py12box

Release 0.2.0

Matt Rigby

Jul 18, 2022

CONTENTS:

CHAPTER

ONE

INSTALLATION

1.1 Installation instructions

The easiest way to install py12box is using pip or conda.

1.1.1 Checking your Python installation

py12box is developed and tested on Linux and MacOS platforms.

To install py12box, you first need to install Python >= 3.7. To check if you have Python 3.7 installed type;

```
python -V
```

1.1.2 Installing py12box

To install using conda:

```
conda install -c mrghg py12box
```

To install using pip:

```
pip install py12box
```

If installation has been successful, you should be able to import py12box in python, e.g.:

```
import py12box
```

CHAPTER

TWO

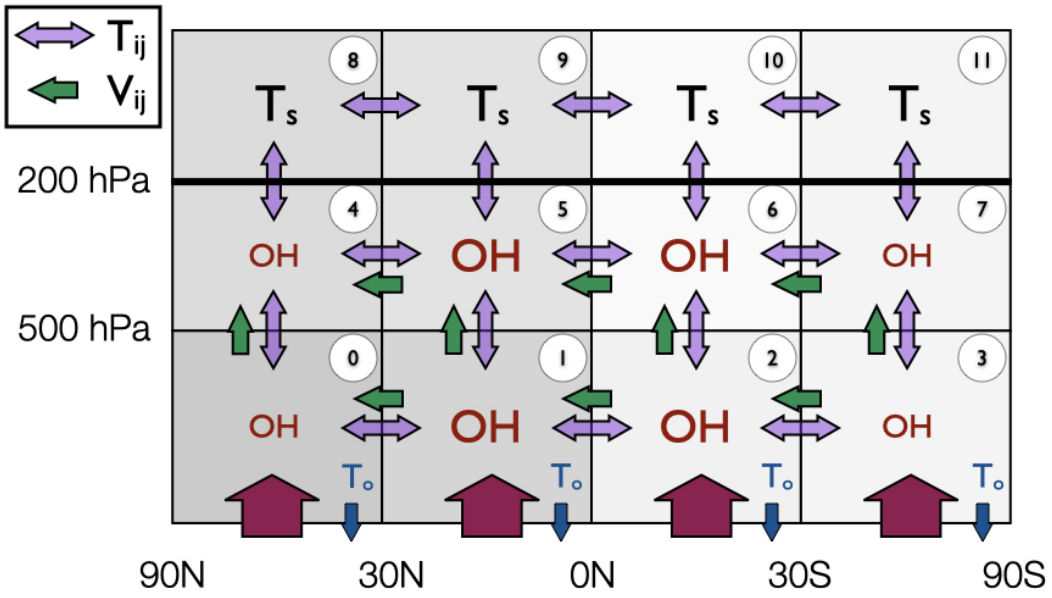
GETTING STARTED

2.1 py12box model usage

This notebook shows how to set up and run the AGAGE 12-box model.

2.1.1 Model schematic

The model uses advection and diffusion parameters to mix gases between boxes. Box indices start at the northern-most box and are as shown in the following schematic:



2.1.2 Model inputs

We will be using some synthetic inputs for CFC-11. Input files are in:

`data/example/CFC-11`

The location of this folder will depend on where you've installed py12box and your system. Perhaps the easiest place to view the contents is in the repository.

In this folder, you will see two files:

`CFC-11_emissions.csv`

`CFC-11_initial_conditions.csv`

As the names suggest, these contain the emissions, initial conditions and lifetimes.

Emissions

The emissions file has four columns: `year, box_1, box_2, box_3, box_4`.

The number of rows in this file determines the length of the box model simulation.

The `year` column should contain a decimal date (e.g. 2000.5 for ~June 2000), and can be monthly or annual resolution.

The other columns specify the emissions in Gg/yr in each surface box.

Initial conditions

The initial conditions file can be used to specify the mole fraction in pmol/mol (~ppt) in each of the 12 boxes.

2.1.3 How to run

Firstly import the `Model` class. This class contains all the input variables (emissions, initial conditions, etc., and run functions).

We are also importing the get_data helper function, only needed for this tutorial, to point to input data files.

[1]:
```
# Import from this package
from py12box.model import Model
from py12box import get_data

# Import matplotlib for some plots
import matplotlib.pyplot as plt
```

The `Model` class takes two arguments, `species` and `project_directory`. The latter is the location of the input files, here just redirecting to the "examples" folder.

The initialisation step may take a few seconds, mainly to compile the model.

