ExoPlaSim

Release 3.2.1post2

Adiv Paradise

May 22, 2023

CONTENTS

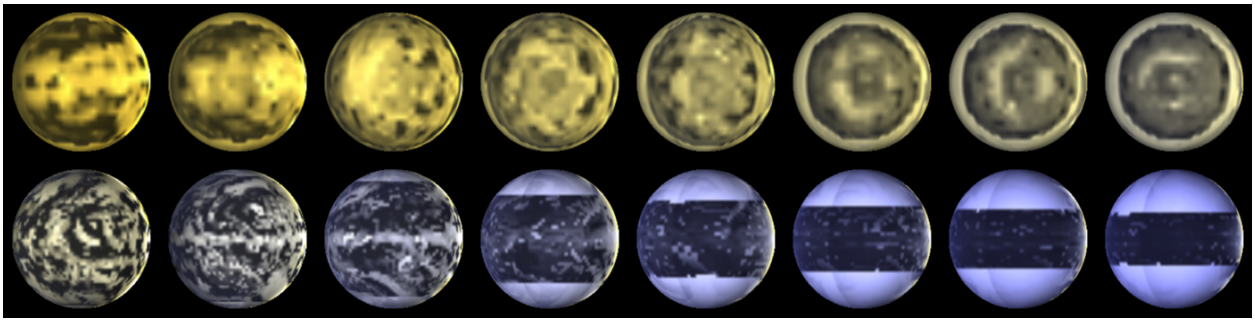


Fig. 1: A range of planets modeled by ExoPlaSim, and postprocessed with SBDART. The top row consists of tidally-locked aquaplanets at T21 orbiting stars ranging from 2500 K to 4000 K, with orbital periods increasing with stellar mass. The bottom row consists of aquaplanets with 24-hour rotation at T42, orbiting stars ranging from 4000 K to 8000 K.

- genindex
- Tutorial
- Postprocessor
- search

CHAPTER

ONE

CONTENTS

1.1 ExoPlaSim Tutorial

In this tutorial, we will model the habitable zone terrestrial planet TOI 700 d, and take a look at some of the data. This tutorial assumes that you have installed ExoPlaSim successfully, and have matplotlib installed.

Additionally to this tutorial, an IPython notebook is included with ExoPlaSim; which demonstrates basic usage. It can be found in the ExoPlaSim installation directory, or downloaded directly here.

1.1.1 Setting Up

First thing's first: we want to import ExoPlaSim, and instantiate our *Model* instance. We want to create our model run in the folder "toi700d_run", and run it at T21 resolution on 4 CPUs. We tell ExoPlaSim to use NumPy's compressed archive format for postprocessed output. Note that a large number of output formats are supported, including netCDF and HDF5, but those two formats require the additional installation of the `netCDF4` and `h5py` Python libraries (which can be done at install-time as optional dependencies for ExoPlaSim).

```
>>> import exoplasim as exo
>>> toi700d = exo.Model(workdir="toi700d_run",modelname="TOI-700d",
>>>                     ncpus=4,resolution="T21",outputtype=".npz")
```

If the appropriate executable does not yet exist, it will be compiled now. If this is the first time an ExoPlaSim model has been created, then a configuration script will be run first to locate the necessary compilers. We assign the model a descriptive name through the `modelname` argument, which is not strictly necessary, but will prove useful later.

Configuring the model for TOI-700d

TOI 700 d was discovered by the TESS telescope in January 2020 (Gilbert, et al 2020). It orbits TOI 700, a 3480 K M2V dwarf just over 100 lightyears away. TOI 700 has a luminosity of 0.0233 ± 0.0011 $L_\odot$, and is relatively quiet. TOI 700 d has the following parameters:

Radius	1.19 ± 0.11 $R_\oplus$
Mass	$1.72^{+1.29}_{-0.63}$ $M_\oplus$
Period	$37.4260^{+0.0007}_{-0.0010}$ days
Semi-major Axis	$0.163^{+0.0026}_{-0.0027}$ AU
Incident Flux	1367 W/m^2 $\left(\frac{L}{a^2}\right) \approx 1199$ W/m^2
Surface Gravity	9.81 m/s^2 $\left(\frac{M}{R^2}\right) \approx 11.9$ m/s^2

We don't know anything else about the planet, so we'll have to make some assumptions about the atmosphere and surface. For simplicity, we'll assume that the surface is entirely ocean-covered, and that the atmospheric mass scales

with planetary mass. We'll also assume that the atmosphere is N_2, CO_2, and H_2 O. The surface pressure relative to Earth can therefore be estimated as follows:

$$p_s \approx \frac{g}{g_\oplus}\left(\frac{M}{M_\oplus}\right)\left(\frac{R_\oplus}{R}\right)^2$$

This gives a surface pressure of approximately 1.47 bars. With that figured out, we can proceed to configure the model (right now it is configured with the barest of defaults–you should always configure the model, even if you pass no non-default arguments).

```
>>> toi700d.configure(startemp=3480.0, flux=1167.0,                          #␣
↪Stellar parameters
>>>                    eccentricity=0.,obliquity=0.,fixedorbit=True,          #␣
↪Orbital parameters
>>>                    synchronous=True,rotationperiod=37.426,                #␣
↪Rotation
>>>                    radius=1.19,gravity=11.9,aquaplanet=True,              # Bulk␣
↪properties
>>>                    pN2=1.47*(1-360e-6),pCO2=1.47*360e-6,ozone=False,      #␣
↪Atmosphere
>>>                    timestep=30.0,snapshots=720,physicsfilter="gp|exp|sp")  # Model␣
↪dynamics
>>> toi700d.exportcfg()
```

This command edits all the namelists and boundary condition files appropriately. The *exportcfg()* command writes a portable text configuration file, by default named `TOI-700d.cfg` using the model's `modelname` parameter, that another user could use to replicate our model by simply running `toi700d.loadconfig("TOI-700d.cfg")`.For a full description of the parameters we could have passed, see *exoplasim.Model.configure()*. Here is a brief overview of what each parameter did:

startemp = 3480.0 Specified the effective blackbody temperature of the star–in this case, 3480 K.

flux = 1167.0 Specified the incident flux (insolation or instellation) at the planet: 1167 W/m^2

eccentricity = 0.0 We set the orbital eccentricity to 0.

obliquity = 0.0 We set the planet's axial tilt to 0.

fixedorbit = True Here, we don't want the orbit precessing or anything, so we keep our orbit fixed.

synchronous = True, By setting this flag, we have told ExoPlaSim that this is a tidally-locked model. The default is for the Sun to be fixed in place over 180° longitude.

rotationperiod = 37.426 Since the planet is tidally-locked, we assume its rotation period matches its orbital period, 37.426 days.

radius = 1.19 We set the planet's radius to 1.19 Earth radii.

gravity = 11.9 We set the surface gravity to 11.9 m/s^2. Note that we do not specify the planet's mass directly, only the radius and surface gravity.

aquaplanet = True Setting this flag deletes all surface boundary condition files and tells ExoPlaSim to initialize an ocean everywhere. The default is to have a mixed-layer depth of 50 meters.

pN2 = 1.47*(1-360e-6) We want 1.47 bars **total**, but we want to include CO_2 as well. The surface pressure is the sum of the partial pressures, so we reduce pN_2 by the amount of CO_2 we want, the TOI 700 d equivalent of 360 μbars. We could also skip the 1.47 scaling and set the pressure directly through its own argument.

pCO2 = 1.47*360e-6 We set the CO_2 partial pressure to its Earth level in bars, scaled up.

ozone = False Since we are not assuming an oxygenated atmosphere (and some studies dispute how much ozone could be produced from an oxygenated atmosphere around an M dwarf anyway), we assume there will be no forcing from ozone. Tidally-locked models in ExoPlaSim are more stable without ozone anyway.

timestep = 30.0 Tidally-locked climates are stlightly more extreme than Earth-like climates, so rather than the default 45-minute timestep, we use 30 minutes.

snapshots = 720 Here we tell ExoPlaSim to write snapshot outputs every 720 timesteps (15 days). These snapshots show us the climate at a particular instant in time, and are therefore necessary for any observational postprocessing (any time-integrated observation is an average of photons that passed through the atmosphere as it was for a brief moment, not through the time-averaged atmosphere–this is mainly important for clouds). It's usually a good idea to write a snapshot every 15 days (twice a month), so scale based on the timestep. The default is to write every 480 timesteps, which is 15 days when a timestep is 15 minutes.

physicsfilter = "gp|exp|sp" Tidally-locked models can be subject to large-scale Gibbs oscillations on the night side, due to the strong dipole moment of the forcing and axial symmetry of the iceline. **All models will struggle to reproduce sharp features accurately**. ExoPlaSim merely struggles in an extremely visible way. Fortunately, we can mitigate this to an acceptable level with the use of *physics filters*. These are mathematical filters included in the dynamical core at the spectral transform stage. Here we have told ExoPlaSim to use an exponential filter, and to apply it both at the transform from gridpoint space to spectral space, and at the transform from spectral space back to gridpoint space. For more details on the choice of filter and how they work, see *exoplasim.Model.configure()*. For Earth-like models that aren't tidally-locked, physics filters are usually not necessary.

1.1.2 Running the Model

Now that we have configured the model, it's time to run it! This demo is intended to be something you can run on your laptop (thus specifying only 4 CPUs), so to make sure you have something to look at when you come back from your lunch break, let's just run for 10 years. On my laptop with 4 cores, a year takes just over 6 minutes. Note that on HPC architecture with 16 cores, a year often takes less than a minute.

```
>>> toi700d.run(years=10,crashifbroken=True)
```

The `crashifbroken` flag simply means that if something goes wrong, the model will crash in a slightly cleaner, Pythonic way. Note that a problem with the postprocessor will get flagged as a crash just like an actual model crash–in most cases, the model is salvageable if you figure out what went wrong with the postprocessor.

1.1.3 Inspecting the Data

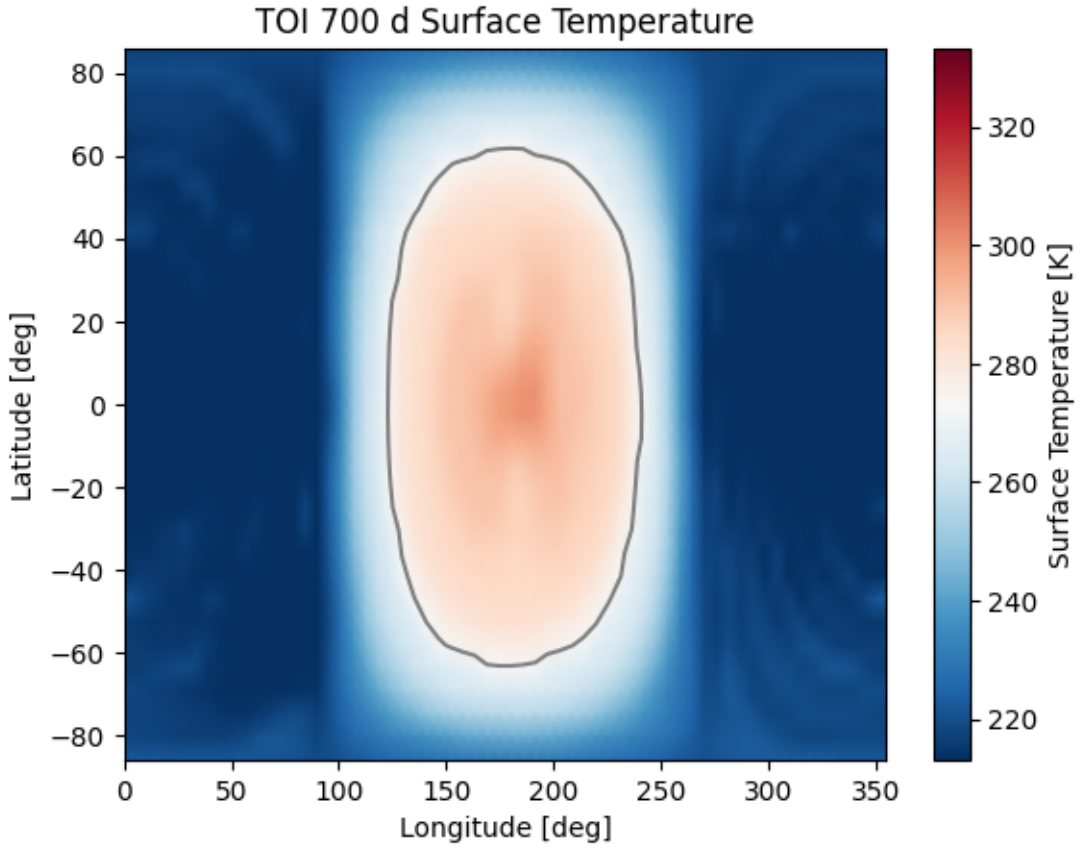
If all went well on that previous step, you should now have a bunch of NetCDF files sitting in the model's working directory. You can now open and analyze those as you wish. However, ExoPlaSim's Python API does provide some data inspection tools. Let's take a look at some of them. First, we'll plot the surface temperature, using matplotlib.

```
>>> import matplotlib.pyplot as plt
>>> lon = toi700d.inspect("lon")
>>> lat = toi700d.inspect("lat")
>>> ts = toi700d.inspect("ts",tavg=True)
>>> im=plt.pcolormesh(lon,lat,ts,cmap="RdBu_r",vmin=273.15-60.0,vmax=273.15+60.0,
↪shading="Gouraud")
>>> plt.contour(lon,lat,ts,[273.15,],colors=['gray',])
>>> plt.colorbar(im,label="Surface Temperature [K]")
```

(continues on next page)

(continued from previous page)

```
>>> plt.xlabel("Longitude [deg]")
>>> plt.ylabel("Latitude [deg]")
>>> plt.title("TOI 700 d Surface Temperature")
>>> plt.show()
```



Neat! That does look like a tidally-locked planet. Note that when we requested the surface temperature, we specified `tavg=True`, but nothing else besides the variable name. That told ExoPlaSim that we wanted a time average, and because we didn't specify otherwise, it gave us the time average of the final year of output. If we hadn't set `tavg`, we would have gotten a 3-dimensional array, with the first dimension being time. If we wanted say the third year, we could have specified `year=2` (remember how Python indexing works). If we wanted to look at 3 years before the model finished, we could use `year=-3`. For more information, refer to the documentation for *`inspect`*.

How about something a bit more complex–say a 3-dimensional field, like wind? Airflow in ExoPlaSim is represented by 3 different fields: `ua` for zonal wind, `va` for meridional wind, and `wa` for vertical wind. In most climates you'll model with ExoPlaSim, wind is almost entirely horizontal, so we'll ignore `wa` for now. To get the overall wind speed, we'll need to combine `ua` and `va`:

```
>>> import numpy as np
>>> ua = toi700d.inspect("ua",layer=5)
>>> va = toi700d.inspect("va",layer=5)
>>> speed = np.nanmean(np.sqrt(ua**2+va**2), axis=0)
>>> ua = np.nanmean(ua, axis=0)
>>> va = np.nanmean(va, axis=0)
```

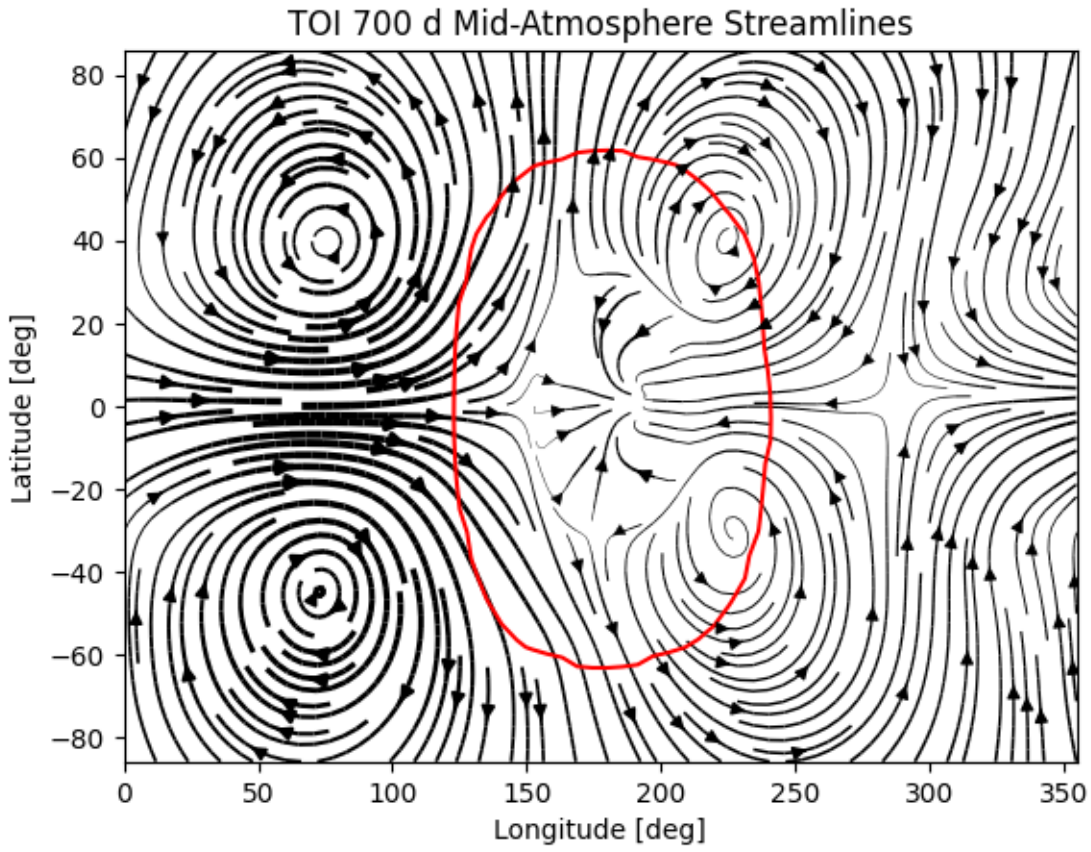
Note that here we do the time-averaging *after* we do math on the variables–the function of an average is not always the average of the function. We've also now specified a `layer` argument, which extracts a particular vertical layer

from a data field that has 3 spatial dimensions. Our model has 10 layers, so we extracted one of the middle layers, to show us the mid-altitude winds.

```
>>> from scipy.interpolate import interp2d
>>> ylat = np.linspace(lat.min(),lat.max(),lat.size) #ExoPlaSim latitudes are not␣
↪evenly-spaced
>>> ux = interp2d(lon, lat, ua)(lon, ylat)
>>> vx = interp2d(lon, lat, va)(lon, ylat)
>>> speedx = interp2d(lon, lat, speed)(lon,ylat)
```

Here we've interpolated our windspeeds onto a new grid with an evenly-spaced y-axis–we have to do this because latitudes in ExoPlaSim are not evenly-spaced, and matplotlib's streamplot routine requires an evenly-spaced grid.

```
>>> linewidth = 3*speedx / speedx.max()
>>> plt.streamplot(lon,ylat,ux,vx, density = 2, color='k', linewidth=linewidth)
>>> plt.contour(lon,lat,ts,[273.15,],colors=['r',])
>>> plt.xlabel("Longitude [deg]")
>>> plt.ylabel("Latitude [deg]")
>>> plt.title("TOI 700 d Mid-Atmosphere Streamlines")
>>> plt.show()
```

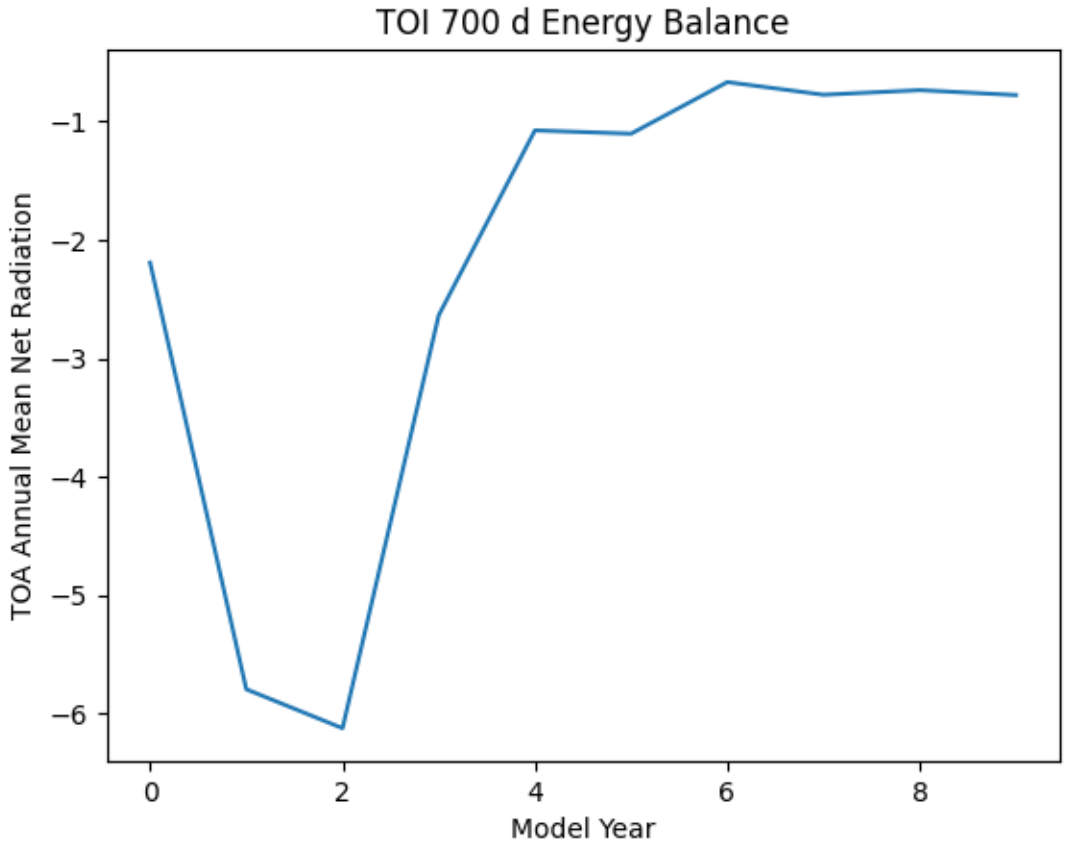


We can pretty clearly see here the night-side gyres, and the complex inflow-outflow behavior at the substellar point.

We've looked up until now only at the current year. What if we wanted to see how, say, the mean top-of-atmosphere energy balance evolved with each model year?

The *Model.gethistory* routine provides the functionality we need. It will return an array of global annual averages for a given variable, for each simulated year:

```
>>> energybalance = toi700d.gethistory(key="ntr")
>>> plt.plot(energybalance)
>>> plt.xlabel("Model Year")
>>> plt.ylabel("TOA Annual Mean Net Radiation")
>>> plt.title("TOI 700 d Energy Balance")
>>> plt.show()
```



You'll notice here that we're not quite in equilibrium yet. That's because we only ran 10 years. Typically, reaching a strict energy balance equilibrium takes many decades, and sometimes up to a few centuries depending on how different the equilibrium is from the initial conditions. For a routine that will automatically run until an energy balance criterion is reached, see *Model.runtobalance*.

