of the dark matter simulation. If enabled, the halo associated with an orphan galaxy loses mass at the same rate as it did between reaching its peak mass and dropping below the resolution limit. Otherwise, the halo mass of an orphan galaxy stays constant at the mass when the halo got lost.

`ORPHAN_NONRADIAL_INFALL`

By default, the position of orphan galaxies is chosen to be on fixed angular coordinates with respect to the central galaxy, while the radius decreases with the dynamical friction time. If this option is set, the angular coordinates are no longer fixed, but chosen randomly on the unit sphere at each time step.

3.3 Data options

`READ_SMF`

If this option is set, the stellar mass functions are read from the corresponding file in the `data` folder, and included in the computation of the model likelihood.

`READ_FQ`

If this option is set, the fractions of quenched galaxies as a function of stellar mass are read from the corresponding file in the `data` folder, and included in the computation of the model likelihood.

`READ_CSFRD`

If this option is set, the cosmic star formation rate densities are read from the corresponding file in the `data` folder, and included in the computation of the model likelihood.

`READ_SSFR`

If this option is set, the specific star formation rates as a function of stellar mass are read from the corresponding file in the `data` folder, and included in the computation of the model likelihood.

`READ_WP`

If this option is set, the projected galaxy auto-correlation functions are read from the corresponding file in the `data` folder, and included in the computation of the model likelihood.

`GLOBAL_SIGMA_SMF`

If this option is set, a global error between different measurements of the stellar mass function can be specified in the parameter file. This value will then be added in quadrature to the uncertainty of each individual data point.

`GLOBAL_SIGMA_FQ`

If this option is set, a global error between different measurements of the quenched fraction can be specified in the parameter file. This value will then be added in quadrature to the uncertainty of each individual data point.

`GLOBAL_SIGMA_CSFRD`

If this option is set, a global error between different measurements of the cosmic star formation rate density can be specified in the parameter file. This value will then be added in quadrature to the uncertainty of each individual data point.

`GLOBAL_SIGMA_SSFR`

If this option is set, a global error between different measurements of the specific star formation rate can be specified in the parameter file. This value will then be added in quadrature to the uncertainty of each individual data point.

`GLOBAL_SIGMA_WP`

If this option is set, a global error between different measurements of the projected galaxy auto-correlation function can be specified in the parameter file. This value will then be added in quadrature to the uncertainty of each individual data point.

3.4 Model options

`WP_RBINS=20`

To compute the projected galaxy auto-correlation function, first the 3D auto-correlation function is computed. This option can be used to override the default number of 20 radial bins.

`WP_RBINS_INT=1000`

The projected auto-correlation function is integrated from the 3D function using interpolation. This option can be used to override the default number of 1000 interpolation points.

`WP_RMAX=0.1`

The correlation function is computed up to a maximum distance of `WP_RMAX` times the box side length. This option can be used to override the default value of 0.1.

`WP_NLEAF_MIN=4`

The pair counting needed to compute the correlation function uses a kd-tree. While building this tree, a node is split if it contains more than 4 objects. This option can be used to override the default value.

`WP_NODE_WIDTH_MIN=0.01`

The splitting of nodes while building the kd-tree needed for the correlation function is also stopped, if the widest dimension is smaller than 0.01 times the box side length. This option can be used to override the default value.

3.5 Fitting options

The HYBRID optimisation algorithm uses a combination of particle swarm optimisation and simulated annealing, and the step size depends on the likelihood of the walker with respect to the other walkers and the starting likelihood. At each MCMC step i, the mean χ^2 value is computed and compared to the mean starting χ^2 value: $q_i = \langle\chi_i^2\rangle/\langle\chi_1^2\rangle$. Moreover, the χ^2 value

of each individual walker j is compared to the mean χ^2 value: $p_j = \langle\chi_j^2\rangle - \chi_j^2$. The step size of each walker is then chosen as $\sigma = \sigma_0(T) \cdot g(q) \cdot f(p)$, where $\sigma(T)$ is a normalisation that can depend on the chain temperature, and g and f are monotonic functions. The dependence on q is implemented as $g(q) = q^\alpha$, while the dependence on p is implemented as $f(p) = 1 - \beta p$ for $p < 0$ and $f(p) = (p+1)^{-\gamma}$ for $p \geq 0$.

`HYBRID_ALPHA=0.4`

This option can be set to override the default value of 0.4 for the slope α of the function $g(q)$. For a larger value, the step size will depend more strongly on the ratio between current and initial mean likelihood.

`HYBRID_BETA=1.0`

This option can be set to override the default value of 1.0 for the slope β of the function $f(p)$ for $p < 0$. For a larger value the step size will depend more strongly on the difference between the likelihood of the walker and the current mean likelihood.

`HYBRID_GAMMA=1.0`

This option can be set to override the default value of 1.0 for the slope γ of the function $f(p)$ for $p \geq 0$. For a larger value the step size will depend more strongly on the difference between the likelihood of the walker and the current mean likelihood.

The parallel-tempering algorithm runs MCMC chains at different temperatures and can swap the temperature of two walkers if one of them has found a location with a good likelihood. In this way the hot walkers can explore a large part of the parameter space quickly, while cold walkers explore the regions with a high likelihood more thoroughly. At the end of the run, the cold walkers with $T = 1$ sample the posterior distribution. A fraction of the walkers is set to $T = 1$, while the other walkers follow a logarithmic temperature distribution with a maximum temperature $T_{\max}$ that can be specified in the parameter file.

`PT_COLD_FRACTION=0.25`

This option can be set to override the default 25% fraction of cold walkers with $T = 1$.

`PT_TARGET_ACCEPTANCE=0.3`

The step size for new walker postions is adjusted on the fly, such that the resulting acceptance fraction has a specified value. With this option the default value for the desired acceptance fraction of 30% can be overridden.

`PT_STEPS_SCALE_ADJUST=10`

The step size for new walker postions is adjusted is adjusted every 10 steps by default. This option can be set to override this choice.

3.6 Output options

`HDF5_SUPPORT`

This option enables support for the HDF5 file format. Note that it can only be used, if the HDF5 library is installed and properly linked in the `Makefile`. If this option is set, the desired output file format can be specified in the parameter file. Otherwise all output files will bin in ASCII fromat.

`WRITE_GALAXY_CATALOG`

If this option is set, Emerge writes galaxy catalogues to the `output` folder. The output redshifts, stellar mass threshold, number of files per redshift, and output format can be specified in the parameter file for each run. For details on the structure of the galaxy catalogues see Galaxy Catalogues.

`WRITE_HALO_CATALOG`

If this option is set, Emerge writes dark matter halo catalogues to the `output` folder for the same redshifts, stellar mass threshold, number of files per redshift, and output format as the galaxy catalogues. For details on the structure of the halo catalogues see Halo Catalogues.

`WRITE_MAINBRANCH`

If this option is set, Emerge writes the main branch, i.e. only main progenitors and descendants for seleced systems to the `output` folder. These galaxies are seleced at a specific redshift and a stellar or halo mass bin in the parameter file.

4 Parameter file

Many aspects of Emerge do not need to be set before compilation, but can be specified at runtime in a parameter file whenever the code is started. Each parameter is set by first specifying a keyword and then a numerical value or a character string, separated by spaces or tabs. A separate line has to be used for each keyword, and the corresponding value has to appear in the same line. It is possible to use an arbitrary amount of space between keywords, including empty lines. The keywords can appear in any order, but can only appear once. If a necessary keyword has been omitted, or a keyword that is not needed has been set, the code will print an error message, and the parameter file has to be adjusted. All keywords are type-sensitive. Empty lines, or lines with a leading `%` are ignored. Comments can be added after the value for the corresponding keyword.

This section discusses the parameters in detail. All keywords marked with an asterisk will be ignored if Emerge is restarting a fitting algorithm, as these values are loaded from the restart file. In this case it is typically not needed to change the other parameters.

4.1 General Setup

`ModelName P100`

This is the name of the model. A folder with the same name will be created in the `output` folder, and all output files will be written to this folder.

