FlowPM

FlowPM Developers

Aug 31, 2021

API DOCUMENTATION

CHAPTER

ONE

FLOWPM PUBLIC API

1.1 flowpm package

1.1.1 flowpm.constants

Created on Jun 12, 2013 @author: Francois Lanusse <francois.lanusse@cea.fr>

C_1: Instrinsic alignment normalisation constant [(h^2 M_sun Mpc^{-3})^{-1}], see Kirk et al 2010. NB: Bridle & King report different units, but is a typo. c : Speed of light in [km/s] eta_nu: ratio of energy density in neutrinos to energy in photons h0: Hubble constant in [km/s/(h^{-1} Mpc)] rh : Hubble radius in [h^{-1} Mpc] rho_crit: Critical density of Universe in units of [h^2 M_sun Mpc^{-3}]. tcmb : Temperature of the CMB today in [K]

1.1.2 flowpm.cosmology

`class flowpm.cosmology.Cosmology(Omega_c, Omega_b, h, n_s, sigma8, Omega_k, w0, wa)`

Bases: `object`

Cosmology object, stores primary and derived cosmological parameters.

Parameters

- **Omega_c** – *float* representing the cold dark matter density fraction.
- **Omega_b** – *float* representing the baryonic matter density fraction.
- **h** – *float* representing Hubble constant divided by 100 km/s/Mpc; unitless.
- **n_s** – *float* representing the primordial scalar perturbation spectral index.
- **sigma8** – *float* representing the variance of matter density perturbations at an 8 Mpc/h scale.
- **Omega_k** – *float* representing the curvature density fraction.
- **w0** – *float* representing the first order term of dark energy equation.
- **wa** – *float* representing the second order term of dark energy equation of state.

`property Omega`

`property Omega_b`

Baryonic matter density fraction.

`property Omega_c`

Cold dark matter density fraction.

property Omega_de

property Omega_k

property Omega_m

property h

property k

property n_s

property sigma8

property sqrtk

to_dict()

property w0

property wa

1.1.3 flowpm.tfbackground

TensorFlow implementation of Cosmology Computations

`flowpm.tfbackground.`**`D1`**(*cosmo*, *a*)

Normalised first order growth factor.

Parameters

- **cosmo** (*dict*) – Cosmology dictionary.
- **a** (*tf.TensorArray*) – Scale factor.

Returns **D1** (*scalar float Tensor*) – Normalised D1.

Notes

The expression for $D_{1norm}(a)$ is:

$$D_{1norm}(a) = \frac{D_1(a)}{D_1(a=1)}$$

`flowpm.tfbackground.`**`D1f`**(*cosmo*, *a*)

Derivative of the first order growth factor respect to scale factor a

Parameters

- **cosmo** (*dict*) – Cosmology dictionary.
- **a** (*tf.TensorArray*) – Scale factor.

Returns *Scalar float Tensor* – normalised derivative D1.

Notes

The expression for $D'_{1norm}(a)$ is:

$$D'_{1norm}(a) = \frac{D'_1(a)}{D_1(a=1)}$$

`flowpm.tfbackground.D2`(*cosmo*, *a*)

Normalised second order growth factor

Parameters

- **cosmo** (*dict*) – Cosmology dictionary.
- **a** (*tf.TensorArray*) – Scale factor.

Returns *Scalar float Tensor* – normalised D2.

Notes

The expression for $D_{2norm}(a)$ is:

$$D_{2norm}(a) = \frac{D_2(a)}{D_2(a=1)}$$

`flowpm.tfbackground.D2f`(*cosmo*, *a*)

Derivative of the second order growth factor respect to scale factor a

Parameters

- **cosmo** (*dict*) – Cosmology dictionary.
- **a** (*tf.TensorArray*) – Scale factor.

Returns *Scalar float Tensor* – normalised derivative D2.

