XIGrM

Zhiwei Shao, Ziqian Hua, Douglas Rennehan

Sep 03, 2019

CONTENTS

CHAPTER

ONE

INTRODUCTION

This is a package for systematically analysing the X-ray properties of cosmological simulations based on pynbody and Liang et al., 2016.

CHAPTER

TWO

CONTENTS

2.1 Pyatomdb Module

Generate a series of cooling rates and spectra tabulated with different atomic numbers (axis 0) and temperatures (axis 1).

Currently used atomdb version: 3.0.9

The temporary version only provides atomic data within [0.01, 100] keV. To check the details, see 2.0.2 release notes in http://www.atomdb.org/download.php

`modules.prepare_pyatomdb.`**`AG89_abundances`**(*atomic_numbers*)

Get AG89 abundances of the given input atomic numbers. Based on pyatomdb.atomdb.get_abundance().

`modules.prepare_pyatomdb.`**`calculate_continuum_emission`**(*energy_bins, specific_elements=array([1, 2, 6, 7, 8, 10, 12, 14, 16, 20, 26]), return_spectra=False*)

Calculate continuum emissions and cooling rates for individual atoms in atomdb.

Parameters

energy_bins Energy_bins to calculate cooling rates and generate spectra on, must be in keV in the range of [0.01, 100].

specfic_elements Atomic numbers of elements to be individually listed in the result. All the other elements in atomdb will also be calculated but will be added together as the last element of the result.

return_spectra Whether to return generated spectra.

Returns

dict A dictionary consists of Cooling rates (key: 'CoolingRate') and spectra (key: 'Emissivity') if chosen contributed by continuum for different elements at different temperatures.

`modules.prepare_pyatomdb.`**`calculate_line_emission`**(*energy_bins, specific_elements=array([1, 2, 6, 7, 8, 10, 12, 14, 16, 20, 26]), return_spectra=False*)

Calculate line emissions and cooling rates for individual atoms in atomdb.

Parameters

energy_bins Energy_bins to calculate cooling rates and generate spectra on, must be in keV in the range of [0.01, 100].

specfic_elements Atomic numbers of elements to be individually listed in the result. All the other elements in atomdb will also be calculated but will be added together as the last element of the result.

return_spectra Whether to return generated spectra.

Returns

dict A dictionary consists of Cooling rates (key: 'CoolingRate') and spectra (key: 'Emissivity') if chosen contributed by emission lines for different elements at different temperatures.

`modules.prepare_pyatomdb.`**`elsymbs_to_z0s`**(*elements*)

Convert element symbols to atomic numbers. Based on pyatomdb.atomic.elsymb_to_z0().

`modules.prepare_pyatomdb.`**`get_atomic_masses`**(*atomic_numbers*)

Get atomic masses of the input atomic numbers. Based on pyatomdb.atomic.Z_to_mass().

`modules.prepare_pyatomdb.`**`get_index`**(*te*, *teunits='K'*, *logscale=False*)

Finds indexes in the calculated table with kT closest ro desired kT.

Parameters

te [numpy.ndarray] Temperatures in keV or K

teunits [{'keV' , 'K'}] Units of te (kev or K, default keV)

logscale [bool] Search on a log scale for nearest temperature if set.

Returns

numpy.adarray Indexes in the Temperature list.

`modules.prepare_pyatomdb.`**`load_emissivity_file`**(*filename*, *specific_elements=None*, *energy_band=[0.5, 2.0]*, *n_bins=1000*)

Load the emissivity file calculated based on pyatomdb. If filename can't be loaded, then will calculate and save the emissivity information based on supplied specific_elements.

Parameters

filename [str] File name of the emissivity file to load.

specific_elements List of element symbols to include in calculation. If set to None, will automatically calculate all elements included in pyatomdb.

energy_band [min, max] energy range in keV to calculate emissivity within.

n_bins [int] Number of bins when calculating emissivity.

2.2 Cosmology Module

Calculate cosmological parameters.

`modules.cosmology.`**`Delta_vir`**(*sim*)

Calculate the virial overdensity factor according to eq. 3 in Liang et al. (2016) and hereafter.

