OMEGA Module

GCE OMEGA (One-zone Model for the Evolution of Galaxies) module

Functionality

This tool allows one to simulate the chemical evolution of single-zone galaxies. Having the star formation history as one of the input parameters, OMEGA can target local galaxies by using observational data found in the literature.

Made by

FEB2015: C. Ritter, B. Cote

MAY2015: B. Cote

The code inherits the chem_evol class, which contains common functions shared by SYGMA and OMEGA. The code in chem_evol has been developed by :

v0.1 NOV2013: C. Fryer, C. Ritter

v0.2 JAN2014: C. Ritter

v0.3 APR2014: C. Ritter, J. F. Navarro, F. Herwig, C. Fryer, E. Starkenburg,
M. Pignatari, S. Jones, K. Venn1, P. A. Denissenkov & the NuGrid collaboration

v0.4 FEB2015: C. Ritter, B. Cote

v0.5 MAR2015: B. Cote

v0.6 OCT2016: B. Cote

Usage

Import the module:

```
>>> import omega as o
```

Get help:

```
>>> help o
```

Get more information:

```
>>> o.omega?
```

Create a custom galaxy (closed box):

```
>>> o1 = o.omega(cte_sfr=1.0, mgal=1.5e10)
```

Simulate a known galaxy (open box):

```
>>> o2 = o.omega(galaxy='sculptor', in_out_control=True, mgal=1e6, mass_loading=8, in_out_rati
```

Analysis functions: See the Sphinx documentation

class `omega.`**omega**(*galaxy='none', in_out_control=False, SF_law=False, DM_evolution=False, Z_trans=1e-20, f_dyn=0.1, sfe=0.1, outflow_rate=-1.0, inflow_rate=-1.0, rand_sfh=0.0, cte_sfr=1.0, m_DM_0=100000000000.0, mass_loading=1.0, t_star=-1.0, sfh_file='none', in_out_ratio=1.0, stellar_mass_0=-1.0, z_dependent=True, exp_ml=2.0, nsmerger_bdys=[8, 100], imf_type='kroupa', alphaimf=2.35, imf_bdys=[0.1, 100], sn1a_rate='power_law', iniZ=0.0, dt=1000000.0, special_timesteps=30, tend=13000000000.0, mgal=10000000000.0, transitionmass=8, iolevel=0, ini_alpha=True, nb_nsm_per_m=-1.0, t_nsm_coal=-1.0, table='yield_tables/isotope_yield_table_MESA_only_fryer12_delay.txt', hardsetZ=-1, sn1a_on=True, nsm_dtd_power=[], sn1a_table='yield_tables/sn1a_t86.txt', ns_merger_on=True, f_binary=1.0, f_merger=0.0008, t_merger_max=10000000000.0, m_ej_nsm=0.025, nsmerger_table='yield_tables/r_process_rosswog_2014.txt', bhns_merger_on=False, m_ej_bhnsm=0.025, bhnsmerger_table='yield_tables/r_process_rosswog_2014.txt', iniabu_table='', extra_source_on=False, extra_source_table='yield_tables/extra_source.txt', f_extra_source=1.0, pop3_table='yield_tables/popIII_heger10.txt', imf_bdys_pop3=[0.1, 100], imf_yields_range_pop3=[10, 30], starbursts=[], beta_pow=-1.0, gauss_dtd=[1000000000.0, 660000000.0], exp_dtd=2000000000.0, nb_1a_per_m=0.001, f_arfo=1, t_merge=-1.0, imf_yields_range=[1, 30], exclude_masses=[], netyields_on=False, wiersmamod=False, skip_zero=False, redshift_f=0.0, print_off=False, long_range_ref=False, f_s_enhance=1.0, m_gas_f=-1.0, cl_SF_law=False, external_control=False, calc_SSP_ej=False, tau_ferrini=False, input_yields=False, popIII_on=True, t_sf_z_dep=1.0, m_crit_on=False, norm_crit_m=8000000000.0, mass_frac_SSP=0.5, sfh_array_norm=-1.0, imf_rnd_sampling=False, out_follows_E_rate=False, r_gas_star=-1.0, cte_m_gas=-1.0, t_dtd_poly_split=-1.0, stellar_param_on=False, bhnsmerger_dtd_array=array([], dtype=float64), DM_array=array([], dtype=float64), nsmerger_dtd_array=array([], dtype=float64), sfh_array=array([], dtype=float64), ism_ini=array([], dtype=float64), mdot_ini=array([], dtype=float64), mdot_ini_t=array([], dtype=float64), ytables_in=array([], dtype=float64), zm_lifetime_grid_nugrid_in=array([], dtype=float64), isotopes_in=array([], dtype=float64), ytables_pop3_in=array([], dtype=float64), zm_lifetime_grid_pop3_in=array([], dtype=float64), ytables_1a_in=array([], dtype=float64), ytables_nsmerger_in=array([], dtype=float64), dt_in=array([], dtype=float64), dt_split_info=array([], dtype=float64), ej_massive=array([], dtype=float64), ej_agb=array([], dtype=float64), ej_sn1a=array([], dtype=float64), ej_massive_coef=array([], dtype=float64), ej_agb_coef=array([], dtype=float64), ej_sn1a_coef=array([], dtype=float64), dt_ssp=array([],* *dtype=float64), yield_interp='lin', mass_sampled=array([], dtype=float64), scale_cor=array([], dtype=float64), mass_sampled_ssp=array([], dtype=float64), scale_cor_ssp=array([], dtype=float64), poly_fit_dtd_5th=array([], dtype=float64), poly_fit_range=array([], dtype=float64), m_tot_ISM_t_in=array([], dtype=float64)*) [source]