Sometimes it can be helpful to examine 3D data in a plane other than latitude-longitude. For this, the *exoplasim.gcmt* module can be useful. Here, for example, we examine meridional average vertical wind, and zonal average wind:

```
>>> import exoplasim.gcmt as gcmt
>>> wa = toi700d.inspect("wa")
>>> ua = toi700d.inspect("ua")
>>> wa = gcmt.make2d(wa,lat="mean")
>>> ua = gcmt.make2d(ua,lon="mean")
```

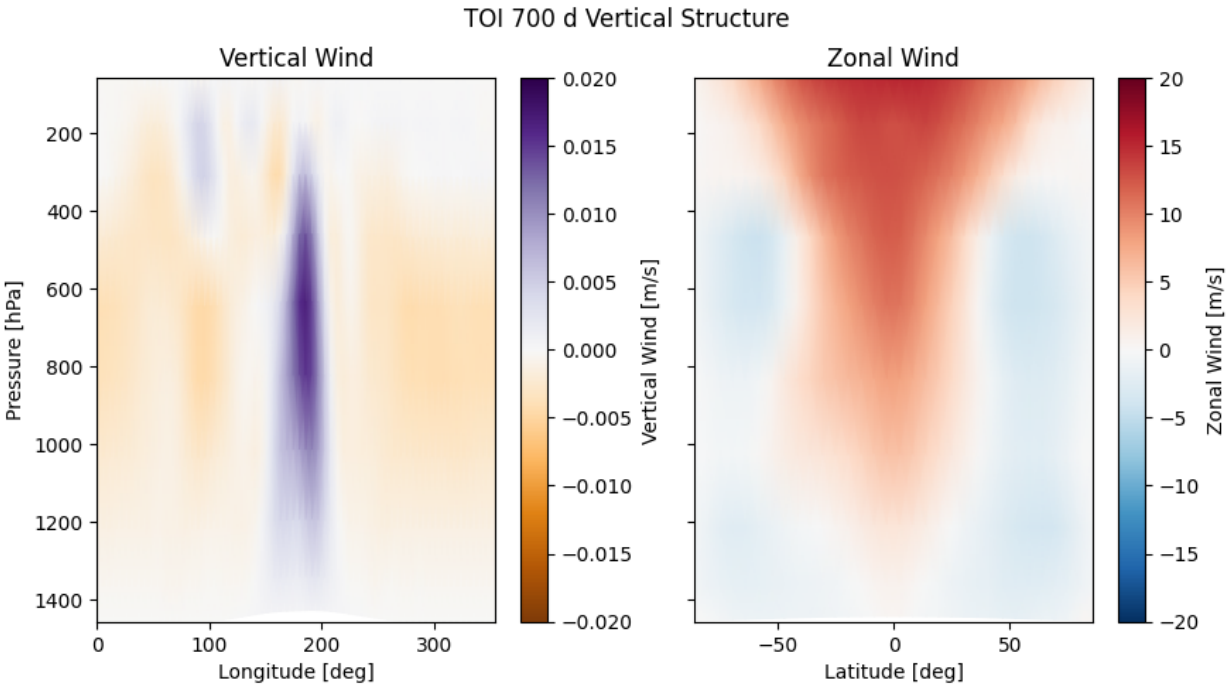
The *make2d* function attempts to reduce an input variable to 2 dimensions. If you specify that a particular dimension (`lat`, `lon`, `lev`) should be averaged (`"mean"`) or summed (`"sum"`), the function will first attempt to reduce along that dimension. If not enough dimensions are specified, or a time slice is not given, the default is to return a time-average. Note that when an average or sum is computed, the different sizes of grid cells **is** taken into account.

For the vertical axis, it may be useful to have pressure levels.

```
>>> sigma = toi700d.inspect("lev")
>>> psurf = toi700d.inspect("ps")
>>> pAir = sigma[np.newaxis,:,np.newaxis,np.newaxis] * psurf[:,np.newaxis,:,:]
>>> pmerid = gcmt.make2d(pAir,lat="mean")
>>> pzonal = gcmt.make2d(pAir,lon="mean")
```

We now have a 2D array of mid-layer pressures for each of our plots, in units of hPa.

```
>>> fig,ax = plt.subplots(1,2,figsize=(10,5),sharey=True)
>>> im1 = ax[0].pcolormesh(lon,pmerid,wa,cmap='PuOr',shading='Gouraud',vmin=-0.02,
↪vmax=0.02)
>>> plt.colorbar(im1,label="Vertical Wind [m/s]",ax=ax[0])
>>> im2 = ax[1].pcolormesh(lat,pzonal,ua,cmap='RdBu_r',shading='Gouraud',vmin=-20,
↪vmax=20)
>>> plt.colorbar(im2,label="Zonal Wind [m/s]",ax=ax[1])
>>> ax[0].invert_yaxis()
>>> ax[0].set_xlabel("Longitude [deg]")
>>> ax[0].set_ylabel("Pressure [hPa]")
>>> ax[1].set_xlabel("Latitude [deg]")
>>> ax[0].set_title("Vertical Wind")
>>> ax[1].set_title("Zonal Wind")
>>> fig.suptitle("TOI 700 d Vertical Structure")
>>> plt.show()
```

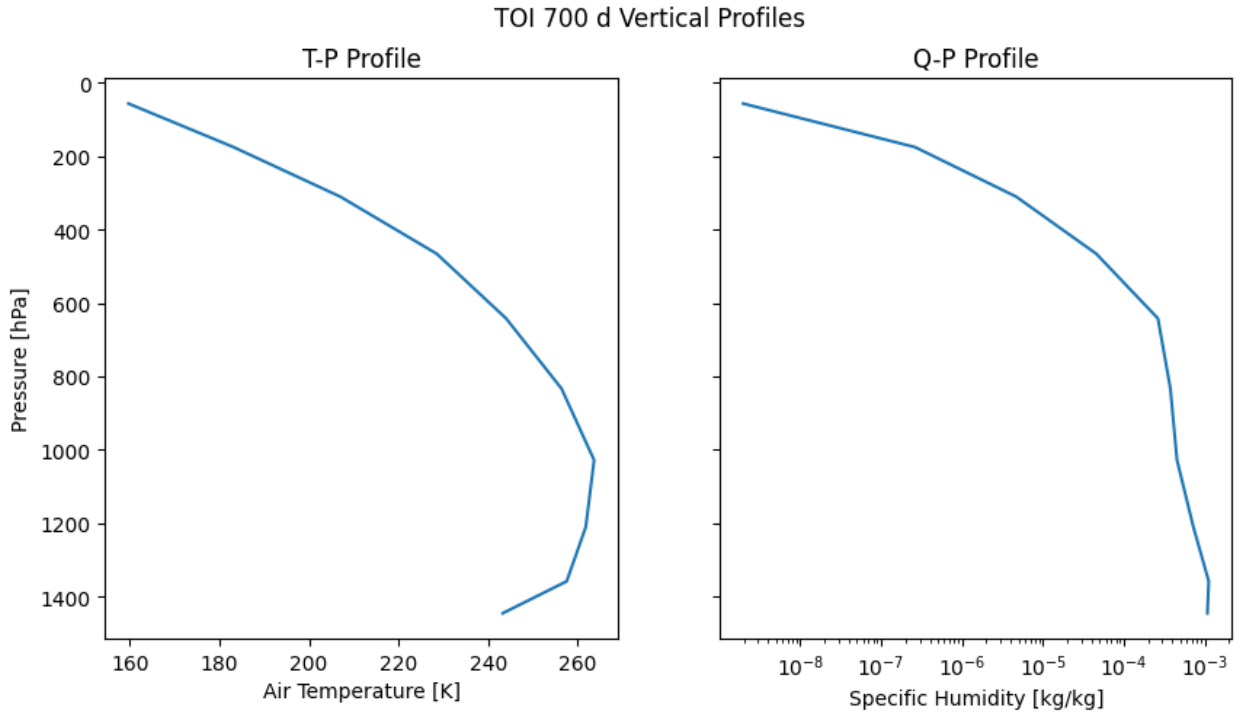


Similarly, we can use the averaging features built into the *inspect* function to extract vertical profiles:

```
>>> ps = toi700d.inspect("ps",savg=True,tavg=True)
>>> pa = ps*sigma
>>> tprofile = toi700d.inspect("ta",savg=True,tavg=True) # Mid-layer air temperature␣
↪[K]
>>> qprofile = toi700d.inspect("hus",savg=True,tavg=True) # Mid-layer specific␣
↪humidity [kg/kg]
```

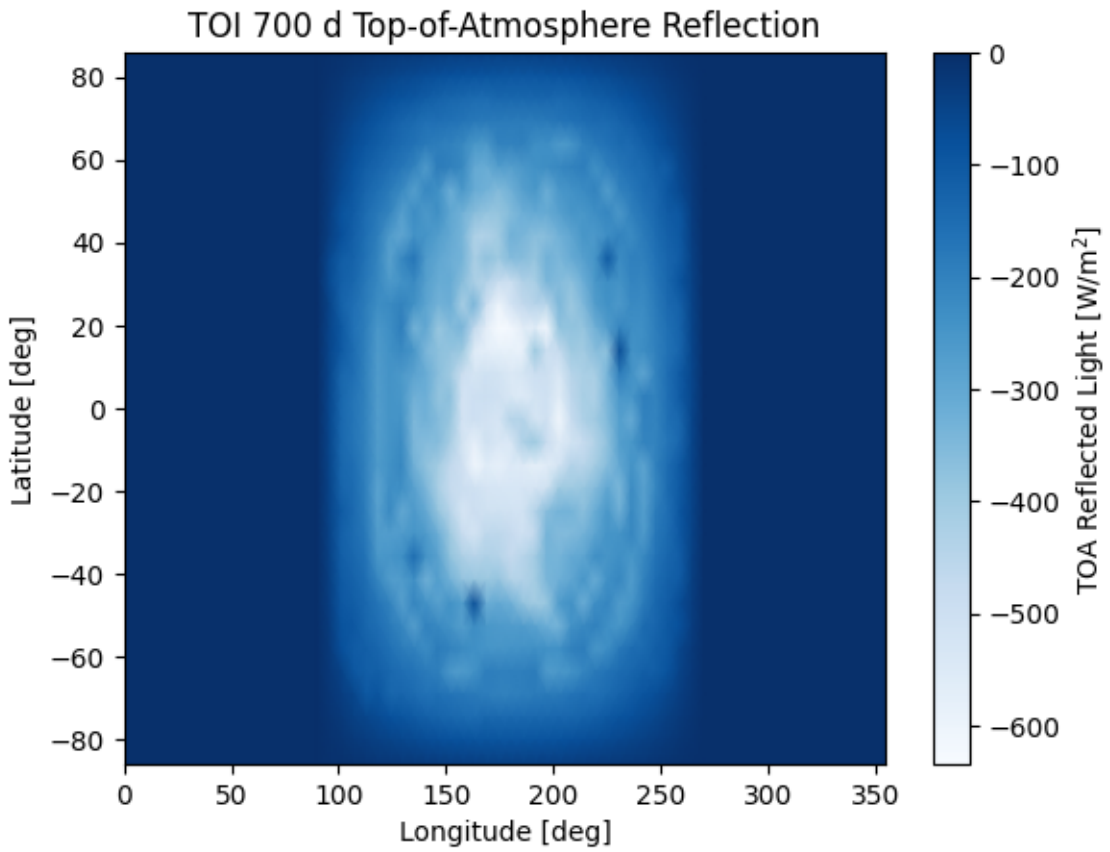
Here, we leverage the `savg` flag to return global means. When the field we want has 3 spatial dimensions, the vertical dimension is preserved, returning an array of the horizontal global mean in each model layer.

```
>>> fig,ax = plt.subplots(1,2,figsize=(10,5),sharey=True)
>>> ax[0].plot(tprofile,pa)
>>> ax[1].plot(qprofile,pa)
>>> ax[1].set_xscale('log')
>>> ax[0].invert_yaxis()
>>> ax[0].set_xlabel("Air Temperature [K]")
>>> ax[0].set_ylabel("Pressure [hPa]")
>>> ax[1].set_xlabel("Specific Humidity [kg/kg]")
>>> ax[0].set_title("T-P Profile")
>>> ax[1].set_title("Q-P Profile")
>>> fig.suptitle("TOI 700 d Vertical Profiles")
>>> plt.show()
```



And of course, it might be nice to see what this planet might look like in reflected light.

```
>>> reflected = toi700d.inspect("rsut",snapshot=True)
>>> im = plt.pcolormesh(lon,lat,reflected[-1],cmap='Blues',shading='Gouraud')
>>> plt.colorbar(im,label="TOA Reflected Light [W/m$^2$]")
>>> plt.xlabel("Longitude [deg]")
>>> plt.ylabel("Latitude [deg]")
>>> plt.title("TOI 700 d Top-of-Atmosphere Reflection")
>>> plt.show()
```



The `snapshot` flag tells exoplasim to pull from the snapshot outputs instead of the time-averaged outputs. This returns an array with many different instances, so we need to specify which one we want. In the plotting command, we select the most recent snapshot that was written.

Finally, to move everything to an output directory:

```
>>> toi700d.finalize("TOI-700d",allyears=True,keeprestarts=True)
>>> toi700d.save() #So we can reload the Model object for data inspection at a later␣
↪date
```

This will move output files, diagnostic files, and restart files to the folder "TOI-700d", delete the run folder (set `clean=False` to avoid this), and then save the `Model` instance to a NumPy save file, from which it can be reloaded at a later date for further data inspection:

```
>>> import numpy as np
>>> toi700d = np.load("TOI-700d/TOI-700d.npy",allow_pickle=True).item()
```

Note that NumPy save files are generally not portable when they've been pickled. If you want to enable somebody else to run your model, give them `TOI-700d.cfg` instead.

1.1.4 A Shortcut for TOI 700 d

Setting up TOI 700 d involved setting several parameters that are probably always going to be set for tidally-locked models. That could get a little repetitive if you set up many models by hand. Fortunately, ExoPlaSim provides a sub-class that would have made configuration much shorter: the *exoplasim.TLaquaplanet* class, along with *exoplasim.TLlandplanet* and *exoplasim.TLmodel*. Using `TLaquaplanet`, we would have done the following:

```
>>> import exoplasim as exo
>>> toi700d = exo.TLaquaplanet(workdir="toi700d_run",modelname="TOI-700d",ncpus=4,
↪resolution="T21")
>>> toi700d.configure(startemp=3480.0, flux=1167.0,          # Stellar parameters
>>>                   rotationperiod=37.426,                 # Rotation
>>>                   radius=1.19,gravity=11.9,              # Bulk properties
>>>                   pN2=1.47*(1-360e-6),pCO2=1.47*360e-6) # Atmosphere
>>> toi700d.exportcfg()
```

All the other parameters we had specified, like the timestep, aquaplanet mode, physics filter, circular orbit, etc are the defaults for a tidally-locked model. Furthermore, there is only one configuration file format–so when you share the configuration file, it can be loaded by any *Model* instance. A similar class exists for tidally-locked land planets, as well as a generic tidally-locked class that does not specify surface type.

And of course, there is an *exoplasim.Earthlike* class, which sets the usual defaults for a planet with more Earth-like rotation, but which for example might have a slightly different surface pressure.

1.2 Postprocessing ExoPlaSim Outputs

1.2.1 The Basics: Formats, Variables, and Math

As of ExoPlaSim 3.0.0, postprocessing can be done using the *exoplasim.pyburn* module. This module exposes an API for setting the variables to be included in postprocessed output, the horizontal mode in which to present them, and any additional math that should be performed, including coordinate transformations, time-averaging, and standard deviations. `pyburn` also supports a large range of output formats: netCDF, HDF5, NumPy's compressed `.npz` archives (the default), and plain-text comma-separated value (CSV) files. The latter can be compressed individually with the `gzip` format, tarballed, or tarballed and compressed (in the latter case with `gzip`, `lzma`, or `bzip2` compression types). Producing netCDF files requires that the netCDF4 python library be present (you can install it with `pip install netCDF4` or at ExoPlaSim's install-time with `pip install exoplasim[netCDF4]`). Similarly, producing HDF5 files requires the presence of the `h5py` Python library, which can be installed via `pip install h5py` or, at ExoPlaSim's install time, with `pip install exoplasim[HDF5]`. Support for both netCDF and HDF5 can be guaranteed at install-time by combining them:

```
pip install exoplasim[netCDF4,HDF5]
```

Format

The choice of output format can be specified either when the postprocessor is called (if being used manually), or as an argument to a *Model* object, by simply providing the file extension:

Format	Supported Extensions
NumPy (default)	.npz
	.npy
netCDF	.nc
HDF5	.h5
	.he5
	.hdf5
Compressed CSV	.gz
	.tar.gz
	.tar.xz
	.tar.bz2
Uncompressed CSV	.csv
	.txt
	.tar

Because the NumPy archive format does not support additional metadata arrays, metadata is stored separately in a file using the `_metadata.npz` suffix. This file is typically a few tens of kiB.

CSV-type files will only contain 2D variable information, so the first N-1 dimensions will be flattened. The original variable shape is included in the file header (prepended with a # character) as the first items in a comma-separated list, with the first non-dimension item given as the 'lll' placeholder. On reading variables from these files, they should be reshaped according to these dimensions. This is true even in tarballs (which contain CSV files). If read in by *gcmt.load()*, this reshaping will be done automatically.

Note that when using the *pyburn.postprocess()* function directly, **a single file must be specified as the output file.** This is true even for formats that produce a large number of files that don't get bound up together, such as `.gz` and `.csv`, which produce a folder containing one file per variable. The file you specify should have the pattern `<subdirectory>.<extension>`. This file will not actually be created, but it will be parsed to determine the desired output format. So, for example, to create an archive consisting of a folder full of CSV files for the raw output file `MOST.00127`, one would use `MOST.00127.csv`. The surface temperature variable, `ts`, would then be found in `MOST.00127/MOST.00127_ts.csv`. **This same combined-format fictional filestring can be passed to** *gcmt.load().* The object returned by that function will access the data in the archive just as if it were a bound archive, such as a tarball, netCDF file, or HDF5 file.

A T21 model output with 10 vertical levels, 12 output times, all supported variables in grid mode,and no standard deviation computation will have the following sizes for each format:

Format	Size
netCDF	12.8 MiB
HDF5	17.2 MiB
NumPy (default)	19.3 MiB
tar.xz	33.6 MiB
tar.bz2	36.8 MiB
gzipped	45.9 MiB
uncompressed	160.2 MiB

Variables

Output variables can be chosen in multiple ways. Either a `burn7`-style namelist can be provided, containing a list of numeric variables codes (listed below), or a list can be passed directly, containing a list of numeri codes, a list of strings of numeric codes, or a list of string variable keys, as indicated in the leftmost-column of the table below.

Variable lists can be specified once for all outputs of a given type ('regular', 'snapshot', or 'highcadence'), with *`Model.cfgpostprocessor()`*, or for each model year with *`Model.postprocess()`*, or manually outside of the ExoPlaSim *`Model`* object, with *`pyburn.postprocess`*.

Optionally, as advanced usage, a dictionary can be passed, with one member per variable (using the same identification rules given above), and *`pyburn.dataset()`* keyword arguments specified for each variable. For example, to create an output file with two variables, surface temperature and streamfunction, both on a horizontal grid, and the streamfunction zonally-averaged and passed through physics filters:

```
{"ts":{"mode":"grid","zonal":False},
 "stf":{"mode":"grid","zonal":True,"physfilter":True}}
```

This can be specified in one of 3 ways. Either it can be set for all outputs of a given type ('regular', 'snapshot', or 'highcadence') as a Model property:

```
>>> myModel.cfgpostprocessor(ftype="regular",extension=".nc",
>>>                          variables={"ts":{"mode":"grid","zonal":False},
>>>                                     "stf":{"mode":"grid","zonal":True,"physfilter
↪":True}})
```

Or it can be set each time *`Model.postprocess()`* is called:

```
>>> myModel.postprocess("MOST.00127",
>>>                      {"ts":{"mode":"grid","zonal":False},
>>>                       "stf":{"mode":"grid","zonal":True,"physfilter":True}},
>>>                      log="burnlog.00127",crashifbroken=True)
```

Or, finally, it can be specified directly to *`pyburn.postprocess()`*:

```
>>> pyburn.postprocess("MOST.00127","MOST.00127.nc",logfile="burnlog.00127",
>>>                    variables={"ts":{"mode":"grid","zonal":False},
>>>                               "stf":{"mode":"grid","zonal":True,"physfilter":True}
↪})
```

Postprocessing Math

`pyburn` provides the ability to perform various mathematical operations on the data as part of the postprocessing step.

Multiple horizontal modes are available (specified with the `mode` keyword), including a Gaussian-spaced latitude-longitude grid (`"grid"`), spherical harmonics (`"spectral"`), Fourier coefficients for each latitude (`"fourier"`), a latitude-longitude grid rotated such that the "North" pole is at the substellar point of a sychronously-rotating planet, and the "equator" is the terminator (`"synchronous"`), and Fourier coefficients computed along lines of constant longitude (including the mirror component on the opposite hemisphere) in that rotated coordinate system (`"syncfourier"`). Additionally, for output modes with discrete latitudes, variables can be zonally-averaged (`zonal=True`).

ExoPlaSim performs some time-averaging on the fly (for "regular"-type outputs) to avoid overloading I/O buffers and creating enormous raw output files, but the number of output times is still often going to be more than you prefer for the postprocessed output data. The default configuration, for example, produces 72 output timestamps per year. `pyburn` can perform time-averaging to reduce this to e.g. monthly output, via the `times` keyword and the `timeaveraging`

keyword. The former specifies either the number of output times or the specific output times requested (as decimal fractions of a model output's timeseries), while the latter is a boolean True/False flag. If specific output times are requested or the number of requested outputs doesn't divide cleanly into the number of timestamps in the raw output, `pyburn` can interpolate between timestamps using linear interpolation. No extrapolation is performed, so you cannot request a time between e.g. the last output of the previous year and the first output of the current year. Whether or not linear interpolation is used or "nearest-neighbor" interpolation (which simply selects the timestamp closest to the target time) can be set with the `interpolatetimes` keyword–if `True`–linear interpolation will be used when necessary. The minimum number of timestamps in the output file is 1; this corresponds to computing an annual average.

Finally, `pyburn` brings the ability to compute the standard deviations of ExoPlaSim variables. Enabling this with `stdev=True` will compute the standard deviation in one of two ways, depending on whether time-averaging is being used. If time averages are being computed, then a standard deviation will be computed **alongside** each average, and the each standard deviation variable (denoted with the `_std` suffix in the variable name, e.g. `ts_std` for the standard deviation of surface temperature) will have the same number of timestamps as the time-averages. If time-averages are **not** being computed, then the standard deviation of the entire file's timeseries will be computed, and there will be one timestamp per standard deviation variable.

1.2.2 Reading Postprocessed Files

While postprocessed files are portable and can be read however you like, ExoPlaSim also provides a native, format-agnostic way to access them via the *`gcmt.load()`* function. This takes the archive filename as its argument, and returns an object analogous to an open netCDF file object. It has two members of interest to the user: `variables` and `metadata`. Both are compatible with all dictionary methods, and individual variables' data can be extracted by using the variable name as the dictionary key. For example:

```
>>> import exoplasim.gcmt as gcmt
>>> myData = gcmt.load("MOST.0127.tar.gz")
>>> surfacetemperature = myData.variables['ts']
>>> surftemp_metadata  = myData.metadata['ts']
```

Note that for CSV-type formats, like the tarball given above, the file is left compressed (except during the initial read), and the whole dataset is *not* loaded into memory. Dimension arrays, such as latitude, longitude, etc, are loaded, as is all metadata. By default, however, only one data array will be loaded into memory. This can be expanded with the `csvbuffersize` keyword, which takes the number of variables to permit to hold in the memory buffer. This buffer uses a first-in, first-out approach, so if a new variable is requested and the buffer is full, the loaded variable which was used the least recently will be purged from memory.

1.2.3 Postprocessor Variable Codes

Note that in addition to the variable codes listed below, if `pyburn` is used with `stdev=True`, there will also be variables that correspond to those listed below, with the `_std` suffix. If time-averaging was performed during post-processing, the standard deviation will be the standard deviation for each averaged time period, and there will be the same number of timestamps for the `_std` variables as for their nominal data counterparts. If time-averaging was not used, then each standard deviation variable will have only one timestamp, corresponding to the standard deviation throughout the entire timeseries present in the file.