[2]:
```
# Initialise the model
mod = Model("CFC-11", get_data("example/CFC-11"))
```

```
Compiling model and tuning lifetime...
... completed in 3 iterations
... stratospheric lifetime: 52.1
... OH lifetime: 1e12
... ocean lifetime: 1e12
... non-OH tropospheric lifetime: 1e12
... overall lifetime: 52.0
... done in 5.5602850914001465 s
```

Assuming this has compiled correctly, you can now check the model inputs by accessing elements of the model class. E.g. to see the emissions:

[3]:
```
mod.emissions
```

[3]:
```
array([[100,  10,   0,   0],
       [100,  10,   0,   0],
       [100,  10,   0,   0],
       ...,
       [  0,   0,   0,   0],
       [  0,   0,   0,   0],
       [  0,   0,   0,   0]])
```

In this case, the emissions should be a 4 x 12*n_years numpy array. If annual emissions were specified in the inputs, the annual mean emissions are repeated each month.

We can now run the model using:

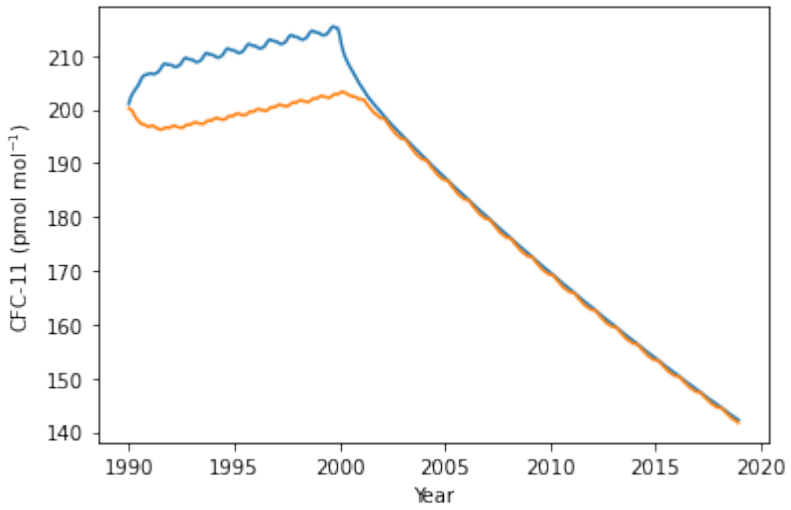
[4]:
```
# Run model
mod.run()
```

```
... done in 0.024969100952148438 s
```

The primary outputs that you'll be interested in are `mf` for the mole fraction (pmol/mol) in each of the 12 boxes at each timestep.

Let's plot this up:

[5]:
```
plt.plot(mod.time, mod.mf[:, 0])
plt.plot(mod.time, mod.mf[:, 3])
plt.ylabel("%s (pmol mol$^{-1}$)" % mod.species)
plt.xlabel("Year")
plt.show()
```

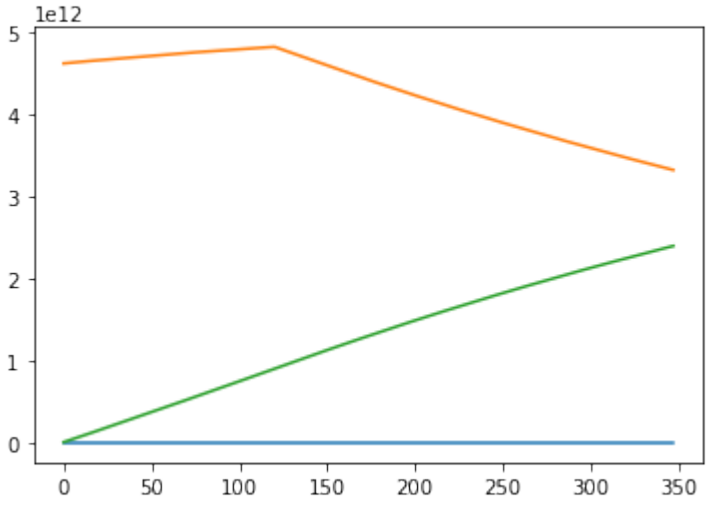


We can also view other outputs such as the burden and loss. Losses are contained in a dictionary, with keys: - `OH` (tropospheric OH losses) - `Cl` (losses via tropospheric chlorine) - `other` (all other first order losses)

For CFC-11, the losses are primarily in the stratosphere, so are contained in `other`:

[6]:
```
plt.plot(mod.emissions.sum(axis = 1).cumsum())
plt.plot(mod.burden.sum(axis = 1))
plt.plot(mod.losses["other"].sum(axis = 1).cumsum())
```

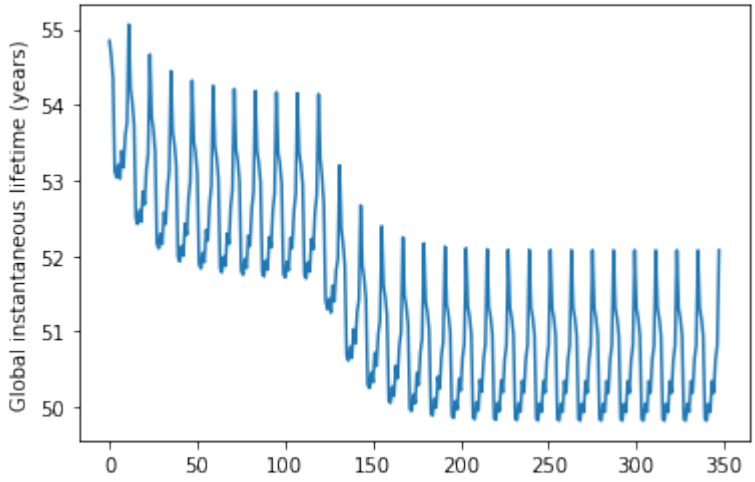
[6]:
```
[<matplotlib.lines.Line2D at 0x7f7824f23070>]
```



Another useful output is the lifetime. This is broken down in a variety of ways. Here we'll plot the global lifetime:

[7]:
```
plt.plot(mod.instantaneous_lifetimes["global_total"])
plt.ylabel("Global instantaneous lifetime (years)")
```

[7]:
```
Text(0, 0.5, 'Global instantaneous lifetime (years)')
```



2.1.4 Setting up your own model run

To create your own project, create a project folder (can be anywhere on your filesystem). The folder must contain two files:

`<species>_emissions.csv`

`<species>_initial_conditions.csv`

To point to the new project, py12box will expect a pathlib.Path object, so make sure you import this first:

```
[ ]: from pathlib import Path
new_model = Model("<SPECIES>", Path("path/to/project/folder"))
```

Once set up, you can run the model using:

```
[ ]: new_model.run()
```

Note that you can modify any of the model inputs in memory by modifying the model class. E.g. to see what happens when you double the emissions:

```
[ ]: new_model.emissions *= 2.
new_model.run()
```

2.1.5 Changing lifetimes

If no user-defined lifetimes are passed to the model, it will use the values in `data/inputs/species_info.csv`

However, you can start the model up with non-standard lifetimes using the following arguments to the `Model` class (all in years):

`lifetime_strat`: stratospheric lifetime

`lifetime_ocean`: lifetime with respect to ocean uptake

`lifetime_trop`: non-OH losses in the troposphere

e.g.:

```
[ ]: new_model = Model("<SPECIES>", Path("path/to/project/folder"), lifetime_strat=100.)
```

To change the tropospheric OH lifetime, you need to modify the `oh_a` or `oh_er` attributes of the `Model` class.

To re-tune the lifetime of the model in-memory, you can use the `tune_lifetime` method of the `Model` class:

```
[ ]: new_model.tune_lifetime(lifetime_strat=50., lifetime_ocean=1e12, lifetime_trop=1e12)
```

CHAPTER

THREE

PY12BOX PACKAGE DETAILS

3.1 py12box

3.1.1 py12box package

Submodules

py12box.core module

py12box.core.**model**(*ic*, *q*, *mol_mass*, *lifetime*, *F*, *temp*, *oh*, *cl*, *arr_oh=array([1.0e-30, - 1.8e+03])*, *arr_cl=array([1.0e-30, - 1.8e+03])*, *dt=172800.0*, *mass=array([6.39625013e+20, 6.39625013e+20, 6.39625013e+20, 6.39625013e+20, 3.83775007e+20, 3.83775007e+20, 3.83775007e+20, 3.83775007e+20, 2.55850005e+20, 2.55850005e+20, 2.55850005e+20, 2.55850005e+20])*, *nsteps=- 1*)