Bases: `chem_evol.chem_evol`

Important : By default, a closed box model is always assumed.

galaxy : *string*

Name of the target galaxy. By using a known galaxy, the code automatically selects the corresponding star formation history, stellar mass, and total mass (when available). By using 'none', the user has perfect control of these three last parameters.

Choices : 'milky_way', 'milky_way_cte', 'sculptor', 'carina', 'fornax', 'none'

Default value : 'none'

Special note : The 'milky_way_cte' option uses the Milky Way's characteristics, but with a constant star formation history.

cte_sfr : *float*

Constant star formation history in [Mo/yr].

Default value : 1.0

rand_sfh : *float*

Maximum possible ratio between the maximum and the minimum values of a star formation history that is randomly generated.

Default value : 0.0 (deactivated)

Special note : A value greater than zero automatically generates a random star formation history, which pypasses the use of the cte_sfr parameter.

sfh_file : *string*

Path to a file containing an input star formation history. The first and second columns must be the age of the galaxy in [yr] and the star formation rate in [Mo/yr].

Default value : 'none' (deactivated)

Special note : When a path is specified, it by passes the cte_sfr and the and_sfh parameters.

stellar_mass_0 : *float*

Current stellar mass of the galaxy, in [Mo], at the end of the simulation.

Default value : -1.0 (you need to specify a value with unknown galaxies)

in_out_control : *boolean*

The in_out_control implementation enables control of the outflow and the inflow rates independently by using constant values (see outflow_rate and inflow_rate) or by using a mass-loading factor that connects the rates to the star formation history (see mass_loading and in_out_ratio).

Default value : False (deactivated)

mass_loading : *float*

Ratio between the outflow rate and the star formation rate.

Default value : 1.0

outflow_rate : *float*

Constant outflow rate in [Mo/yr].

Default value : -1.0 (deactivated)

Special note : A value greater or equal to zero activates the constant utflow mode, which bypasses the use of the mass_loading parameter.

in_out_ratio : *float*

Used in : in_out_control mode

Ratio between the inflow rate and the outflow rate. This parameter is used to calculate the inflow rate, not the outflow rate.

Default value : 1.0

inflow_rate : *float*

Used in : in_out_control mode Constant inflow rate in [Mo/yr].

Default value : -1.0 (deactivated)

Special note : A value greater or equal to zero activates the constant inflow mode, which bypasses the use of the in_out_ratio parameter.

SF_law : *boolean*

The SF_law inplementation assumes a Kennicutt-Schmidt star formation law and combines it to the known input star formation history in order to derive the mass of the gas reservoir at every timestep.

Default value : False (deactivated)

sfe : *float*

Used in : SF_law and DM_evolution modes

Star formation efficiency present in the Kennicutt-Schmidt law.