Notes

The expression for $D'_{2norm}(a)$ is:

$$D'_{2norm}(a) = \frac{D'_2(a)}{D_2(a=1)}$$

`flowpm.tfbackground.E`(*cosmo*, *a*)

The scale factor dependent factor E(a) in the Hubble parameter.

Parameters

- **cosmo** (`Cosmology`) – Cosmological parameters structure
- **a** (*array_like or tf.TensorArray*) – Scale factor

Returns **E^2** (*Scalar float Tensor*) – Square of the scaling of the Hubble constant as a function of scale factor

Notes

The Hubble parameter at scale factor a is given by $H^2(a) = E^2(a)H_o^2$ where E^2 is obtained through Friedman's Equation (see **:cite:`2005:Percival`**) :

$$E(a) = sqrt(\Omega_m a^{-3} + \Omega_k a^{-2} + \Omega_{de} a^{f(a)})$$

where $f(a)$ is the Dark Energy evolution parameter computed by `f_de()`.

`flowpm.tfbackground.`**`Gf`**`(`*cosmo*, *a*`)`
FastPM growth factor function

Parameters

- **`cosmo`** (*`dict`*) – Cosmology dictionary.
- **`a`** (*`tf.TensorArray`*) – Scale factor.

Returns **Scalar float Tensor** (*FastPM growth factor function.*)

Notes

The expression for $Gf(a)$ is:

$$Gf(a) = D'_{1norm} * a * *3 * E(a)$$

`flowpm.tfbackground.`**`Gf2`**`(`*cosmo*, *a*`)`
FastPM second order growth factor function

Parameters

- **`cosmo`** (*`dict`*) – Cosmology dictionary.
- **`a`** (*`tf.TensorArray`*) – Scale factor.

Returns **Scalar float Tensor** (*FastPM second order growth factor function.*)

Notes

The expression for $Gf_2(a)$ is:

$$Gf_2(a) = D'_{2norm} * a * *3 * E(a)$$

`flowpm.tfbackground.`**`H`**`(`*cosmo*, *a*`)`
Hubble parameter [km/s/(Mpc/h)] at scale factor *a*

Parameters

- **`cosmo`** (`Cosmology`) – Cosmological parameters structure
- **`a`** (*`array_like or tf.TensorArray`*) – Scale factor

Returns **H** (*Scalar float Tensor*) – Hubble parameter at the requested scale factor.

`flowpm.tfbackground.`**`Omega_de_a`**`(`*cosmo*, *a*`)`
Dark Energy density at scale factor *a*.

Parameters

- **`cosmo`** (`Cosmology`) – Cosmological parameters structure

- **a** (*array_like or tf.TensorArray*) – Scale factor

Returns **Omega_de** (*Scalar float Tensor*) – Dark Energy density at the requested scale factor

Notes

The evolution of Dark Energy density $\Omega_{de}(a)$ is given by:

$$\Omega_{de}(a) = \frac{\Omega_{0,de} a^{f(a)}}{E^2(a)}$$

where $f(a)$ is the Dark Energy evolution parameter computed by `f_de()` (see **:cite:`2005:Percival`** Eq. (6)).

`flowpm.tfbackground.Omega_m_a(cosmo, a)`

Matter density at scale factor *a*.

Parameters

- **cosmo** (`Cosmology`) – Cosmological parameters structure
- **a** (*array_like or tf.TensorArray*) – Scale factor

Returns **Omega_m** (*Scalar float Tensor*) – Non-relativistic matter density at the requested scale factor

Notes

The evolution of matter density $\Omega_m(a)$ is given by:

$$\Omega_m(a) = \frac{\Omega_{0,m} a^{-3}}{E^2(a)}$$

see **:cite:`2005:Percival`** Eq. (6)

`flowpm.tfbackground.a_of_chi(cosmo, chi)`

Computes the scale factor for corresponding (array) of radial comoving distance by reverse linear interpolation.