Main model code

Parameters

- **ic** (*ndarray*) – 1d, n_box Initial conditions of each boxes (pmol/mol) (~pptv).
- **q** (*ndarray*) – 2d, n_months x n_box_surface Monthly emissions into the surface boxes (Gg/yr). The length of the simulation is determined by the length of q.
- **mol_mass** (*float*) – Single value of molecular mass (g/mol).
- **lifetime** (*ndarray*) – 2d, n_months x n_box Non-OH first-order lifetimes for each month in each box (years).
- **F** (*ndarray*) – 3d, month x n_box x n_box Transport parameter matrix.
- **temp** (*ndarray*) – 2d, month x n_box Temperature (K).
- **oh** (*ndarray*) – 2d, month x n_box Chlorine and OH radical concentrations (molec cm^-3).
- **cl** (*ndarray*) – 2d, month x n_box Chlorine and OH radical concentrations (molec cm^-3).
- **arr_OH** (*ndarray, optional*) – 1d, 2 Arrhenius A and E/R constants for (X + sink) reactions.
- **arr_Cl** (*ndarray, optional*) – 1d, 2 Arrhenius A and E/R constants for (X + sink) reactions.
- **mass** (*ndarray*) – 1d, n_box Air mass of individual boxes.
- **dt** (*float, optional*) – Delta time (s).

- **nsteps** (*int*) – Number of timesteps to run since simulation start (default=-1, which ignores this argument)

Returns

- *ndarray* – 2d, n_months x output_boxes Monthly mean mole fractions (pmol/mol).
- *ndarray* – 2d, n_months x output_boxes Instantaneous mole fraction at final step of each month (pmol/mol).
- *ndarray* – 2d, n_months x n_box The monthly-average global burden (g).
- *ndarray* – 2d, n_months x n_box The monthly-average mass emissions (g).
- *ndarray* – 3d, n_resolved_losses x n_months x n_box. The monthly-average mass loss (g).
- *dict* – 3d, (n_resolved_losses+total) x n_months x n_box. Lifetimes calculated from individual time steps (year).

py12box.core.**model_solver_rk4**(*chi*, *F*, *dt*)

Vectorized Runge-Kutta

Parameters

- **chi** (*ndarray*) – 1d Burden (g).
- **F** (*ndarray*) – 2d, n_box Transport matrix.
- **dt** (*float*) – Delta t (s).

Returns **chi** – 1d Burden (g).

Return type ndarray

py12box.core.**model_transport_matrix**(*i_t*, *i_v1*, *t_in*, *v1_in*)

Calculate transport matrix

Based on equations in: Cunnold, D. M. et al. (1983). The Atmospheric Lifetime Experiment 3. Lifetime Methodology and Application to Three Years of CFCl3 Data. Journal of Geophysical Research, 88(C13), 8379-8400.

This function outputs a 12x12 matrix (F), calculated by collecting terms in the full equation scheme written out in doc_F_equation.txt. model transport is then calculated as dc/dt=F##c.

Parameters

- **i_t** (*ndarray*) – 2d, n_box_stag x 2 Intersections to apply to the mixing timescales for staggered grids.
- **i_v1** (*ndarray*) – 2d, n_box_stag_es x 2 Intersections to apply to the velocity timescales for staggered grids excluding stratosphere.
- **t** (*ndarray*) – 1d, n_box_stag Mixing timescales for staggered grids (s).
- **v1** (*ndarray*) – 1d, n_box_stag_es Velocity timescales for staggered grids excluding stratosphere (s).

Returns **F** – 2d, n_box x n_box Transport matrix

Return type ndarray

py12box.model module

class py12box.model.**Model**(*species*, *project_directory*, *species_param_file=None*, *lifetime_strat=None*, *lifetime_ocean=None*, *lifetime_trop=None*, *start_year=None*)

Bases: object

AGAGE 12-box model class

This class contains inputs and outputs of the 12-box model, for a particular species, emissions, initial conditions, etc.

mass
1d, 12 Mass of atmosphere in g in each box

Type ndarray

inputs_path
Path to the model parameter input directory

Type pathlib.Path

__init__(*species*, *project_directory*, *species_param_file=None*, *lifetime_strat=None*, *lifetime_ocean=None*, *lifetime_trop=None*, *start_year=None*)
Set up model class

Parameters:

species [str] Species name (e.g. "CFC-11") Must match string in data/inputs/species_info.csv

project_directory [pathlib.Path] Path to project directory, which contains emissions, lifetimes, etc.

species_param_file [str, optional] Species parameter file. Defaults to data/inputs/species_info.csv, by default None

lifetime_strat [float, optional] Stratospheric lifetime in years, by default None

lifetime_ocean: float, optional Lifetime with respect to loss to the ocean in years, by default None

lifetime_trop [float, optional] Lifetime with respect to non-OH tropospheric loss in years (e.g. photolysis), by default None

start_year [flt, optional] Optional year to start the model run. Must be after first year in emissions file. If specified, model will run using emissions and initial conditions value from file. Initial conditions will be updated to the new start year from model run.