Default value : 0.1

f_dyn : *float*

Used in : SF_law and DM_evolution modes Scaling factor used to calculate the star formation timescale present in the Kennicutt-Schmidt law. We assume that this timescale is equal to a fraction of the dynamical timescale of the virialized system (dark and baryonic matter), t_star = f_dyn * t_dyn.

Default value : 0.1

m_DM_0 : *float*

Used in : SF_law and DM_evolution modes Current dark matter halo mass of the galaxy, in [Mo], at the end of the simulations.

Default value : 1.0e+11

t_star : *float*

Used in : SF_law and DM_evolution modes Star formation timescale, in [yr], used in the Kennicutt-Schmidt law. Default value = -1.0 (deactivated)

Special note : A positive value activates the use of this parameter, which bypasses the f_dyn parameter.

DM_evolution : *boolean*

The DM_evolution implementation is an extension of the SF_law option. In addition to using a Kennicutt-Schmidt star formation law, it assumes an evolution in the total mass of the galaxy as function of time. With this prescription, the mass-loading factor has a mass dependency. The mass_loading parameter then only represents the final value at the end of the simulation.

Default value : False (deactivated)

exp_ml : *float*

Used in : DM_evolution mode

Exponent of the mass dependency of the mass-loading factor. This last factor is proportional to M_vir**(-exp_ml/3), where M_vir is the sum of dark and baryonic matter.

Default value : 2.0

special_timesteps : *integer*

Number of special timesteps. This option (already activated by default) is activated when special_timesteps > 0. It uses a logarithm timestep scheme which increases the duration of timesteps throughout the simulation.

Default value : 30

dt : *float*

Duration of the first timestep [yr] if special_timesteps is activated. Duration of each timestep if special_timesteps is desactivated.

Default value : 1.0e6

tend : *float*

Total duration of the simulation [yr].

Default value : 13.0e9

imf_bdys : *list*

Upper and lower mass limits of the initial mass function (IMF) [Mo].

Default value : [0.1,100]

imf_yields_range : *list*

Initial mass of stars that contribute to stellar ejecta [Mo].

Default value : [1,30]

imf_type : *string*

Choices : ‘salpeter’, ‘chabrier’, ‘kroupa’, ‘alphaimf’ ‘alphaimf’ creates a custom IMF with a single power-law covering imf_bdys.

Default value : ‘kroupa’

alphaimf : *float*

Aplha index of the custom IMF, dN/dM = Constant * M^-alphaimf

Default value : 2.35

imf_bdys_pop3 : *list*

Upper and lower mass limits of the IMF of PopIII stars [Mo].

Default value : [0.1,100]

imf_yields_range_pop3 : *list*

Initial mass of stars that contribute to PopIII stellar ejecta [Mo]. PopIII stars ejecta taken from Heger et al. (2010)

Default value : [10,30]

iniZ : *float*

Initial metallicity of the gas in mass fraction (e.g. Solar Z = 0.02).

Choices : *0.0, 0.0001, 0.001, 0.006, 0.01, 0.02*

(-1.0 to use non-default yield tables)

Default value : 0.0

Z_trans : *float*

Variable used when interpolating stellar yields as a function of Z. Transition Z below which PopIII yields are used, and above which default yields are used.

Default value : -1 (not active)

mgal : *float*

Initial mass of gas in the simulation [Mo].

Default value : 1.6e11

sn1a_on : *boolean*

True or False to include or exclude the contribution of SNe Ia.

Default value : True

sn1a_rate : *string*

SN Ia delay-time distribution function used to calculate the SN Ia rate.

Choices :

‘power_law’ - custom power law, set parameter with beta_pow (similar to Maoz & Mannucci 2012)

‘gauss’ - gaussian DTD, set parameter with gauss_dtd

‘exp’ - exponential DTD, set parameter with exp_dtd

‘maoz’ - specific power law from Maoz & Mannucci (2012)

Default value : ‘power_law’

sn1a_energy : *float*

Energy ejected by single SNIa event. Units in erg.

Default value : 1e51

ns_merger_on : *boolean*

True or False to include or exclude the contribution of neutron star mergers.

Default value: True

beta_pow : *float*

Slope of the power law for custom SN Ia rate, R = Constant * t^-beta_pow.

Default value : -1.0

gauss_dtd : *list*

Contains parameter for the gaussian DTD: first the characteristic time [yrs] (gaussian center) and then the width of the distribution [yrs].