Variable	Code	Description	Units	Notes
nu	50	orbital true anomaly	deg	
lambda	51	solar ecliptic longitude	deg	
zdec	52	solar declination angle	deg	
rdist	53	planet-star distance modulus	nondimensional	
mld	110	mixed layer depth	m	

continues on next page

Table 1 – continued from previous page

Variable	Code	Description	Units	Notes
sg	129	surface geopotential	m^2 s^{-2}	
ta	130	air temperature	K	
ua	131	eastward wind	m s^{-1}	
va	132	northward wind	m s^{-1}	
hus	133	specific humidity	kg/kg	
ps	134	surface air pressure	hPa	
wap	135	vertical air velocity	Pa s-1	
wa	137	upward wind	m s^{-1}	
zeta	138	atm relative vorticity	s^{-1}	
ts	139	surface temperature	K	
mrso	140	lwe of soil moisture content	m	
snd	141	surface snow thickness	m	
prl	142	lwe of large scale precipitation	m s^{-1}	
prc	143	convective precipitation rate	m s^{-1}	
prsn	144	lwe of snowfall amount	m s^{-1}	
bld	145	dissipation in boundary layer	W m^{-2}	
hfss	146	surface sensible heat flux	W m^{-2}	
hfls	147	surface latent heat flux	W m^{-2}	
stf	148	streamfunction	m^2 s^{-2}	
psi	149	velocity potential	m^2 s^{-2}	
psl	151	air pressure at sea level	hPa	
pl	152	log surface pressure	nondimensional	
d	155	divergence of wind	s^{-1}	
zg	156	geopotential height	m	
hur	157	relative humidity	nondimensional	
tps	158	tendency of surface air pressure	Pa s-1	
u3	159	u*	m^3 s^{-3}	
mrro	160	surface runoff	m s^{-1}	
clw	161	liquid water content	nondimensional	
cl	162	cloud area fraction in layer	nondimensional	
tcc	163	total cloud cover	nondimensional	
clt	164	cloud area fraction	nondimensional	
uas	165	eastward wind 10m	m s^{-1}	
vas	166	northward wind 10m	m s^{-1}	
tas	167	air temperature 2m	K	
td2m	168	dew point temperature 2m	K	
tsa	169	surface temperature accumulated	K	
tsod	170	deep soil temperature	K	
dsw	171	deep soil wetness	nondimensional	
lsm	172	land binary mask	nondimensional	
z0	173	surface roughness length	m	
alb	174	surface albedo	nondimensional	
as	175	surface albedo	nondimensional	
rss	176	surface net shortwave flux	W m^{-2}	
rls	177	surface net longwave flux	W m^{-2}	
rst	178	toa net shortwave flux	W m^{-2}	
rlut	179	toa net longwave flux	W m^{-2}	
tauu	180	surface eastward stress	Pa	
tauv	181	surface northward stress	Pa	

continues on next page

Table 1 – continued from previous page

Variable	Code	Description	Units	Notes
evap	182	lwe of water evaporation	m s^{-1}	
tso	183	climate deep soil temperature	K	
wsoi	184	climate deep soil wetness	nondimensional	
vegc	199	vegetation cover	nondimensional	
rsut	203	toa outgoing shortwave flux	W m^{-2}	
ssru	204	surface solar radiation upward	W m^{-2}	
stru	205	surface thermal radiation upward	W m^{-2}	
tso2	207	soil temperature level 2	K	
tso3	208	soil temperature level 3	K	
tso4	209	soil temperature level 4	K	
sic	210	sea ice cover	nondimensional	
sit	211	sea ice thickness	m	
vegf	212	forest cover	nondimensional	
snm	218	snow melt	m s^{-1}	
sndc	221	snow depth change	m s^{-1}	
prw	230	atmosphere water vapor content	kg m^{-2}	
glac	232	glacier cover	nondimensional	
tsn	238	snow temperature	K	
spd	259	wind speed	m s^{-1}	
pr	260	total precipitation	m s^{-1}	
ntr	261	net top radiation	W m^{-2}	
nbr	262	net bottom radiation	W m^{-2}	
hfns	263	surface downward heat flux	W m^{-2}	
wfn	264	net water flux	m s^{-1}	
lwth	266	local weathering	W earth	
grnz	267	ground geopotential	m^2 s^{-2}	
icez	301	glacier geopotential	m^2 s^{-2}	
netz	302	net geopotential	m^2 s^{-2}	
dpdx	273	d(ps)/dx	Pa m^{-1}	
dpdy	274	d(ps)/dy	Pa m^{-1}	
hlpr	277	half level pressure	Pa	
flpr	278	full level pressure	Pa	
thetah	279	half level potential temperature	K	
theta	280	full level potential temperature	K	
czen	318	cosine solar zenith angle	nondimensional	
wthpr	319	weatherable precipitation	mm day^{-1}	
mint	320	minimum temperature	K	
maxt	321	maximum temperature	K	
cape	322	convective available potential energy	J kg^{-1}	Storm Clim.
lnb	323	level of neutral buoyancy	hPa	Storm Clim.
sdef	324	troposphere entropy deficit	nondimensional	Storm Clim.
absz	325	sigma-0.85 abs vorticity	s^{-1}	Storm Clim.
umax	326	maximum potential intensity	m s^{-1}	Storm Clim.
vent	327	ventilation index	nondimensional	Storm Clim.
vrumax	328	ventilation-reduced maximum wind	m s^{-1}	Storm Clim.
gpi	329	genesis potential index	nondimensional	Storm Clim.
dfu	404	shortwave up	W m^{-2}	Snapshot Only
dfd	405	shortwave down	W m^{-2}	Snapshot Only
dftu	406	longwave up	W m^{-2}	Snapshot Only

continues on next page

Table 1 – continued from previous page

Variable	Code	Description	Units	Notes
dftd	407	longwave down	W m^{-2}	Snapshot Only
dtdt	408	radiative heating rate	K s^{-1}	Snapshot Only
dfdz	409	flux convergence	W m^{-3}	Snapshot Only

1.2.4 Burn7 Postprocessor Options

The C++ `burn7` postprocessor is now deprecated and unsupported. It is only available via the `exoplasim-legacy` package.

1.3 exoplasim package

1.3.1 Module contents

class exoplasim.**Earthlike**(*resolution='T21'*, *layers=10*, *ncpus=4*, *precision=8*, *debug=False*, *inityear=0*, *recompile=False*, *optimization=None*, *mars=False*, *workdir='most'*, *source=None*, *force991=False*, *modelname='MOST_EXP'*, *outputtype='.npz'*, *crashtolerant=False*)

Bases: *exoplasim.Model*

Create an Earth-like model, but more flexible.

Identical to *Model*, except configuration options common for Earth-like models requiring slightly more flexibility are the default when configure is called–specifically, 45-minute timestep, snapshot output reporting every 480 timesteps, and a model top pinned to 50 mbar. All these defaults can be overridden.

configure(*timestep=45.0*, *snapshots=480*, *vtype=4*, *modeltop=50.0*, ***kwargs*)

Configure the model's namelists and boundary conditions.

The defaults here are appropriate for an Earth model.

Model Operation

noutput [bool, optional] True/False. Whether or not model output should be written.

restartfile [str, optional] Path to a restart file to use for initial conditions. Can be None.

writefrequency [int, optional] How many times per day ExoPlaSim should write output. Ignored by default–default is to write time-averaged output once every 5 days.

timestep [float, optional] Model timestep. Defaults to 45 minutes.

runscript [function , optional] A Python function that accepts a Model object as its first argument. This is the routine that will be run when you issue the Model.run() command. Any keyword arguments passed to run() will be forwarded to the specified function. If not set, the default internal routine will be used.

snapshots [int, optional] How many timesteps should elapse between snapshot outputs. If not set, no snapshots will be written.

restartfile [string, optional] Path to a restart file to use.

highcadence [dict, optional] A dictionary containing the following arguments:

'toggle' [{0,1}] Whether or not high-cadence output should be written (1=yes).

'start' [int] Timestep at which high-cadence output should begin.

'end' [int] Timestep at which high-cadence output should end.

'interval' [int] How many timesteps should elapse between high-cadence outputs.

threshold [float, optional] Energy balance threshold model should run to, if using *runtobalance()*. Default is <0.05 W/m^2/yr average drift in TOA and surface energy balance over 45-year timescales.

resources [list, optional] A list of paths to any additional files that should be available in the run directory.

runsteps [integer, optional] The number of timesteps to run each 'year'. By default, this is tuned to 360 Earth days. If set, this will override other controls setting the length of each modelled year.

otherargs [dict, optional] Any namelist parameters not included by default in the configuration options. These should be passed as a dictionary, with "PARAMETER@namelist" as the form of the dictionary key, and the parameter value passed as a string. e.g. `otherargs={"N_RUN_MONTHS@plasim_namelist":'4', "NGUI@plasim_namelist:'1'}`

Model Dynamics

columnmode [{None,"-","clear","static","static|clear","clear|static"}, optional] The inclusion of 'static' will disable horizontal advection, forcing ExoPlaSim into a column-only mode of operation. The inclusion of 'clear' will disable the radiative effects of clouds.

drycore [bool, optional] True/False. If True, evaporation is turned off, and a dry atmosphere will be used.

physicsfilter [str, optional] If not an empty string, specifies the physics filter(s) to be used. Filters can be used during the transform from gridpoint to spectral (`"gp"`), and/or during the transform from spectral to gridpoint (`"sp"`). Filter types are "none", "cesaro", "exp", or "lh" (see the Notes for more details). Combinations of filter types and times should be combined with a `|`, e.g. `physicsfilter="gp|exp|sp"` or `physicsfilter="gp|cesaro"`.

filterkappa [float, optional] A constant to be used with the exponential filter. Default is 8.0.

filterpower [int, optional] A constant integer to be used with the exponential filter. Default is 8.

filterLHN0 [float, optional] The constant used in the denominator of the Lander-Hoskins Filter. Default is 15; typically chosen so f(N)=0.1.

diffusionwaven [int, optional] The critical wavenumber beyond which hyperdiffusion is applied. Default is 15 for T21.

qdiffusion [float, optional] Timescale for humidity hyperdiffusion in days. Default for T21 is 0.1.

tdiffusion [float, optional] Timescale for temperature hyperdiffusion in days. Default for T21 is 5.6.

zdiffusion [float, optional] Timescale for vorticity hyperdiffusion in days. Default for T21 is 1.1.

ddiffusion [float, optional] Timescale for divergence hyperdiffusion in days.. Default for T21 is 0.2.

diffusionpower [int, optional] integer exponent used in hyperdiffusion. Default is 2 for T21.

Radiation

flux [float, optional] Incident stellar flux in W/m^2. Default 1367 for Earth.

startemp [float, optional] Effective blackbody temperature for the star. Not used if not set.

starradius [float, optional] Radius of the parent star in solar radii. Currently only used for the optional petitRADTRANS direct imaging postprocessor.

starspec [str, optional] Spectral file for the stellar spectrum. Should have two columns and 965 rows, with wavelength in the first column and radiance or intensity in the second. A similarly-named file with the "_hr.dat" suffix must also exist and have 2048 wavelengths. Appropriately-formatted files can be created with *makestellarspec.py*.

twobandalbedo [bool, optional] True/False. If True, separate albedos will be calculated for each of the two shortwave bands. If False (default), a single broadband albedo will be computed and used for both.

synchronous [bool, optional] True/False. If True, the Sun is fixed to one longitude in the sky.

desync [float, optional] The rate of drift of the substellar point in degrees per minute. May be positive or negative.

substellarlon [float, optional] The longitude of the substellar point, if synchronous==True. Default 180°

pressurebroaden [bool, optional] True/False. If False, pressure-broadening of absorbers no longer depends on surface pressure. Default is True

ozone [bool or dict, optional] True/False/dict. Whether or not forcing from stratospheric ozone should be included. If a dict is provided, it should contain the keys "height", "spread", "amount","varlat","varseason", and "seasonoffset", which correspond to the height in meters of peak O3 concentration, the width of the gaussian distribution in meters, the baseline column amount of ozone in cm-STP, the latitudinal amplitude, the magnitude of seasonal variation, and the time offset of the seasonal variation in fraction of a year. The three amounts are additive. To set a uniform, unvarying O3 distribution, ,place all the ozone in "amount", and set "varlat" and "varseason" to 0.

snowicealbedo [float, optional] A uniform albedo to use for all snow and ice.

soilalbedo [float, optional] A uniform albedo to use for all land.

wetsoil [bool, optional] True/False. If True, land albedo depends on soil moisture (wet=darker).

oceanalbedo [float, optional] A uniform albedo to use for the ocean.

oceanzenith [{"ECHAM-3","ECHAM-6","Lambertian}, optional] The zenith-angle dependence to use for blue-light reflectance from the ocean. Can be `'Lambertian'/'uniform'`, `'ECHAM-3'/'plasim'/'default'`, or `'ECHAM-6'`. The default is `'ECHAM-3'` (synonymous with `'plasim'` and `'default'`), which is the dependence used in the ECHAM-3 model.

Orbital Parameters

year [float, optional] Number of 24-hour days in a sidereal year. Not necessary if eccentricity and obliquity are zero. Defaults if not set to ~365.25 days

rotationperiod [float, optional] Planetary rotation period, in days. Default is 1.0.

eccentricity [float, optional] Orbital eccentricity. If not set, defaults to Earth's (0.016715)

obliquity [float, optional] Axial tilt, in degrees. If not set, defaults to Earth's obliquity (23.441°).

lonvernaleq [float, optional] Longitude of periapse, measured from vernal equinox, in degrees. If not set, defaults to Earth's (102.7°).

fixedorbit [bool, optional] True/False. If True, orbital parameters do not vary over time. If False, variations such as Milankovich cycles will be computed by PlaSim.

keplerian [bool, optional] True/False. If True, a generic Keplerian orbital calculation will be performed. This means no orbital precession, Milankovich cycles, etc, but does allow for accurate calculation of a wide diversity of orbits, including with higher eccentricity. Note that extreme orbits may have extreme results, including extreme crashes.

meananomaly0 [float, optional] The initial mean anomaly in degrees. Only used if *keplerian=True*.

Planet Parameters

gravity [float, optional] Surface gravity, in m/s^2. Defaults to 9.80665 m/s^2.

radius [float, optional] Planet radius in Earth radii. Default is 1.0.

orography [float, optional] If set, a scaling factor for topographic relief. If `orography=0.0`, topography will be zeroed-out.

aquaplanet [bool, optional] True/False. If True, the surface will be entirely ocean-covered.

desertplanet [bool, optional] True/False. If True, the surface will be entirely land-covered.

tlcontrast [float, optional] The initial surface temperature contrast between fixedlon and the anterior point. Default is 0.0 K.

seaice [bool, optional] True/False. If False, disables radiative effects of sea ice (although sea ice itself is still computed).

landmap [str, optional] Path to a `.sra` file containing a land mask for the chosen resolution.

topomap [str, optional] Path to a `.sra` file containing geopotential height map. Must include landmap.

Atmosphere

gascon [float, optional] Effective gas constant. Defaults to 287.0 (Earth), or the gas constant corresponding to the composition specified by partial pressures.

vtype [{0,1,2,3,4,5}, optional] Type of vertical discretization. Can be: 0 Pseudolinear scaling with pressure that maintains resolution near the ground. 1 Linear scaling with pressure. 2 Logarithmic scaling with pressure (resolves high altitudes) 3 Pseudologarithmic scaling with pressure that preserves resolution near the ground. 4

Pseudolinear scaling with pressure, pinned to a specified top pressure. 5 If >10 layers, bottom 10 as if `vtype=4`, and upper layers as if `vtype=2`.

modeltop [float, optional] Pressure of the top layer

tropopause [float, optional] If stratosphere is being included, pressure of the 10th layer (where scheme switches from linear to logarithmic).

stratosphere [bool, optional] True/False. If True, vtype=5 is used, and model is discretized to include a stratosphere.

pressure: float, optional Surface pressure in bars, if not specified through partial pressures.

Gas Partial Pressures

Partial pressures of individual gases can be specified. If pressure and gascon are not explicitly set, these will determine surface pressure, mean molecular weight, and effective gas constant. Note however that Rayleigh scattering assumes an Earth-like composition, and the only absorbers explicitly included in the radiation scheme are CO2 and H2O.

pH2 [float, optional] H2 partial pressure in bars.

pHe [float, optional] He partial pressure in bars.

pN2 [float, optional] N2 partial pressure in bars.

pO2 [float, optional] O2 partial pressure in bars.

pH2 [float, optional] H2 partial pressure in bars.

pAr [float, optional] Ar partial pressure in bars.

pNe [float, optional] Ne partial pressure in bars.

pKr [float, optional] Kr partial pressure in bars.

pCO2 [float, optional] CO2 partial pressure in bars. This gets translated into a ppmv concentration, so if you want to specify/vary CO2 but don't need the other gases, specifying pCO2, pressure, and gascon will do the trick. In most use cases, however, just specifying pN2 and pCO2 will give good enough behavior.

pH2O [float, optional] H2O partial pressure in bars. This is only useful in setting the gas constant and surface pressure; it will have no effect on actual moist processes.

Surface Parameters

mldepth [float, optional] Depth of the mixed-layer ocean. Default is 50 meters.

soildepth [float, optional] Scaling factor for the depth of soil layers (default total of 12.4 meters)

cpsoil [float, optional] Heat capacity of the soil, in J/m^3/K. Default is 2.4*10^6.

soilwatercap [float, optional] Water capacity of the soil, in meters. Defaults to 0.5 meters

soilsaturation [float, optional] Initial fractional saturation of the soil. Default is 0.0 (dry).

maxsnow [float, optional] Maximum snow depth (Default is 5 meters; set to -1 to have no limit).

Additional Physics

Carbon-Silicate Weathering

co2weathering [bool, optional] True/False. Toggles whether or not carbon-silicate weathering should be computed. Default is False.

evolveco2 [bool, optional] True/False. If co2weathering==True, toggles whether or not the CO2 partial pressure should be updated every year. Usually the change in pCO2 will be extremely small, so this is not necessary, and weathering experiments try to estimate the average weathering rate for a given climate in order to interpolate timescales between climates, rather than modelling changes in CO2 over time directly.

outgassing [float, optional] The assumed CO2 outgassing rate in units of Earth outgassing. Default is 1.0.

erosionsupplylimit [float, optional] If set, the maximum CO2 weathering rate per year permitted by erosion, in ubars/year. This is not simply a hard cutoff, but follows Foley 2015 so high weathering below the cutoff is also reduced.

Vegetation

vegetation [bool or int, optional] Can be True/False, or 0/1/2. If True or 1, then diagnostic vegetation is turned on. If 2, then coupled vegetation is turned on. Vegetation is computed via the SimBA module.

vegaccel [int, optional] Integer factor by which to accelerate vegetation growth

nforestgrowth: float, optional Biomass growth

initgrowth [float, optional] Initial above-ground growth

initstomcond [float, optional] Initial stomatal conductance

initrough [float, optional] Initial vegetative surface roughness

initsoilcarbon [float, optional] Initial soil carbon content

initplantcarbon [float, optional] Initial vegetative carbon content

See [1]_ for details on the implementation of supply-limited weathering.

Glaciology

glaciers [dict, optional] A dictionary containing the following arguments: toggle : bool

True/False. Whether or not glaciers should be allowed to grow or shrink in thickness, or be formed from persistent snow on land.

mindepth [float] The minimum snow depth in meters of liquid water equivalent that must persist year-round before the grid cell is considered glaciated. Default is 2 meters.

initialh [float] If >=0, covers the land surface with ice sheets of a height given in meterss. If -1, no initial ice sheets are assumed.

Storm Climatology

stormclim [bool, optional] True/False. Toggles whether or not storm climatology (convective available potential energy, maximum potential intensity, ventilation index, etc) should be computed. If True, output fields related to storm climatology will be added to standard output files. Enabling this mode currently roughly

doubles the computational cost of the model. This may improve in future updates. Refer to Paradise, et al 2021 for implementation description.

stormcapture [dict, optional] A dictionary containing arguments controlling when high-cadence output is triggered by storm activity. This dictionary must contain 'toggle', which can be either 1 or 0 (yes or no). It may also contain any namelist parameters accepted by hurricanemod.f90, including the following:

toggle [{0,1}] Whether (1) or not (0) to write high-cadence output when storms occur

NKTRIGGER [{0,1}, optional] (0/1=no/yes). Whether or not to use the Komacek, et al 2020 conditions for hurricane cyclogenesis as the output trigger. Default is no.

VITHRESH [float, optional] (nktrigger) Ventilation index threshold for nktrigger output. Default 0.145

VMXTHRESH [float, optional] (nktrigger) Max potential intensity threshold for nktrigger output.Default 33 m/s

LAVTHRESH [float, optional] (nktrigger) Lower-atmosphere vorticity threshold for nktrigger output. Default 1.2*10^-5 s^-1

VRMTHRESH [float, optional] (unused) Ventilation-reduced maximum intensity threshold. Default 0.577

GPITHRESH [float, optional] (default) Genesis Potential Index threshold. Default 0.37.

MINSURFTEMP [float, optional] (default) Min. surface temperature for storm activity. Default 25C

MAXSURFTEMP [float, optional] (default) Max. surface temperature for storm activity. Default 100C

WINDTHRESH [float, optional] (default) Lower-atmosphere maximum wind threshold for storm activity. Default 33 m/s

SWINDTHRESH [float, optional] (default) Minimum surface windspeed for storm activity. Default 20.5 m/s

SIZETHRESH [float, optional] (default) Minimum number of cells that must trigger to start outputDefault 30

ENDTHRESH [float, optional] (default) Minimum number of cells at which point storm output ends.Default 16

MINSTORMLEN [float, optional] (default) Minimum number of timesteps to write output. Default 256

MAXSTORMLEN [float, optional] (default) Maximum number of timesteps to write output. Default 1024

Note that actual number of writes will be stormlen/interval, as set in highcadence. This interval defaults to 4, so 64 writes minimum, 256 max. For more details on the storm climatology factors considered here, see [5]_.

Notes

In some cases, it may be necessary to include physics filters. This typically becomes necessary when sharp features are projected on the model's smallest spectral modes, causing Gibbs "ripples". Earth-like models typically do not require filtering, but tidally-locked models do. Filtering may be beneficial for Earth-like models at very high resolutions as well, or if there is sharp topography.

Three filter functional forms are included in ExoPlaSim: Cesaro, exponential, and Lander-Hoskins. Their functional forms are given below, where *n* is the wavenumber, and *N* is the truncation wavenumber (e.g. 21 for T21):

Cesaro: $f(n) = 1 - \frac{n}{N+1}$ [2]_

Exponential: $f(n) = \exp\left[-\kappa\left(\frac{n}{N}\right)^{\gamma}\right]$ [3]_

Lander-Hoskins: $f(n) = \exp\left[-\left(\frac{n(n+1)}{n_0(n_0+1}\right)^2\right]$ [3]_ [4]_

κ is exposed to the user through `filterkappa`, γ is exposed through `filterpower`, and n_0 is exposed through `filterLHN0`.

Physics filters can be applied at two different points; either at the transform from gridpoint to spectral, or the reverse. We find that in most cases, the ideal usage is to use both. Generally, a filter at the gridpoint->spectral transform is good for dealing with oscillations caused by sharp jumps and small features in the gridpoint tendencies. Conversely, a filter at the spectral->gridpoint transform is good for dealing with oscillations that come from small-scale features in the spectral fields causing small-scale features to appear in the gridpoint tendencies [3]_. Since we deal with climate systems where everything is coupled, any oscillations not removed by one filter will be amplified through physical feedbacks if not suppressed by the other filter.

References

class exoplasim.**Model**(*resolution='T21'*, *layers=10*, *ncpus=4*, *precision=8*, *debug=False*, *inityear=0*, *recompile=False*, *optimization=None*, *mars=False*, *workdir='most'*, *source=None*, *force991=False*, *modelname='MOST_EXP'*, *outputtype='.npz'*, *crashtolerant=False*)

Bases: `object`

Create an ExoPlaSim model in a particular directory.