`UniversesInParallel 1`

This specifies the number of universes, i.e. walkers, that will be computed in parallel. If more than one universe are computed in parallel, the initial setup of the halo merger trees will be done for the first universe and then duplicated for the other universes. For a simple generation of mock catalogues, this parameter can be set to `1`. For parameter space exploration, ideally the number of computing cores per universe, i.e. the total number of available cores divided by the number of parallel universes, is chosen such that it fits in the available memory per core. This setup results in as many parallel universes as possible and will typically lead to the best performance. For example, if 400 cores are available, and at least 10 cores are needed to hold one universe, `UniversesInParallel` should be set to 40. Note that the number of cores must be a multiple of `UniversesInParallel`.

`NumFilesInParallel 4`

The code can read halo merger tree files and write output files in parallel. This parameter controls how many files are processed in parallel, and is ideally set to a subdivision of the number of halo merger tree files.

`Verbose 1`

This parameter regulates how much screen output is produced. Currently it can be set to 1 (minimal screen output), and 2 (maximal screen output).

`MaxMemSize 1500`

This parameter specifies the amount of memory in MByte that the dynamic memory manager allocates for each computing core. This should typically be set to the amount of available memory divided by the number of cores. If `MaxMemSize` has been chosen too low, the code will print an error message and inform how much memory is needed for the specified setup. In case the available memory per core is lower than this value, either the number of parallel universes must be decreased, or more memory is needed, i.e. additional computing nodes must be added.

`BufferSize 100`

This parameter specifies the size in Mbyte for the multi-purpose communication buffer used by the code in various parts of the parallel algorithms. It should be large enough to have minimal work-load imbalance losses, though in practice a few to 100 MBytes are sufficient.

4.2 Input files

`TreefileName trees/P100`

This sets the filename of the halo merger tree files to be read in at start-up. The trees can be distributed into several files. In this case, only the base name without the tailing .n number should be specified. The code will recognise the number of files that contain the halo merger trees automatically, and load all of these files accordingly. In the example above, the code will read the files `P100.0`, `P100.1`, and so on, in the `trees` folder.

`SMFfileName data/smf.dat`

This sets the location and the name of the file that contains the observed stellar mass functions. For details on the structure of this file, see Stellar Mass Functions.

`FQfileName data/fq.dat`

This sets the location and the name of the file that contains the observed fractions of quenched galaxies as function of stellar mass and redshift. For details on the structure of this file, see Quenched Fractions.

`CSFRDfileName data/csfrd.dat`

This sets the location and the name of the file that contains the observed cosmic star formation rate densities. For details on the structure of this file, see Cosmic Star Formation Rate Density.

`SSFRfileName data/ssfr.dat`

This sets the location and the name of the file that contains the observed specific star formation rates as function of stellar mass and redshift. For details on the structure of this file, see Specific Star Formation Rates.

`WPfileName data/wp.dat`

This sets the location and the name of the file that contains the observed projected auto-correlation functions. For details on the structure of this file, see Projected correlation functions.

4.3 Cosmological parameters

These parameters specify the cosmology and have to take the same values that have been used to simulated the halo merger trees.

`HubbleParam 0.6781`

This sets the value of the Hubble constant in units of 100 kms $^{-1}$ Mpc $^{-1}$.

`Omega0 0.3080`

This sets the value of the cosmological matter density parameter in units of the critical density at $z = 0$.

`OmegaLambda 0.6781`

This sets the value of the cosmological dark energy density parameter for a cosmological constant in units of the critical density at $z = 0$. A geometrically flat universe has `Omega0 + OmegaLambda = 1`.

`OmegaBaryon 0.0484`

This sets the value of the cosmological baryon density parameter in units of the critical density at $z = 0$. The resulting value for universal baryon fraction is given by `Omega0 / OmegaLambda`.

4.4 Simulation parameters

`BoxSize 67.81`

This parameter specifies the side length of the simulation box from which the dark matter halo merger trees have been extracted. Note that it should have the same unit as the tree files, e.g. if the side length is 100 Mpc, but the unit used for the merger trees is Mpc/h, the value given here should be `67.81` (for $h = 0.6781$).

4.5 Units

These parameters set the internal units. All simulated and observed data that is read in will be converted to these internal units. The output data will be converted to convenient units (see Output files for more details).

`UnitLength_in_Mpc 1.0`

This sets the internal length unit in Mpc.

`UnitTime_in_yr 1.0e9`

This sets the internal time unit in years. The above choice is convenient for cosmological boxes as it sets the time unit to one Gyr.

`UnitMass_in_Msun 1.0e9`

This sets the internal mass unit in solar masses. The above choice is convenient, as the interal unit for the star formation rate is then one solar mass per year.

4.6 Conversion efficiency parameters

These parameters regulate the instantaneous baryon conversion efficiency. If Emerge is run with a mode for which only one model is computed, the exact values specified are used. If Emerge is run with a mode for which walkers with random initial positions are evolved, the values specify the mean of the bin from which the inital values for the walkers are sampled.

`Eff_MassPeak 11.35`

This sets the $z = 0$ value M_0 for the parameter M_1 of the conversion efficiency, which specifies the halo mass close to which the conversion efficiency peaks.

`Eff_Normalisation 0.009`

This sets the $z = 0$ value ϵ_0 for the parameter ϵ_N of the conversion efficiency, which specifies its normalisation.

`Eff_LowMassSlope 3.09`

This sets the $z = 0$ value β_0 for the parameter β of the conversion efficiency, which specifies its low-mass slope.

`Eff_HighMassSlope 1.11`

This sets the $z = 0$ value γ_0 for the parameter γ of the conversion efficiency, which specifies its high-mass slope. This parameter is not present, if the code has been compiled with the option `SFE_SAME_SLOPE`.

`Eff_MassPeak_Z 0.65`

This sets the redshift scaling M_z for the parameter M_1 of the conversion efficiency. This parameter is only present if the code has been compiled with the option `SF↩E_MPEAK_ZEVOLV`.

`Eff_Normalisation_Z 0.60`

This sets the redshift scaling ϵ_z for the parameter ϵ_N of the conversion efficiency. This parameter is only present if the code has been compiled with the option `SF↩E_NORM_ZEVOLV`.

`Eff_LowMassSlope_Z -2.02`

This sets the redshift scaling β_z for the parameter β of the conversion efficiency. This parameter is only present if the code has been compiled with the option `SF↩E_BETA_ZEVOLV`.

`Eff_HighMassSlope_Z 0.0`

This sets the redshift scaling γ_z for the parameter γ of the conversion efficiency. This parameter is only present if the code has been compiled with the option `SF↩E_GAMMA_ZEVOLV`, and the option `SFE_SAME_SLOPE` has not been set.

4.7 Satellite galaxy parameters

These parameters regulate the evolution of satellite galaxies. If Emerge is run with a mode for which only one model is computed, the exact values specified are used. If Emerge is run with a mode for which walkers with random initial positions are evolved, the values specify the mean of the bin from which the inital values for the walkers are sampled.

`Fraction_Escape_ICM 0.56`

This sets the escape fraction for galaxy mergers, i.e. the fraction of the stellar mass of the satellite galaxy that gets ejected into the ICM during a galaxy merger.

`Fraction_Stripping 0.004`

This parameter defines the threshold for the subhalo mass below which satellites are instantly stripped to the ICM. If the option `SAT_STRIP_USE_MSTAR` is set, satellite galaxies are stripped if the mass of their subhalo drops below `Fraction_Stripping` times their own stellar mass, otherwise they are stripped if the mass of their subhalo drops below `Fraction_Stripping` times it maximum mass through history.

`Timescale_Quenching 4.46`

This sets the timescale for satellite quenching. If a subhalo has dropped below its maximum mass through history for more than `Timescale_Quenching` times the halo dynamical time, the SFR in the satellite galaxy is set to zero.

`Timescale_Quenching 0.35`

If the option `SAT_QUENCH_MASS_DEPENDENT` has been set, this parameter defines the slope τ_s of the mass dependence of the quenching timescale

$$\tau = \tau_0 \cdot t_\mathrm{dyn} \cdot \max[(m_*/10^{10}\mathrm{M}_\odot)^{-\tau_\mathrm{s}}, 1] \; .$$

`Decay_Quenching 0.0`

If the option `SAT_SFR_EXP_DECAY` has been set, this parameter sets the timescale for the exponential decay of the SFR after the subhalo falls below its maximum mass. After the quenching timescale τ has elapsed, the SFR is still set to zero.