Default value : [3.3e9,6.6e8]

exp_dtd : *float*

Characteristic delay time [yrs] for the e-folding DTD.

nb_1a_per_m : *float*

Number of SNe Ia per stellar mass formed in a simple stellar population.

Default value : 1.0e-03

direct_norm_1a : *float*

Normalization coefficient for SNIa rate integral.

Default: deactived but replaces the usage of teh nb_1a_per_m when its value is larger than zero.

transitionmass : *float*

Initial mass which marks the transition from AGB to massive stars [Mo].

Default value : 8.0

exclude_masses : *list*

Contains initial masses in yield tables to be excluded from the simulation;

Default value : []

table : *string*

Path pointing toward the stellar yield tables for massive and AGB stars.

Default value : ‘yield_tables/isotope_yield_table_MESA_only_fryer12_delay.txt’ (NuGrid)

sn1a_table : *string*

Path pointing toward the stellar yield table for SNe Ia.

Default value : ‘yield_tables/sn1a_t86.txt’ (Tielemann et al. 1986)

nsmerger_table : *string*

Path pointing toward the r-process yield tables for neutron star mergers

Default value : ‘yield_tables/r_process_rosswog_2014.txt’ (Rosswog et al. 2013)

iniabu_table : *string*

Path pointing toward the table of initial abuncances in mass fraction.

Default value : ‘yield_tables/iniabu/iniab2.0E-02GN93.ppn’

yield_interp : *string*

if ‘None’ : no yield interpolation, no interpolation of total ejecta

if ‘lin’ - Simple linear yield interpolation.

if ‘wiersma’ - Interpolation method which makes use of net yields as used e.g. in Wiersma+ (2009); Does not require net yields.

if netyields_on is true than makes use of given net yields
else calculates net yields from given X0 in yield table.

Default : ‘lin’

netyields_on : *boolean*

if true assumes that yields (input from table parameter) are net yields.

Default : false.

total_ejecta_interp : *boolean*

if true then interpolates total ejecta given in yield tables
over initial mass range.

Default : True

stellar_param_on : *boolean*

if true reads in additional stellar parameter given in table stellar_param_table.

Default : true in sygma and false in omega

stellar_param_table: string

Path pointoing toward the table hosting the evolution of stellar parameter derived from stellar evolution calculations.

Default table : ‘yield_tables/isotope_yield_table_MESA_only_param.txt’

iolevel : *int*

Specifies the amount of output for testing purposes (up to 3).

Default value : 0

poly_fit_dtd : *list*

Array of polynomial coefficients of a customized delay-time distribution function for SNe Ia. The polynome can be of any order. Example : [0.2, 0.3, 0.1] for rate_snIa(t) = 0.2*t**2 + 0.3*t + 0.1 Note : Must be used with the poly_fit_range parameter (see below)

Default value : np.array([]) –> Desactivated

poly_fit_range : *list –> [t_min,t_max]*

Time range where the customized delay-time distribution function for SNe Ia will be applied for a simple stellar population.

Default value : np.array([]) –> Desactivated

mass_sampled : *list*

Stellar masses that are sampled to eject yields in a stellar population. Warning : The use of this parameter bypasses the IMF calculation and do not ensure a correlation with the star formation rate. Each sampled mass will eject the exact amount of mass give in the stellar yields.

Default value : np.array([]) –> Desactivated

scale_cor : *2D list*

Determine the fraction of yields ejected for any given stellar mass bin. Example : [[1.0,8], [0.5,100]] means that stars with initial mass between 0 and 8 Msu will eject 100% of their yields, and stars with initial mass between 8 and 100 will eject 50% of their yields. There is no limit for the number of [%,M_upper_limit] arrays used. Default value : np.array([]) –> Desactivated

Methods

`plot_abu_distr`(*species, source='all', norm='', label='', shape='', marker='', color='', markevery=20, multiplot=False, return_x_y=False, fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

Plot abundance distribution at point in time normalized to solar system distribution.