Parameters

- **cosmo** (`Cosmology`) – Cosmological parameters
- **chi** (*array_like or tf.TensorArray*) – radial comoving distance to query.

Returns **a** (*tf.TensorArray*) – Scale factors corresponding to requested distances

`flowpm.tfbackground.angular_diameter_distance(cosmo, a)`

Angular diameter distance in [Mpc/h] for a given scale factor.

Parameters

- **cosmo** (`Cosmology`) – Cosmological parameters structure
- **a** (*tf.TensorArray*) – Scale factor

Returns **d_A** (*tf.TensorArray*)

Notes

Angular diameter distance is expressed in terms of the transverse comoving distance as:

$$d_A(a) = a f_k(a)$$

`flowpm.tfbackground.dEa(cosmo, a)`

Derivative of the scale factor dependent factor E(a) in the Hubble parameter with respect to the scale factor.

Parameters

- **cosmo** (Cosmology) – Cosmological parameters structure
- **a** (*array_like or tf.TensorArray*) – Scale factor

Returns **dE(a)/da** (*Scalar float Tensor*) – Derivative of the scale factor dependent factor in the Hubble parameter with respect to the scale factor.

Notes

The expression for $\frac{dE}{da}$ is:

$$\frac{dE(a)}{da} = \frac{-3a^{-4}\Omega_{0m} - 2a^{-3}\Omega_{0k} + f'_{de}\Omega_{0de}a^{f_{de}(a)}}{2E(a)}$$

`flowpm.tfbackground.dchioverda(cosmo, a)`

Derivative of the radial comoving distance with respect to the scale factor.

Parameters

- **cosmo** (Cosmology) – Cosmological parameters structure
- **a** (*array_like or tf.TensorArray*) – Scale factor

Returns **dchi/da** (*tf.TensorArray*) – Derivative of the radial comoving distance with respect to the scale factor at the specified scale factor.

Notes

The expression for $\frac{d\chi}{da}$ is:

$$\frac{d\chi}{da}(a) = \frac{R_H}{a^2 E(a)}$$

`flowpm.tfbackground.dfde(cosmo, a, epsilon=1e-05)`

Derivative of the evolution parameter for the Dark Energy density f(a) with respect to the scale factor.

Parameters

- **cosmo** (Cosmology) – Cosmological parameters structure
- **a** (*array_like or tf.TensorArray*) – Scale factor
- **epsilon** (*float value*) – Small number to make sure we are not dividing by 0 and avoid a singularity

Returns **df(a)/da** (*Scalar float Tensor*) – Derivative of the evolution parameter for the Dark Energy density with respect to the scale factor.

Notes

The expression for $\frac{df(a)}{da}$ is:

`flowpm.tfbackground.f1(cosmo, a)`
Linear order growth rate

Parameters

- **cosmo** (*dict*) – Cosmology dictionary.
- **a** (*tf.TensorArray*) – Scale factor.

Returns *Scalar float Tensor* – Linear order growth rate.

Notes

The expression for $f_1(a)$ is:

$$f1(a) = \frac{D_1'(a)}{D_1(a=1)} * a$$

`flowpm.tfbackground.f2(cosmo, a)`
Second order growth rate.

Parameters

- **cosmo** (*dict*) – Cosmology dictionary.
- **a** (*tf.TensorArray*) – Scale factor.

Returns *Scalar float Tensor* – Linear order growth rate.

Notes

The expression for $f_2(a)$ is:

$$f2(a) = \frac{D_2'(a)}{D_2(a=1)} * a$$

`flowpm.tfbackground.fde(cosmo, a, epsilon=1e-05)`
Evolution parameter for the Dark Energy density.