4.8 Adjustable parameters

`Mstar_min 7.0`

Minimum logarithmic stellar mass in solar masses that is used to compute global statistics such as stellar mass functions, quenched fractions and specific star formation rates. All observational data below this mass are ignored in the fit.

`Mstar_max 12.5`

Maximum logarithmic stellar mass in solar masses that is used to compute global statistics such as stellar mass functions, quenched fractions and specific star formation rates. All observational data above this mass are ignored in the fit.

`Delta_Mstar 0.2`

Stellar mass bin size in dex used to compute global statistics. During the fitting process the model values corresponding to individual oberved data will be interpolated from these bins.

`Observation_Error_0 0.08`

In Emerge, each galaxy has an *intrinsic* stellar mass, i.e. the physically self-consistent stellar mass given by the integral of past star formation and mass loss, and an *observed* stellar mass that is computed from the intrinsic mass and a gaussian distribution for the observational error. The standard deviation of this distribution linearly depends on redshift as $\sigma = \sigma_0 + \sigma_z \cdot z$. The $z = 0$ scatter σ_0 is set by `Observation_Error_0`.

`Observation_Error_z 0.06`

The scatter between *intrinsic* and *observed* stellar mass increases linearly with redshift. The slope σ_z is set by `Observation_Error_z`. The scatter increases only up to redshift of $z = 4$, after which it is held constant. This choice can be changed by modifying the value of `ZMAX_SMFERR` in the `allvals.h` file.

`Global_Sigma_SMF_LZ 0.15`

If the option `GLOBAL_SIGMA_SMF` has been set, this parameter is added in quadrature to the observational uncertainty of each data point of the stellar mass function below $z = 0.1$. This redshift threshold can be changed by modifying the value of `SIG↩ MASMFZTHRESH` in the `allvals.h` file. This increased uncertainty can be sensible if different selected observational data sets are not in agreement with each other given their original uncertainties.

`Global_Sigma_SMF_HZ 0.30`

If the option `GLOBAL_SIGMA_SMF` has been set, this parameter is added in quadrature to the observational uncertainty of each data point of the stellar mass function above $z = 0.1$.

`Global_Sigma_FQ 0.10`

If the option `GLOBAL_SIGMA_FQ` has been set, this parameter is added in quadrature to the observational uncertainty of each data point of the quenched fractions.

`GLOBAL_SIGMA_CSFRD 0.10`

If the option `GLOBAL_SIGMA_CSFRD` has been set, this parameter is added in quadrature to the observational uncertainty of each data point of the cosmic SFR density.

`GLOBAL_SIGMA_SSFR 0.15`

If the option `GLOBAL_SIGMA_SSFR` has been set, this parameter is added in quadrature to the observational uncertainty of each data point of the specific SFRs.

`GLOBAL_SIGMA_WP 0.40`

If the option `GLOBAL_SIGMA_WP` has been set, this parameter is added in quadrature to the observational uncertainty of each data point of the projected galaxy auto-correlation functions.

4.9 Fitting parameters

When the model parameters are fit using any of the fitting algorithms (code modes 1, 2, or 3), the walkers are initialised uniformly in a hyper-cube centred around the values given above. The width of the cube in each dimension is given by the following parameters, which have the same names as the fitted parameters plus an extension of `_Range`. For example, if the fitted parameter has a value of 1.0 and the `_Range` parameter has a value of 0.5, the walkers will be initialised uniformly between 0.75 and 1.25 in this dimension. Note that some parameters are fit in logarithmic space, as indicated below.

`Eff_MassPeak_Range 0.2`

Range of the initial distribution of the parameter `Eff_MassPeak`.

`Eff_Normalisation_Range 0.5`

Range of the initial distribution of the parameter `Eff_Normalisation`. This parameter is fit in logarithmic space, and the range is given in dex. For example, if `Eff_Normalisation` is 0.01 and `Eff_Normalisation_Range` is 0.5, the walkers will be distributed logarithmically between $10^{[\log_{10}(0.01)-0.5/2]} = 0.0056$ and $10^{[\log_{10}(0.01)+0.5/2]} = 0.0178$.

`Eff_LowMassSlope_Range 0.5`

Range of the initial distribution of the parameter `Eff_LowMassSlope`.

`Eff_HighMassSlope_Range 0.2`

Range of the initial distribution of the parameter `Eff_HighMassSlope`.

`Eff_MassPeak_Z_Range 0.2`

Range of the initial distribution of the parameter `Eff_MassPeak_Z`.

`Eff_Normalisation_Z_Range 0.2`

Range of the initial distribution of the parameter `Eff_Normalisation_Z`.

`Eff_LowMassSlope_Z_Range 0.5`

Range of the initial distribution of the parameter `Eff_LowMassSlope_Z`.

`Eff_HighMassSlope_Z_Range 0.2`

Range of the initial distribution of the parameter `Eff_HighMassSlope_Z`.

`Fraction_Escape_ICM_Range 0.3`

Range of the initial distribution of the parameter `Fraction_Escape_ICM` in dex, distributed logarithmically.

`Fraction_Stripping_Range 0.3`

Range of the initial distribution of the parameter `Fraction_Stripping` in dex, distributed logarithmically.

`Timescale_Quenching_Range 0.5`

Range of the initial distribution of the parameter `Timescale_Quenching` in dex, distributed logarithmically.

`Slope_Quenching_Range 0.4`

Range of the initial distribution of the parameter `Slope_Quenching`.

`Decay_Quenching_Range 0.2`

Range of the initial distribution of the parameter `Decay_Quenching`.

4.10 MCMC parameters

For parameter space exploration, three different code modes can be used. The following hyperparameters regulate these algorithms.

`NumberOfMCMCWalkers 8`

This sets the number of walkers used for all parameter space exploration methods. Each walker then moves through parameter space for each step of the algorithm, forming a chain. As all parameter space exploration algorithms are ensemble methods, a large number here is recommended, ideally several hundreds of walkers.

`MCMCScaleParameter 2.0`

This sets the step size with which new walker positions will be suggested. For the affine-invariant MCMC sampler (code mode 1), `MCMCScaleParameter` defines the a parameter and is suggested to be chosen around 2. For the HYBRID method (code mode 2) and the parallel tempering algorithm (code mode 3), this parameter sets the initial gaussian trial step, based on the parameter range divided by two to the power of `MCMCScaleParameter`. For example, if the parameter range `Eff_Mass↩ Peak_Range` is set to 0.2, and `MCMCScaleParameter` is set to 4, a new trial step size will be drawn from a gaussian distribution with a sigma of $0.2/2^4 = 0.0125$ for the dimension corresponding to the parameter `Eff_MassPeak`. A higher value leads to a larger step size, and consequently to a lower acceptance rate. For code modes 1 and 2 this parameter can be adjusted to obtain the desired acceptance rate, while for code mode 3, the step size is adjusted on the fly to get an acceptance rate as defined by the option `PT_TARGET_ACCEPTANCE`.

`ChainTemperature 100.0`

This sets the initial temperature for the chains. If the code is run with mode 1 (MCMC), this parameter is ignored. If the code is run with mode 2 (HYBRID), `ChainTemperature` sets the initial temperature of all chains. For a positive value, this temperature will decrease on a logarithmic timescale. For a negative value, the temperature will be held constant at the absolute value of `ChainTemperature`. If the code is run with mode 3 (parallel tempering), `ChainTemperature` sets the temperature of the hottest chain. A fraction of the walkers (set with the option `PT_COLD_FRACTION`) is initialised at a temperature of $T = 1$, while the remaining walkers will have temperatures spaced logarithmically up to the value of `Chain↩ Temperature`.

`TimeLimit 180.0`

This sets the time limit of the fitting procedure in seconds. The algorithm is stopped, if the next chain would be completed after the time limit. This parameter should be adapted to the computing time limit of the system Emerge is running on, taking into account the time the code needs to set up everything before the fitting starts. In this way, the code can be terminated before the load managing system stops it while running.