Parameters: **species** : array

isotopes or element names, in the form ['C'] or ['C-12']

time : float

Time in yrs telling when to extract abundances, e.g. birth SS 9.2Gyr

source : string

Specifies if yields come from all sources ('all'), including AGB+SN1a, massive stars. Or from distinctive sources: only agb stars ('agb'), only SN1a ('SN1a'), or only massive stars ('massive')

norm : string

Choose the solar abundances, e.g. 'Grevesse_Noels_1993', same notation as in stellab?

label : string

figure label

marker : string

figure marker

shape : string

line style

color : string

color of line

fig : string,float

to name the plot figure

Examples

```
>>> s.plot_abu_distr(species=['C-12','O-16'],time=9.2e9,norm='Asplund_et_al_2009')
```

plot_abun(*fig=20, age=9200000000.0, solar_norm=False, iso_on=False, list_elem=[], list_iso=[], return_x_y=False, over_plot_solar=False, solar_ab_m='yield_tables/iniabu/iniab2.0E-02GN93.ppn', f_y_annotate=0.9, marker='o', marker_s='^', shape='', shape_s='-', color='b', color_s='r', label='Prediction', label_s='solar', fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14, markersize=5*) [source]

This function plots the abundance distribution in mass fraction at any given point in time or at any given metallicity.

Parameters: **age** : float

Time passed since the beginning of the simulation at which abundance distribution is taken. Default: Time until the formation of the pre-solar cloud.

solar_norm: boolean

If True, the abundances will be divide by the solar abundances. If False, there is no normalization. Default: False

iso_on : boolean

> If True, the abundances will be in terms of isotopes. If False, the abundances will be in terms of elements.

list_elem : array of strings

> Contain the list of elements included in the abundances distribution. Default: [], all elements will be included. Example: list_elem=['H','He','C','N','O','Fe']

list_iso : array of strings

> Contain the list of isotopes included in the abundances distribution. Default: [], all isotopes will be included. Example: list_iso=['H-1','C-12','C-13','N-14','N-15'] Note: list_iso dominates over list_elem is both are used.

return_x_y : boolean

> If False, show the plot. If True, return two arrays containing the X and Y axis data, respectively. If True and iso_list=True, return a multi-D array [i][j][:] where j=0 -> list of mass numbers A for the isotopes associated with the element i, j=1 -> list of abundances

over_plot_solar : boolean

> If True, plot the solar abundances distribution along with the predictions Note: This option is desactivated if solar_norm=True

solar_ab_m : string

> Path of the solar normalization containing the mass fraction of each isotope Default: solar_ab_m='yield_tables/iniabu/iniab2.0E-02GN93.ppn'

f_y_annotate : float (between 0 and 1)

> Fraction of the Y axis where the name of the element will be shown Default: f_y_annotate=0.9

fig : string, float

> Figure name.

marker : string

> Figure marker.

marker_s : string

Marker of the solar abundance distribution.

shape : string

Line style.

shape_s : string

Line style of the solar abundance distribution.

color : string

Line color.

color_s : string

Line color of the solar abundances pattern.

label : string

Figure label.

label_s : string

Label of the solar abundance distribution.

fsize : 2D float array

Figure dimension/size.

fontsize : integer

Font size of the numbers on the X and Y axis.

rspace : float

Extra space on the right for the legend.

bspace : float

Extra space at the bottom for the Y axis label.

labelsize : integer

Font size of the X and Y axis labels.

legend_fontsize : integer

Font size of the legend.

markersize : float

Size of the markers Default: markersize=5

`plot_dark_matter`(*fig=11, marker='', shape='', color='', label='Dark matter', fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

This function plots the dark matter halo mass of the galaxy as a function of time.

Parameters:

fig : string, float

Figure name.

marker : string

Figure marker.

shape : string

Line style.

color : string

Line color.

label : string

Figure label.

fsize : 2D float array

Figure dimension/size.

fontsize : integer

Font size of the numbers on the X and Y axis.

rspace : float

Extra space on the right for the legend.

bspace : float

Extra space at the bottom for the Y axis label.

labelsize : integer

Font size of the X and Y axis labels.

legend_fontsize : integer

Font size of the legend.

Examples

```
>>> o1.plot_dark_matter(label='Dark Matter')
```

plot_inflow_rate(*fig=10, marker='', shape='', color='', label='Inflow', fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

This function plots the mass inflow rate as a function of time in units of [Mo/yr].