Initialize an ExoPlaSim model in a particular directory. If the necessary executable does not yet exist, compile it.

Parameters

- **resolution** (*str, optional*) – The resolution of the model. Options are T21, T42, T63, T85, T106, T127, and T170, corresponding to 32, 64, 96, 128, 160, 192, and 256 latitudes respectively, and twice as many longitudes. ExoPlaSim has been tested and validated most extensively at T21 and T42. Higher resolutions will take considerable time to run.
- **layers** (*int, optional*) – The number of vertical layers in the model atmosphere. The default is 10, but PlaSim has been used with 5 layers in many studies. More layers are supported, but not recommended except at higher resolutions.
- **ncpus** (*int, optional*) – The number of MPI processes to use, typically the number of cores available. If ncpus=1, MPI will not be used.
- **precision** (*int, optional*) – Either 4 or 8–specifies the number of bytes for a Fortran real.

- **debug** (*bool, optional*) – If True, compiler optimizations are disabled and the code is compiled with debugging flags enabled that will allow line-by-line tracebacks if ExoPlaSim crashes. Only use for development purposes.
- **inityear** (*int, optional*) – The number to use for the initial model year (default 0).
- **recompile** (*bool, optional*) – True/False flag used to force a recompile. Cannot force the model to skip compilation if the executable does not exist or compilation-inducing flags are set.
- **optimization** (*str, optional*) – Fortran compiler arguments for optimization. ANY compiler flags can be passed here, but it's intended for optimization flags. Setting this will trigger a recompile.
- **mars** (*bool, optional*) – True/False. If True, will use Mars-specific routines.
- **workdir** (*str, optional*) – The directory in which to construct the model.
- **source** (*str, optional*) – The directory in which to look for executables, namelists, boundary conditions, etc. If not set, will default to exoplasim/plasim/run/.
- **force991** (*bool, optional*) – Force the use of the FFT991 library instead of the default FFT library. Recommended for advanced use only.
- **modelname** (*str, optional*) – The name to use for the model and its output files when finished.
- **outputtype** (*str, optional*) – File extension to use for the output, if using the pyburn postprocessor. Supported extensions are *.nc*, *.npy*, *.npz*, *.hdf5*, *.he5*, *.h5*, *.csv*, *.gz*, *.txt*, *.tar*, *.tar.gz*, *.tar.xz*, and *.tar.bz2*. If using *.nc*, *netcdf4-python* must be installed. If using any of *.hdf5*, *.he5*, or *.h5*, then *h5py* must be installed. The default is the numpy compressed format, *.npz*.
- **crashtolerant** (*bool, optional*) – If True, then on a crash, ExoPlaSim will rewind 10 years and resume from there. If fewer than 10 years have elapsed, ExoPlaSim will simply crash.

Returns An instantiated Model object that resides in a directory with the namelists and executable necessary to run ExoPlaSim.

Return type *Model*

Examples

```
>>> import exoplasim as exo
>>> mymodel = exo.Model(workdir="mymodel_testrun",modelname="mymodel",resolution=
↪"T21",layers=10,ncpus=8)
>>> mymodel.configure()
>>> mymodel.exportcfg()
>>> mymodel.run(years=100,crashifbroken=True)
>>> mymodel.finalize("mymodel_output")
```

In this example, we initialize a model that will run in the directory "mymodel_testrun", and has the name "mymodel", which will be used to label output and error logs. The model has T21 resolution, or 32x64, 10 layers, and will run on 8 CPUs. By default, the compiler will use 8-byte precision. 4-byte may run slightly faster, but possibly at the cost of reduced stability. If there are machine-specific optimization flags you would like to use when compiling, you may specify them as a string to the optimization argument, e.g. `optimization='mavx'`. ExoPlaSim will check to see if an appropriate executable has already been created, and if not (or if flags indicating special compiler behavior such as debug=True or an optimization flag are set) it will compile one. We

then configure the model with all the default parameter choices, which means we will get a model of Earth. We then export the model configurations to a `.cfg` file (named automatically after the model), which will allow the model configuration to be recreated exactly by other users. We run the model for 100 years, with error-handling enabled. Finally, we tell the model to clean up after itself. It will take the most recent output files and rename them after the model name we chose, and delete all the intermediate output and configuration files.

cfgpostprocessor(*ftype='regular'*, *extension='.npz'*, *namelist=None*, *variables=['50', '51', '52', '53', '54', '110', '129', '130', '131', '132', '133', '134', '135', '137', '138', '139', '140', '141', '142', '143', '144', '145', '146', '147', '148', '149', '151', '152', '155', '156', '157', '158', '159', '160', '161', '162', '163', '164', '165', '166', '167', '168', '169', '170', '171', '172', '173', '174', '175', '176', '177', '178', '179', '180', '181', '182', '183', '184', '203', '204', '205', '207', '208', '209', '210', '211', '218', '221', '230', '232', '238', '259', '260', '261', '262', '263', '264', '265', '266', '267', '268', '269', '273', '274', '277', '278', '279', '280', '298', '299', '300', '301', '302', '303', '304', '305', '306', '307', '308', '318', '319', '320', '321', '322', '323', '324', '325', '326', '327', '328', '329', '404', '405', '406', '407', '408', '409']*, *mode='grid'*, *zonal=False*, *substellarlon=180.0*, *physfilter=False*, *timeaverage=True*, *stdev=False*, *times=12*, *interpolatetimes=True*, *transit=False*, *image=False*, *h2o_linelist='Exomol'*, *cloudfunc=None*, *smooth=False*, *smoothweight=0.95*)

Configure postprocessor options for pyburn.

Output format is determined by the file extension of outfile. Current supported formats are NetCDF (*.nc), numpy's ``np.savez_compressed`` format (*.npz), and CSV format. If NumPy's single-array .npy extension is used, .npz will be substituted–this is a compressed ZIP archive containing .npy files. Additionally, the CSV output format can be used in compressed form either individually by using the .gz file extension, or collectively via tarballs (compressed or uncompressed).

If a tarball format (e.g. *.tar or *.tar.gz) is used, output files will be packed into a tarball. gzip (.gz), bzip2 (.bz2), and lzma (.xz) compression types are supported. If a tarball format is not used, then accepted file extensions are .csv, .txt, or .gz. All three will produce a directory named following the filename pattern, with one file per variable in the directory. If the .gz extension is used, NumPy will compress each output file using gzip compression.

CSV-type files will only contain 2D variable information, so the first N-1 dimensions will be flattened. The original variable shape is included in the file header (prepended with a # character) as the first items in a comma- separated list, with the first non-dimension item given as the '|||' placeholder. On reading variables from these files, they should be reshaped according to these dimensions. This is true even in tarballs (which contain CSV files).

A T21 model output with 10 vertical levels, 12 output times, all supported variables in grid mode,and no standard deviation computation will have the following sizes for each format:

Format	Size
netCDF	12.8 MiB
HDF5	17.2 MiB
NumPy (default)	19.3 MiB
tar.xz	33.6 MiB
tar.bz2	36.8 MiB
gzipped	45.9 MiB
uncompressed	160.2 MiB

Using the NetCDF (.nc) format requires the netCDF4 python package.

Using the HDF4 format (.h5, .hdf5, .he5) requires the h5py python package.

All supported formats can be read by *exoplasim.gcmt.load()* and will return identical data objects analogous to netCDF4 archives.

Parameters

- **ftype** (*str, optional*) – Which type of output to set for this–is this a regular output file ('regular'), a snapshot output file ('snapshot'), or high-cadence ('highcadence')?
- **extension** (*str, optional*) – Output format to use, specified via file extension. Supported formats are netCDF (`.nc`), NumPy compressed archives (`.npy`, `.npz`), HDF5 archives (`.hdf5`, `.he5`, `.h5`), or plain-text comma-separated value files, which may be compressed individually or as a tarball (`.csv`, `.gz`, `.txt`, `.tar`, `.tar.gz`, `.tar.xz`, and `.tar.bz2`). If using netCDF, `netcdf4-python` must be installed. If using HDF5, then `h5py` must be installed. The default is the numpy compressed format, `.npz`.
- **namelist** (*str, optional*) – Path to a burn7 postprocessor namelist file. If not given, then *variables* must be set.
- **variables** (*list or dict, optional*) – If a list is given, a list of either variable keycodes (integers or strings), or the abbreviated variable name (e.g. 'ts' for surface temperature). If a dict is given, each item in the dictionary should have the keycode or variable name as the key, and the desired horizontal mode and additional options for that variable as a sub-dict. Each member of the sub-dict should be passable as **kwargs to :py:func`pyburn.advancedDataset() <exoplasim.pyburn.advancedDataset>`. If None, then `namelist` must be set.
- **mode** (*str, optional*) – Horizontal output mode, if modes are not specified for individual variables. Options are 'grid', meaning the Gaussian latitude-longitude grid used in ExoPlaSim, 'spectral', meaning spherical harmonics, 'fourier', meaning Fourier coefficients and latitudes, 'synchronous', meaning a Gaussian latitude-longitude grid in the synchronous coordinate system defined in Paradise, et al (2021), with the north pole centered on the substellar point, or 'syncfourier', meaning Fourier coefficients computed along the dipolar meridians in the synchronous coordinate system (e.g. the substellar-antistellar-polar meridian, which is 0 degrees, or the substellar-evening-antistellar-morning equatorial meridian, which is 90 degrees). Because this will get assigned to the original latitude array, that will become -90 degrees for the polar meridian, and 0 degrees for the equatorial meridian, identical to the typical equatorial coordinate system.
- **zonal** (*bool, optional*) – Whether zonal means should be computed for applicable variables.
- **substellarlon** (*float, optional*) – Longitude of the substellar point. Only relevant if a synchronous coordinate output mode is chosen.
- **physfilter** (*bool, optional*) – Whether or not a physics filter should be used in spectral transforms.
- **times** (*int or array-like or None, optional*) – Either the number of timestamps by which to divide the output, or a list of times given as a fraction of the output file duration (which enables e.g. a higher frequency of outputs during periapse of an eccentric orbit, when insolation is changing more rapidly). Note that if a list is given, all members of the list MUST be between 0 and 1, inclusive. If None, the timestamps in the raw output will be written directly to file.
- **timeaverage** (*bool, optional*) – Whether or not timestamps in the output

file should be averaged to produce the requested number of output timestamps. Timestamps for averaged outputs will correspond to the middle of the averaged time period.

- **stdev** (*bool, optional*) – Whether or not standard deviations should be computed. If timeaverage is True, this will be the standard deviation over the averaged time period; if False, then it will be the standard deviation over the whole duration of the output file
- **interpolatetimes** (*bool, optional*) – If true, then if the times requested don't correspond to existing timestamps, outputs will be linearly interpolated to those times. If false, then nearest-neighbor interpolation will be used.

configure(*noutput=True*, *flux=1367.0*, *startemp=None*, *starradius=1.0*, *starspec=None*, *pH2=None*, *pHe=None*, *pN2=None*, *pO2=None*, *pCO2=None*, *pAr=None*, *pNe=None*, *pKr=None*, *pH2O=None*, *gascon=None*, *pressure=None*, *pressurebroaden=True*, *vtype=0*, *rotationperiod=1.0*, *synchronous=False*, *substellarlon=180.0*, *keplerian=False*, *meananomaly0=None*, *year=None*, *glaciers={'initialh': - 1.0, 'mindepth': 2.0, 'toggle': False}*, *restartfile=None*, *gravity=None*, *radius=None*, *eccentricity=None*, *obliquity=None*, *lonvernaleq=None*, *fixedorbit=False*, *orography=None*, *seaice=True*, *co2weathering=False*, *evolveco2=False*, *physicsfilter=None*, *filterkappa=8.0*, *filterpower=8*, *filterLHN0=15.0*, *diffusionwaven=None*, *qdiffusion=None*, *tdiffusion=None*, *zdiffusion=None*, *ddiffusion=None*, *diffusionpower=None*, *erosionsupplylimit=None*, *outgassing=50.0*, *snowicealbedo=None*, *twobandalbedo=False*, *maxsnow=None*, *soilalbedo=None*, *oceanalbedo=None*, *oceanzenith='ECHAM-3'*, *wetsoil=False*, *soilwatercap=None*, *vegetation=False*, *vegaccel=1*, *nforestgrowth=1.0*, *initgrowth=0.5*, *initstomcond=1.0*, *initrough=2.0*, *initsoilcarbon=0.0*, *initplantcarbon=0.0*, *aquaplanet=False*, *desertplanet=False*, *soilsaturation=None*, *drycore=False*, *ozone=True*, *cpsoil=None*, *soildepth=1.0*, *mldepth=50.0*, *tlcontrast=0.0*, *desync=0.0*, *writefrequency=None*, *modeltop=None*, *stratosphere=False*, *top_restoretime=None*, *tropopause=None*, *timestep=45.0*, *runscript=None*, *columnmode=None*, *runsteps=None*, *highcadence={'end': 576, 'interval': 4, 'start': 320, 'toggle': 0}*, *snapshots=None*, *resources=[]*, *landmap=None*, *stormclim=False*, *nstorms=4*, *stormcapture={'ENDTHRESH': 16, 'GPITHRESH': 0.37, 'LAVTHRESH': 1.2e-05, 'MAXSTORMLEN': 1024, 'MAXSURFTEMP': 373.15, 'MINSTORMLEN': 256, 'MINSURFTEMP': 298.15, 'NKTRIGGER': 0, 'SIZETHRESH': 30, 'SWINDTHRESH': 20.5, 'VITHRESH': 0.145, 'VMXTHRESH': 33.0, 'VRMTHRESH': 0.577, 'WINDTHRESH': 33.0, 'toggle': 0}*, *topomap=None*, *threshold=0.0005*, *otherargs={}*)

Configure the model's namelists and boundary conditions.

The defaults here are appropriate for an Earth model.

Model Operation

noutput [bool, optional] True/False. Whether or not model output should be written.

restartfile [str, optional] Path to a restart file to use for initial conditions. Can be None.

writefrequency [int, optional] How many times per day ExoPlaSim should write output. Ignored by default–default is to write time-averaged output once every 5 days.

timestep [float, optional] Model timestep. Defaults to 45 minutes.

runscript [function , optional] A Python function that accepts a Model object as its first argument. This is the routine that will be run when you issue the Model.run() command. Any keyword arguments passed to run() will be forwarded to the specified function. If not set, the default internal routine will be used.

snapshots [int, optional] How many timesteps should elapse between snapshot outputs. If not set, no snapshots will be written.

restartfile [string, optional] Path to a restart file to use.

highcadence [dict, optional] A dictionary containing the following arguments:

- **`'toggle'`** [{0,1}] Whether or not high-cadence output should be written (1=yes).
- **`'start'`** [int] Timestep at which high-cadence output should begin.
- **`'end'`** [int] Timestep at which high-cadence output should end.
- **`'interval'`** [int] How many timesteps should elapse between high-cadence outputs.

threshold [float, optional] Energy balance threshold model should run to, if using *`runtobalance()`*. Default is <0.05 W/m^2/yr average drift in TOA and surface energy balance over 45-year timescales.

resources [list, optional] A list of paths to any additional files that should be available in the run directory.

runsteps [integer, optional] The number of timesteps to run each 'year'. By default, this is tuned to 360 Earth days. If set, this will override other controls setting the length of each modelled year.

otherargs [dict, optional] Any namelist parameters not included by default in the configuration options. These should be passed as a dictionary, with "PARAMETER@namelist" as the form of the dictionary key, and the parameter value passed as a string. e.g. `otherargs={"N_RUN_MONTHS@plasim_namelist":'4', "NGUI@plasim_namelist:'1'}`

Model Dynamics

columnmode [{None,"-","clear","static","static|clear","clear|static"}, optional] The inclusion of 'static' will disable horizontal advection, forcing ExoPlaSim into a column-only mode of operation. The inclusion of 'clear' will disable the radiative effects of clouds.

drycore [bool, optional] True/False. If True, evaporation is turned off, and a dry atmosphere will be used.

physicsfilter [str, optional] If not an empty string, specifies the physics filter(s) to be used. Filters can be used during the transform from gridpoint to spectral (`"gp"`), and/or during the transform from spectral to gridpoint (`"sp"`). Filter types are "none", "cesaro", "exp", or "lh" (see the Notes for more details). Combinations of filter types and times should be combined with a `|`, e.g. `physicsfilter="gp|exp|sp"` or `physicsfilter="gp|cesaro"`.

filterkappa [float, optional] A constant to be used with the exponential filter. Default is 8.0.

filterpower [int, optional] A constant integer to be used with the exponential filter. Default is 8.

filterLHN0 [float, optional] The constant used in the denominator of the Lander-Hoskins Filter. Default is 15; typically chosen so f(N)=0.1.

diffusionwaven [int, optional] The critical wavenumber beyond which hyperdiffusion is applied. Default is 15 for T21.

qdiffusion [float, optional] Timescale for humidity hyperdiffusion in days. Default for T21 is 0.1.

tdiffusion [float, optional] Timescale for temperature hyperdiffusion in days. Default for T21 is 5.6.

zdiffusion [float, optional] Timescale for vorticity hyperdiffusion in days. Default for T21 is 1.1.

ddiffusion [float, optional] Timescale for divergence hyperdiffusion in days.. Default for T21 is 0.2.

diffusionpower [int, optional] integer exponent used in hyperdiffusion. Default is 2 for T21.

Radiation

flux [float, optional] Incident stellar flux in W/m^2. Default 1367 for Earth.

startemp [float, optional] Effective blackbody temperature for the star. Not used if not set.

starradius [float, optional] Radius of the parent star in solar radii. Currently only used for the optional petitRADTRANS direct imaging postprocessor.

starspec [str, optional] Spectral file for the stellar spectrum. Should have two columns and 965 rows, with wavelength in the first column and radiance or intensity in the second. A similarly-named file with the "_hr.dat" suffix must also exist and have 2048 wavelengths. Appropriately-formatted files can be created with *makestellarspec.py*.

twobandalbedo [bool, optional] True/False. If True, separate albedos will be calculated for each of the two shortwave bands. If False (default), a single broadband albedo will be computed and used for both.

synchronous [bool, optional] True/False. If True, the Sun is fixed to one longitude in the sky.

desync [float, optional] The rate of drift of the substellar point in degrees per minute. May be positive or negative.

substellarlon [float, optional] The longitude of the substellar point, if synchronous==True. Default 180°

pressurebroaden [bool, optional] True/False. If False, pressure-broadening of absorbers no longer depends on surface pressure. Default is True

ozone [bool or dict, optional] True/False/dict. Whether or not forcing from stratospheric ozone should be included. If a dict is provided, it should contain the keys "height", "spread", "amount","varlat","varseason", and "seasonoffset", which correspond to the height in meters of peak O3 concentration, the width of the gaussian distribution in meters, the baseline column amount of ozone in cm-STP, the latitudinal amplitude, the magnitude of seasonal variation, and the time offset of the seasonal variation in fraction of a year. The three amounts are additive. To set a uniform, unvarying O3 distribution, ,place all the ozone in "amount", and set "varlat" and "varseason" to 0.

snowicealbedo [float, optional] A uniform albedo to use for all snow and ice.

soilalbedo [float, optional] A uniform albedo to use for all land.

wetsoil [bool, optional] True/False. If True, land albedo depends on soil moisture (wet=darker).

oceanalbedo [float, optional] A uniform albedo to use for the ocean.

oceanzenith [{"ECHAM-3","ECHAM-6","Lambertian}, optional] The zenith-angle dependence to use for blue-light reflectance from the ocean. Can be `'Lambertian'/'uniform'`, `'ECHAM-3'/'plasim'/'default'`, or `'ECHAM-6'`. The default is `'ECHAM-3'` (synonymous with `'plasim'` and `'default'`), which is the dependence used in the ECHAM-3 model.

Orbital Parameters

year [float, optional] Number of 24-hour days in a sidereal year. Not necessary if eccentricity and obliquity are zero. Defaults if not set to ~365.25 days

rotationperiod [float, optional] Planetary rotation period, in days. Default is 1.0.

eccentricity [float, optional] Orbital eccentricity. If not set, defaults to Earth's (0.016715)

obliquity [float, optional] Axial tilt, in degrees. If not set, defaults to Earth's obliquity (23.441°).

lonvernaleq [float, optional] Longitude of periapse, measured from vernal equinox, in degrees. If not set, defaults to Earth's (102.7°).

fixedorbit [bool, optional] True/False. If True, orbital parameters do not vary over time. If False, variations such as Milankovich cycles will be computed by PlaSim.

keplerian [bool, optional] True/False. If True, a generic Keplerian orbital calculation will be performed. This means no orbital precession, Milankovich cycles, etc, but does allow for accurate calculation of a wide diversity of orbits, including with higher eccentricity. Note that extreme orbits may have extreme results, including extreme crashes.

meananomaly0 [float, optional] The initial mean anomaly in degrees. Only used if *keplerian=True*.

Planet Parameters

gravity [float, optional] Surface gravity, in m/s^2. Defaults to 9.80665 m/s^2.

radius [float, optional] Planet radius in Earth radii. Default is 1.0.

orography [float, optional] If set, a scaling factor for topographic relief. If `orography=0.0`, topography will be zeroed-out.

aquaplanet [bool, optional] True/False. If True, the surface will be entirely ocean-covered.

desertplanet [bool, optional] True/False. If True, the surface will be entirely land-covered.

tlcontrast [float, optional] The initial surface temperature contrast between fixedlon and the anterior point. Default is 0.0 K.

seaice [bool, optional] True/False. If False, disables radiative effects of sea ice (although sea ice itself is still computed).

landmap [str, optional] Path to a `.sra` file containing a land mask for the chosen resolution.

topomap [str, optional] Path to a `.sra` file containing geopotential height map. Must include landmap.

Atmosphere

gascon [float, optional] Effective gas constant. Defaults to 287.0 (Earth), or the gas constant corresponding to the composition specified by partial pressures.

vtype [{0,1,2,3,4,5}, optional] Type of vertical discretization. Can be: 0 Pseudolinear scaling with pressure that maintains resolution near the ground. 1 Linear scaling with pressure. 2 Logarithmic scaling with pressure (resolves high altitudes) 3 Pseudologarithmic scaling with pressure that preserves resolution near the ground. 4 Pseudolinear scaling with pressure, pinned to a specified top pressure. 5 If >10 layers, bottom 10 as if `vtype=4`, and upper layers as if `vtype=2`.

modeltop [float, optional] Pressure of the top layer

tropopause [float, optional] If stratosphere is being included, pressure of the 10th layer (where scheme switches from linear to logarithmic).

stratosphere [bool, optional] True/False. If True, vtype=5 is used, and model is discretized to include a stratosphere.

pressure: float, optional Surface pressure in bars, if not specified through partial pressures.

Gas Partial Pressures

Partial pressures of individual gases can be specified. If pressure and gascon are not explicitly set, these will determine surface pressure, mean molecular weight, and effective gas constant. Note however that Rayleigh scattering assumes an Earth-like composition, and the only absorbers explicitly included in the radiation scheme are CO2 and H2O.

pH2 [float, optional] H2 partial pressure in bars.

pHe [float, optional] He partial pressure in bars.

pN2 [float, optional] N2 partial pressure in bars.

pO2 [float, optional] O2 partial pressure in bars.

pH2 [float, optional] H2 partial pressure in bars.

pAr [float, optional] Ar partial pressure in bars.

pNe [float, optional] Ne partial pressure in bars.

pKr [float, optional] Kr partial pressure in bars.

pCO2 [float, optional] CO2 partial pressure in bars. This gets translated into a ppmv concentration, so if you want to specify/vary CO2 but don't need the other gases, specifying pCO2, pressure, and gascon will do the trick. In most use cases, however, just specifying pN2 and pCO2 will give good enough behavior.

pH2O [float, optional] H2O partial pressure in bars. This is only useful in setting the gas constant and surface pressure; it will have no effect on actual moist processes.