`MCMCseed 12345`

This integer value sets the seed for the fitting algorithm. For the same seed value, the algorithm will produce the same walker chains. If the code is run to start a new fitting procedure, a file containing the seed value and a `000` ending will be created, e.g. `mcmc12345.000.out`. If this file already exists, it will be overridden. If fitting is resumed with `1` added to the flag behind the parameter file (e.g. 11 to resume the MCMC fitting), a new file will be created with an incremented number. For example, if the fitting is resumed for the first time, the file `mcmc12345.001.out`, will be created. If the fitting is restarted again, the file `mcmc12345.002.out` will be created, and so on. Changing the seed value will result in completely different chains, such that several ensembles of walkers with different seeds can be run at the same time.

4.11 Output parameters

`NumOutputFiles 1`

This sets the number of files each output product is split into. This applies to galaxy and catalogues at a given redshift, while catalogues at different redshifts will be already separated into different files.

`OutputFormat 1`

This parameter indicates which output format will be used. A value of 1 will produce output in ASCII format, while for a value of 2, the HDF5 format is used. For more details on the specific structure of the output files, see Output files.

`OutputRedshifts 0.1,0.5,1.0`

This comma-separated list specifies the output redshifts for which galaxy and halo catalogues are created. If only a single value is given, only one output catalogue at this redshift will be written. If several values are given, separated by commas, output files will be written for each redshift. Note that there must not be any space between values and commas.

`OutputMassThreshold 7.0`

This sets the minimum stellar mass for which a system is included in the output catalogues. The value is given in logarithmic solar masses.

`MainBranchMasses 10.0,11.0,12.0`

This sets the mean of the mass bin for which galaxy main branches will be written. The parameter `MainBranchMassType` controls if these are given in halo masses or stellar masses. If only a single value is given, only one mass bin will be written. If several values are given, separated by commas, each mass bin will be written. Note that there must not be any space between values and commas.

`MainBranchBinSize 0.1,0.2,0.5`

This sets the mass bin width for which main branches will be written. The number of values given in this list must be the same as for the `MainBranchMasses` list.

`MainBranchMassType 1`

This defines which mass will be used for the selection. For a `0` the mass bin corresponds to the halo mass, while for `1` the mass bin corresponds to the (intrinsic) stellar mass.

`MainBranchRedshift 0.0`

This value sets the redshift at which the systems are selected, i.e. they must have a (halo or stellar) mass that falls into the specified bins at this redshift.

5 Observed data files

In Emerge the properties of simulated dark matter haloes are linked to the observed properties of galaxies. This is achieved by using global statistics, such as the stellar mass function and galaxy clustering. Each time a model with a set of parameters is run, mock observations are computed and compared to observed data. This data is stored in ASCII format in several files in the `data` folder. To fit the model to different data sets, these files can be simply modified to suit any configuration. Typically, they contain blocks of individual data points representing data sets reported by different publications. The following sections provide an overview of the different statistical data, the format of the files, and how to modify them.

5.1 Stellar Mass Functions

The stellar mass functions are stored in the `data/smf.dat` file, and are read and used in the fit only if the option `READ_SMF` has been set. This file starts with a general header line and is followed by several blocks of data, which in turn consist of a header line and individual data. The general header starts with a `#` and then lists the entries that each data block header needs to consist of. These are a `#`, which identifies the line as a block header, the number of individual data points in this block, the minimum and the maximum redshift of the stellar mass function, a correction value for the (universal) initial mass function (IMF) of the data, the Hubble parameter, and a string that identifies the publication from which the data is taken. A typical block header thus has the format

```
# 39 0.0 0.2 0.0 1.0 Li & White 2009,
```

which signifies that the data block has 39 entries (excluding this header), ranges from $z = 0.0$ to $z = 0.2$, has no IMF correction, uses a Hubble parameter of $h = 1.0$, and is from the paper by Li & White (2009).

The IMF can be normalised to any choice, as long as this choice is consistent for all data files. For example, if the model shall be calibrated to a Chabrier IMF, the correction for the data blocks that are given for a Chabrier IMF is 0, while for the data blocks that are given for a Salpeter IMF, the correction value would be 0.24, which is then subtracted from the logarithmic stellar masses. The block header will then look like `# 19 0.6 1.0 0.24 0.7 Santini 2012`. If instead the masses shall be calibrated to a Salpeter IMF, data using a Chabrier IMF will need to have a block header with a IMF correction value of -0.24, while data using a Salpeter IMF would have a correction value of 0.

Each data block consists of several lines that form the stellar mass function in the given redshift interval. These lines consist of the logarithmic stellar mass, the logarithm of the number density per dex, the upper 1σ number density, the lower 1σ number density, and a weight factor. This factor will be multiplied to the χ^2 value of the corresponding data point. For a complete parameter space exploration it should be set to 1 for each data point. However, for testing purposes it can be set to zero to ignore some data, or to a larger value to see how the model changes if one observation is emphasised. A typical data block will look like:

```
# 39 0.0 0.2 0.0 1.0 Li & White 2009
8.05 -1.216 -1.174 -1.263 1
8.15 -1.312 -1.264 -1.366 1
...  [35 more lines] ...
11.45 -4.745 -4.702 -4.794 1
11.55 -5.312 -5.237 -5.403 1
```

New data blocks can be simply added to the end of the file, taking into account that each block needs a header as described above, and that the number of data points in the header correspond to the number of lines in that block. Note that there must not be empty lines before the end of the file.

5.2 Quenched Fractions

The fraction of quenched galaxies as a function of stellar mass are stored in the `data/fq.dat` file, and are read and used in the fit only if the option `READ_FQ` has been set. This file starts

with a general header line and is followed by several blocks of data, which in turn consist of a header line and individual data. The headers have the same format as in the stellar mass function file. The IMF can be normalised with a correction value in the same way as for the stellar mass functions. Note that the IMF choice has to be the same everywhere.

The data blocks consist of several lines that give the quenched fractions in the given redshift interval. These lines consist of the logarithmic stellar mass, the fraction of quenched galaxies, the 1σ uncertainty, and a weight factor that will be multiplied to the χ^2 value of the corresponding data point. A typical data block will look like:

```
# 32 0.2 0.5 0.03 0.7 Muzzin 2013
8.4 0.142 0.157 1
8.5 0.122 0.100 1
...  [28 more lines] ...
11.4 0.943 0.237 1
11.5 0.920 0.285 1
```

New data blocks can be added to the end of the file in the same way as for the SMFs.

5.3 Cosmic Star Formation Rate Density

The cosmic star formation rate densities are stored in the `data/csfrd.dat` file, and are read and used in the fit only if the option `READ_CSFRD` has been set. This file starts with a general header line and is followed by several blocks of data, which in turn consist of a header line and individual data. The general header starts with a `#` and then lists the entries that each data block header needs to consist of. These are a `#`, which identifies the line as a block header, the number of individual data points in this block, the IMF correction value, the Hubble parameter, and a string that identifies the publication from which the data is taken and the wavelength band that has been used. A typical block header thus has the format

```
# 1 0.24 1.0 Robotham & Driver 2011 UV,
```

which signifies that the data block has 1 entry, wil be converted from a Salpeter to a Chabrier IMF, uses a Hubble parameter of $h = 1.0$, and is from the paper by Robotham & Driver (2011). Note that the IMF choice has to be the same everywhere.

The data blocks consist of several lines that give the cosmic star formation rate density at several redshifts. These lines consist of the redshift, the logarithmic cosmic star formation rate density in solar masses per year and cubic Megaparsec, the upper 1σ uncertainty in dex, the lower 1σ uncertainty in dex, and a weight factor that will be multiplied to the χ^2 value of the corresponding data point. A typical data block will look like:

```
# 4 0.0 0.7 Zheng 2006 UV/IR
0.3 -1.7060 0.1261 0.1821 1
0.5 -1.4575 0.0977 0.1255 1
0.7 -1.2438 0.0710 0.0801 1
0.9 -1.2103 0.0776 0.0944 1
```

New data blocks can be simply added to the end of the file, taking into account that each block needs a header as described above, the number of data points in the header corresponds to the number of lines in that block, and there are no empty lines before the end of the file.

5.4 Specific Star Formation Rates

The specific star formation rates are stored in the `data/ssfr.dat` file, and are read and used in the fit only if the option `READ_SSFR` has been set. This file starts with a general header line and is followed by several blocks of data, which in turn consist of a header line and individual data. The headers have the same format as in the cosmic star formation rate density file. The IMF can be normalised with a correction value, but the IMF choice has to be the same everywhere.