Parameters: **fig** : string, float

Figure name.

marker : string

Figure marker.

shape : string

Line style.

color : string

Line color.

label : string

Figure label.

fsize : 2D float array

Figure dimension/size.

fontsize : integer

Font size of the numbers on the X and Y axis.

rspace : float

Extra space on the right for the legend.

bspace : float

Extra space at the bottom for the Y axis label.

labelsize : integer

Font size of the X and Y axis labels.

legend_fontsize : integer

Font size of the legend.

Examples

```
>>> o1.plot_inflow_rate(label='Inflow Rate')
```

plot_iso_ratio(*return_x_y=False, xaxis='age', yaxis='C-12/C-13', solar_ab='yield_tables/iniabu/iniab2.0E-02GN93.ppn', solar_iso='solar_normalization/Asplund_et_al_2009_iso.txt', fig=18, source='all', marker='', shape='', color='', label='', fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

This function plots the evolution of an isotopic ratio as a function of time, as a function of an abundance ratio, or as a function of another isotopic ratio. Isotopic ratios are given in the delta notation and abundances ratios are given in spectroscopic notation.

Parameters: **return_x_y** : boolean

If False, show the plot. If True, return two arrays containing the X and Y axis data, respectively.

xaxis : string

X axis, either ‘age’, an abundance ratio such as ‘[Fe/H]’, or an isotopic ratio such as ‘C-12/C-13’.

yaxis : string

Y axis, isotopic ratio such as ‘C-12/C-13’.

solar_ab : string

Path to the solar abundances used to normalize abundance ratios.

solar_iso : string

Path to the solar isotopes used to normalize the isotopic ratios.

fig : string, float

Figure name.

source : string

Specificies if yields come from all sources ('all') or from distinctive sources: only AGB stars ('agb'), only SN Ia ('sn1a'), or only massive stars ('massive')

marker : string

Figure marker.

shape : string

Line style.

color : string

Line color.

label : string

Figure label.

fsize : 2D float array

Figure dimension/size.

fontsize : integer

Font size of the numbers on the X and Y axis.

rspace : float

Extra space on the right for the legend.

bspace : float

Extra space at the bottom for the Y axis label.

labelsize : integer

Font size of the X and Y axis labels.

legend_fontsize : integer

Font size of the legend.

Examples

```
>>> o1.plot_iso_ratio(xaxis='[Fe/H]', yaxis='C-12/C-13')
```

plot_mass(*fig=0, specie='C', source='all', norm=False, label='', shape='', marker='', color='', markevery=20, multiplot=False, return_x_y=False, fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14, show_legend=True*) [source]

mass evolution (in Msun) of an element or isotope vs time.

Parameters:

specie : string

isotope or element name, in the form 'C' or 'C-12'

source : string

Specifies if yields come from all sources ('all'), including AGB+SN1a, massive stars. Or from distinctive sources: only agb stars ('agb'), only SN1a ('SN1a'), or only massive stars ('massive')

norm : boolean

If True, normalize to current total ISM mass

label : string

figure label

marker : string

figure marker

shape : string

line style

color : string

color of line

fig : string,float

to name the plot figure

show_legend : boolean

Default True. Show or not the legend

Examples

```
>>> s.plot('C-12')
```

`plot_mass_loading`(*fig=8, marker='', shape='', color='', label='Mass-loading', fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

This function plots the mass-loading factor, which is the ratio between the mass outflow rate and the star formation rate, as a function of time.

Parameters:

fig : string, float

Figure name.

marker : string

Figure marker.

shape : string

Line style.

color : string

Line color.

label : string

Figure label.

fsize : 2D float array

Figure dimension/size.

fontsize : integer

Font size of the numbers on the X and Y axis.

rspace : float

Extra space on the right for the legend.

bspace : float

Extra space at the bottom for the Y axis label.

labelsize : integer

Font size of the X and Y axis labels.

legend_fontsize : integer

Font size of the legend.

Examples

```
>>> o1.plot_mass_loading(label='Mass-Loading Factor')
```

`plot_massfrac`(*fig=2, xaxis='age', yaxis='O-16', source='all', norm='no', label='', shape='', marker='', color='', markevery=20, fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

Plots mass fraction of isotope or element vs either time or other isotope or element.