Surface Parameters

mldepth [float, optional] Depth of the mixed-layer ocean. Default is 50 meters.

soildepth [float, optional] Scaling factor for the depth of soil layers (default total of 12.4 meters)

cpsoil [float, optional] Heat capacity of the soil, in J/m^3/K. Default is 2.4*10^6.

soilwatercap [float, optional] Water capacity of the soil, in meters. Defaults to 0.5 meters

soilsaturation [float, optional] Initial fractional saturation of the soil. Default is 0.0 (dry).

maxsnow [float, optional] Maximum snow depth (Default is 5 meters; set to -1 to have no limit).

Additional Physics

Carbon-Silicate Weathering

co2weathering [bool, optional] True/False. Toggles whether or not carbon-silicate weathering should be computed. Default is False.

evolveco2 [bool, optional] True/False. If co2weathering==True, toggles whether or not the CO2 partial pressure should be updated every year. Usually the change in pCO2 will be extremely small, so this is not necessary, and weathering experiments try to estimate the average weathering rate for a given climate in order to interpolate timescales between climates, rather than modelling changes in CO2 over time directly.

outgassing [float, optional] The assumed CO2 outgassing rate in units of Earth outgassing. Default is 1.0.

erosionsupplylimit [float, optional] If set, the maximum CO2 weathering rate per year permitted by erosion, in ubars/year. This is not simply a hard cutoff, but follows Foley 2015 so high weathering below the cutoff is also reduced.

Vegetation

vegetation [bool or int, optional] Can be True/False, or 0/1/2. If True or 1, then diagnostic vegetation is turned on. If 2, then coupled vegetation is turned on. Vegetation is computed via the SimBA module.

vegaccel [int, optional] Integer factor by which to accelerate vegetation growth

nforestgrowth: float, optional Biomass growth

initgrowth [float, optional] Initial above-ground growth

initstomcond [float, optional] Initial stomatal conductance

initrough [float, optional] Initial vegetative surface roughness

initsoilcarbon [float, optional] Initial soil carbon content

initplantcarbon [float, optional] Initial vegetative carbon content

See [1]_ for details on the implementation of supply-limited weathering.

Glaciology

glaciers [dict, optional] A dictionary containing the following arguments: toggle : bool

True/False. Whether or not glaciers should be allowed to grow or shrink in thickness, or be formed from persistent snow on land.

mindepth [float] The minimum snow depth in meters of liquid water equivalent that must persist year-round before the grid cell is considered glaciated. Default is 2 meters.

initialh [float] If >=0, covers the land surface with ice sheets of a height given in meterss. If -1, no initial ice sheets are assumed.

Storm Climatology

stormclim [bool, optional] True/False. Toggles whether or not storm climatology (convective available potential energy, maximum potential intensity, ventilation index, etc) should be computed. If True, output fields related to storm climatology will be added to standard output files. Enabling this mode currently roughly doubles the computational cost of the model. This may improve in future updates. Refer to Paradise, et al 2021 for implementation description.

stormcapture [dict, optional] A dictionary containing arguments controlling when high-cadence output is triggered by storm activity. This dictionary must contain 'toggle', which can be either 1 or 0 (yes or no). It may also contain any namelist parameters accepted by hurricanemod.f90, including the following:

toggle [{0,1}] Whether (1) or not (0) to write high-cadence output when storms occur

NKTRIGGER [{0,1}, optional] (0/1=no/yes). Whether or not to use the Komacek, et al 2020 conditions for hurricane cyclogenesis as the output trigger. Default is no.

VITHRESH [float, optional] (nktrigger) Ventilation index threshold for nktrigger output. Default 0.145

VMXTHRESH [float, optional] (nktrigger) Max potential intensity threshold for nktrigger output.Default 33 m/s

LAVTHRESH [float, optional] (nktrigger) Lower-atmosphere vorticity threshold for nktrigger output. Default 1.2*10^-5 s^-1

VRMTHRESH [float, optional] (unused) Ventilation-reduced maximum intensity threshold. Default 0.577

GPITHRESH [float, optional] (default) Genesis Potential Index threshold. Default 0.37.

MINSURFTEMP [float, optional] (default) Min. surface temperature for storm activity. Default 25C

MAXSURFTEMP [float, optional] (default) Max. surface temperature for storm activity. Default 100C

WINDTHRESH [float, optional] (default) Lower-atmosphere maximum wind threshold for storm activity. Default 33 m/s

SWINDTHRESH [float, optional] (default) Minimum surface windspeed for storm activity. Default 20.5 m/s

SIZETHRESH [float, optional] (default) Minimum number of cells that must trigger to start outputDefault 30

ENDTHRESH [float, optional] (default) Minimum number of cells at which point storm output ends.Default 16

MINSTORMLEN [float, optional] (default) Minimum number of timesteps to write output. Default 256

MAXSTORMLEN [float, optional] (default) Maximum number of timesteps to write output. Default 1024

Note that actual number of writes will be stormlen/interval, as set in highcadence. This interval defaults to 4, so 64 writes minimum, 256 max. For more details on the storm climatology factors considered here, see [5]_.

Notes

In some cases, it may be necessary to include physics filters. This typically becomes necessary when sharp features are projected on the model's smallest spectral modes, causing Gibbs "ripples". Earth-like models typically do not require filtering, but tidally-locked models do. Filtering may be beneficial for Earth-like models at very high resolutions as well, or if there is sharp topography.

Three filter functional forms are included in ExoPlaSim: Cesaro, exponential, and Lander-Hoskins. Their functional forms are given below, where *n* is the wavenumber, and *N* is the truncation wavenumber (e.g. 21 for T21):

Cesaro: $f(n) = 1 - \frac{n}{N+1}$ [2]_

Exponential: $f(n) = \exp\left[-\kappa\left(\frac{n}{N}\right)^{\gamma}\right]$ [3]_

Lander-Hoskins: $f(n) = \exp\left[-\left(\frac{n(n+1)}{n_0(n_0+1}\right)^2\right]$ [3]_ [4]_

κ is exposed to the user through `filterkappa`, γ is exposed through `filterpower`, and n_0 is exposed through `filterLHN0`.

Physics filters can be applied at two different points; either at the transform from gridpoint to spectral, or the reverse. We find that in most cases, the ideal usage is to use both. Generally, a filter at the gridpoint->spectral transform is good for dealing with oscillations caused by sharp jumps and small features in the gridpoint tendencies. Conversely, a filter at the spectral->gridpoint transform is good for dealing with oscillations that come from small-scale features in the spectral fields causing small-scale features to appear in the gridpoint tendencies [3]_. Since we deal with climate systems where everything is coupled, any oscillations not removed by one filter will be amplified through physical feedbacks if not suppressed by the other filter.

References

emergencyabort()
A problem has been encountered by an external script, and the model needs to crash gracefully

exportcfg(*filename=None*)
Export model configuration to a text file that can be used as configuration input

Write the current model configuration to a text file. This file can be shared and used by other users to recreate your model configuration.

Parameters **filename** (*str, optional*) – Path to the file that should be written. If None (default), <modelname>.cfg will be created in the working directory.

See also:

loadconfig : Load a saved configuration.

finalize(*outputdir*, *allyears=False*, *keeprestarts=False*, *clean=True*)
Move outputs and optionally restarts to a specified output directory.

If more than the final year of output is being kept, a folder will be created in the output directory using the model name. Otherwise, finalized files will be renamed using the model name.

Parameters

- **outputdir** (*str*) – Directory in which to put output.
- **allyears** (*bool, optional*) – True/False. If True, output from all years will be kept, in a directory in outputdir named with the model name. Otherwise, the most recent year will be kept in outputdir, using the model name. Default False.
- **keeprestarts** (*bool, optional*) – True/False: If True, restart files will be kept as well as output files. Default False.
- **clean** (*bool, optional*) – True/False. If True, the original working directory will be deleted after files are moved. Default True.

get(*year*, *snapshot=False*, *highcadence=False*)

Return an open NetCDF data object for the given year. Defaults is to return time-averaged output.

Parameters

- **year** (*int*) – Integer year of output to return
- **snapshot** (*bool, optional*) – True/False. If True, return the snapshot version.
- **highcadence** (*bool, optional*) – True/False. If True, return the high-cadence version.

Returns An open netCDF4 data opject

Return type netCDF4.Dataset

getbalance(*key*, *year=- 1*)

Return the global annual mean of a given variable for a given year

Parameters

- **key** (*str*) – The output variable string to return
- **year** (*int, optional*) – Which year to go to for output

Returns Global annual mean of requested quantity

Return type float

gethistory(*key='ts'*, *mean=True*, *layer=- 1*)

Return the an array of global annual means of a given variable for each year

Parameters

- **key** (*str, optional*) – The output variable string to return
- **mean** (*bool, optional*) – Toggle whether we return the mean or the sum
- **year** (*int, optional*) – Which year to go to for output

Returns 1-D Array of global annual means

Return type numpy.ndarray

image(*year*, *times*, *obsv_coords*, *snapshot=True*, *highcadence=False*, *h2o_linelist='Exomol'*, *num_cpus=None*, *cloudfunc=None*, *smooth=True*, *smoothweight=0.95*, *filldry=1e-06*, *oren-nayar=True*, *debug=False*, *logfile=None*, *filename=None*, *inputfile=None*, *baremoun-tainz=50000.0*, *colorspace='sRGB'*, *gamma=True*, *consistency=True*, *vegpowerlaw=1.0*)

Compute reflection+emission spectra for snapshot output

This routine computes the reflection+emission spectrum for the planet at each indicated time.

Note that deciding what the observer coordinates ought to be may not be a trivial operation. Simply setting them to always be the same is fine for a 1:1 synchronously-rotating planet, where the insolation pattern never changes. But for an Earth-like rotator, you will need to be mindful of rotation rate and the local time

when snapshots are written. Perhaps you would like to see how things look as the local time changes, as a geosynchronous satellite might observe, or maybe you'd like to only observe in secondary eclipse or in quadrature, and so the observer-facing coordinates may not be the same each time.

Parameters

- **year** (*int*) – Year of output that should be imaged.
- **times** (*list(int)*) – List of time indices at which the image should be computed.
- **obsv_coords** (*numpy.ndarray (3D)*) – List of observer (lat,lon) coordinates for each observing time. First axis is time, second axis is for each observer; the third axis is for lat and lon. Should have shape (time,observers,lat-lon). These are the surface coordinates that are directly facing the observer.
- **snapshot** (*bool, optional*) – Whether snapshot output should be used.
- **highcadence** (*bool, optional*) – Whether high-cadence output should be used.
- **h2o_linelist** (*{'HITEMP','EXOMOL'}, optional*) – Either 'HITEMP' or 'EXOMOL'–the line list from which H2O absorption should be sourced
- **num_cpus** (*int, optional*) – The number of CPUs to use
- **cloudfunc** (*function, optional*) – A routine which takes pressure, temperature, and cloud water content as arguments, and returns keyword arguments to be unpacked into calc_flux_transm. If not specified, *basicclouds* will be used.
- **smooth** (*bool, optional*) – Whether or not to smooth humidity and cloud columns. As of Nov 12, 2021, it is recommended that you use smooth=True for well-behaved spectra. This is a conservative smoothing operation, meaning the water and cloud column mass should be conserved–what this does is move some water from the water-rich layers into the layers directly above and below.
- **smoothweight** (*float, optional*) – The fraction of the water in a layer that should be retained during smoothing. A higher value means the smoothing is less severe. 0.95 is probably the upper limit for well-behaved spectra.
- **filldry** (*float, optional*) – If nonzero, the floor value for water humidity when moist layers are present above dry layers. Columns will be adjusted in a mass-conserving manner with excess humidity accounted for in layers *above* the filled layer, such that total optical depth from TOA is maintained at the dry layer.
- **orennayar** (*bool, optional*) – If True, compute true-colour intensity using Oren-Nayar scattering instead of Lambertian scattering. Most solar system bodies do not exhibit Lambertian scattering.
- **debug** (*bool, optional*) – Optional debugging mode, that outputs intermediate quantities used in the imaging process.
- **logfile** (*str, optional*) – Optional log file to write diagnostics to.
- **filename** (*str, optional*) – Output filename; will be auto-generated if None.
- **inputfile** (*str, optional*) – If provided, ignore the year argument and image the provided output file.
- **baremountainz** (*float, optional*) – If vegetation is present, the geopotential above which mountains become bare rock instead of eroded vegetative regolith. Functionally, this means gray rock instead of brown/tan ground.

- **colorspace** (*str or np.ndarray(3,3)*) – Color gamut to be used. For available built-in color gamuts, see colormatch.colorgamuts.
- **gamma** (*bool or float, optional*) – If True, use the piecewise gamma-function defined for sRGB; otherwise if a float, use rgb^(1/gamma). If None, gamma=1.0 is used.
- **consistency** (*bool, optional*) – If True, force surface albedo to match model output
- **vegpowerlaw** (*float, optional*) – Scale the apparent vegetation fraction by a power law. Setting this to 0.1, for example, will increase the area that appears partially-vegetated, while setting it to 1.0 leaves vegetation unchanged.

Returns pRT Atmosphere object, filename the output file generated. Output file can be stored in any of ExoPlaSim's standard supported output formats.

Return type petitRADTRANS.Atmosphere, str

inspect(*variable*, *year=-1*, *ignoreNaNs=True*, *snapshot=False*, *highcadence=False*, *savg=False*, *tavg=False*, *layer=None*)

Return a given output variable from a given year or list of years, with optional averaging parameters.

Parameters

- **variable** (*str*) – The name of the variable to return.
- **year** (*int, optional OR array-like*) – Which year of output to return. Year indexing follows Pythonic rules. If the model has been finalized, only the final year of output will be returned. If year is an array-like with length>1, the years implied by the list will be concatenated into a single output, along the time axis.
- **ignoreNaNs** (*bool, optional*) – True/False. If True, use NaN-tolerant numpy functions.
- **snapshot** (*bool, optional*) – True/False. If True, use snapshot output instead of time-averaged.
- **highcadence** (*bool, optional*) – True/False. If True, use high-cadednce output instead of time-averaged.
- **savg** (*bool, optional*) – True/False. If True, compute the spatial average. Default False
- **tavg** (*bool, optional*) – True/False. If True, compute the annual average. Default False
- **layer** (*int, optional*) – If specified and data has 3 spatial dimensions, extract the specified layer. If unspecified and data has 3 spatial dimensions, the vertical dimension will be preserved (even if spatial averages are being computed).

Returns The requested data, averaged if that was requested.

Return type float or numpy.ndarray

integritycheck(*ncfile*)

Check an output file to see it contains the expected variables and isn't full of NaNs.

If the file does not exist, exoplasim will attempt to create it using the postprocessor. If the file does not have the expected variables or is full of trash, an exception will be raised. If the file is fine, this function returns a 1. If the file did not exist and cannot be created, this function will return a 0.

Parameters **ncfile** (*str*) – The output file to check.

Returns 0 or 1 depending on failure or success respectively

Return type int

loadconfig(*configfile*)
Load a previously-exported configuration file and configure the model accordingly.

Parameters **configfile** (*str*) – Path to the configuration file to load

See also:

exportcfg : Export model configuration to a text file.

modify(***kwargs*)
Modify any already-configured parameters. All parameters accepted by *configure()* can be passed as arguments.

See also:

configure : Set model parameters and boundary conditions

postprocess(*inputfile*, *variables*, *ftype='regular'*, *log='postprocess.log'*, *crashifbroken=False*, *transit=False*, *image=False*, ***kwargs*)
Produce NetCDF output from an input file, using a specified postprocessing namelist.

Parameters

- **inputfile** (*str*) – The raw output file to be processed
- **variables** (*str or list or dict or None*) – Can be a path to a burn7-style namelist, a list of variable codes or keys, or a dictionary containing output options for each variable. If None, then a variable set pre-configured with :py:func`Model.cfgpostprocessor() <exoplasim.Model.cfgpostprocessor>` will be used. If the postprocessor was not pre-configured, this will prompt pyburn to use the default set.
- **ftype** (*str, optional*) – Which type of output to set for this–is this a regular output file ('regular'), a snapshot output file ('snapshot'), or high-cadence ('highcadence')?
- **log** (*str, optional*) – The log file to which pyburn should output standard output and errors
- **crashifbroken** (*bool, optional*) – True/False. If True, exoplasim will run .integritycheck() on the file.
- ****kwargs** (*keyword arguments*) – Keyword arguments accepted by pyburn.postprocess. Do not specify radius, gravity, or gascon. These are set by the model configuration. Specifying additional keywords here will override any options set via :py:func`Model.cfgpostprocessor() <exoplasim.Model.cfgpostprocessor>`

Returns 1 if successful, 0 if not

Return type int

run(***kwargs*)
Run the Model's designated run routine.

This may have been passed as runscript when the model was created, or it could be the model's internal ._run() routine. That method takes the following arguments:

Parameters

- **years** (*int, optional*) – Number of years to run
- **postprocess** (*bool, optional*) – True/False. Whether or not output files should be produced on-the-fly

- **crashifbroken** (*bool, optional*) – True/False. If True, use Pythonic error handling
- **clean** (*bool, optional*) – True/False. If True, delete raw output files once output files are made

runtobalance(*threshold=None*, *baseline=50*, *maxyears=300*, *minyears=75*, *timelimit=None*, *crashifbroken=True*, *clean=True*, *diagnosticvars=None*)

Run the model until energy balance equilibrium is reached at the top and surface.

Parameters

- **threshold** (*float, optional*) – If specified, overrides the threshold set by `.config()`. The model will run until the energy balance at the top and surface drifts by less than this amount per year over a given baseline.
- **baseline** (*int, optional*) – The number of years over which to evaluate energy balance drift. Default 50
- **maxyears** (*int, optional*) – The maximum number of years to run before returning. Default 300. This is useful if you are running on a scratch disk with limited space.
- **minyears** (*int, optional*) – The minimum number of years to run before determining that the model is in equilibrium.
- **timelimit** (*float, optional*) – If set, maxyears will be revised each year based on the average minutes per year thus far, to try to avoid going over the time limit, which should be given in minutes.
- **crashifbroken** (*bool, optional*) – True/False. If True, Pythonic error handling is enabled. Default True.
- **clean** (*bool, optional*) – True/False. If True, raw output is deleted once post-processed. Default True.
- **diagnosticvars** (*array-like, optional*) – List of output variables for which global annual means should be computed and printed to standard output each year.

Returns True if the model reached equilibrium, False if not.

Return type bool

save(*filename=None*)

Save the current Model object to a NumPy save file.

The model object can then be reinstantiated using `numpy.load(savefile).item()`.

Parameters **filename** (*str, optional*) – Filename to save to. If unspecified, will default to <modelname>.npy.

Notes

Note that these files are often not portable between versions of Python or machine architectures, so their use is only recommended internally. For sharing with other users, it is recommended that you use the *exportcfg* function.

See also:

exportcfg : Export model configuration to a portable text file.

transit(*year*, *times*, *inputfile=None*, *snapshot=True*, *highcadence=False*, *h2o_linelist='Exomol'*, *num_cpus=1*, *cloudfunc=None*, *smooth=False*, *smoothweight=0.95*, *logfile=None*, *filename=None*)

Compute transmission spectra for snapshot output

This routine computes the transmission spectrum for each atmospheric column along the terminator, for each time in transittimes.

Note: This routine does not currently include emission from atmospheric layers.

Parameters

- **year** (*int*) – Year of output that should be imaged.
- **times** (*list (int)*) – List of time indices at which the image should be computed.
- **inputfile** (*str, optional*) – If provided, ignore the year argument and image the provided output file.
- **snapshot** (*bool, optional*) – Whether snapshot output should be used.
- **highcadence** (*bool, optional*) – Whether high-cadence output should be used.
- **h2o_lines** (*{'HITEMP','EXOMOL'}, optional*) – Either 'HITEMP' or 'EXOMOL'–the line list from which H2O absorption should be sourced
- **num_cpus** (*int, optional*) – The number of CPUs to use
- **cloudfunc** (*function, optional*) – A routine which takes pressure, temperature, and cloud water content as arguments, and returns keyword arguments to be unpacked into calc_flux_transm. If not specified, *basicclouds* will be used.
- **smooth** (*bool, optional*) – Whether or not to smooth humidity and cloud columns. As of Nov 12, 2021, it is recommended that you use smooth=True for well-behaved spectra. This is a conservative smoothing operation, meaning the water and cloud column mass should be conserved–what this does is move some water from the water-rich layers into the layers directly above and below.
- **smoothweight** (*float, optional*) – The fraction of the water in a layer that should be retained during smoothing. A higher value means the smoothing is less severe. 0.95 is probably the upper limit for well-behaved spectra.
- **logfile** (*str, optional*) – Optional log file to which diagnostic info will be written.
- **filename** (*str, optional*) – Output filename; will be auto-generated if None.

Returns pRT Atmosphere object, filename the output file generated. Output file can be stored in any of ExoPlaSim's standard supported output formats. Transit radius is in km.

Return type petitRADTRANS.Atmosphere, str

class exoplasim.**TLaquaplanet**(*resolution='T21'*, *layers=10*, *ncpus=4*, *precision=8*, *debug=False*, *inityear=0*, *recompile=False*, *optimization=None*, *mars=False*, *workdir='most'*, *source=None*, *force991=False*, *modelname='MOST_EXP'*, *outputtype='.npz'*, *crashtolerant=False*)

Bases: *exoplasim.Model*

Create a tidally-locked planet with no land.

Identical to *Model*, except configuration options suitable for tidally-locked models are the default when configure() is called, and the surface is entirely ocean-covered. Specifically, a 30-minute timestep, snapshot outputs every 720 timesteps, eccentricity=0.0, 0-degree obliquity, exponential physics filtering, fixed orbital parameters, and no ozone. All these defaults can be overridden.

configure(*timestep=30.0*, *snapshots=720*, *eccentricity=0.0*, *ozone=False*, *obliquity=0.0*, *physicsfilter='gp|exp|sp'*, *tlcontrast=100.0*, ***kwargs*)

Configure the model's namelists and boundary conditions.

The defaults here are appropriate for an Earth model.

Model Operation

noutput [bool, optional] True/False. Whether or not model output should be written.

restartfile [str, optional] Path to a restart file to use for initial conditions. Can be None.

writefrequency [int, optional] How many times per day ExoPlaSim should write output. Ignored by default–default is to write time-averaged output once every 5 days.

timestep [float, optional] Model timestep. Defaults to 45 minutes.

runscript [function , optional] A Python function that accepts a Model object as its first argument. This is the routine that will be run when you issue the Model.run() command. Any keyword arguments passed to run() will be forwarded to the specified function. If not set, the default internal routine will be used.

snapshots [int, optional] How many timesteps should elapse between snapshot outputs. If not set, no snapshots will be written.

restartfile [string, optional] Path to a restart file to use.

highcadence [dict, optional] A dictionary containing the following arguments:

'toggle' [{0,1}] Whether or not high-cadence output should be written (1=yes).

'start' [int] Timestep at which high-cadence output should begin.

'end' [int] Timestep at which high-cadence output should end.