The data blocks consist of several lines that give the specific star formation rate as a function of stellar mass and redshift. These lines consist of the redshift, logarithmic stellar mass, the logarithm of the specific star formation rate in yr^{-1}, the upper 1σ uncertainty in dex, the lower 1σ uncertainty in dex, and a weight factor that will be multiplied to the χ^2 value of the corresponding data point. A typical data block will look like:

```
# 60 0.0 0.7 Whitaker 2012
0.25 8.58 -9.31 0.3 0.3 1
0.25 8.69 -9.29 0.3 0.3 1
...  [56 more lines] ...
2.25 11.00 -8.78 0.3 0.3 1
2.25 11.25 -8.90 0.3 0.3 1
```

New data blocks can be simply be added to the end of the file, although there must not be empty lines before the end of the file.

5.5 Projected correlation functions

The projected galaxy auto-correltion function are stored in the `data/wp.dat` file, and are read and used in the fit only if the option `READ_WP` has been set. In the current implementation, all observed correlation functions must be at the same redshift, which is specified at the beginning of the file. The following line is a general header that starts with a `#` and then lists the entries that each data block header needs to consist of. These are a `#`, which identifies the line as a block header, the number of individual data points in this block, the minimum logarithmic stellar mass for this bock, the maximum logarithmic stellar mass for this block, the maximum line-of-sight projection distance $\pi_{\rm max}$, the IMF correction value, the Hubble parameter, and a string that identifies the publication from which the data is taken. A typical block header thus has the format

```
# 25 9.0 9.5 10.0 0.0 1.0 Li 2006,
```

which signifies that the data block has 25 entries (excluding this header), ranges from $\log m_* = 9.0$ to $\log m_* = 9.5$, has a projection distance of $\pi_{\rm max} = 10{\rm Mpc}/h$, no IMF correction, uses a Hubble parameter of $h = 1.0$, and is from the paper by Li et al. (2006).

Each data block consists of several lines that give the projected correlation function in the given stellar mass bin. These lines consist of the projected distance in Mpc/h, the projected correlation function in Mpc/h, the 1σ uncertainty, and a weight factor that will be multiplied to the χ^2 value of the corresponding data point. A typical data block will look like:

```
# 25 9.0 9.5 10.0 0.0 1.0 Li 2006
0.11 411.3 133.7 1
0.14 428.6 94.3 1
```

```
...  [21 more lines] ...
22.54 0.9 1.8 1
28.37 0.5 1.8 1
```

New data blocks can be simply be added to the end of the file. Note that there must not be empty lines before the end of the file, and the redshift for all correlation functions must be the same (given at the start of the file).

6 Halo merger tree files

In addition to observed galaxy data, Emerge relies on simulated dark matter halo merger trees that will be populated with galaxies. The files that contain these merger trees and their locations can be specified in the parameter file with the parameter `TreefileName`. The suggested location for the merger tree files is the `trees` folder. If the trees are located in a single file, `TreefileName` corresponds to the name of this file including folder, e.g. `trees/P100`. If the trees are stored within multiple files, `TreefileName` gives the file base name, which will be appended with the file index, e.g. `trees/P100` corresponds to the files `P100.0`, `P100.1`, `P100.2`, `...`, in the `trees` folder. There is no restriction on the number of files that the trees are split into, although an individual tree cannot be split into different files. The code will automatically detect the number of files. Merger tree files extracted from a simulation with a side length of 100 Mpc can be downloaded from the code website http://www.usm.lmu.de/emerge.

Each merger tree file has the same binary format. It starts with header giving the basic information contained in the file, which is followed by the properties for each halo. The files use the common 'unformatted binary' convention, for which the data of each read or write statement is bracketed by fields that give the length of the data block in bytes. One such 4-byte integer field is stored before and after each data block. This is useful to check the consistency of the file structure, to make sure that the data is read at the correct place, and to skip individual blocks quickly by using the length information of the block size fields to fast forward in the file without actually having to read the data. The sequence of blocks in a file has the following format:

#	Block name	Type	Bit size	Elements	Comment
0	`Ntrees`	int	32	1	Number of merger trees in this file
1	`Nhalos`	int	32	`Ntrees`	Number of haloes in each merger tree
2	`TreeID`	IDtype	32 or 64	`Ntrees`	ID of each tree
3	`Halo`	struct	480 or 576	`Nhalostot`	Properties of each halo in this file

The data type `IDtype` is `unsigned int` by default. However, for very large simulations the total number of haloes can be larger than 2 billion. In this case `IDtype` needs to be able to cover this range, and can be set to the type `unsigned long long int` if the `LONGIDS` option is enabled. In total the `Halo` block contains `Nhalostot = Sum_i Nhalos_i` elements, i.e. the sum of the haloes in all trees. The halo properties are stored in the form of a structure for each halo. This structure has the following format:

#	Block name	Type	Bit size	Example	Comment
0	`haloid`	IDtype	32 or 64	123456788	ID of the halo (unique across simulation and snapshots)
1	`descid`	IDtype	32 or 64	123456789	ID of the halo's descendant in the next snapshot
2	`upid`	IDtype	32 or 64	223456788	ID of the most massive host halo
3	`np`	unsigned short	16	5	Number of progenitors in the previous snapshot
4	`mmp`	unsigned short	16	1	Flag indicating if the halo is the most massive progenitor (1) or not (0)
5	`scale`	float	32	0.99	Scale factor a of the halo
6	`mvir`	float	32	1.1e12	Virial mass in $M_\odot/h$
7	`rvir`	float	32	220.5	Virial radius in kpc/h
8	`c`	float	32	11.5	Halo concentration ($R_{\rm vir}/R_{\rm s}$)
9	`lambda`	float	32	0.033	Halo spin parameter
10	`pos[3]`	float (x3)	96	3.1, 2.4, 7.1	Halo position in Mpc/h (x, y, z)
11	`vel[3]`	float (x3)	96	302, -19, 212	Halo velocity in km/s (v_x, v_y, v_z)

The halo IDs may not be smaller than 1. If a halo has no descendant (e.g. for $z = 0$), the descendant ID `descid` should be 0. Similarly, if a halo is a host halo, the ID of the most massive host halo `upid` should be 0. Both the position and the velocity are stored as 3D arrays.

For some applications, for example to compute the amount of stellar mass in the ICM, it is necessary that the trees are ordered within forests, such that all merger trees that reference the same host ID need to be grouped together. In practice, this is achieved by providing a separate file in the same folder as the tree files, which lists for each tree ID (left column) the corresponding forest ID (right column). The name of the file has to consist of the tree file base name and a `.forests` extension, e.g. `P100.forests`.

While merger trees that have been created with any software can easily be converted to this format, a code that converts merger trees created with the Rockstar / consistent-trees software into the Emerge format is provided in the `tools` folder. Compiling Emerge should automatically also compile the `convert_CT_to_emerge.c` file and create the `convert_CT_to_emerge.c` executable, which converts a single ASCII tree file created with consistent-trees. Before compiling, it should be verified that the column numbers specified at the beginning of this file correspond to the correct halo properties for the adopted version of consistent-trees. The executable must be called with the input file and output file as arguments. If consistent-trees has divided the merger trees into several files (boxes) with the names `tree_0_0_0.dat`, `tree_0_0_1.dat`, ..., the shell script `consistent_trees_to_emerge.sh` can convert all files to the appropriate Emerge files. It needs to be called with the output base name and the number of box divisions as arguments. The command `consistent_trees_to_emerge.sh P100 2` converts all 8 files from `tree_0_0_0.dat` to `tree_1_1_1.dat` into the files `P100.0` to `P100.7`.

7 Output files

At the end of each run, Emerge can write different data products to files. This includes the model predictions for the statistics that have been used in the fit, mock galaxy catalogues, halo catalogues, and the main branch evolution of each system. In future releases, Emerge will also be able to write the full galaxy merger trees, and support lightcone output. All files can either be written in standard ASCII format, or, if the option `HDF5_SUPPORT` has been set and the proper libraries have been installed, the HDF5 format can be used. The output format can be selected before each run in the parameter file with the `OutputFormat` parameter. All files will be written to the `output` folder, within the subfolder corresponding to the parameter `Model↩ Name`. Emerge supports parallel output by distributing the catalogues into several files, each written by a group of processors, which may simplify the handling of very large simulations.