Parameters: **xaxis** : string

either 'age' for time or isotope name, in the form e.g. 'C-12'

yaxis : string

isotope name, in the same form as for xaxis

source : string

Specifies if yields come from all sources ('all'), including AGB+SN1a, massive stars. Or from distinctive sources: only agb stars ('agb'), only SN1a ('SN1a'), or only massive stars ('massive')

norm : string

if 'no', no normalization of mass fraction if 'ini', normalization ot initial abundance

label : string

figure label

marker : string

figure marker

shape : string

line style

color : string

color of line

fig : string,float

to name the plot figure

Examples

```
>>> s.plot_massfrac('age','C-12')
```

`plot_mdf`(*fig=19, return_x_y=False, axis_mdf='[Fe/H]', dx=0.05, solar_ab='yield_tables/iniabu/iniab2.0E-02GN93.ppn', sigma_gauss=0.0, nb_sigma=3.0, marker='', shape='', norm=True, color='', label='', fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15,* *labelsize=15, legend_fontsize=14*) [source]

This function plots the stellar metallicity distribution function (MDF), or the distribution function of any other abundance ratio.

Parameters:

return_x_y : boolean

If False, show the plot. If True, return two arrays containing the X and Y axis data, respectively.

axis_mdf : string

Abundance ratio for the distribution such as '[Fe/H]' or '[Mg/H]'.

dx : float

Bin resolution of the distribution.

solar_ab : string

Path to the solar abundances used to normalize abundance ratios.

sigma_gauss : float

Each point in the MDF can be associated with a gaussian. This implies that in reality, the metallicity should have a certain dispersion when stars form at each timestep (instead of using only a single average value). The sigma_guass parameter sets the sigma value eac gaussian function.

nb_sigma : float

When sigma_gauss is greater than zero, nb_sigma is the number of sigma by which the X axis range is expanded.

norm : boolean

Normalize the MDF is True Default : True

fig : string, float

Figure name.

marker : string

Figure marker.

shape : string

Line style.

color : string

Line color.

label : string

Figure label.

fsize : 2D float array

Figure dimension/size.

fontsize : integer

Font size of the numbers on the X and Y axis.

rspace : float

Extra space on the right for the legend.

bspace : float

Extra space at the bottom for the Y axis label.

labelsize : integer

Font size of the X and Y axis labels.

legend_fontsize : integer

Font size of the legend.

Examples

```
>>> o1.plot_mdf(axis_mdf='[Fe/H]', sigma_gauss=0.2)
```

plot_outflow_rate(*fig=9, marker='', shape='', color='', label='Outflow', fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

This function plots the mass outflow rate as a function of time in units of [Mo/yr].

Parameters: **fig** : string, float

Figure name.

marker : string

Figure marker.

shape : string

Line style.

color : string

Line color.

label : string

Figure label.

fsize : 2D float array

Figure dimension/size.

fontsize : integer

Font size of the numbers on the X and Y axis.

rspace : float

Extra space on the right for the legend.

bspace : float

Extra space at the bottom for the Y axis label.

labelsize : integer

Font size of the X and Y axis labels.

legend_fontsize : integer

Font size of the legend.

Examples

```
>>> o1.plot_outflow_rate(label='Outflow Rate')
```

`plot_redshift`(*fig=16, marker='', shape='', color='', label='Redshift', fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

This function plots the redshift a function of time.

Parameters: **fig** : string, float

Figure name.

marker : string

Figure marker.

shape : string

Line style.

color : string

Line color.

label : string

Figure label.

fsize : 2D float array

Figure dimension/size.

fontsize : integer

Font size of the numbers on the X and Y axis.

rspace : float

Extra space on the right for the legend.

bspace : float

Extra space at the bottom for the Y axis label.

labelsize : integer

Font size of the X and Y axis labels.

legend_fontsize : integer

Font size of the legend.

Examples

```
>>> o1.plot_redshift(label='Redshift')
```

`plot_sf_timescale`(*fig=15, marker='', shape='', color='', label='', fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

This function plots the star formation timescale as a function of time in units of [yr].

Parameters: **fig** : string, float

Figure name.

marker : string

Figure marker.

shape : string

Line style.

color : string

Line color.

label : string

Figure label.

fsize : 2D float array

Figure dimension/size.

fontsize : integer

Font size of the numbers on the X and Y axis.

rspace : float

Extra space on the right for the legend.

bspace : float

Extra space at the bottom for the Y axis label.

labelsize : integer

Font size of the X and Y axis labels.

legend_fontsize : integer

Font size of the legend.

