growin

Release 0.1-beta.1

Aug 05, 2021

Contents:

CHAPTER 1

API

1.1 Growin

functions and variables to run the requisite models and functions to calculate the drifts and and from them run the model and then compute the rtgr

`growin._core.`**`fit_fejer`**(*year*, *day*, *lon*)

Compute the fourier coefficients for the Fejer-Scherliess model

Parameters

year [(int)] year to use for Fejer-Scherliess drifts

day [int] julian day

lon [int or double] longitude in degrees

`growin._core.`**`get_drifts`**(*start=2008*, *stop=2014*, *clean_level='none'*, *drift_inst=None*, *drift_key='ionVelocityZ'*, *season_names: list = None*, *season_bounds: list = None*, *zone_names: list = None*, *zone_bounds: list = None*, *offset: int = None*)

create/load the instrument and obtain the drifts then save the drifts

Parameters

start [(int)] start year of the survey

stop [(int)] stop year of the survey

clean_level [(string)] specify cleaning routine for pysat

drift_inst: (growin.fourier_exb.DriftInstrument) drift instrument if different custom modifier functions are desired

drift_key [(int)] dictionary key for the pysat instrument drift values to use

season_names [(array-like of strings)] array-like containing the names of the specified seasons

season_bounds [(array-like of int or float)] array-like ocntaining the days that delineate the season bounds

zone_names [(array-like of strings)] array-like of stings specifying the names of longitude zones used

zone_bounds [(array-like of int or float)] array-like of longitudes in degrees that delineate the zone bounds

`growin._core.`**`get_growth`**(*tag*, *day*, *year*, *lon*, *exb_drifts=None*, *ve01=0*, *f10=120.0*)

get the sami instrument with growth rates calculated checks if there is an existing sami instrument with the appropriate tag and loads it. Otherwise it runs the growth rate calculation.

Parameters

tag [(string)] name of run where growth is/will be archived

day [(int)] day of year for SAMI run

year [(int)] year for SAMI run

lon [(int)] geo longitude in degrees for SAMI run

exb_drifts [(10x2 ndarray of floats)] Matrix of Fourier series coefficients dependent on solar local time (SLT) in hours where exb_total = exv_drifts[i,0]*cos((i+1)*pi*SLT/12) + exb_drifts[i,1]*sin((i+1)*pi*SLT/12)

ve01 [(float)] offset for Fourier exb drifts, not used by default therefore we are assuming net zero vertical drift

`growin._core.`**`get_growth_rates_survey`**(*start: int*, *stop: int*, *clean_level: str*, *drift_key: str*, *season_names: list*, *season_bounds: list*, *season_days: dict*, *zone_names: list*, *zone_bounds: list*, *zone_lons: dict*)

calculate the growth rate from the sami model using computed drifts run the model for each year and season compute the growth rate and plot

Parameters

start [(int)] start year of the survey

stop [(int)] stop year of the survey

clean_level [(string)] specify cleaning routine for pysat

drift_key [(int)] dictionary key for the pysat instrument drift values to use a good default for cnofs is 'IonVelmeridional'

season_names [(array-like of strings)] array-like containing the names of the specified seasons

season_bounds [(array-like of int or float)] array-like ocntaining the days that delineate the season bounds

season_days [(dict)] dictionary with season names as keys, and as values the day to be used by SAMI

zone_names [(array-like of strings)] array-like of stings specifying the names of longitude zones used

zone_bounds [(array-like of int or float)] array-like of longitudes in degrees that delineate the zone bounds

zone_lons [(dict)] dictionary with zone names as keys, and as values the longitude to be used by SAMI

1.2 Growth Rate

Calculation of altitude profiles of flux tube integrated growth rates of the Rayleigh Taylor instability using SAMI2 output for ion densities, and electron temperatures

1.2.1 Author

1.2.2 Jonathon Smith (JS), 20 Sep 2018, Goddard Space Flight Center (GSFC)

class growin.growth_rate.**FluxTube**(*sami_data*, *ft*, *max_alt*, *exb*)

flux tube integrated (fti) quantities all of these have the same shape that is the number of altitude bins by the number of time steps

class growin.growth_rate.**FluxTubeCell**(*sami_data*, *ftl*, *ft*, *iyd*, *d_str*, *t_step*)

one cell or bin of a flux tube from the sami model and all of its attributes are stored in an object for easy reference and use this includes all of the density values and the location coordinates to specify this cell