7.1 Global statistics

The global statistics that have been used to fit the model will be written to individual files in the `statistics` subfolder for ASCII output, and to individual datasets within the file `statistics.h5` for HDF5 outout. For each seperate universe that has been run in parallel as specified by the `UniversesInParallel` parameter, separate statistics will be written. For the ASCII format, the index of the universe is given in the file names, so that the `*.0.out` files correspond to the first universe. For the HDF5 output, the `statistics.h5` file contains a group that holds all data for each universe.

7.1.1 Stellar Mass Functions

For the ASCII format, the observed stellar mass functions, converted to the globally adopted IMF and Hubble parameter, are stored in the `smfobs.*.out` files (one for each universe). These files contain one data block for each observational data set, separated by empty lines. Each data block is preceded by a header that starts with a `#`, followed by the number of data points, the redshift bin edges and the publication reference. Each line in the data block contains the logarithmic stellar mass, the observed logarithmic number density, the observational uncertainty, and the model prediction for this data point. The `smfmod.*.out` files contain the model stellar mass function at each redshift for the stellar mass bins selected in the parameter file. Each line starts with the logarithmic stellar mass bin followed by a model value for the stellar mass function at each redshift.

For the HDF5 format, the stellar mass functions are stored in the `SMF` group. The `Sets` HDF5 compound gives an overview of the different data sets. It has the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	`Ndata`	int	32	37	The number of data points in this data set
1	`Offset`	int	32	604	The offset with respect to the first data set
2	`Redshift_min`	float	32	0.2	The minimum redshift
3	`Redshift_max`	float	32	0.5	The maximum redshift
4	`Tag`	string	400	Muzzin 2013	A tag referencing the publication

All of the observed data sets are located in the `data` subgroup, and are organised in a HDF5 compound for each data set. The name of each compound reflects the publication reference and the redshift bins. The observational data compounds have the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	`Stellar_mass`	float	32	8.39761	The logarithmic stellar mass in solar masses
1	`Phi_observed`	float	32	-1.6404	The logarithm of the observed number of galaxies per cubic Megaparsec and dex
2	`Sigma_observed`	float	32	0.31167	The observational uncertainty in dex
3	`Mean_ScaleFactor`	float	32	0.75	The mean scale factor a of the redshift bin
4	`Phi_model`	float	32	-1.6724	The logarithm of the model prediction for the number density of galaxies
5	`Sigma_model`	float	32	0.00661	The model uncertainty in dex resulting from Poisson noise

The model stellar mass function at each redshift is stored in the table `Model`. The first row contains the scale factors that correspond to each column, while the first column contains the logarithmic stellar masses that correspond to each row. The remaining table entries give the logarithm of the stellar mass function (in $\mathrm{dex}^{-1}\mathrm{Mpc}^{-3}$) at the given stellar mass and scale factor.

7.1.2 Quenched Fractions

For the ASCII format, the observed quenched fractions, converted to the globally adopted IMF and Hubble parameter, are stored in the `fqobs.*.out` files (one for each universe). These files contain one data block for each observational data set, separated by empty lines. Each data block is preceded by a header that starts with a `#`, followed by the number of data points, the redshift bin edges and the publication reference. Each line in the data block contains the logarithmic stellar mass, the observed quenched fraction, the observational uncertainty, and the model prediction for this data point. The `fqmod.*.out` files contain the model quenched fractions for the stellar mass bins selected in the parameter file. Each line starts with the logarithmic stellar mass bin followed by a model value for the quenched fraction at each redshift.

For the HDF5 format, the quenched fractions are stored in the `FQ` group. The `Sets` HDF5 compound gives an overview of the different data sets. It has the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	`Ndata`	int	32	32	The number of data points in this data set
1	`Offset`	int	32	266	The offset with respect to the first data set
2	`Redshift_min`	float	32	0.2	The minimum redshift
3	`Redshift_max`	float	32	0.5	The maximum redshift
4	`Tag`	string	400	Muzzin 2013	A tag referencing the publication

All of the observed data sets are located in the `data` subgroup, and are organised in a HDF5 compound for each data set. The name of each compound reflects the publication reference and the redshift bins. The observational data compounds have the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	`Stellar_mass`	float	32	10.5976	The logarithmic stellar mass in solar masses
1	`Fq_observed`	float	32	0.50100	The observed fraction of quenched galaxies
2	`Sigma_observed`	float	32	0.11413	The observational uncertainty
3	`Mean_ScaleFactor`	float	32	0.75	The mean scale factor a of the redshift bin
4	`Fq_model`	float	32	0.47875	The model prediction for the fraction of quenched galaxies
5	`Sigma_model`	float	32	0.01463	The model uncertainty resulting from Poisson noise

The model quenched fractions at each redshift are stored in the table `Model`. The first row contains the scale factors that correspond to each column, while the first column contains the logarithmic stellar masses that correspond to each row. The remaining table entries give the fraction of quenched galaxies at the given stellar mass and scale factor.

7.1.3 Cosmic Star Formation Rate Density

For the ASCII format, the observed cosmic star formation rate densities, converted to the globally adopted IMF and Hubble parameter, are stored in the `csfrdobs.*.out` files (one for each universe). These files contain one data block for each observational data set, separated by empty lines. Each data block is preceded by a header that starts with a `#`, followed by the number of data points and the publication reference. Each line in the data block contains the redshift, the observed logarithmic cosmic star formation rate density in solar masses per year and cubic Megaparsec, the observational uncertainty, and the model prediction for this data point. The `csfrdmod.*.out` files contain the redshift (first column) and the corresponding model cosmic star formation rate density (second column).

For the HDF5 format, the cosmic star formation rate densities are stored in the `CSFRD` group. The `Sets` HDF5 compound gives an overview of the data sets. It has the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	`Ndata`	int	32	4	The number of data points in this data set
1	`Offset`	int	32	106	The offset with respect to the first data set
2	`Tag`	string	400	Duncan 2014 UV	A tag referencing the publication

All of the observed data sets are located in the `data` subgroup, and are organised in a HDF5 compound for each data set. The name of each compound reflects the publication reference and the redshift bins. The observational data compounds have the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	Redshift	float	32	4.0	The redshift
1	Csfrd_observed	float	32	-1.4438	The observed cosmic star formation rate density in solar masses per year and cubic Megaparsec
2	Sigma_observed	float	32	0.12806	The observational uncertainty
3	Csfrd_model	float	32	-0.9831	The model prediction for the cosmic star formation rate density

The model predictions are stored in the table Model. The first column contains the redshift, and the second column contains the corresponding cosmic star formation rate density.

7.1.4 Specific Star Formation Rates

For the ASCII format, the observed specific star formation rates, converted to the globally adopted IMF and Hubble parameter, are stored in the ssfrobs.*.out files (one for each universe). These files contain one data block for each observational data set, separated by empty lines. Each data block is preceded by a header that starts with a #, followed by the number of data points and the publication reference. Each line in the data block contains the redshift, the logarithmic stellar mass, the observed specific star formation rate, the observational uncertainty, and the model prediction for this data point. The ssfrmod.*.out files contain the model specific star formation rates at each redshift for the stellar mass bins selected in the parameter file. Each line starts with the logarithmic stellar mass bin followed by a model value for the specific star formation rate at each redshift.

For the HDF5 format, the stellar mass functions are stored in the SSFR group. The Sets HDF5 compound gives an overview of the different data sets. It has the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	Ndata	int	32	46	The number of data points in this data set
1	Offset	int	32	24	The offset with respect to the first data set
2	Tag	string	400	Noeske 2007	A tag referencing the publication

All of the observed data sets are located in the data subgroup, and are organised in a HDF5 compound for each data set. The name of each compound reflects the publication reference and the redshift bins. The observational data compounds have the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	Redshift	float	32	0.775	The redshift
1	Ssfr_observed	float	32	-9.480	The logarithm of the observed specific star formation rate in yr^{-1}
2	Sigma_observed	float	32	0.335	The observational uncertainty in dex
3	Stellar_mass	float	32	10.397	The logarithmic stellar mass in solar masses

#	HDF5 name	Type	Bit size	Example	Comment
4	Ssfr_model	float	32	-9.597	The logarithm of the model prediction for the specific star formation rate
5	Sigma_model	float	32	0.012	The model uncertainty in dex resulting from Poisson noise

The model specific star formation rates at each stellar mass and redshift are stored in the table Model. The first row contains the logarithmic stellar masses that correspond to each column, while the first column contains the redshifts that correspond to each row. The remaining table entries give the logarithm of the specific star formation rate (in yr^{-1}) at the given stellar mass and scale factor.