Returns:

self : returns an instance of self.

Attributes:

mol_mass [float] Molecular mass

oh_a [float] OH "A" Arrhenuis parameter

oh_er [float] OH "E/R" Arrhenuis parameter

units [float] units for mole fraction (currently all stored at 1e-12 for ppt)

time [array] Array containing decimal times (1 x ntimesteps)

emissions [array] Array containing emissions (12 x ntimesteps)

ic [array] Initial conditions in each box

oh [array] OH concentration in each box for each month

cl [array] Cl concentration in each box for each month

temperature : Temperature in each box for each month

F [array] Transport matrix

lifetime [array] Global lifetime in each box (years)

steady_state_lifetime_strat : Global steady state stratospheric lifetime (years)

steady_state_lifetime_ocean : Global ocean steady state lifetime (years)

steady_state_lifetime_oh : Global steady state lifetime with respect OH loss (years)

steady_state_lifetime_cl : Global steady state lifetime with respect Cl loss (years)

steady_state_lifetime_othertrop : Global steady state lifetime with respect Cl loss (years)

steady_state_lifetime : Global steady state lifetime (years)

change_end_year(*end_year*)
Change last model year

Parameters **end_year** (*flt*) – New end year for simulation. Note that the simulation will be trimmed before the beginning of end_year. I.e. end_year=2001. will curtail the simulation at the end of December 2000.

change_start_year(*start_year*)
Change first model year

Parameters **start_year** (*flt*) – New first year for simulation

inputs_path = PosixPath('/home/docs/checkouts/readthedocs.org/user_builds/py12box/envs

mass = array([6.39625013e+20, 6.39625013e+20, 6.39625013e+20, 6.39625013e+20, 3.837750

run(*nsteps=- 1*, *verbose=True*)
Run 12-box model

Parameters:

nsteps [int, optional] Number of timesteps. Ignored if set to a negative value, by default -1

verbose [bool, optional] Toggle verbose output, by default True

Returns:

self : returns an instance of self.

Attributes:

mf [array] Monthly mean mole fractions (pmol/mol).

mf_restart : Instantaneous mole fraction at final step of each month (pmol/mol).

burden [array] The monthly-average global burden (g).

instantaneous_lifetimes [dict] Dictionary of monthly instantaneous lifetimes

losses [dict] Loss in each box for each month due to OH, Cl and other

emissions_model : The monthly-average mass emissions (g).

tune_lifetime(*lifetime_strat*, *lifetime_ocean*, *lifetime_trop*, *lifetime_relative_strat_file=PosixPath('/home/docs/checkouts/re*
packages/py12box-0.2.0-py3.8.egg/py12box/data/inputs/invlifetime_relative_strat.npy'),
threshold=10000.0, *tune_years=100*)
Tune the local non-OH lifetimes to a set of given global values

Parameters

- **lifetime_strat** (*float*) – Target global steady state stratospheric lifetime (years)
- **lifetime_ocean** (*float*) – Target global steady state lifetime with respect to ocean uptake (years)
- **lifetime_trop** (*float*) – Target global steady state lifetime with respect to non-OH tropospheric loss (years)
- **lifetime_relative_strat_file** (*pathlib, optional*) – File containing monthly relative loss rates in stratospheric boxes, by default get_data("inputs/invlifetime_relative_strat.npy")
- **threshold** (*float, optional*) – Above this threshold, lifetimes are ignored, and negligible loss is asssumed (years), by default 1e4
- **tune_years** (*int, optional*) – Number of years assumed to spin the model up to steady state, by default 100

Raises **Exception** – If an ocean and tropospheric lifetime are both specified. This isn't implemented yet

py12box.startup module

py12box.startup.**get_emissions**(*species*, *project_directory*)
Get emissions from project's emissions file

Parameters

- **species** (*str*) – Species name to look up emissions file in project folder (e.g. "CFC-11_emissions.csv")
- **project_directory** (*pathlib.Path*) – Path to 12-box model project