Parameters

- **cosmo** (Cosmology) – Cosmological parameters structure
- **a** (*array_like or tf.TensorArray*) – Scale factor
- **epsilon** (*float value*) – Small number to make sure we are not dividing by 0 and avoid a singularity

Returns **f** (*Scalar float Tensor.*) – The evolution parameter of the Dark Energy density as a function of scale factor

Notes

For a given parametrisation of the Dark Energy equation of state, the scaling of the Dark Energy density with time can be written as:

$$\rho_{de}(a) \propto a^{f(a)}$$

(see **:cite:`2005:Percival`**) where $f(a)$ is computed as $f(a) = \frac{-3}{\ln(a)} \int_0^{\ln(a)} [1 + w(a')] d \ln(a')$. In the case of Linder's parametrisation for the dark energy $f(a)$ becomes:

$$f(a) = -3(1 + w_0) + 3w \left[\frac{a-1}{\ln(a)} - 1 \right]$$

`flowpm.tfbackground.`**`gf`**(*cosmo*, *a*)

Derivative of Gf against a

Parameters

- **cosmo** (*dict*) – Cosmology dictionary.
- **a** (*tf.TensorArray*) – Scale factor.

Returns **Scalar float Tensor** (*the derivative of Gf against a.*)

Notes

The expression for $gf(a)$ is:

$$gf(a) = \frac{dGF}{da} = D_1^{''} * a * *3 * E(a) + D'_{1norm} * a * *3 * E'(a) + 3 * a * *2 * E(a) * D'_{1norm}$$

`flowpm.tfbackground.`**`gf2`**(*cosmo*, *a*)

Derivative of Gf2 against a

Parameters

- **cosmo** (*dict*) – Cosmology dictionary.
- **a** (*tf.TensorArray*) – Scale factor.

Returns **Scalar float Tensor** (*the derivative of Gf2 against a.*)

Notes

The expression for $gf2(a)$ is:

$$gf_2(a) = \frac{dGF_2}{da} = D_2^{''} * a * *3 * E(a) + D'_{2norm} * a * *3 * E'(a) + 3 * a * *2 * E(a) * D'_{2norm}$$

`flowpm.tfbackground.`**`growth_ode`**(*a*, *y*, ***kwcosmo*)

Define the ode functions that will be used to compute the linear growth factor D_1(a) and second-order growth factor D_2(a) at a given scale factor.

Parameters

- **a** (*Contain the value of y for each desired scale factors in*) – Scale factor

- **y** (*tf.TensorArray*) –
- **a** –
- **row** (*with the initial value y0 in the first*) –
- **cosmo** (Cosmology) – Cosmological parameters structure

Notes

Linear growth factor D_1(a) is given by .. math:: a^2frac{d^2 D_1}{da^2}+ left(Omega_{Lambda}(a)- frac{ Omega_{m}(a)}{2} +2 right) a frac{dD_1}{da}=frac{3}{2} Omega_{m}(a)D_1

(see **:cite:`Florent Leclercq thesis`** Eq. (1.96))

`flowpm.tfbackground.`**`maybe_compute_ODE`**(*cosmo*, *log10_amin=- 2*, *steps=256*)
Either computes or returns the cached ODE solution

`flowpm.tfbackground.`**`odesolve_func`**(*a*, *rtol=0.0001*, ***kwcosmo*)
Solves the growth ODE system for a given cosmology at the requested scale factors.

Parameters

- **a** (*array_like*) – Output scale factors, note that the ODE is initialized at a[0]
- **rtol** (*float, optional*) – Parameters determing the error control performed by the solver
- **kwcosmo** (*keyword args*) – Cosmological parameter values.

Returns **(D1, D1f), (D2, D2f)** (*dictionary*) – First and second order growth factors, and their derivatives, computed at the requested scale factors.

`flowpm.tfbackground.`**`rad_comoving_distance`**(*cosmo*, *a*, *log10_amin=- 3*, *steps=256*, *rtol=0.001*)
Radial comoving distance in [Mpc/h] for a given scale factor.