growin.growth_rate.**calc_growth_rate**(*tube*)

the growth rate equation from Sultan 96

Parameters

tube [(FluxTube)] flux tube object

Variables used

———-

sig_F_P [(float)] flux tube integrated pedersen cond. F region in mho

sig_total [(float)] flux tube integrated pedersen cond. total in mho

V_P [(float)] flux tube integrated vertical drift or drift at apex altitude m/s

U_L [(float)] flux tube integrated neutral wind perp. B in L direction in m/s

g_e [(float)] gravtiy at apex altitude in m/s^2

nu_eff [(float)] collision frequency in s-1

K_F [(float)] altitude gradient in density in m-1

growin.growth_rate.**eval_tubes**(*sami*, *exb*, *t_step=0*)

calculate the flux tube integrated quantities for each flux tube needed for the growth rate calculation

Parameters

sami [(sami2py.Model)] sami2py model object

t_step [(int)] array index for sami2py object timestep variable

growin.growth_rate.**exb_calc**(*coefficients*, *ve01*, *t*)

Parameters

coefficients [(array)] 10x2 array of Fourier coefficients

ve01 [(float)] flat offset for fourier function 0 by default

t [(float)] time in hours

`growin.growth_rate.`**`format_dates`**(*sami*, *t_step*)

returns the date in all the required formats for different packages

Parameters

sami [(sami2py.Model)] the sami model being used to calculate growth rates

t_step [(int)] time step for the sami model object

`growin.growth_rate.`**`ft_bin_loc`**(*sami_data*, *ftl*, *ft*)

returns the location and spatial extent of current bin

Parameters

sami_data [(xarray.core.dataset.Dataset)] the sami model being used to calculate growth rates

ftl [(float)] 'flux tube length' index along the length of a flux tube for SAMI

ft [(int)] 'flux tube' flux tube index for SAMI

`growin.growth_rate.`**`ft_length`**(*alt_1*, *alt_2*, *lat_1*, *lat_2*)

law of cosines for determining linear extent of flux tube (ft) bin

Parameters

alt_1 [(float)] altitude of current flux tube cell in km

alt_2 [(float)] altitude of next flux tube cell on same flux tube in km

lat_1 [(float)] latitude of current flux tube cell in degrees

lat_2 [(float)] latitude of next flux tube cell on same flux tube in degrees

`growin.growth_rate.`**`g_e_L`**(*sami_data*, *ft*)

gravity at the bin altitude L is geocentric distance in earth radii the L shell at the local altitude not apex.

Parameters

sami_data [(xarray.dataset.Dataset)] the sami model being used to calculate growth rates

ft [(int)] 'flux tube' flux tube index for SAMI

`growin.growth_rate.`**`get_n_n`**(*nn*)

Parameters

nn [(array)] neutral number densities in cm-3 there was an old note here that they were in m-3, but this was likely before this quantity was build into sami2py

`growin.growth_rate.`**`nu_e`**(*n_n*, *n_e*, *T_e*)

approximate calculation of electron collision frequency from Kelly 89

Parameters

n_n [(float)] neutral density cm-3

n_e [(float)] electron density cm-3

T_e [(float)] electron temperature K

`growin.growth_rate.`**`nu_i`**(*n_i*, *n_n*, *A*)

approximate calculation of ion collision frequency from Kelley 89

Parameters

n_i [(float)] ion density cm-3

n_n [(float)] neutral density cm-3

A [(int)] mean neutral molecular mass in atomic mass units

growin.growth_rate.**omega**(*B*, *particle*)

Parameters

B [(float)] total electron density cm-3

particle [(str)] particle name to get the correct gyrofrequency

growin.growth_rate.**r_local**(*denis*, *alt*)

Local recombination from Sultan eq 21 alpha*n_mol Risbeth & Garriott '69: Dissociative recombination is the principal E and F region loss mechansim. Huba '96: RTI not damped by recombination in F region n_mol is the concentration of molecular ions alpha = 2*10**(-7) according to Sultan '92 alpha ~ 10**(-7) according to Risbeth & Garriott '69

growin.growth_rate.**rt_growth_rate**(*sami*, *exb*, *t_step=0*)

calculate flux tube integrated electron density altitude gradient and flux tube integrated growth rate. These are done together to avoid iterating over all of the flux tubes twice to do each of these calculations individually. Further this is because K cannot be calculated using this method in the eval_tubes function.