Parameters

sim [pynbody.snapshot.SimSnap]

`modules.cosmology.`**`Ez`**(*sim*)

Calculate E(z)=H(z)/H_0.

Parameters

sim [pynbody.snapshot.SimSnap]

2.3 Gas Properties Module

Tools to generate basic information of gas particles required by following analysing.

`modules.gas_properties.abundance_to_solar`(*mass_fraction, elements=['H', 'He', 'C', 'N', 'O', 'Ne', 'Mg', 'Si', 'S', 'Ca', 'Fe']*)

Convert elements mass fraction to abundance relative to AG89 which is accepted by pyatomdb. (AG89: Anders, E. and Grevesse, N. 1989, Geochimica et Cosmochimica Acta, 53, 197)

Parameters

mass_fraction Mass fractions of different elements. In the shape of (n_particles, n_elements). The order of element list must be sorted as atomic number from small to large. Hydrogen must be included.

elements List of elements symbols included.

Returns

numpy.ndarrays Abundance with respect to AG89 results. In the shape of (n_particles, n_elements).

`modules.gas_properties.calcu_luminosity`(*gas, filename, mode='total', elements=['H', 'He', 'C', 'N', 'O', 'Ne', 'Mg', 'Si', 'S', 'Ca', 'Fe'], band=[0.5, 2.0], bins=1000*)

Calculate X-ray luminosity of gas particles.

Parameters

gas [pynbody.snapshot.SimSnap] SubSnap of gas particles to be calculated.

filename [emissivity file]

mode [str] If set to 'total', both continuum and line emission will be taken into account. If set to 'cont', only continuum emission will be considered.

elements, band, bins Required by prepare_pyatomdb.load_emissivity_file(). See load_emissivity_file() docmentation for details.

Returns

list List of luminosities.

`modules.gas_properties.nh`(*sim*)

Calculating hydrogen number density from density.

`modules.gas_properties.temp`(*sim*)

Convert internal energy to temperature, following the instructions from GIZMO documentation

2.4 X-ray Properties Module

Tools for analysing halo X-ray properties using pynbody. Assume any necessary quantity is already prepared as the derived arrays of the snapshot.

`modules.X_properties.cal_tspec`(*hdgas, cal_f, datatype*)

Calculate the Tspec of hot diffuse gas particles.

Parameters

hdgas [pynbody.snapshot.SubSnap] SubSnap of the hot diffuse gas.

cal_f [str] Calibration file.

datatype [str] Simulation data type.

`modules.X_properties.`**`cal_tweight`**(*halo*, *weight_type='Lx'*)

Calculate luminosity weighted or mass weighted temperatures.

Parameters

halo [pynbody.snapshot.SubSnap] SubSnap of the halo.

weight_type [str] Type of weight to take. Related to the available properties of the gas. Now available: luminousity weighted (starts with 'l') and mass weighted (starts with 'm')

2.5 Calculating Radii Module

This module contains codes for caluclating radii R200, R500, etc (and corresponding mass) of halos.

Based on Douglas Rennehan's code on HaloAnalysis.

`modules.calculate_R.`**`get_radius`**(*halo*, *overdensities=array([], dtype=float64)*, *rho_crit=None*, *precision=0.01*, *rmax=None*, *cen=array([], dtype=float64)*, *prop=None*)

Calculate different radii of a given halo with a decreasing sphere method.

Parameters

halo Halo to be calculated, SimSnap in pynbody. Paramaters need to be in physical_units.

overdensity Overdensity factor $Delta$s. Must be a list!

rho_crit Critical density of the universe at the redshift of current SimSnap. Must be in units of Msol/kpc**3.

precision Precision for calculating radius.

rmax The radius at which start using decreasing sphere method. If 0, then use 2 * distance of farthest particle. If set, must be in units of kpc or convertible to kpc via pynbody.

cen center coordinates of calculated halo. If not provided, then will load from property dictionary or calculate via pynbody.analysis.halo.center().

prop halo.properties dictionary. If set to 0 then will automatically load it. Only used for getting boxsize and redshift.