7.1.5 Galaxy Clustering

For the ASCII format, the observed projected correlation functions, converted to the globally adopted IMF and Hubble parameter, are stored in the ssfrobs.*.out files (one for each universe). These files contain one data block for each observational data set, separated by empty lines. Each data block is preceded by a header that starts with a #, followed by the number of data points, the stellar mass bin edges, maximum line-of-sight projection distance $\pi_{\max}$, and the publication reference. Each line in the data block contains the projected distance, the observed projected correlation function, the observational uncertainty, the model prediction for this data point, and the model error, based on Poisson noise. The ximod.*.out files contain the model predictions for the 3D correlation functions in the same stellar mass bins as the observations. The first line starts with a # followed by all stellar mass bins. The subsequent lines contain the 3D distance, the model 3D correlation function, and the model uncertainty for each stellar mass bin.

For the HDF5 format, the clustering data are stored in the Clustering group. The Sets HDF5 compound gives an overview of the different data sets. It has the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	Ndata	int	32	27	The number of data points in this data set
1	Offset	int	32	50	The offset with respect to the first data set
2	Minimum_Mass	float	32	10.3374	The minimum logarithmic stellar mass
3	Maximum_Mass	float	32	10.8374	The maximum logarithmic stellar mass
4	Pi_max	float	32	14.7471	The maximum projected distance in Megaparsecs
5	Tag	string	400	Li 2006	A tag referencing the publication

All of the observed data sets are located in the data subgroup, and are organised in a HDF5 compound for each data set. The name of each compound reflects the publication reference and the stellar mass bins. The observational data compounds have the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	Radius	float	32	0.3391	The projected radius in Megaparsecs
1	Wp_observed	float	32	322.66	The projected galaxy auto-correltion function in Megaparsecs
2	Sigma_observed	float	32	135.46	The observational uncertainty in Megaparsecs
3	Wp_model	float	32	303.63	The model prediction for the projected galaxy auto-correltion function
4	Sigma_model	float	32	16.79	The model uncertainty in Megaparsecs resulting from Poisson noise

The model 3D auto-correlation functions for each stellar mass bin are stored in the Model↩ _3D subgroup, and are organised in a HDF5 compound for each data set. The name of each compound reflects the publication reference and the stellar mass bins. The data compounds have the following structure:

#	HDF5 name	Type	Bit size	Example	Comment
0	Radius	float	32	1.0015	The 3D radius in Megaparsecs
1	Xi	float	32	41.049	The 3D galaxy auto-correltion function
2	Sigma_xi	float	32	1.9703	The model uncertainty

7.1.6 Model likelihoods

To check how well the model fits the observed data, Emerge writes out the calculated χ^2 values. For the ASCII output, the values are stored in the chi2.out file. The header line starts with a # and then lists all global statistics that have been used in the fit. The following lines then list the χ^2 values for each universe. Each line starts with the universe index, and then gives the total χ^2 value, followed by the χ^2 values for the individual statistics. For the HDF5 output, the χ^2 values are stored in the Chi2 table in the statistics.h5 file for each universe.

7.1.7 Adopted model parameters

The model parameters that have been used to run each universe are stored for the HDF5 output format in the statistics.h5 file. The HDF5 compound Model_Parameters within each universe group contains all parameters as double (64 bit) values.

7.2 Galaxy Catalogues

If Emerge is not called with a flag behind the parameter file, which corresponds to the code mode 0, mock galaxy catalogue are written out as long as the option WRITE_GALAXY_CATA↩ LOG has been set. The output format (ASCII or HDF5) can be chosen with the parameter OutputFormat in the parameter file. For each output redshift that has been specified with the

parameter `OutputRedshifts`, a file containing all galaxies in the simulation box that are more massive than the threshold `OutputMassThreshold` will be written written. The name of each file begins with `galaxies.S`, followed by the snapshot number that corresponds to its redshift, and a file extension, e.g. `galaxies.S90.out` for snapshot 90 and ASCII format. Note that the mock catalogue is only written for the first universe if `UniversesInParallel` is larger than 1.

All ASCII files have a header starting with a `#` which lists the contents of each column, as well as several key parameters that have been chosen to create the catalogue. The table following this header then contains the galaxy and halo properties of one system in each line. The HDF5 files contain a compound dataset that lists the properties of each system. The key parameters are stored as attributes of this dataset:

#	HDF5 name	Type	Bit size	Example	Comment
0	`Scale Factor`	float	32	1.0	Scale factor of the catalogue
1	`Box Size`	double	64	100.0	Box side length in Mpc
2	`Hubble Parameter`	double	64	0.6781	Hubble parameter $h = H/(100\mathrm{kms}^{-1}\mathrm{Mpc}^{-1})$
3	`Omega_0`	double	64	0.308	Matter density parameter.
4	`Omega_Lambda`	double	64	0.692	Dark energy density parameter
5	`Omega_Baryon_0`	double	64	0.0484	Baryon density parameter
6	`Minimum Stellar Mass`	double	64	7.0	Logarithm of the minimum stellar mass in $\mathrm{M}_\odot$

Each system in the dataset has several properties:

#	HDF5 name	Type	Bit size	Example	Comment
0	`Halo_mass`	float	32	12.41	Virial mass of the halo in log $\mathrm{M}_\odot$
1	`Halo_growth_rate`	float	32	471.76	Growth rate of the halo in $\mathrm{M}_\odot\mathrm{yr}^{-1}$
2	`Halo_mass_peak`	float	32	12.41	Peak virial mass over history in log $\mathrm{M}_\odot$
3	`Halo_mass_host`	float	32	12.41	Virial mass of the host halo in log $\mathrm{M}_\odot$
4	`Scale_peak_mass`	float	32	1.00	Scale factor when peak virial mass was reached
5	`Scale_half_mass`	float	32	0.69	Scale factor when half of the current virial mass was reached
6	`Halo_radius`	float	32	373.81	Virial radius in kpc
7	`Concentration`	float	32	4.41	Concentration parameter ($c = R_{\mathrm{vir}}/R_{\mathrm{s}}$)

#	HDF5 name	Type	Bit size	Example	Comment
8	`Halo_spin`	float	32	0.038	Halo spin parameter λ
9	`Stellar_mass`	float	32	10.52	Stellar mass of the galaxy in log $M_\odot$
10	`SFR`	float	32	0.72	Star formation rate of the galaxy in $M_\odot yr^{-1}$
11	`Intra_cluster_mass`	float	32	9.051	Stellar mass of the halo (excluding satellites) in log $M_\odot$
12	`Stellar_mass_obs`	float	32	10.71	Observed stellar mass of the galaxy in log $M_\odot$
13	`SFR_obs`	float	32	0.59	Observed star formation rate of the galaxy in $M_\odot yr^{-1}$
14	`X_pos`	float	32	8.68	X-Position of the system in the simulation box in Mpc
15	`Y_pos`	float	32	86.19	Y-Position of the system in the simulation box in Mpc
16	`Z_pos`	float	32	7.81	Z-Position of the system in the simulation box in Mpc
17	`X_vel`	float	32	-213.27	X-Velocity of the system in kms^{-1}
18	`Y_vel`	float	32	132.13	Y-Velocity of the system in kms^{-1}
19	`Z_vel`	float	32	-343.57	Z-Velocity of the system in kms^{-1}
20	`Type`	unsigned short	16	0	System type (0: central, 1: satellite, 2: orphan)
21	`Halo_ID`	long long int	64	12345	ID of the halo
22	`Desc_ID`	long long int	64	-1	ID of the descendant (-1 if none)
23	`Up_ID`	long long int	64	-1	ID of the host halo (-1 if halo is a host halo itself)

If the parameter `NumOutputFiles` is set to a number larger than 1, each mock catalogue at a fixed redshift will be split into several files. The file index will be appended to the filename before the file extension, e.g. `galaxies.S90.0.h5`, `galaxies.S90.1.h5`, ...