'interval' [int] How many timesteps should elapse between high-cadence outputs.

threshold [float, optional] Energy balance threshold model should run to, if using *runtobalance()*. Default is <0.05 W/m^2/yr average drift in TOA and surface energy balance over 45-year timescales.

resources [list, optional] A list of paths to any additional files that should be available in the run directory.

runsteps [integer, optional] The number of timesteps to run each 'year'. By default, this is tuned to 360 Earth days. If set, this will override other controls setting the length of each modelled year.

otherargs [dict, optional] Any namelist parameters not included by default in the configuration options. These should be passed as a dictionary, with "PARAMETER@namelist" as the form of the dictionary key, and the parameter value passed as a string. e.g. `otherargs={"N_RUN_MONTHS@plasim_namelist":'4', "NGUI@plasim_namelist:'1'}`

Model Dynamics

columnmode [{None,"-","clear","static","static|clear","clear|static"}, optional] The inclusion of 'static' will disable horizontal advection, forcing ExoPlaSim into a column-only mode of operation. The inclusion of 'clear' will disable the radiative effects of clouds.

drycore [bool, optional] True/False. If True, evaporation is turned off, and a dry atmosphere will be used.

physicsfilter [str, optional] If not an empty string, specifies the physics filter(s) to be used. Filters can be used during the transform from gridpoint to spectral (`"gp"`), and/or during the transform from spectral to gridpoint (`"sp"`). Filter types are "none", "cesaro", "exp", or "lh" (see the Notes for more details). Combinations of filter types and times should be combined with a `|`, e.g. `physicsfilter="gp|exp|sp"` or `physicsfilter="gp|cesaro"`.

filterkappa [float, optional] A constant to be used with the exponential filter. Default is 8.0.

filterpower [int, optional] A constant integer to be used with the exponential filter. Default is 8.

filterLHN0 [float, optional] The constant used in the denominator of the Lander-Hoskins Filter. Default is 15; typically chosen so f(N)=0.1.

diffusionwaven [int, optional] The critical wavenumber beyond which hyperdiffusion is applied. Default is 15 for T21.

qdiffusion [float, optional] Timescale for humidity hyperdiffusion in days. Default for T21 is 0.1.

tdiffusion [float, optional] Timescale for temperature hyperdiffusion in days. Default for T21 is 5.6.

zdiffusion [float, optional] Timescale for vorticity hyperdiffusion in days. Default for T21 is 1.1.

ddiffusion [float, optional] Timescale for divergence hyperdiffusion in days.. Default for T21 is 0.2.

diffusionpower [int, optional] integer exponent used in hyperdiffusion. Default is 2 for T21.

Radiation

flux [float, optional] Incident stellar flux in W/m^2. Default 1367 for Earth.

startemp [float, optional] Effective blackbody temperature for the star. Not used if not set.

starradius [float, optional] Radius of the parent star in solar radii. Currently only used for the optional petitRADTRANS direct imaging postprocessor.

starspec [str, optional] Spectral file for the stellar spectrum. Should have two columns and 965 rows, with wavelength in the first column and radiance or intensity in the second. A similarly-named file with the "_hr.dat" suffix must also exist and have 2048 wavelengths. Appropriately-formatted files can be created with *makestellarspec.py*.

twobandalbedo [bool, optional] True/False. If True, separate albedos will be calculated for each of the two shortwave bands. If False (default), a single broadband albedo will be computed and used for both.

synchronous [bool, optional] True/False. If True, the Sun is fixed to one longitude in the sky.

desync [float, optional] The rate of drift of the substellar point in degrees per minute. May be positive or negative.

substellarlon [float, optional] The longitude of the substellar point, if synchronous==True. Default 180°

pressurebroaden [bool, optional] True/False. If False, pressure-broadening of absorbers no longer depends on surface pressure. Default is True

ozone [bool or dict, optional] True/False/dict. Whether or not forcing from stratospheric ozone should be included. If a dict is provided, it should contain the keys "height", "spread", "amount","varlat","varseason", and "seasonoffset", which correspond to the height in meters of peak O3 concentration, the width of the gaussian distribution in meters, the baseline column amount of ozone in cm-STP, the latitudinal amplitude, the magnitude of seasonal variation, and the time offset of the seasonal variation in fraction of a year. The three amounts are additive. To set a uniform, unvarying O3 distribution, ,place all the ozone in "amount", and set "varlat" and "varseason" to 0.

snowicealbedo [float, optional] A uniform albedo to use for all snow and ice.

soilalbedo [float, optional] A uniform albedo to use for all land.

wetsoil [bool, optional] True/False. If True, land albedo depends on soil moisture (wet=darker).

oceanalbedo [float, optional] A uniform albedo to use for the ocean.

oceanzenith [{"ECHAM-3","ECHAM-6","Lambertian}, optional] The zenith-angle dependence to use for blue-light reflectance from the ocean. Can be `'Lambertian'/'uniform'`, `'ECHAM-3'/'plasim'/'default'`, or `'ECHAM-6'`. The default is `'ECHAM-3'` (synonymous with `'plasim'` and `'default'`), which is the dependence used in the ECHAM-3 model.

Orbital Parameters

year [float, optional] Number of 24-hour days in a sidereal year. Not necessary if eccentricity and obliquity are zero. Defaults if not set to ~365.25 days

rotationperiod [float, optional] Planetary rotation period, in days. Default is 1.0.

eccentricity [float, optional] Orbital eccentricity. If not set, defaults to Earth's (0.016715)

obliquity [float, optional] Axial tilt, in degrees. If not set, defaults to Earth's obliquity (23.441°).

lonvernaleq [float, optional] Longitude of periapse, measured from vernal equinox, in degrees. If not set, defaults to Earth's (102.7°).

fixedorbit [bool, optional] True/False. If True, orbital parameters do not vary over time. If False, variations such as Milankovich cycles will be computed by PlaSim.

keplerian [bool, optional] True/False. If True, a generic Keplerian orbital calculation will be performed. This means no orbital precession, Milankovich cycles, etc, but does allow for accurate calculation of a wide diversity of orbits, including with higher eccentricity. Note that extreme orbits may have extreme results, including extreme crashes.

meananomaly0 [float, optional] The initial mean anomaly in degrees. Only used if *keplerian=True*.

Planet Parameters

gravity [float, optional] Surface gravity, in m/s^2. Defaults to 9.80665 m/s^2.

radius [float, optional] Planet radius in Earth radii. Default is 1.0.

orography [float, optional] If set, a scaling factor for topographic relief. If `orography=0.0`, topography will be zeroed-out.

aquaplanet [bool, optional] True/False. If True, the surface will be entirely ocean-covered.

desertplanet [bool, optional] True/False. If True, the surface will be entirely land-covered.

tlcontrast [float, optional] The initial surface temperature contrast between fixedlon and the anterior point. Default is 0.0 K.

seaice [bool, optional] True/False. If False, disables radiative effects of sea ice (although sea ice itself is still computed).

landmap [str, optional] Path to a `.sra` file containing a land mask for the chosen resolution.

topomap [str, optional] Path to a `.sra` file containing geopotential height map. Must include landmap.

Atmosphere

gascon [float, optional] Effective gas constant. Defaults to 287.0 (Earth), or the gas constant corresponding to the composition specified by partial pressures.

vtype [{0,1,2,3,4,5}, optional] Type of vertical discretization. Can be: 0 Pseudo-linear scaling with pressure that maintains resolution near the ground. 1 Linear scaling with pressure. 2 Logarithmic scaling with pressure (resolves high altitudes) 3 Pseudologarithmic scaling with pressure that preserves resolution near the ground. 4 Pseudolinear scaling with pressure, pinned to a specified top pressure. 5 If >10 layers, bottom 10 as if `vtype=4`, and upper layers as if `vtype=2`.

modeltop [float, optional] Pressure of the top layer

tropopause [float, optional] If stratosphere is being included, pressure of the 10th layer (where scheme switches from linear to logarithmic).

stratosphere [bool, optional] True/False. If True, vtype=5 is used, and model is discretized to include a stratosphere.

pressure: float, optional Surface pressure in bars, if not specified through partial pressures.

Gas Partial Pressures

Partial pressures of individual gases can be specified. If pressure and gascon are not explicitly set, these will determine surface pressure, mean molecular weight, and effective gas constant. Note however that Rayleigh scattering assumes an Earth-like composition, and the only absorbers explicitly included in the radiation scheme are CO2 and H2O.

pH2 [float, optional] H2 partial pressure in bars.

pHe [float, optional] He partial pressure in bars.

pN2 [float, optional] N2 partial pressure in bars.

pO2 [float, optional] O2 partial pressure in bars.

pH2 [float, optional] H2 partial pressure in bars.

pAr [float, optional] Ar partial pressure in bars.

pNe [float, optional] Ne partial pressure in bars.

pKr [float, optional] Kr partial pressure in bars.

pCO2 [float, optional] CO2 partial pressure in bars. This gets translated into a ppmv concentration, so if you want to specify/vary CO2 but don't need the other gases, specifying pCO2, pressure, and gascon will do the trick. In most use cases, however, just specifying pN2 and pCO2 will give good enough behavior.

pH2O [float, optional] H2O partial pressure in bars. This is only useful in setting the gas constant and surface pressure; it will have no effect on actual moist processes.

Surface Parameters

mldepth [float, optional] Depth of the mixed-layer ocean. Default is 50 meters.

soildepth [float, optional] Scaling factor for the depth of soil layers (default total of 12.4 meters)

cpsoil [float, optional] Heat capacity of the soil, in J/m^3/K. Default is 2.4*10^6.

soilwatercap [float, optional] Water capacity of the soil, in meters. Defaults to 0.5 meters

soilsaturation [float, optional] Initial fractional saturation of the soil. Default is 0.0 (dry).

maxsnow [float, optional] Maximum snow depth (Default is 5 meters; set to -1 to have no limit).

Additional Physics

Carbon-Silicate Weathering

co2weathering [bool, optional] True/False. Toggles whether or not carbon-silicate weathering should be computed. Default is False.

evolveco2 [bool, optional] True/False. If co2weathering==True, toggles whether or not the CO2 partial pressure should be updated every year. Usually the change in pCO2 will be extremely small, so this is not necessary, and weathering experiments try to estimate the average weathering rate for a given climate in order to interpolate timescales between climates, rather than modelling changes in CO2 over time directly.

outgassing [float, optional] The assumed CO2 outgassing rate in units of Earth outgassing. Default is 1.0.

erosionsupplylimit [float, optional] If set, the maximum CO2 weathering rate per year permitted by erosion, in ubars/year. This is not simply a hard cutoff, but follows Foley 2015 so high weathering below the cutoff is also reduced.

Vegetation

vegetation [bool or int, optional] Can be True/False, or 0/1/2. If True or 1, then diagnostic vegetation is turned on. If 2, then coupled vegetation is turned on. Vegetation is computed via the SimBA module.

vegaccel [int, optional] Integer factor by which to accelerate vegetation growth

nforestgrowth: float, optional Biomass growth

initgrowth [float, optional] Initial above-ground growth

initstomcond [float, optional] Initial stomatal conductance

initrough [float, optional] Initial vegetative surface roughness

initsoilcarbon [float, optional] Initial soil carbon content

initplantcarbon [float, optional] Initial vegetative carbon content

See [1]_ for details on the implementation of supply-limited weathering.

Glaciology

glaciers [dict, optional] A dictionary containing the following arguments: toggle : bool

True/False. Whether or not glaciers should be allowed to grow or shrink in thickness, or be formed from persistent snow on land.

mindepth [float] The minimum snow depth in meters of liquid water equivalent that must persist year-round before the grid cell is considered glaciated. Default is 2 meters.

initialh [float] If >=0, covers the land surface with ice sheets of a height given in meterss. If -1, no initial ice sheets are assumed.

Storm Climatology

stormclim [bool, optional] True/False. Toggles whether or not storm climatology (convective available potential energy, maximum potential intensity, ventilation index, etc) should be computed. If True, output fields related to storm climatology will be added to standard output files. Enabling this mode currently roughly doubles the computational cost of the model. This may improve in future updates. Refer to Paradise, et al 2021 for implementation description.

stormcapture [dict, optional] A dictionary containing arguments controlling when high-cadence output is triggered by storm activity. This dictionary must contain 'toggle', which can be either 1 or 0 (yes or no). It may also contain any namelist parameters accepted by hurricanemod.f90, including the following:

toggle [{0,1}] Whether (1) or not (0) to write high-cadence output when storms occur

NKTRIGGER [{0,1}, optional] (0/1=no/yes). Whether or not to use the Komacek, et al 2020 conditions for hurricane cyclogenesis as the output trigger. Default is no.

VITHRESH [float, optional] (nktrigger) Ventilation index threshold for nktrigger output. Default 0.145

VMXTHRESH [float, optional] (nktrigger) Max potential intensity threshold for nktrigger output.Default 33 m/s

LAVTHRESH [float, optional] (nktrigger) Lower-atmosphere vorticity threshold for nktrigger output. Default 1.2*10^-5 s^-1

VRMTHRESH [float, optional] (unused) Ventilation-reduced maximum intensity threshold. Default 0.577

GPITHRESH [float, optional] (default) Genesis Potential Index threshold. Default 0.37.

MINSURFTEMP [float, optional] (default) Min. surface temperature for storm activity. Default 25C

MAXSURFTEMP [float, optional] (default) Max. surface temperature for storm activity. Default 100C

WINDTHRESH [float, optional] (default) Lower-atmosphere maximum wind threshold for storm activity. Default 33 m/s

SWINDTHRESH [float, optional] (default) Minimum surface windspeed for storm activity. Default 20.5 m/s

SIZETHRESH [float, optional] (default) Minimum number of cells that must trigger to start outputDefault 30

ENDTHRESH [float, optional] (default) Minimum number of cells at which point storm output ends.Default 16

MINSTORMLEN [float, optional] (default) Minimum number of timesteps to write output. Default 256

MAXSTORMLEN [float, optional] (default) Maximum number of timesteps to write output. Default 1024

Note that actual number of writes will be stormlen/interval, as set in highcadence. This interval defaults to 4, so 64 writes minimum, 256 max. For more details on the storm climatology factors considered here, see [5]_.

Notes

In some cases, it may be necessary to include physics filters. This typically becomes necessary when sharp features are projected on the model's smallest spectral modes, causing Gibbs "ripples". Earth-like models typically do not require filtering, but tidally-locked models do. Filtering may be beneficial for Earth-like models at very high resolutions as well, or if there is sharp topography.

Three filter functional forms are included in ExoPlaSim: Cesaro, exponential, and Lander-Hoskins. Their functional forms are given below, where *n* is the wavenumber, and *N* is the truncation wavenumber (e.g. 21 for T21):

Cesaro: $f(n) = 1 - \frac{n}{N+1}$ [2]_

Exponential: $f(n) = \exp\left[-\kappa\left(\frac{n}{N}\right)^{\gamma}\right]$ [3]_

Lander-Hoskins: $f(n) = \exp\left[-\left(\frac{n(n+1)}{n_0(n_0+1}\right)^2\right]$ [3]_ [4]_

κ is exposed to the user through `filterkappa`, γ is exposed through `filterpower`, and n_0 is exposed through `filterLHN0`.

Physics filters can be applied at two different points; either at the transform from gridpoint to spectral, or the reverse. We find that in most cases, the ideal usage is to use both. Generally, a filter at the gridpoint->spectral transform is good for dealing with oscillations caused by sharp jumps and small features in the gridpoint tendencies. Conversely, a filter at the spectral->gridpoint transform is good for dealing with oscillations that come from small-scale features in the spectral fields causing small-scale features to appear in the gridpoint tendencies [3]_. Since we deal with climate systems where everything is coupled, any oscillations not removed by one filter will be amplified through physical feedbacks if not suppressed by the other filter.

References

class exoplasim.**TLlandplanet**(*resolution='T21'*, *layers=10*, *ncpus=4*, *precision=8*, *debug=False*, *inityear=0*, *recompile=False*, *optimization=None*, *mars=False*, *workdir='most'*, *source=None*, *force991=False*, *modelname='MOST_EXP'*, *outputtype='.npz'*, *crashtolerant=False*)

Bases: *exoplasim.Model*

Create a tidally-locked model with no oceans.

Identical to *Model*, except configuration options suitable for tidally-locked models are the default when configure() is called, and the surface is entirely land-covered. Specifically, a 30-minute timestep, snapshot outputs every 720 timesteps, eccentricity=0.0, 0-degree obliquity, exponential physics filtering, fixed orbital parameters, and no ozone. All these defaults can be overridden.

Notes

The default is to include zero soil water initially. This will result in a completely dry model. Set soilsaturation to something nonzero if you want groundwater.

configure(*timestep=30.0*, *snapshots=720*, *eccentricity=0.0*, *ozone=False*, *obliquity=0.0*, *physicsfilter='gp|exp|sp'*, *tlcontrast=100.0*, ***kwargs*)

Configure the model's namelists and boundary conditions.

The defaults here are appropriate for an Earth model.

Model Operation

noutput [bool, optional] True/False. Whether or not model output should be written.

restartfile [str, optional] Path to a restart file to use for initial conditions. Can be None.

writefrequency [int, optional] How many times per day ExoPlaSim should write output. Ignored by default–default is to write time-averaged output once every 5 days.

timestep [float, optional] Model timestep. Defaults to 45 minutes.

runscript [function , optional] A Python function that accepts a Model object as its first argument. This is the routine that will be run when you issue the Model.run() command. Any keyword arguments passed to run() will be forwarded to the specified function. If not set, the default internal routine will be used.

snapshots [int, optional] How many timesteps should elapse between snapshot outputs. If not set, no snapshots will be written.

restartfile [string, optional] Path to a restart file to use.

highcadence [dict, optional] A dictionary containing the following arguments:

- **'toggle'** [{0,1}] Whether or not high-cadence output should be written (1=yes).
- **'start'** [int] Timestep at which high-cadence output should begin.
- **'end'** [int] Timestep at which high-cadence output should end.
- **'interval'** [int] How many timesteps should elapse between high-cadence outputs.

threshold [float, optional] Energy balance threshold model should run to, if using *runtobalance()*. Default is <0.05 W/m^2/yr average drift in TOA and surface energy balance over 45-year timescales.

resources [list, optional] A list of paths to any additional files that should be available in the run directory.

runsteps [integer, optional] The number of timesteps to run each 'year'. By default, this is tuned to 360 Earth days. If set, this will override other controls setting the length of each modelled year.

otherargs [dict, optional] Any namelist parameters not included by default in the configuration options. These should be passed as a dictionary, with "PARAMETER@namelist" as the form of the dictionary key, and the parameter value passed as a string. e.g. `otherargs={"N_RUN_MONTHS@plasim_namelist":'4', "NGUI@plasim_namelist:'1'}`

Model Dynamics

columnmode [{None,"-","clear","static","static|clear","clear|static"}, optional] The inclusion of 'static' will disable horizontal advection, forcing ExoPlaSim into a column-only mode of operation. The inclusion of 'clear' will disable the radiative effects of clouds.

drycore [bool, optional] True/False. If True, evaporation is turned off, and a dry atmosphere will be used.

physicsfilter [str, optional] If not an empty string, specifies the physics filter(s) to be used. Filters can be used during the transform from gridpoint to spectral (`"gp"`), and/or during the transform from spectral to gridpoint (`"sp"`). Filter types are "none", "cesaro", "exp", or "lh" (see the Notes for more details). Combinations of filter types and times should be combined with a `|`, e.g. `physicsfilter="gp|exp|sp"` or `physicsfilter="gp|cesaro"`.

filterkappa [float, optional] A constant to be used with the exponential filter. Default is 8.0.

filterpower [int, optional] A constant integer to be used with the exponential filter. Default is 8.

filterLHN0 [float, optional] The constant used in the denominator of the Lander-Hoskins Filter. Default is 15; typically chosen so f(N)=0.1.

diffusionwaven [int, optional] The critical wavenumber beyond which hyperdiffusion is applied. Default is 15 for T21.

qdiffusion [float, optional] Timescale for humidity hyperdiffusion in days. Default for T21 is 0.1.

tdiffusion [float, optional] Timescale for temperature hyperdiffusion in days. Default for T21 is 5.6.

zdiffusion [float, optional] Timescale for vorticity hyperdiffusion in days. Default for T21 is 1.1.

ddiffusion [float, optional] Timescale for divergence hyperdiffusion in days.. Default for T21 is 0.2.

diffusionpower [int, optional] integer exponent used in hyperdiffusion. Default is 2 for T21.

Radiation

flux [float, optional] Incident stellar flux in W/m^2. Default 1367 for Earth.

startemp [float, optional] Effective blackbody temperature for the star. Not used if not set.

starradius [float, optional] Radius of the parent star in solar radii. Currently only used for the optional petitRADTRANS direct imaging postprocessor.

starspec [str, optional] Spectral file for the stellar spectrum. Should have two columns and 965 rows, with wavelength in the first column and radiance or intensity in the second. A similarly-named file with the "_hr.dat" suffix must also exist and have 2048 wavelengths. Appropriately-formatted files can be created with *makestellarspec.py*.

twobandalbedo [bool, optional] True/False. If True, separate albedos will be calculated for each of the two shortwave bands. If False (default), a single broadband albedo will be computed and used for both.

synchronous [bool, optional] True/False. If True, the Sun is fixed to one longitude in the sky.

desync [float, optional] The rate of drift of the substellar point in degrees per minute. May be positive or negative.

substellarlon [float, optional] The longitude of the substellar point, if synchronous==True. Default 180°

pressurebroaden [bool, optional] True/False. If False, pressure-broadening of absorbers no longer depends on surface pressure. Default is True

ozone [bool or dict, optional] True/False/dict. Whether or not forcing from stratospheric ozone should be included. If a dict is provided, it should contain the keys "height", "spread", "amount","varlat","varseason", and "seasonoffset", which correspond to the height in meters of peak O3 concentration, the width of the gaussian distribution in meters, the baseline column amount of ozone in cm-STP, the latitudinal amplitude, the magnitude of seasonal variation, and the time offset of the seasonal variation in fraction of a year. The three amounts are additive. To set a uniform, unvarying O3 distribution, ,place all the ozone in "amount", and set "varlat" and "varseason" to 0.

snowicealbedo [float, optional] A uniform albedo to use for all snow and ice.

soilalbedo [float, optional] A uniform albedo to use for all land.

wetsoil [bool, optional] True/False. If True, land albedo depends on soil moisture (wet=darker).

oceanalbedo [float, optional] A uniform albedo to use for the ocean.

oceanzenith [{"ECHAM-3","ECHAM-6","Lambertian}, optional] The zenith-angle dependence to use for blue-light reflectance from the ocean. Can be `'Lambertian'/'uniform'`, `'ECHAM-3'/'plasim'/'default'`, or `'ECHAM-6'`. The default is `'ECHAM-3'` (synonymous with `'plasim'` and `'default'`), which is the dependence used in the ECHAM-3 model.

Orbital Parameters

year [float, optional] Number of 24-hour days in a sidereal year. Not necessary if eccentricity and obliquity are zero. Defaults if not set to ~365.25 days

rotationperiod [float, optional] Planetary rotation period, in days. Default is 1.0.

eccentricity [float, optional] Orbital eccentricity. If not set, defaults to Earth's (0.016715)

obliquity [float, optional] Axial tilt, in degrees. If not set, defaults to Earth's obliquity (23.441°).

lonvernaleq [float, optional] Longitude of periapse, measured from vernal equinox, in degrees. If not set, defaults to Earth's (102.7°).

fixedorbit [bool, optional] True/False. If True, orbital parameters do not vary over time. If False, variations such as Milankovich cycles will be computed by PlaSim.

keplerian [bool, optional] True/False. If True, a generic Keplerian orbital calculation will be performed. This means no orbital precession, Milankovich cycles, etc, but does allow for accurate calculation of a wide diversity of orbits, including with higher eccentricity. Note that extreme orbits may have extreme results, including extreme crashes.

meananomaly0 [float, optional] The initial mean anomaly in degrees. Only used if *keplerian=True*.

Planet Parameters

gravity [float, optional] Surface gravity, in m/s^2. Defaults to 9.80665 m/s^2.

radius [float, optional] Planet radius in Earth radii. Default is 1.0.

orography [float, optional] If set, a scaling factor for topographic relief. If `orography=0.0`, topography will be zeroed-out.

aquaplanet [bool, optional] True/False. If True, the surface will be entirely ocean-covered.

desertplanet [bool, optional] True/False. If True, the surface will be entirely land-covered.

tlcontrast [float, optional] The initial surface temperature contrast between fixedlon and the anterior point. Default is 0.0 K.

seaice [bool, optional] True/False. If False, disables radiative effects of sea ice (although sea ice itself is still computed).

landmap [str, optional] Path to a `.sra` file containing a land mask for the chosen resolution.

topomap [str, optional] Path to a `.sra` file containing geopotential height map. Must include landmap.