Returns

mass Dict of masses of the halo within calculated radii.

radius Dict of radii of the halo at given overdensities.

`modules.calculate_R.`**`get_radius_bisection`**(*halo*, *overdensities=array([], dtype=float64)*, *rho_crit=0*, *precision=0.01*, *prop=0*)

Calculate different radii of a given halo with a bisection method. This is a prototype without much optimization. And you will need to modify the source code to make it compatible with the module.

Parameters

halo

Halo to be calculated, SimSnap in pynbody. Paramaters need to be in physical_units.

overdensity Overdensity factor $Delta$s.

rho_crit Critical density of the universe at the redshift of current SimSnap. Must be in units of halo['mass'].units / halo['pos'].units**3.

precision Precision within which radii is calculated.

Returns

mass Dict of masses of the halo within calculated radii.

radius Dict of radii of the halo at given overdensities.

2.6 Halo Analysis Module

Tools for analysing halo properties. Assume any necessary basic quantities is already prepared as the derived arrays of the snapshot.

`modules.halo_analysis.`**`get_union`**(*catalogue*, *list*)

Calculate the union of the particles listed in the list.

Parameters

catalogue [pynbody.halo.HaloCatalogue]

list List of halos in the catalogue to get union.

`class` `modules.halo_analysis.`**`halo_props`**(*halocatalogue, datatype, field={'L': ['x', 'x_cont', 'xb', 'xb_cont'], 'M': ['vir', '200', '500', '2500', 'star200', 'gas200', 'bar200', 'ism200', 'cold200', 'igrm200', 'star500', 'gas500', 'bar500', 'ism500', 'cold500', 'igrm500', 'total_star', 'self_star'], 'R': ['vir', '200', '500', '2500'], 'S': ['500', '2500'], 'T': ['x', 'x_cont', 'mass', 'spec', 'spec_corr', 'x_corr', 'x_corr_cont', 'mass_corr']}, host_id_of_top_level=0*)

Bases: `object`

Systematically analyse the halo X-ray properties based on other modules.

Attributes

datatype [str] A copy of the input type of simulation data.

catalogue_original [pynbody.halo.HaloCatalogue] The input halo catalogue.

length [length of the input catalogue.]

host_id_of_top_level How catalogue record "hostHalo" for those halos without a host halo. Default is 0.

errorlist [list] Record when the host halo ID of a certain subhalo is not recorded in the catalogue (weird but will happen in ahf sometimes).

rho_crit Critical density of the current snapshot in Msol kpc**-3.

ovdens Virial overdensity factor $Delta$ of the current snapshot.

dict [astropy.table.Table] A copy of the halo.properties dictionary but in a table form to make future reference more convenient.

haloid List of halo_id given by property dictionary.

IDlist Table of halo_id and corresponding #ID given in the property dictionary.

hostid List of the halo_id of the host halo of each halo (originally recorded in the property dictionary in the form of #ID).

new_catalogue [dict] The new catalogue which includes all the subhalo particles in its host halo. The keys of the dictionary are the indexes of halos in *catalogue_original*.

prop Table of quantities corresponding to input field.

host_list List of host halos.

tophost halo_ids of the top-level host halo for each halo.

children [list of sets] Each set corresponds to the one-level down children of each halo.

galaxy_list List of all galaxies (as long as n_star > 0).

lumi_galaxy_list List of all luminous galaxies (self_m_star > galaxy_low_limit).

galaxies [list of sets] Each set corresponds to the embeded galaxies of each halo. All the subhalos will not be considered and will have an empty set. And for host halos it will include all the galaxies within it, including the galaxies actually embedded in the subhalo (i.e., the children of subhalo).

lumi_galaxies Each set corresponds to the embeded luminous galaxies of each halo. Same as *galaxies*, only care about host halo and include all the luminous galaxies within.

n_lgal Number of total luminous galaxies embedded in each halo. Again, only care about host halos and the galaxies within subhalos (i.e., subhalos themselves) will also be taken into account.

group_list halo_id of the halo identified as group in the catalogue.

calcu_entropy(*self, cal_file, halo_id_list=array([], dtype=float64), calcu_field=['500', '2500'], thickness=SimArray(1, 'kpc')*)

Calculate all entropy within a thin spherical shell centered at halo.