7.3 Halo Catalogues

Similar to the galaxy catalogue output, Emerge can create halo catalogues if called without a flag (code mode 0), and the option `WRITE_HALO_CATALOG` has been set. These halo catalogues include the subhaloes that correspond to orphan galaxies, which can be useful if the catalogues are used to link galaxies to haloes, e.g. for subhalo abundance matching. The output format (ASCII or HDF5) is again chosen with the parameter `OutputFormat` and the output redshifts are specified with `OutputRedshifts`. The files start with `haloes.S`, followed by the snapshot number and a file extension, e.g. `haloes.S90.h5`. The mock catalogue is again only written for the first universe. The ASCII files have a similar header as the galaxy catalogue files, and are followed by a table that lists the properties of one halo in each line. For the HDF5 files, the halo properties are stored in a compound dataset, with key parameters stored as attributes. The following halo properties are written:

#	HDF5 name	Type	Bit size	Example	Comment
0	`Halo_mass`	float	32	12.41	Virial mass of the halo in log $M_\odot$
1	`Halo_growth_rate`	float	32	471.76	Growth rate of the halo in $M_\odot yr^{-1}$
2	`Halo_mass_peak`	float	32	12.41	Peak virial mass over history in log $M_\odot$
3	`Halo_mass_host`	float	32	12.41	Virial mass of the host halo in log $M_\odot$
4	`Scale_peak_mass`	float	32	1.00	Scale factor when peak virial mass was reached
5	`Scale_half_mass`	float	32	0.69	Scale factor when half of the current virial mass was reached
6	`Halo_radius`	float	32	373.81	Virial radius in kpc
7	`Concentration`	float	32	4.41	Concentration parameter ($c = R_{\rm vir}/R_{\rm s}$)
8	`Halo_spin`	float	32	0.038	Halo spin parameter λ
9	`X_pos`	float	32	8.68	X-Position of the system in the simulation box in Mpc
10	`Y_pos`	float	32	86.19	Y-Position of the system in the simulation box in Mpc
11	`Z_pos`	float	32	7.81	Z-Position of the system in the simulation box in Mpc
12	`X_vel`	float	32	-213.27	X-Velocity of the system in kms^{-1}
13	`Y_vel`	float	32	132.13	Y-Velocity of the system in kms^{-1}
14	`Z_vel`	float	32	-343.57	Z-Velocity of the system in kms^{-1}

#	HDF5 name	Type	Bit size	Example	Comment
15	Type	unsigned short	16	0	System type (0: central, 1: satellite, 2: orphan)
16	Halo_ID	long long int	64	12345	ID of the halo
17	Desc_ID	long long int	64	-1	ID of the descendant (-1 if none)
18	Up_ID	long long int	64	-1	ID of the host halo (-1 if halo is a host halo itself)

To generate halo catalogues with haloes that correspond to orphan galaxies, it is recommended to set the normalisation of the conversion efficiency (ϵ_0 and ϵ_z), and the stripping parameter f_s to zero. In this case, the stellar mass in each halo will be zero, and the dynamical friction times will be based only on the halo mass.

7.4 Main Branches

To track galaxies and haloes through cosmic time, Emerge can write the properties of systems on the main branch of the merger tree. For this, Emerge must be called without a flag (code mode 0), and the option `WRITE_MAINBRANCH` has to be set. The output format (ASCII or HDF5) is chosen with the parameter `OutputFormat`. Galaxies are selected at a specific redshift with `MainBranchRedshift`, and in several mass bins with `MainBranchMasses` and `MainBranchBinSize`, while `MainBranchMassType` specifies which mass is used for this selection (0: halo mass, 1: stellar mass). For each galaxy that fulfils this criterion, all main progenitors and descendants are written as a data block. The file name starts with `mainbranches.S`, followed by the snapshot number that corresponds the selection redshift, and a file extension, e.g. `mainbranches.S90.out` for the main branches that have been selected to fulfil the mass criterion at snapshot 90.

The ASCII files have a header block starting with a `#`, which lists the contents of each column, as well as several key parameters, such as the selection mass bins and the box size. After the header block, each main branch is written as a data block, separated by empty lines. Each data block starts with an index given by the mass bin and the main branch number, e.g. `#000_0000000` corresponds to the first mass bin and the first main branch in that bin. After this index, a table lists the properties of each galaxy on the main branch of the merger tree.

The HDF5 files include an attribute `Selection_Redshift` listing the redshift for which galaxies have been selected. The main branches are stored within HDF5 groups for each mass bin, e.g. `MainBranch_M000` for the first mass bin. Each of these groups has the following attributes:

#	HDF5 name	Type	Bit size	Example	Comment
0	Mass_bin	double	64	10.0	Selection mass in log $M_\odot$
1	Mass_bin_size	double	64	0.1	Selection mass bin size in dex
2	Mass_Type	string	72 or 96	Stellar_Mass	Which mass was used in the selection (Halo_Mass or Stellar_Mass)

The main branches are written as HDF5 compound datasets in each mass bin group. The name of each dataset stars with `Tree_` followed by the Index of the main branch. Each system in the

main branch has the following properties:

#	HDF5 name	Type	Bit size	Example	Comment
0	`Scale_factor	float	32	1.00	Scale factor a of the system
1	Halo_mass	float	32	12.41	Virial mass of the halo in log $M_\odot$
2	Halo_growth_rate	float	32	471.76	Growth rate of the halo in $M_\odot yr^{-1}$
3	Halo_mass_peak	float	32	12.41	Peak virial mass over history in log $M_\odot$
4	Scale_peak_mass	float	32	1.00	Scale factor when peak virial mass was reached
5	Halo_radius	float	32	373.81	Virial radius in kpc
6	Concentration	float	32	4.41	Concentration parameter ($c = R_{vir}/R_s$)
7	Halo_spin	float	32	0.038	Halo spin parameter λ
8	Stellar_mass	float	32	10.52	Stellar mass of the galaxy in log $M_\odot$
9	SFR	float	32	0.72	Star formation rate of the galaxy in $M_\odot yr^{-1}$
10	Intra_cluster_↩ mass	float	32	9.051	Stellar mass of the halo (excluding satellites) in log $M_\odot$
11	X_pos	float	32	8.68	X-Position of the system in the simulation box in Mpc
12	Y_pos	float	32	86.19	Y-Position of the system in the simulation box in Mpc
13	Z_pos	float	32	7.81	Z-Position of the system in the simulation box in Mpc
14	X_vel	float	32	-213.27	X-Velocity of the system in kms^{-1}
15	Y_vel	float	32	132.13	Y-Velocity of the system in kms^{-1}
16	Z_vel	float	32	-343.57	Z-Velocity of the system in kms^{-1}
17	Type	unsigned short	16	0	System type (0: central, 1: satellite, 2↩ : orphan)

#	HDF5 name	Type	Bit size	Example	Comment
18	`Halo_ID`	long long int	64	12345	ID of the halo
19	`Desc_ID`	long long int	64	-1	ID of the descendant (-1 if none)
20	`Up_ID`	long long int	64	-1	ID of the host halo (-1 if halo is a host halo itself)

A single dataset always traces a single system from the root at $z = 0$ up to its leaf at high redshift, i.e. each scale factor is listed only once. The output allways follows the main progenitor in halo mass, i.e. the most massive progenitor halo is used.

8 Disclaimer

The performance and accuracy of Emerge strongly depend on a number of variables, such as the $N-$body simulations and the derived halo merger trees, the observational data used to fit the model, and the code parameters. Emerge comes without any warranty, and without any guarantee that it produces correct results, although the resulting galaxies will undoubtedly be the most elaborate on the market in any case. The numerical parameter values used in the provided parameter file and in this manual do not represent a specific recommendation by the author, but are merely a combination that has been found to work for a particular setup. Exploring parameter space is a difficult task, and it is never guaranteed that an algorithm will converge towards the global maximum of the likelihood function. The user is therefore encouraged to test different parameters to achieve sufficient accuracy for any given setup. If in doubt about some aspect of Emerge, a sensible strategy is to read the source code and the scientific paper to understand the details.