Returns

- *ndarray* – 1d, ntimesteps Array containing decimal times (1 x ntimesteps)
- *ndarray* – 2d, 12 x ntimesteps Array containing emissions (12 x ntimesteps)

py12box.startup.**get_initial_conditions**(*species*, *project_directory*)
Read initial conditions from file

Parameters

- **species** (*str*) – Species name
- **project_directory** (*pathlib.Path*) – Path to project folder, containing <species>_initial_conditions.csv file

Returns 1d, 12 Initial conditions in each box

Return type ndarray

py12box.startup.**get_model_parameters**(*n_years*, *input_dir=PosixPath('/home/docs/checkouts/readthedocs.org/user_buil packages/py12box-0.2.0-py3.8.egg/py12box/data/inputs')*)
Retrieve model transport parameters, OH, Cl and temperature and repeat annually

Parameters

- **n_years** (*int*) – Number of years over which to repeat parameters
- **input_dir** (*pathlib.Path, optional*) – Directory containing parameter files, by default get_data("inputs")

Returns

- *ndarray* – 3d, n_years, n_box_intersections_diffusion x 2 Intersections to apply to the mixing timescales for staggered grids.
- *ndarray* – 3d, n_years, n_box_intersections_advection x 2 Intersections to apply to the velocity timescales for staggered grids excluding stratosphere.
- *ndarray* – 2d, n_years, n_box_intersections_diffusion Mixing timescales for staggered grids (s).
- *ndarray* – 2d, n_years, n_box_intersections_diffusion_advection Velocity timescales for staggered grids excluding stratosphere (s).
- *ndarray* – 2d, n_years*12, 12 OH concentration in each box for each month
- *ndarray* – 2d, n_years*12, 12 Cl concentration in each box for each month
- *ndarray* – 2d, n_years*12, 12 Temperature in each box for each month

py12box.startup.**get_species_lifetime**(*species*, *which_lifetime*, *param_file=None*)

Get lifetimes for a specific species

Parameters

- **species** (*str*) – Species name. Must match species_info.csv
- **which_lifetime** (*str*) – Either "strat", "ocean" or "trop"
- **param_file** (*str, optional*) – Name of species info file, by default None, which sets species_info.csv

Returns Lifetime value

Return type float

Raises **Exception** – If which_lifetime is not a valid input

py12box.startup.**get_species_parameters**(*species*, *param_file=None*)

Get parameters for a specific species (e.g. mol_mass, etc.)

Parameters

- **species** (*str*) – Species name. Must match species_info.csv
- **param_file** (*str, optional*) – Name of species info file, by default None, which sets species_info.csv

Returns

- *float* – Molecular mass
- *float* – OH "A" Arrhenuis parameter
- *float* – OH "E/R" Arrhenius parameter
- *float* – unit (currently all stored at 1e-12 for ppt)

py12box.startup.**transport_matrix**(*i_t*, *i_v1*, *t*, *v1*)

Construct transport matrix from transport parameters

Parameters

- **i_t** (*ndarray*) – 3d, n_years x n_box_intersections_diffusion x 2 Intersections to apply to the mixing timescales for staggered grids.
- **i_v1** (*ndarray*) – 3d, n_years x n_box_intersections_advection x 2 Intersections to apply to the velocity timescales for staggered grids excluding stratosphere.

- **t** (*ndarray*) – 2d, n_years x n_box_intersections_diffusion Mixing timescales for staggered grids (s).
- **v1** (*ndarray*) – 2d, n_years x n_box_intersections_diffusion_advection Velocity timescales for staggered grids excluding stratosphere (s).

Returns n_months x 12 x 12 Transport matrix

Return type ndarray

py12box.startup.**zero_initial_conditions**()
Make an initial conditions dataframe with all boxes 1e-12

Returns Dataframe of initial conditions equal to 1e-12

Return type pandas.DataFrame

py12box.util module

py12box.util.**io_r_npy**(*fpath*, *mmap_mode='r'*)
Read npy file

Parameters **fpath** (*file-like object, string, or pathlib.Path*) – Path of the data file.

Returns **f** – Output data.

Return type ndarray

py12box.util.**io_r_npz**(*fpath*)
Read npz file

Parameters **fpath** (*file-like object, string, or pathlib.Path*) – Path of the data file.

Returns **f** – Output data.

Return type ndarray

Module contents

py12box.**get_data**(*sub_path*)
Get path to data files

Parameters **sub_path** (*str*) – path to data files, relative to py12box/data directory

Returns pathlib Path to data folder/file

Return type pathlib.Path

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

Symbols

C

G

I

M

P

R

T

Z