Parameters

cal_file Calibration file used for calculating Tspec.

halo_id_list List of halo_ids to calculate entropies. If set to None, then will use self.group_list.

calcu_field Radii of the thin shell to calculate entropies.

thickness Thickness of the spherical shell.

calcu_radii_masses(*self, halo_id_list=array([], dtype=float64), rdict=None, precision=0.01, rmax=None*)

Calculate radii (Rvir, R200, etc) and corresponding masses.

Parameters

halo_id_list List of halo_ids to calculate radii and masses. If set to None, then will use self.group_list.

rdict names and values for overdensity factors. Default is: {'vir': self.ovdens, '200': 200, '500': 500, '2500': 2500}

precision Precision for calculate radius. See get_index() in calculate_R.py documentation for detail.

rmax Maximum value for the shrinking sphere method. See get_index() in calculate_R.py documentation for detail.

calcu_specific_masses (*self, halo_id_list=array([], dtype=float64), calcu_field=['200', '500']*)
Calculate some specific masses, such as baryon, IGrM, etc.

Parameters

halo_id_list List of halo_ids to calculate masses. If set to None, then will use self.group_list.

calcu_field Radii to calculate specific masses within.

calcu_temp_lumi (*self, cal_file, halo_id_list=array([], dtype=float64), core_corr_factor=0.15, calcu_field='500'*)
Calculate all the temperatures and luminosities listed in temp_field and luminosity_field.

Parameters

cal_file Calibration file used for calculating Tspec.

halo_id_list List of halo_ids to calculate temperatures and luminosities. If set to None, then will use self.group_list.

core_corr_factor Inner radius for calculating core-corrected temperatures. Gas particles within (core_corr_factor*R, R) will be used for calculation.

calcu_field Radius to calculate temperatures and luminosities within. Must be in radius_field. Default: R_500.

calcu_tspec (*self, cal_file, halo_id_list=array([], dtype=float64), core_corr_factor=0.15, calcu_field='500'*)
Calculate spectroscopic temperatures based on Douglas's pytspec module.

Parameters

cal_file Calibration file used for calculating Tspec.

halo_id_list List of halo_ids to calculate temperatures and luminosities. If set to None, then will use self.group_list.

core_corr_factor Inner radius for calculating core-corrected temperatures. Gas particles within (core_corr_factor*R, R) will be used for calculation.

calcu_field Radius to calculate temperatures and luminosities within. Must be in radius_field. Default: R_500.

calcu_tx_lx (*self, halo_id_list=array([], dtype=float64), core_corr_factor=0.15, calcu_field='500'*)
Calculate X-ray luminosities and emission weighted temperatures listed in temp_field and luminosity_field.

Parameters

halo_id_list List of halo_ids to calculate temperatures on. If set to None, then will use self.group_list.

core_corr_factor Inner radius for calculating core-corrected temperatures. Gas particles within (core_corr_factor*R, R) will be used for calculation.

calcu_field Radius to calculate temperatures and luminosities within. Must be in radius_field. Default: R_500.

get_center (*self*)
Calculate the center of the halos if an ahfcatalogue is provided, then will automatically load the results in ahf. Otherwise it will try to calculate the center coordinates via gravitional potential or center of mass.

Notes

Due to a bug in pynbody, calculating center of mass will lead to an incorrect result for the halos crossing the periodical boundary of the simulation box. Make sure pynbody has fixed it before you use.

get_children (*self*)
Generate list of children (subhalos) for each halo. Subhalo itself can also have children. And this list will not contain "grandchildren" (i.e., the children of children).

get_galaxy (*self*, *g_low_limit*)
Generate list of galaxies for each host halo. The subsubhalo will also be included in the hosthalo galaxy list. And it won't generate list for the subhalos even if there are galaxies within.

get_group_list (*self*, *N_galaxy*)
halo_id of the halo identified as group in the catalogue.