Parameters

- **cosmo** (Cosmology) – Cosmological parameters structure
- **a** (*array_like or tf.TensorArray*) – Scale factor
- **log10_amin** (*integer*) – Starting value of the log-scale spaced sequence.
- **steps** (*integer, optional*) – Number of samples to generate.
- **rtol** (*float, optional*) – Parameters determing the error control performed by the solver

Returns **chi** (*tf.TensorArray,*) – Radial comoving distance corresponding to the specified scale factor.

Notes

The radial comoving distance is computed by performing the following integration:

$$\chi(a) = R_H \int_a^1 \frac{da'}{a'^2 E(a')}$$

`flowpm.tfbackground.transverse_comoving_distance(cosmo, a)`

Transverse comoving distance in [Mpc/h] for a given scale factor.

Parameters

- **cosmo** (`Cosmology`) – Cosmological parameters
- **a** (*tf.TensorArray*) – Scale factor

Returns **f_k** (*tf.TensorArray*) – Transverse comoving distance corresponding to the specified scale factor.

Notes

The transverse comoving distance depends on the curvature of the universe and is related to the radial comoving distance through:

$$f_k(a) = \begin{cases} R_H \frac{1}{\sqrt{\Omega_k}} \sinh(\sqrt{|\Omega_k|}\chi(a) R_H) & \text{for } \Omega_k > 0 \\ \chi(a) & \text{for } \Omega_k = 0 \\ R_H \frac{1}{\sqrt{\Omega_k}} \sin(\sqrt{|\Omega_k|}\chi(a) R_H) & \text{for } \Omega_k < 0 \end{cases}$$

`flowpm.tfbackground.w(cosmo, a)`

Dark Energy equation of state parameter using the Linder parametrisation.

Parameters

- **cosmo** (`Cosmology`) – Cosmological parameters structure
- **a** (*array_like or tf.TensorArray*) – Scale factor

Returns **w** (*Scalar float Tensor*) – The Dark Energy equation of state parameter at the specified scale factor

Notes

The Linder parametrization **:cite:`2003:Linder`** for the Dark Energy equation of state $p = w\rho$ is given by:

$$w(a) = w_0 + w(1 - a)$$

1.1.4 flowpm.tfpm

Core FastPM elements

`flowpm.tfpm.linear_field`(*nc*, *boxsize*, *pk*, *kvec=None*, *batch_size=1*, *seed=None*, *dtype=tf.float32*, *name='LinearField'*)

Generates a linear field with a given linear power spectrum.

nc: int, or list of ints Number of cells in the field. If a list is provided, number of cells per dimension.

boxsize: float, or list of floats Physical size of the cube, in Mpc/h.

pk: interpolator Power spectrum to use for the field

kvec: array k_vector corresponding to the cube, optional

batch_size: int Size of batches

seed: int Seed to initialize the gaussian random field

dtype: tf.dtype Type of the sampled field, e.g. tf.float32 or tf.float64

Returns **linfield** (*tensor (batch_size, nc, nc, nc)*) – Realization of the linear field with requested power spectrum

`flowpm.tfpm.lpt_init`(*cosmo*, *linear*, *a*, *order=2*, *kvec=None*, *name='LPTInit'*)

Estimate the initial LPT displacement given an input linear (real) field

TODO: documentation

`flowpm.tfpm.nbody`(*cosmo*, *state*, *stages*, *nc*, *pm_nc_factor=1*, *return_intermediate_states=False*, *name='NBody'*)

Integrate the evolution of the state across the givent stages

cosmo: cosmology Cosmological parameter object

state: tensor (3, batch_size, npart, 3) Input state

stages: array Array of scale factors

nc: int, or list of ints Number of cells

pm_nc_factor: int Upsampling factor for computing

return_intermediate_states: boolean If true, the fruction will return each intermediate states, not only the last one.

Returns **state** (*tensor (3, batch_size, npart, 3), or list of states*) – Integrated state to final condition, or list of intermediate steps

1.1.5 flowpm.tfpower

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

INDEX

W