Atmosphere

gascon [float, optional] Effective gas constant. Defaults to 287.0 (Earth), or the gas constant corresponding to the composition specified by partial pressures.

vtype [{0,1,2,3,4,5}, optional] Type of vertical discretization. Can be: 0 Pseudo-linear scaling with pressure that maintains resolution near the ground. 1 Linear scaling with pressure. 2 Logarithmic scaling with pressure (resolves high altitudes) 3 Pseudologarithmic scaling with pressure that preserves resolution near the ground. 4 Pseudolinear scaling with pressure, pinned to a specified top pressure. 5 If >10 layers, bottom 10 as if `vtype=4`, and upper layers as if `vtype=2`.

modeltop [float, optional] Pressure of the top layer

tropopause [float, optional] If stratosphere is being included, pressure of the 10th layer (where scheme switches from linear to logarithmic).

stratosphere [bool, optional] True/False. If True, vtype=5 is used, and model is discretized to include a stratosphere.

pressure: float, optional Surface pressure in bars, if not specified through partial pressures.

Gas Partial Pressures

Partial pressures of individual gases can be specified. If pressure and gascon are not explicitly set, these will determine surface pressure, mean molecular weight, and effective gas constant. Note however that Rayleigh scattering assumes an Earth-like composition, and the only absorbers explicitly included in the radiation scheme are CO2 and H2O.

pH2 [float, optional] H2 partial pressure in bars.

pHe [float, optional] He partial pressure in bars.

pN2 [float, optional] N2 partial pressure in bars.

pO2 [float, optional] O2 partial pressure in bars.

pH2 [float, optional] H2 partial pressure in bars.

pAr [float, optional] Ar partial pressure in bars.

pNe [float, optional] Ne partial pressure in bars.

pKr [float, optional] Kr partial pressure in bars.

pCO2 [float, optional] CO2 partial pressure in bars. This gets translated into a ppmv concentration, so if you want to specify/vary CO2 but don't need the other gases, specifying pCO2, pressure, and gascon will do the trick. In most use cases, however, just specifying pN2 and pCO2 will give good enough behavior.

pH2O [float, optional] H2O partial pressure in bars. This is only useful in setting the gas constant and surface pressure; it will have no effect on actual moist processes.

Surface Parameters

mldepth [float, optional] Depth of the mixed-layer ocean. Default is 50 meters.

soildepth [float, optional] Scaling factor for the depth of soil layers (default total of 12.4 meters)

cpsoil [float, optional] Heat capacity of the soil, in J/m^3/K. Default is 2.4*10^6.

soilwatercap [float, optional] Water capacity of the soil, in meters. Defaults to 0.5 meters

soilsaturation [float, optional] Initial fractional saturation of the soil. Default is 0.0 (dry).

maxsnow [float, optional] Maximum snow depth (Default is 5 meters; set to -1 to have no limit).

Additional Physics

Carbon-Silicate Weathering

co2weathering [bool, optional] True/False. Toggles whether or not carbon-silicate weathering should be computed. Default is False.

evolveco2 [bool, optional] True/False. If co2weathering==True, toggles whether or not the CO2 partial pressure should be updated every year. Usually the change in pCO2 will be extremely small, so this is not necessary, and weathering experiments try to estimate the average weathering rate for a given climate in order to interpolate timescales between climates, rather than modelling changes in CO2 over time directly.

outgassing [float, optional] The assumed CO2 outgassing rate in units of Earth outgassing. Default is 1.0.

erosionsupplylimit [float, optional] If set, the maximum CO2 weathering rate per year permitted by erosion, in ubars/year. This is not simply a hard cutoff, but follows Foley 2015 so high weathering below the cutoff is also reduced.

Vegetation

vegetation [bool or int, optional] Can be True/False, or 0/1/2. If True or 1, then diagnostic vegetation is turned on. If 2, then coupled vegetation is turned on. Vegetation is computed via the SimBA module.

vegaccel [int, optional] Integer factor by which to accelerate vegetation growth

nforestgrowth: float, optional Biomass growth

initgrowth [float, optional] Initial above-ground growth

initstomcond [float, optional] Initial stomatal conductance

initrough [float, optional] Initial vegetative surface roughness

initsoilcarbon [float, optional] Initial soil carbon content

initplantcarbon [float, optional] Initial vegetative carbon content

See [1]_ for details on the implementation of supply-limited weathering.

Glaciology

glaciers [dict, optional] A dictionary containing the following arguments: toggle : bool

True/False. Whether or not glaciers should be allowed to grow or shrink in thickness, or be formed from persistent snow on land.

mindepth [float] The minimum snow depth in meters of liquid water equivalent that must persist year-round before the grid cell is considered glaciated. Default is 2 meters.

initialh [float] If >=0, covers the land surface with ice sheets of a height given in meterss. If -1, no initial ice sheets are assumed.

Storm Climatology

stormclim [bool, optional] True/False. Toggles whether or not storm climatology (convective available potential energy, maximum potential intensity, ventilation index, etc) should be computed. If True, output fields related to storm climatology will be added to standard output files. Enabling this mode currently roughly doubles the computational cost of the model. This may improve in future updates. Refer to Paradise, et al 2021 for implementation description.

stormcapture [dict, optional] A dictionary containing arguments controlling when high-cadence output is triggered by storm activity. This dictionary must contain 'toggle', which can be either 1 or 0 (yes or no). It may also contain any namelist parameters accepted by hurricanemod.f90, including the following:

toggle [{0,1}] Whether (1) or not (0) to write high-cadence output when storms occur

NKTRIGGER [{0,1}, optional] (0/1=no/yes). Whether or not to use the Komacek, et al 2020 conditions for hurricane cyclogenesis as the output trigger. Default is no.

VITHRESH [float, optional] (nktrigger) Ventilation index threshold for nktrigger output. Default 0.145

VMXTHRESH [float, optional] (nktrigger) Max potential intensity threshold for nktrigger output.Default 33 m/s

LAVTHRESH [float, optional] (nktrigger) Lower-atmosphere vorticity threshold for nktrigger output. Default 1.2*10^-5 s^-1

VRMTHRESH [float, optional] (unused) Ventilation-reduced maximum intensity threshold. Default 0.577

GPITHRESH [float, optional] (default) Genesis Potential Index threshold. Default 0.37.

MINSURFTEMP [float, optional] (default) Min. surface temperature for storm activity. Default 25C

MAXSURFTEMP [float, optional] (default) Max. surface temperature for storm activity. Default 100C

WINDTHRESH [float, optional] (default) Lower-atmosphere maximum wind threshold for storm activity. Default 33 m/s

SWINDTHRESH [float, optional] (default) Minimum surface windspeed for storm activity. Default 20.5 m/s

SIZETHRESH [float, optional] (default) Minimum number of cells that must trigger to start outputDefault 30

ENDTHRESH [float, optional] (default) Minimum number of cells at which point storm output ends.Default 16

MINSTORMLEN [float, optional] (default) Minimum number of timesteps to write output. Default 256

MAXSTORMLEN [float, optional] (default) Maximum number of timesteps to write output. Default 1024

Note that actual number of writes will be stormlen/interval, as set in highcadence. This interval defaults to 4, so 64 writes minimum, 256 max. For more details on the storm climatology factors considered here, see [5]_.

Notes

In some cases, it may be necessary to include physics filters. This typically becomes necessary when sharp features are projected on the model's smallest spectral modes, causing Gibbs "ripples". Earth-like models typically do not require filtering, but tidally-locked models do. Filtering may be beneficial for Earth-like models at very high resolutions as well, or if there is sharp topography.

Three filter functional forms are included in ExoPlaSim: Cesaro, exponential, and Lander-Hoskins. Their functional forms are given below, where *n* is the wavenumber, and *N* is the truncation wavenumber (e.g. 21 for T21):

Cesaro: $f(n) = 1 - \frac{n}{N+1}$ [2]_

Exponential: $f(n) = \exp\left[-\kappa\left(\frac{n}{N}\right)^{\gamma}\right]$ [3]_

Lander-Hoskins: $f(n) = \exp\left[-\left(\frac{n(n+1)}{n_0(n_0+1}\right)^2\right]$ [3]_ [4]_

κ is exposed to the user through `filterkappa`, γ is exposed through `filterpower`, and n_0 is exposed through `filterLHN0`.

Physics filters can be applied at two different points; either at the transform from gridpoint to spectral, or the reverse. We find that in most cases, the ideal usage is to use both. Generally, a filter at the gridpoint->spectral transform is good for dealing with oscillations caused by sharp jumps and small features in the gridpoint tendencies. Conversely, a filter at the spectral->gridpoint transform is good for dealing with oscillations that come from small-scale features in the spectral fields causing small-scale features to appear in the gridpoint tendencies [3]_. Since we deal with climate systems where everything is coupled, any oscillations not removed by one filter will be amplified through physical feedbacks if not suppressed by the other filter.

References

class exoplasim.**TLmodel**(*resolution='T21'*, *layers=10*, *ncpus=4*, *precision=8*, *debug=False*, *inityear=0*, *recompile=False*, *optimization=None*, *mars=False*, *workdir='most'*, *source=None*, *force991=False*, *modelname='MOST_EXP'*, *outputtype='.npz'*, *crashtolerant=False*)

Bases: *exoplasim.Model*

Create a tidally-locked model.

Identical to *Model*, except configuration options suitable for tidally-locked models are the default when configure() is called.

configure(*timestep=30.0*, *snapshots=720*, *eccentricity=0.0*, *ozone=False*, *obliquity=0.0*, *physicsfilter='gp|exp|sp'*, *tlcontrast=100.0*, ***kwargs*)

Configure the model's namelists and boundary conditions.

The defaults here are appropriate for an Earth model.

Model Operation

noutput [bool, optional] True/False. Whether or not model output should be written.

restartfile [str, optional] Path to a restart file to use for initial conditions. Can be None.

writefrequency [int, optional] How many times per day ExoPlaSim should write output. Ignored by default–default is to write time-averaged output once every 5 days.

timestep [float, optional] Model timestep. Defaults to 45 minutes.

runscript [function , optional] A Python function that accepts a Model object as its first argument. This is the routine that will be run when you issue the Model.run() command. Any keyword arguments passed to run() will be forwarded to the specified function. If not set, the default internal routine will be used.

snapshots [int, optional] How many timesteps should elapse between snapshot outputs. If not set, no snapshots will be written.

restartfile [string, optional] Path to a restart file to use.

highcadence [dict, optional] A dictionary containing the following arguments:

`'toggle'` [{0,1}] Whether or not high-cadence output should be written (1=yes).

`'start'` [int] Timestep at which high-cadence output should begin.

`'end'` [int] Timestep at which high-cadence output should end.

`'interval'` [int] How many timesteps should elapse between high-cadence outputs.

threshold [float, optional] Energy balance threshold model should run to, if using *`runtobalance()`*. Default is <0.05 W/m^2/yr average drift in TOA and surface energy balance over 45-year timescales.

resources [list, optional] A list of paths to any additional files that should be available in the run directory.

runsteps [integer, optional] The number of timesteps to run each 'year'. By default, this is tuned to 360 Earth days. If set, this will override other controls setting the length of each modelled year.

otherargs [dict, optional] Any namelist parameters not included by default in the configuration options. These should be passed as a dictionary, with "PARAMETER@namelist" as the form of the dictionary key, and the parameter value passed as a string. e.g. `otherargs={"N_RUN_MONTHS@plasim_namelist":'4', "NGUI@plasim_namelist:'1'}`

Model Dynamics

columnmode [{None,"-","clear","static","static|clear","clear|static"}, optional] The inclusion of 'static' will disable horizontal advection, forcing ExoPlaSim into a column-only mode of operation. The inclusion of 'clear' will disable the radiative effects of clouds.

drycore [bool, optional] True/False. If True, evaporation is turned off, and a dry atmosphere will be used.

physicsfilter [str, optional] If not an empty string, specifies the physics filter(s) to be used. Filters can be used during the transform from gridpoint to spectral (`"gp"`), and/or during the transform from spectral to gridpoint (`"sp"`). Filter types are "none", "cesaro", "exp", or "lh" (see the Notes for more details). Combinations of filter types and times should be combined with a `|`, e.g. `physicsfilter="gp|exp|sp"` or `physicsfilter="gp|cesaro"`.

filterkappa [float, optional] A constant to be used with the exponential filter. Default is 8.0.

filterpower [int, optional] A constant integer to be used with the exponential filter. Default is 8.

filterLHN0 [float, optional] The constant used in the denominator of the Lander-Hoskins Filter. Default is 15; typically chosen so f(N)=0.1.

diffusionwaven [int, optional] The critical wavenumber beyond which hyperdiffusion is applied. Default is 15 for T21.

qdiffusion [float, optional] Timescale for humidity hyperdiffusion in days. Default for T21 is 0.1.

tdiffusion [float, optional] Timescale for temperature hyperdiffusion in days. Default for T21 is 5.6.

zdiffusion [float, optional] Timescale for vorticity hyperdiffusion in days. Default for T21 is 1.1.

ddiffusion [float, optional] Timescale for divergence hyperdiffusion in days.. Default for T21 is 0.2.

diffusionpower [int, optional] integer exponent used in hyperdiffusion. Default is 2 for T21.

Radiation

flux [float, optional] Incident stellar flux in W/m^2. Default 1367 for Earth.

startemp [float, optional] Effective blackbody temperature for the star. Not used if not set.

starradius [float, optional] Radius of the parent star in solar radii. Currently only used for the optional petitRADTRANS direct imaging postprocessor.

starspec [str, optional] Spectral file for the stellar spectrum. Should have two columns and 965 rows, with wavelength in the first column and radiance or intensity in the second. A similarly-named file with the "_hr.dat" suffix must also exist and have 2048 wavelengths. Appropriately-formatted files can be created with *makestellarspec.py*.

twobandalbedo [bool, optional] True/False. If True, separate albedos will be calculated for each of the two shortwave bands. If False (default), a single broadband albedo will be computed and used for both.

synchronous [bool, optional] True/False. If True, the Sun is fixed to one longitude in the sky.

desync [float, optional] The rate of drift of the substellar point in degrees per minute. May be positive or negative.

substellarlon [float, optional] The longitude of the substellar point, if synchronous==True. Default 180°

pressurebroaden [bool, optional] True/False. If False, pressure-broadening of absorbers no longer depends on surface pressure. Default is True

ozone [bool or dict, optional] True/False/dict. Whether or not forcing from stratospheric ozone should be included. If a dict is provided, it should contain the keys "height", "spread", "amount","varlat","varseason", and "seasonoffset", which correspond to the height in meters of peak O3 concentration, the width of the

gaussian distribution in meters, the baseline column amount of ozone in cm-STP, the latitudinal amplitude, the magnitude of seasonal variation, and the time offset of the seasonal variation in fraction of a year. The three amounts are additive. To set a uniform, unvarying O3 distribution, ,place all the ozone in "amount", and set "varlat" and "varseason" to 0.

snowicealbedo [float, optional] A uniform albedo to use for all snow and ice.

soilalbedo [float, optional] A uniform albedo to use for all land.

wetsoil [bool, optional] True/False. If True, land albedo depends on soil moisture (wet=darker).

oceanalbedo [float, optional] A uniform albedo to use for the ocean.

oceanzenith [{"ECHAM-3","ECHAM-6","Lambertian}, optional] The zenith-angle dependence to use for blue-light reflectance from the ocean. Can be `'Lambertian'/'uniform'`, `'ECHAM-3'/'plasim'/'default'`, or `'ECHAM-6'`. The default is `'ECHAM-3'` (synonymous with `'plasim'` and `'default'`), which is the dependence used in the ECHAM-3 model.

Orbital Parameters

year [float, optional] Number of 24-hour days in a sidereal year. Not necessary if eccentricity and obliquity are zero. Defaults if not set to ~365.25 days

rotationperiod [float, optional] Planetary rotation period, in days. Default is 1.0.

eccentricity [float, optional] Orbital eccentricity. If not set, defaults to Earth's (0.016715)

obliquity [float, optional] Axial tilt, in degrees. If not set, defaults to Earth's obliquity (23.441°).

lonvernaleq [float, optional] Longitude of periapse, measured from vernal equinox, in degrees. If not set, defaults to Earth's (102.7°).

fixedorbit [bool, optional] True/False. If True, orbital parameters do not vary over time. If False, variations such as Milankovich cycles will be computed by PlaSim.

keplerian [bool, optional] True/False. If True, a generic Keplerian orbital calculation will be performed. This means no orbital precession, Milankovich cycles, etc, but does allow for accurate calculation of a wide diversity of orbits, including with higher eccentricity. Note that extreme orbits may have extreme results, including extreme crashes.

meananomaly0 [float, optional] The initial mean anomaly in degrees. Only used if *keplerian=True*.

Planet Parameters

gravity [float, optional] Surface gravity, in m/s^2. Defaults to 9.80665 m/s^2.

radius [float, optional] Planet radius in Earth radii. Default is 1.0.

orography [float, optional] If set, a scaling factor for topographic relief. If `orography=0.0`, topography will be zeroed-out.

aquaplanet [bool, optional] True/False. If True, the surface will be entirely ocean-covered.

desertplanet [bool, optional] True/False. If True, the surface will be entirely land-covered.

tlcontrast [float, optional] The initial surface temperature contrast between fixedlon and the anterior point. Default is 0.0 K.

seaice [bool, optional] True/False. If False, disables radiative effects of sea ice (although sea ice itself is still computed).

landmap [str, optional] Path to a `.sra` file containing a land mask for the chosen resolution.

topomap [str, optional] Path to a `.sra` file containing geopotential height map. Must include landmap.

Atmosphere

gascon [float, optional] Effective gas constant. Defaults to 287.0 (Earth), or the gas constant corresponding to the composition specified by partial pressures.

vtype [{0,1,2,3,4,5}, optional] Type of vertical discretization. Can be: 0 Pseudolinear scaling with pressure that maintains resolution near the ground. 1 Linear scaling with pressure. 2 Logarithmic scaling with pressure (resolves high altitudes) 3 Pseudologarithmic scaling with pressure that preserves resolution near the ground. 4 Pseudolinear scaling with pressure, pinned to a specified top pressure. 5 If >10 layers, bottom 10 as if `vtype=4`, and upper layers as if `vtype=2`.

modeltop [float, optional] Pressure of the top layer

tropopause [float, optional] If stratosphere is being included, pressure of the 10th layer (where scheme switches from linear to logarithmic).

stratosphere [bool, optional] True/False. If True, vtype=5 is used, and model is discretized to include a stratosphere.

pressure: float, optional Surface pressure in bars, if not specified through partial pressures.

Gas Partial Pressures

Partial pressures of individual gases can be specified. If pressure and gascon are not explicitly set, these will determine surface pressure, mean molecular weight, and effective gas constant. Note however that Rayleigh scattering assumes an Earth-like composition, and the only absorbers explicitly included in the radiation scheme are CO2 and H2O.

pH2 [float, optional] H2 partial pressure in bars.

pHe [float, optional] He partial pressure in bars.

pN2 [float, optional] N2 partial pressure in bars.

pO2 [float, optional] O2 partial pressure in bars.

pH2 [float, optional] H2 partial pressure in bars.

pAr [float, optional] Ar partial pressure in bars.

pNe [float, optional] Ne partial pressure in bars.

pKr [float, optional] Kr partial pressure in bars.

pCO2 [float, optional] CO2 partial pressure in bars. This gets translated into a ppmv concentration, so if you want to specify/vary CO2 but don't need the other gases, specifying pCO2, pressure, and gascon will do the trick. In most use cases, however, just specifying pN2 and pCO2 will give good enough behavior.

pH2O [float, optional] H2O partial pressure in bars. This is only useful in setting the gas constant and surface pressure; it will have no effect on actual moist processes.

Surface Parameters

mldepth [float, optional] Depth of the mixed-layer ocean. Default is 50 meters.

soildepth [float, optional] Scaling factor for the depth of soil layers (default total of 12.4 meters)

cpsoil [float, optional] Heat capacity of the soil, in J/m^3/K. Default is 2.4*10^6.

soilwatercap [float, optional] Water capacity of the soil, in meters. Defaults to 0.5 meters

soilsaturation [float, optional] Initial fractional saturation of the soil. Default is 0.0 (dry).

maxsnow [float, optional] Maximum snow depth (Default is 5 meters; set to -1 to have no limit).

Additional Physics

Carbon-Silicate Weathering

co2weathering [bool, optional] True/False. Toggles whether or not carbon-silicate weathering should be computed. Default is False.

evolveco2 [bool, optional] True/False. If co2weathering==True, toggles whether or not the CO2 partial pressure should be updated every year. Usually the change in pCO2 will be extremely small, so this is not necessary, and weathering experiments try to estimate the average weathering rate for a given climate in order to interpolate timescales between climates, rather than modelling changes in CO2 over time directly.

outgassing [float, optional] The assumed CO2 outgassing rate in units of Earth outgassing. Default is 1.0.

erosionsupplylimit [float, optional] If set, the maximum CO2 weathering rate per year permitted by erosion, in ubars/year. This is not simply a hard cutoff, but follows Foley 2015 so high weathering below the cutoff is also reduced.

Vegetation

vegetation [bool or int, optional] Can be True/False, or 0/1/2. If True or 1, then diagnostic vegetation is turned on. If 2, then coupled vegetation is turned on. Vegetation is computed via the SimBA module.

vegaccel [int, optional] Integer factor by which to accelerate vegetation growth

nforestgrowth: float, optional Biomass growth

initgrowth [float, optional] Initial above-ground growth

initstomcond [float, optional] Initial stomatal conductance

initrough [float, optional] Initial vegetative surface roughness

initsoilcarbon [float, optional] Initial soil carbon content

initplantcarbon [float, optional] Initial vegetative carbon content

See [1]_ for details on the implementation of supply-limited weathering.

Glaciology

glaciers [dict, optional] A dictionary containing the following arguments: toggle : bool

True/False. Whether or not glaciers should be allowed to grow or shrink in thickness, or be formed from persistent snow on land.

mindepth [float] The minimum snow depth in meters of liquid water equivalent that must persist year-round before the grid cell is considered glaciated. Default is 2 meters.

initialh [float] If >=0, covers the land surface with ice sheets of a height given in meterss. If -1, no initial ice sheets are assumed.

Storm Climatology

stormclim [bool, optional] True/False. Toggles whether or not storm climatology (convective available potential energy, maximum potential intensity, ventilation index, etc) should be computed. If True, output fields related to storm climatology will be added to standard output files. Enabling this mode currently roughly doubles the computational cost of the model. This may improve in future updates. Refer to Paradise, et al 2021 for implementation description.

stormcapture [dict, optional] A dictionary containing arguments controlling when high-cadence output is triggered by storm activity. This dictionary must contain 'toggle', which can be either 1 or 0 (yes or no). It may also contain any namelist parameters accepted by hurricanemod.f90, including the following:

toggle [{0,1}] Whether (1) or not (0) to write high-cadence output when storms occur

NKTRIGGER [{0,1}, optional] (0/1=no/yes). Whether or not to use the Komacek, et al 2020 conditions for hurricane cyclogenesis as the output trigger. Default is no.

VITHRESH [float, optional] (nktrigger) Ventilation index threshold for nktrigger output. Default 0.145

VMXTHRESH [float, optional] (nktrigger) Max potential intensity threshold for nktrigger output.Default 33 m/s

LAVTHRESH [float, optional] (nktrigger) Lower-atmosphere vorticity threshold for nktrigger output. Default 1.2*10^-5 s^-1

VRMTHRESH [float, optional] (unused) Ventilation-reduced maximum intensity threshold. Default 0.577

GPITHRESH [float, optional] (default) Genesis Potential Index threshold. Default 0.37.