Parameters

sami_out [(sami2py.Model)]

exb [(float)] vertical plasma drift at the apex of flux tube

t_step [(int)] time step for SAMI2

growin.growth_rate.**run_growth_calc**(*sami*, *coefficients=None*, *ve01=0*)

runs the growth rate calculation for a sami2 run. Requires external drift information until exb drifts from sami2 are an archived data prod.

Parameters

sami [(sami2py.Model)] sami2py model object

coefficients [(array)] 10x2 array of fourier coefficients describe the vertical drift function

ve01: (float) offset or 0th term of fourier fit

growin.growth_rate.**run_models**(*sami*, *lat*, *lon*, *alt*, *cell*, *flux_tube*, *d_str*, *t_step*)

run all required models to get quantities not contained in SAMI2 vestigial inclusion of neutral density here from before SAMI2 offered it

Parameters

sami [(sami2py.Model)] sami2 model output

lat [(float)] latitude where model is to be run

lon [(float)] longitude where model is to be run

alt [(float)] altitude where model is to be run

cell [(int)] index for the cell along the flux tube

flux_tube [(int)] index for the flux tube

d_str [(string)] string versionn of date

t_step [(int)] time step for sami2

growin.growth_rate.**sigma_tot**(*denis*, *n_n*, *n_e*, *B*, *A*, *T_e*)

calculate thetotal Pedersen conductivity at location in mho/m

Parameters

denis [(list)] ion densities from sami cm-3

n_n [(float)] total neutral density cm-3

n_e [(float)] total electron density cm-3

B [(float)] magnetic field in Teslas

A [(float)] average neutral density in amus

T_e: electron temperature in Kelvin

1.3 Drifts

class growin.fourier_exb.**DriftInstrument**(**args*, ***kwargs*)

Class that inherits from a pysat instrument that will have an attribute corresponding to the median drifts, their deviation, and the fourier curve fit for use in the SAMI2 model

Methods

`__call__`(*args, **kw)	Call self as a function.
compile_drifts()	Builds larger Dataset containg all of the drifts and fits over the entire user-specified annual range.
fit_drifts(start, stop, coords)	This gets the median drifts and the coefficients and puts them in
get_drifts([drift_key, num_co, lon_bins, . . .])	The Big Function. This is the main function that generates all of

compile_drifts()

Builds larger Dataset containg all of the drifts and fits over the entire user-specified annual range.

fit_drifts(*start*, *stop*, *coords*)

This gets the median drifts and the coefficients and puts them in an xarray Dataset. The drifts are first obtained using the pysat function pysat.ssnl.avg.median2D and then the fits are performed on the output and all of it is placed in xarray.DataArrays that are then combined into a data set for this Year/Season. The longitude coordinate is just the longitude values but will/can be later specified as names if desired.

Parameters

start [(datetime)] start date for median2d

stop [(datetime)] stop date for median2d

get_drifts(*drift_key=None*, *num_co=10*, *lon_bins=<sphinx.ext.autodoc.importer._MockObject object>*, *slt_bins=<sphinx.ext.autodoc.importer._MockObject object>*, *season_bins=<sphinx.ext.autodoc.importer._MockObject object>*, *season_names=None*, *zone_labels=None*, *start_year=None*, *stop_year=None*)

The Big Function. This is the main function that generates all of the data and organizes it for the drift attribute in the object.

Parameters

- **drift_key** [(string)] instrument key for vertical drift
- **num_co** [(int)] how many sin/cosine pairs for the fitdownload
- **slt_bins** [(array-like)] bin edges in local time
- **lon_bins** [(array-like)] bin edges in longitude
- **season_bins** [(array-like)] months used as edges of the season "bins"
- **season_names** [(array-like)] string names for season xarray coords
- **zone_labels** [(array-like)] string names for longitude xarray coords
- **start_year** [(int)] the year to start the seasonal averaging
- **stop_year** [(int)] the year to stop the seasonal averaging

growin.fourier_exb.**fourier_fit**(*local_times*, *median_drifts*, *num_co*)

Here the terms in the fourier fit are actually determined

Parameters

- **local_times** [(array-like)] xdim for fit; local time values
- **median_drifts** [(array-like)] ydim for fit; median drift values from data
- **num_co** [(int)] 'number of coefficients) how many sin/cosine pairs for the fit

growin.fourier_exb.**make_fourier**(*na*, *nb*)

The function for the curve fit

Parameters

- **na: (int)** number of cosine terms/coefficients
- **nb: (int)** number of sin terms/coefficients

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index