Examples

```
>>> o1.plot_sf_timescale(label='Star Formation Timescale')
```

plot_sn_distr(*fig=5, rate=True, rate_only='', xaxis='time', fraction=False, label1='SNIa', label2='SN2', shape1=':', shape2='--', marker1='o', marker2='s', color1='k', color2='b', markevery=20, fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

Plots the SN1a distribution: The evolution of the number of SN1a and SN2 #Add numbers/dt numbers/year...

Parameters: **rate** : boolean

if true, calculate rate [1/century] else calculate numbers

fraction ; boolean

if true, ignorate rate and calculate number fraction of SNIa per WD

rate_only : string

if empty string, plot both rates (default)
if 'sn1a', plot only SN1a rate
if 'sn2', plot only SN2 rate

xaxis: string

if 'time' : time evolution if 'redshift': experimental! use with caution; redshift evolution

label : string

figure label

marker : string

figure marker

shape : string

line style

color : string

color of line

fig : string,float

to name the plot figure

Examples

```
>>> s.plot_sn_distr()
```

plot_spectro(*fig=3, xaxis='age', yaxis='[Fe/H]', source='all', label='', shape='-', marker='o', color='k', markevery=100, show_data=False, show_sculptor=False, show_legend=True, return_x_y=False, sub_plot=False, linewidth=3, sub=1, plot_data=False, fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14, only_one_iso=False, solar_ab='', sfr_thresh=0.0, m_formed_thresh=1.0, solar_norm=''*) [source]

Plots elements in spectroscopic notation:

Parameters: **xaxis** : string

Elements spectroscopic notation e.g. [Fe/H] if 'age': time evolution in years

yaxis : string

Elements in spectroscopic notation, e.g. [C/Fe]

source : string

If yields come from all sources use 'all' (include AGB+SN1a, massive stars.)
If yields come from distinctive source: only agb stars use 'agb', only SN1a ('SN1a'), or only massive stars ('massive')

label : string

figure label

marker : string

figure marker

shape : string

line style

color : string

color of line

fig : string,float

to name the plot figure

Examples

```
>>> plt.plot_spectro('[Fe/H]','[C/Fe]')
```

`plot_star_formation_rate`(*fig=6, fraction=False, source='all', marker='', shape='', color='', label='', abs_unit=True, fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

Plots the star formation rate over time. Shows fraction of ISM mass which transforms into stars.

Parameters: **fraction** : boolean

if true: fraction of ISM which transforms into stars; else: mass of ISM which goes into stars

source : string

either 'all' for total star formation rate; 'agb' for AGB and 'massive' for massive stars

marker : string

marker type

shape : string

line shape type

fig: figure id

Examples

```
>>> s.star_formation_rate()
```

`plot_totmasses`(*fig=4, mass='gas', source='all', norm='no', label='', shape='', marker='', color='', markevery=20, log=True, fsize=[10, 4.5], fontsize=14, rspace=0.6, bspace=0.15, labelsize=15, legend_fontsize=14*) [source]

Plots either gas or star mass in fraction of total mass vs time.

Parameters: **mass** : string

either 'gas' for ISM gas mass or 'stars' for gas locked away in stars (totalgas - ISM gas)

norm : string

normalization, either 'no' for no normalization (total gass mass in solar masses),

for normalization to the initial gas mass (mgal) with 'ini',
for normalization to the current total gas mass 'current'. The latter case makes sense when comparing different sources (see below)

source : string

specificies if yields come from all sources ('all'), including AGB+SN1a, massive stars. Or from distinctive sources: only agb stars ('agb'), only SN1a ('sn1a'), or only massive stars ('massive')

log : boolean

if true plot logarithmic y axis

label : string

figure label

marker : string

figure marker

shape : string

line style

color : string

color of line

fig : string,float

to name the plot figure

Examples

```
>>> s.plot_totmasses()
```

`run_step`(*i*, *sfr_rs*, *m_added=array([], dtype=float64)*, *m_lost=0.0*, *mass_sampled=array([], dtype=float64)*, *scale_cor=array([], dtype=float64)*) [source]

This function calculates the evolution of one single step in the chemical evolution.

`save_data`(*header=[]*, *data=[]*, *filename='plot_data.txt'*) [source]

Writes data into a text file. data entries can have different lengths