Parameters

N_galaxy [int] Number of luminous galaxies above which host halos are considered as groups.

get_new_catalogue (*self*)
Generate a new catalogue based on catalogue_original, the new catalogue will include all the subhalo particles in its host halo.

init_relationship (*self*, *galaxy_low_limit*, *N_galaxy=3*)
Get basic information regarding groups, hosts, children, etc.

savedata (*self, filename, field={'R': ['vir', '200', '500', '2500'], 'M': ['vir', '200', '500', '2500', 'star200', 'gas200', 'bar200', 'ism200', 'cold200', 'igrm200', 'star500', 'gas500', 'bar500', 'ism500', 'cold500', 'igrm500', 'total_star', 'self_star'], 'T': ['x', 'x_cont', 'mass', 'spec', 'spec_corr', 'x_corr', 'x_corr_cont', 'mass_corr'], 'S': ['500', '2500'], 'L': ['x', 'x_cont', 'xb', 'xb_cont']}, halo_id_list=array([], dtype=float64), units={'T': 'keV', 'L': 'erg s**-1', 'R': 'kpc', 'M': 'Msol', 'S': 'keV cm**2'}*)
Save the data in hdf5 format. Will save halo_id_list (key: 'halo_id') and the quantities listed in field.

Parameters

filename Filename of the hdf5 file.

field Type of information to save.

halo_id_list List of halo_ids to save.If set to None, then will use self.group_list.

units Convert the data into specified inits and save.

2.7 How to Use

2.7.1 Data Structure

Most useful halo information is stored in the *prop* attribute of the *class halo_props*. And to modify what to be calculated, one will need to provide a new field dictionary similarly organized as the default field dictionary. Note that quantities related to temperature and luminosity are mostly hard coded, so you may need to modify the source code to meet your own requirements.

Generally there are four types of properties stored in the *prop* attribute: R (radius), M (mass), T (temperature) and S (entropy). And the following list explains the physical meaning of the terms in the default field dictionary.

R:

Key name	Description
vir	Virial radius R_{vir}
200	R_{200}
500	R_{500}
2500	R_{2500}

M ($X = 200, 500$):

Key name	Description
vir	Mass enclosed within R_{vir}
200	Mass enclosed within R_{200}
500	Mass enclosed within R_{500}
2500	Mass enclosed within R_{2500}
starX	Stellar mass enclosed within R_X
gasX	Gas mass enclosed within R_X
barX	Baryon mass enclosed within R_X
ismX	ISM mass enclosed within R_X
coldX	Cold gas mass enclosed within R_X
igrmX	IGrM mass enclosed within R_X
total_star	Stellar mass within halo, including contribution from subhalo
self_star	Stellar mass within halo, excluding contribution from subhalo

Before introducing temperature and entropy, I will first talk about the luminosities used during calculation:

Name	Key name	Description
L_X	Lx	0.5-2.0keV X-ray luminosity, including both continuum and line emission
$L_{X,cont}$	Lx_cont	0.5-2.0keV X-ray luminosity, only including continuum emission
$L_{X,broad}$	Lxb	0.1-2.4keV X-ray luminosity, including both continuum and line emission
$L_{X,broad,cont}$	Lxb_cont	0.1-2.4keV X-ray luminosity, only including continuum emission

T (only hot diffuse gas (IGrM) is taken into account when calculating temperature):

Key name	Description
x	Emission weighted temperature using L_X
x_cont	Emission weighted temperature using $L_{X,cont}$
mass	Mass weighted temperature
spec	Spectroscopic temperature (See Vikhlinin, 2006)
spec_corr	Spectroscopic (core-corrected) temperature
x_corr	Emission weighted (core-corrected) temperature using L_X
x_corr_cont	Emission weighted (core-corrected) temperature using $L_{X,cont}$
mass_corr	Mass weighted (core-corrected) temperature

S (calculated within a thin spherical shell with thickness = 1 kpc):

Key name	Description
500	Entropy of the shell at R_{500}
2500	Entropy of the shell at R_{2500}

2.7.2 Examples

Comming soon… For now, see "Shao's Final Tutorial.ipynb" for details.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

S

T