MINSURFTEMP [float, optional] (default) Min. surface temperature for storm activity. Default 25C

MAXSURFTEMP [float, optional] (default) Max. surface temperature for storm activity. Default 100C

WINDTHRESH [float, optional] (default) Lower-atmosphere maximum wind threshold for storm activity. Default 33 m/s

SWINDTHRESH [float, optional] (default) Minimum surface windspeed for storm activity. Default 20.5 m/s

SIZETHRESH [float, optional] (default) Minimum number of cells that must trigger to start outputDefault 30

ENDTHRESH [float, optional] (default) Minimum number of cells at which point storm output ends.Default 16

MINSTORMLEN [float, optional] (default) Minimum number of timesteps to write output. Default 256

MAXSTORMLEN [float, optional] (default) Maximum number of timesteps to write output. Default 1024

Note that actual number of writes will be stormlen/interval, as set in highcadence. This interval defaults to 4, so 64 writes minimum, 256 max. For more details on the storm climatology factors considered here, see [5]_.

Notes

In some cases, it may be necessary to include physics filters. This typically becomes necessary when sharp features are projected on the model's smallest spectral modes, causing Gibbs "ripples". Earth-like models typically do not require filtering, but tidally-locked models do. Filtering may be beneficial for Earth-like models at very high resolutions as well, or if there is sharp topography.

Three filter functional forms are included in ExoPlaSim: Cesaro, exponential, and Lander-Hoskins. Their functional forms are given below, where *n* is the wavenumber, and *N* is the truncation wavenumber (e.g. 21 for T21):

Cesaro: $f(n) = 1 - \frac{n}{N+1}$ [2]_

Exponential: $f(n) = \exp\left[-\kappa\left(\frac{n}{N}\right)^{\gamma}\right]$ [3]_

Lander-Hoskins: $f(n) = \exp\left[-\left(\frac{n(n+1)}{n_0(n_0+1}\right)^2\right]$ [3]_ [4]_

κ is exposed to the user through `filterkappa`, γ is exposed through `filterpower`, and n_0 is exposed through `filterLHN0`.

Physics filters can be applied at two different points; either at the transform from gridpoint to spectral, or the reverse. We find that in most cases, the ideal usage is to use both. Generally, a filter at the gridpoint->spectral transform is good for dealing with oscillations caused by sharp jumps and small features in the gridpoint tendencies. Conversely, a filter at the spectral->gridpoint transform is good for dealing with oscillations that come from small-scale features in the spectral fields causing small-scale features to appear in the gridpoint tendencies [3]_. Since we deal with climate systems where everything is coupled, any oscillations not removed by one filter will be amplified through physical feedbacks if not suppressed by the other filter.

References

1.3.2 Submodules

1.3.3 exoplasim.gcmt module

exception exoplasim.gcmt.**DatafileError**
Bases: Exception

exception exoplasim.gcmt.**DimensionError**
Bases: Exception

exception exoplasim.gcmt.**UnitError**
Bases: Exception

exoplasim.gcmt.**adist**(*lon1*, *lat1*, *lon2*, *lat2*)
Return angular distance(s) in degrees between two points (or sets of points) on a sphere.

Parameters

- **lon1** (*float or numpy.ndarray*) – Longitudes of first point(s)
- **lat1** (*float or numpy.ndarray*) – Latitudes of first point(s)
- **lon2** (*float or numpy.ndarray*) – Longitudes of second point(s)
- **lat2** (*float or numpy.ndarray*) – Latitudes of second point(s)

Returns Angular distance(s) between given points

Return type float or numpy.ndarray

exoplasim.gcmt.**blackbody**(*wavelengths*, *temperature*)
Compute the Planck function for a set of wavelengths and a given effective temperature.

Parameters

- **wavelengths** (*array-like*) – Wavelengths in microns
- **temperature** (*float*) – Effective temperature in Kelvins

Returns Spectral radiance F_lambda(lambda,T) for the provided wavelengths assuming a perfect blackbody.

Return type array-like

exoplasim.gcmt.**cspatialmath**(*variable*, *lat=None*, *lon=None*, *file=None*, *mean=True*, *time=None*, *ignoreNaNs=True*, *lev=None*, *radius=6371000.0*, *poles=False*)
Compute spatial means or sums of data, but optionally don't go all the way to the poles.

Sometimes, saying that the latitudes covered go all the way to $\pm 90^\circ$ results in errors, and accurate accounting requires excluding the poles themselves. This function is identical to spatialmath, except that it provides that option.

Parameters

- **variable** (*str, numpy.ndarray*) – The variable to operate on. Can either be a data array, or the name of a variable. If the latter, file must be specified.
- **lat** (*numpy.ndarray, optional*) – Latitude and longitude arrays. If file is provided and lat and lon are not, they will be extracted from the file.
- **lon** (*numpy.ndarray, optional*) – Latitude and longitude arrays. If file is provided and lat and lon are not, they will be extracted from the file.
- **file** (*str, optional*) – Path to a NetCDF output file to open and extract data from.
- **mean** (*bool, optional*) – If True, compute a global mean. If False, compute a global sum.
- **time** (*int, optional*) – The time index on which to slice. If unspecified, a time average will be returned.
- **ignoreNaNs** (*bool, optional*) – If True, use NaN-safe numpy operators.
- **lev** (*int, optional*) – If set, slice a 3D spatial array at the specified level.
- **radius** (*float, optional*) – Radius of the planet in meters. Only used if mean=False.

- **poles** (*bool, optional*) – If False (default), exclude the poles.

Returns

Return type float

exoplasim.gcmt.**eq2tl**(*variable*, *lon*, *lat*, *substellar=0.0*, *polemethod='interp'*)
Transform a variable to tidally-locked coordinates

Note that in our tidally-locked coordinate system, 0 degrees longitude is the substellar-south pole-antistellar meridian, and 90 degrees latitude is the substellar point, such that the evening hemisphere is 0-180 degrees longitude, the morning hemisphere is 180-360 degrees longitude, the north equatorial pole is at (0, 180), and easterly flow is counter-clockwise. Note that this differs from the coordinate system introduced in Koll & Abbot (2015) in that theirs is a left-handed coordinate system, with the south pole at (0, 180) and counter-clockwise easterly flow, which represents a south-facing observer inside the sphere, while ours is a right-handed coordinate system, representing a south-facing observer outside the sphere, which is the usual convention for spherical coordinate systems.

Parameters

- **variable** (*numpy.ndarray (2D, 3D, or 4D)*) – N-D data array to be transformed. Final two dimensions must be (lat,lon)
- **lon** (*numpy.ndarray*) – 1D array of longitudes [deg]
- **lat** (*numpy.ndarray*) – 1D array of latitudes [deg]
- **substellar** (*float, optional*) – Longitude of the substellar point (defaults to 0 degrees)
- **polemethod** (*str, optional*) – Interpolation method for polar latitudes. If "nearest", then instead of inverse-distance linear interpolation, will use nearest-neighbor. This is recommended for vector variables. For scalars, leave as "interp".

Returns Transformed longitudes, latitudes, and data array.

Return type numpy.ndarray, numpy.ndarray, numpy.ndarray

exoplasim.gcmt.**eq2tl_coords**(*lon*, *lat*, *substellar=0.0*)
Compute tidally-locked coordinates of a set of equatorial lat-lon coordinates.

Transforms equatorial coordinates into a tidally-locked coordinate system where 0 degrees longitude is the substellar-south pole-antistellar meridian, and 90 degrees latitude is the substellar point, such that the evening hemisphere is 0-180 degrees longitude, the morning hemisphere is 180-360 degrees longitude, the north equatorial pole is at (0, 180), and easterly flow is counter-clockwise. Note that this differs from the coordinate system introduced in Koll & Abbot (2015) in that theirs is a left-handed coordinate system, with the south pole at (0, 180) and counter-clockwise easterly flow, which represents a south-facing observer inside the sphere, while ours is a right-handed coordinate system, representing a south-facing observer outside the sphere, which is the usual convention for spherical coordinate systems.

Parameters

- **lon** (*numpy.ndarray*) – Longitudes in equatorial coordinates [degrees]
- **lat** (*numpy.ndarray*) – Latitudes in equatorial coordinates [degrees]
- **substellar** (*float, optional*) – Longitude of the substellar point. [degrees]

Returns Transformed longitudes and latitudes [degrees]

Return type numpy.ndarray, numpy.ndarray

exoplasim.gcmt.**eq2tl_uv**(*u*, *v*, *lon*, *lat*, *substellar=0.0*)
Transform velocity variables to tidally-locked coordinates

Parameters

- **u** (*numpy.ndarray (2D, 3D, or 4D)*) – N-D data array of zonal velocities to be transformed. Final two dimensions must be (lat,lon)
- **v** (*numpy.ndarray (2D, 3D, or 4D)*) – N-D data array of meridional velocities to be transformed. Final two dimensions must be (lat,lon)
- **lon** (*numpy.ndarray*) – 1D array of longitudes [deg]
- **lat** (*numpy.ndarray*) – 1D array of latitudes [deg]
- **substellar** (*float, optional*) – Longitude of the substellar point (defaults to 0 degrees)

Returns Transformed longitudes, latitudes, and velocity data arrays.

Return type numpy.ndarray, numpy.ndarray, numpy.ndarray, numpy.ndarray

exoplasim.gcmt.**eqstream**(*file*, *radius=6371000.0*, *gravity=9.80665*)

Compute the tidally-locked streamfunction

Parameters

- **dataset** (*str or ExoPlaSim Dataset*) – Either path to ExoPlaSim Dataset of model output or an instance of the dataset.
- **plarad** (*float, optional*) – Planetary radius [m]
- **grav** (*float, optional*) – Surface gravity [m/s^2]

Returns tidally-locked latitude, layer interface pressures, and TL streamfunction

Return type numpy.ndarray(1D), numpy.ndarray(1D), numpy.ndarray(2D)

exoplasim.gcmt.**latmean**(*variable*, *latitudes*)

Compute meriodional mean (i.e. the variable that changes is latitude).

Compute the area-weighted mean of a latitude array x, such that:

$$\bar{x} = \frac{\sum_{i=1}^{N} |\sin(\phi_{i-1/2}) - \sin(\phi_{i+1/2})| x_i}{\sum_{i=1}^{N} |\sin(\phi_{i-1/2}) - \sin(\phi_{i+1/2})|}$$

Parameters

- **variable** (*numpy.ndarray*) – Array to be averaged. Assumption is that if 2D, lat is the first dimension, if 3D, the second dimension, and if 4D. the 3rd dimension.
- **latitudes** (*array-like*) – Array or list of latitudes

Returns Depending on the dimensionality of the input array, output may have 0, 1, or 2 dimensions.

Return type scalar or numpy.ndarray

exoplasim.gcmt.**latsum**(*variable*, *latitudes*, *dlon=360.0*, *radius=6371000.0*)

Compute meriodional sum (i.e. the variable that changes is latitude).

Compute the area-weighted sum of a latitude array x given a longitude span $\Delta\theta$ and planet radius R, such that:

$$X = \sum_{i=1}^{N} |\sin(\phi_{i-1/2}) - \sin(\phi_{i+1/2})| \Delta\theta R^2 x_i$$

Parameters

- **variable** (*numpy.ndarray*) – Array to be summed. Assumption is that if 2D, lat is the first dimension, if 3D, the second dimension, and if 4D. the 3rd dimension.
- **latitudes** (*array-like*) – Array or list of latitudes
- **dlon** (*float, optional*) – Longitude span in degrees.
- **radius** (*float, optional*) – Planet radius in meters.

Returns Depending on the dimensionality of the input array, output may have 0, 1, or 2 dimensions.

Return type scalar or numpy.ndarray

exoplasim.gcmt.**load**(*filename*, *csvbuffersize=1*)

Open a postprocessed ExoPlaSim output file.

Supported formats include netCDF, CSV/TXT (can be compressed), NumPy, and HDF5. If the data archive is a group of files that are not tarballed, such as a directory of CSV/TXT or gzipped files, then the filename should be the name of the directory with the final file extension.

For example, if the dataset is a group of CSV files in a folder called "MOST_output.002", then *filename* ought to be "MOST_output.002.csv", even though no such file exists.

When accessing a file archive comprised of CSV/TXT files such as that described above, only part of the archive will be extracted/read into memory at once, with the exception of the first read, when the entire archive is extracted to read header information. Dimensional arrays, such as latitude, longitude, etc will be ready into memory and stored as attributes of the returned object (but are accessed with the usual dictionary pattern). Other data arrays however may need to be extracted and read from the archive. A memory buffer exists to hold recently-accessed arrays in memory, which will prioritize the most recently-accessed variables. The number of variables that can be stored in memory can be set with the *csvbuffersize* keyword. The default is 1. This means that the first time the variable is accessed, access times will be roughly the time it takes to extract the file and read it into memory. Subsequent accesses, however, will use RAM speeds. Once the variable has left the buffer, due to other variables being accessed, the next access will return to file access speeds. This behavior is intended to mimic the npz, netcdf, and hdf5 protocols.

Parameters

- **filename** (*str*) – Path to the file
- **csvbuffersize** (*int, optional*) – If the file (or group of files) is a file archive such as a directory, tarball, etc, this is the number of variables to keep in a memory buffer when the archive is accessed.

Returns gmct._Dataset object that can be queried like a netCDF file.

Return type object

exoplasim.gcmt.**lonmean**(*variable*, *longitudes*)

Compute zonal mean (i.e. the variable that changes is longitude).

Compute the area-weighted mean of a longitude array x, such that:

$$\bar{x} = \frac{\sum_{i=1}^{N} |\theta_{i-1/2} - \theta_{i+1/2}| x_i}{\sum_{i=1}^{N} |\theta_{i-1/2} - \theta_{i+1/2}|}$$

Parameters **variable** (*numpy.ndarray*) – Array to be summed. Assumption is that longitude is always the last dimension.

Returns Depending on the dimensionality of the input array, output may be a scalar or have N-1 dimensions.

Return type scalar or numpy.ndarray

exoplasim.gcmt.**lonsum**(*variable*, *longitudes*, *dsinlat=2.0*, *radius=6371000.0*)

Compute zonal sum (i.e. the variable that changes is longitude).

Compute the area-weighted sum of a longitude array x given a latitude span $\Delta \sin \phi$ and planet radius R, such that:

$$X = \sum_{i=1}^{N} |\theta_{i-1/2} - \theta_{i+1/2}| \Delta \sin \phi R^2 x_i$$

Parameters

- **variable** (*numpy.ndarray*) – Array to be summed. Assumption is that longitude is always the last dimension.
- **longitudes** (*array-like*) – Array or list of longitudes
- **dsinlat** (*float, optional*) – The sine-latitude span for the longitude span considered. The default is 2, corresponding to -90 degrees to 90 degrees.
- **radius** (*float, optional*) – Planet radius in meters.

Returns Depending on the dimensionality of the input array, output may have 0, 1, or 2 dimensions.

Return type scalar or numpy.ndarray

exoplasim.gcmt.**make2d**(*variable*, *lat=None*, *lon=None*, *time=None*, *lev=None*, *ignoreNaNs=True*, *radius=6371000.0*, *latitudes=None*, *longitudes=None*)

Compress a variable in two dimensions by slicing or averaging.

Parameters

- **variable** (*numpy.ndarray*) – The variable to operate on
- **lat** (*int, str, optional*) – Either an index on which to slice, or either of "sum" or "mean", indicating what should be done along that axis.
- **lon** (*int, str, optional*) – Either an index on which to slice, or either of "sum" or "mean", indicating what should be done along that axis.
- **lev** (*int, str, optional*) – Either an index on which to slice, or either of "sum" or "mean", indicating what should be done along that axis.
- **time** (*int, optional*) – The time index on which to slice. If unspecified, a time average will be returned.
- **ignoreNaNs** (*bool, optional*) – If set, will use NaN-safe numpy operators.
- **radius** (*float, optional*) – Planet radius in meters (only used for summation)
- **latitudes** (*numpy.ndarray, optional*) – Latitude array–required if lat is "mean", or if either lat or lon is "sum"
- **longitudes** (*numpy.ndarray, optional*) – Longitude array–required if lon is "mean" or if either lat or lon is "sum"

Returns A 2-D array

Return type numpy.ndarray

exoplasim.gcmt.**orthographic**(*lon*, *lat*, *imap*, *central_longitude=0*, *central_latitude=0*, *ny=200*, *nx=200*, *interp='bilinear'*)

Perform an orthographic projection.

Parameters

- **lon** (*numpy.ndarray (1D)*) – Longitude array [degrees]
- **lat** (*numpy.ndarray (1D)*) – Latitude array [degrees]
- **imap** (*numpy.ndarray*) – Data array to be projected. The first two dimensions should be (lat,lon)
- **central_longitude** (*float, optional*) – Longitude in degrees to be centered beneath the observer
- **central_latitude** (*float, optional*) – Latitude in degrees to be centered beneath the observer
- **ny** (*int, optional*) – Number of pixels in the Y direction for the output projection
- **nx** (*int, optional*) – Number of pixels in the X direction for the output projection
- **interp** (*str, optional*) – Interpolation to use. Currently only 'bilinear' is accepted; otherwise nearest-neighbor will be used.

Returns The projected output

Return type numpy.ndarray (ny,nx)

exoplasim.gcmt.**parse**(*file*, *variable*, *lat=None*, *lon=None*)

Retrieve a variable from a NetCDF file

Parameters

- **file** (*str*) – Path to a NetCDF file
- **variable** (*str*) – Name of the variable to extract
- **lat** (*str, optional*) – If the latitude and longitude arrays have non-standard names, specify them here.
- **lon** (*str, optional*) – If the latitude and longitude arrays have non-standard names, specify them here.

Returns Requested output field

Return type numpy.ndarray

exoplasim.gcmt.**spatialmath**(*variable*, *lat=None*, *lon=None*, *file=None*, *mean=True*, *time=None*, *ignoreNaNs=True*, *lev=None*, *radius=6371000.0*)

Compute spatial means or sums of data

Parameters

- **variable** (*str, numpy.ndarray*) – The variable to operate on. Can either be a data array, or the name of a variable. If the latter, file must be specified.
- **lat** (*numpy.ndarray, optional*) – Latitude and longitude arrays. If file is provided and lat and lon are not, they will be extracted from the file.
- **lon** (*numpy.ndarray, optional*) – Latitude and longitude arrays. If file is provided and lat and lon are not, they will be extracted from the file.
- **file** (*str, optional*) – Path to a NetCDF output file to open and extract data from.
- **mean** (*bool, optional*) – If True, compute a global mean. If False, compute a global sum.
- **time** (*int, optional, or "all"*) – The time index on which to slice. If unspecified, a time average will be returned. If set to "all", the time axis will be preserved.
- **ignoreNaNs** (*bool, optional*) – If True, use NaN-safe numpy operators.

- **lev** (*int, optional*) – If set, slice a 3D spatial array at the specified level.
- **radius** (*float, optional*) – Radius of the planet in meters. Only used if mean=False.

Returns

Return type float

exoplasim.gcmt.**streamfxn**(*file*, *time=None*)
Deprecated. Passes args to eqstream().

exoplasim.gcmt.**tl2eq**(*variable*, *lon*, *lat*, *substellar=0.0*)
Transform a tidally-locked variable to standard equatorial coordinates

Note that in our tidally-locked coordinate system, 0 degrees longitude is the substellar-south pole-antistellar meridian, and 90 degrees latitude is the substellar point, such that the evening hemisphere is 0-180 degrees longitude, the morning hemisphere is 180-360 degrees longitude, the north equatorial pole is at (0, 180), and easterly flow is counter-clockwise. Note that this differs from the coordinate system introduced in Koll & Abbot (2015) in that theirs is a left-handed coordinate system, with the south pole at (0, 180) and counter-clockwise easterly flow, which represents a south-facing observer inside the sphere, while ours is a right-handed coordinate system, representing a south-facing observer outside the sphere, which is the usual convention for spherical coordinate systems.

Parameters

- **variable** (*numpy.ndarray (2D, 3D, or 4D)*) – N-D data array to be transformed. Final two dimensions must be (lat,lon)
- **lon** (*numpy.ndarray*) – 1D array of longitudes [deg]
- **lat** (*numpy.ndarray*) – 1D array of latitudes [deg]
- **substellar** (*float, optional*) – Longitude of the substellar point (defaults to 0 degrees)

Returns Transformed longitudes, latitudes, and data array.

Return type numpy.ndarray, numpy.ndarray, numpy.ndarray

exoplasim.gcmt.**tl2eq_coords**(*lon*, *lat*, *substellar=0.0*)
Compute equatorial coordinates of a set of tidally-locked lat-lon coordinates.

Transforms tidally-locked coordinates into the standard equatorial coordinate system. Note that in our tidally-locked coordinate system, 0 degrees longitude is the substellar-south pole-antistellar meridian, and 90 degrees latitude is the substellar point, such that the evening hemisphere is 0-180 degrees longitude, the morning hemisphere is 180-360 degrees longitude, the north equatorial pole is at (0, 180), and easterly flow is counter-clockwise. Note that this differs from the coordinate system introduced in Koll & Abbot (2015) in that theirs is a left-handed coordinate system, with the south pole at (0, 180) and counter-clockwise easterly flow, which represents a south-facing observer inside the sphere, while ours is a right-handed coordinate system, representing a south-facing observer outside the sphere, which is the usual convention for spherical coordinate systems.

Parameters

- **lon** (*numpy.ndarray*) – Longitudes in tidally-locked coordinates [degrees]
- **lat** (*numpy.ndarray*) – Latitudes in tidally-locked coordinates [degrees]
- **substellar** (*float, optional*) – Longitude of the substellar point. [degrees]

Returns Transformed longitudes and latitudes [degrees]

Return type numpy.ndarray, numpy.ndarray

exoplasim.gcmt.**tlstream**(*dataset*, *radius=6371000.0*, *gravity=9.80665*, *substellar=0.0*)
Compute the tidally-locked streamfunction

Parameters

- **dataset** (*str or ExoPlaSim Dataset*) – Either path to ExoPlaSim Dataset of model output or an instance of the dataset.
- **radius** (*float, optional*) – Planetary radius [m]
- **gravity** (*float, optional*) – Surface gravity [m/s^2]
- **substellar** (*float, optional*) – Longitude of the substellar point in degrees.

Returns tidally-locked latitude, layer interface pressures, and TL streamfunction

Return type numpy.ndarray(1D), numpy.ndarray(1D), numpy.ndarray(2D)

exoplasim.gcmt.**wrap2d**(*var*)
Add one element to the longitude axis to allow for wrapping

exoplasim.gcmt.**xcolorbar**(*mappable*, *fontsize=None*, *ticksize=None*, ***kwargs*)

1.3.4 exoplasim.pyburn module

Read raw exoplasim output files and postprocess them into netCDF output files.

exoplasim.pyburn.**advancedDataset**(*filename*, *variablecodes*, *substellarlon=180.0*, *radius=1.0*, *gravity=9.80665*, *gascon=287.0*, *logfile=None*)
Read a raw output file, and construct a dataset.

Parameters

- **filename** (*str*) – Path to the raw output file
- **variablecodes** (*dict*) – Variables to include. Each member must use the variable name as the key, and contain a sub-dict with the horizontal mode, zonal averaging, and physics filtering options optionall set as members. For example:

```
{"ts":{"mode":"grid","zonal":False},
 "stf":{"mode":"grid","zonal":True,"physfilter":True}}
```

  Options that are not set take on their default values from *dataset()*.

- **mode** (*str, optional*) – Horizontal output mode. Can be 'grid', meaning the Gaussian latitude-longitude grid used in ExoPlaSim, 'spectral', meaning spherical harmonics, 'fourier', meaning Fourier coefficients and latitudes, 'synchronous', meaning a Gaussian latitude-longitude grid in the synchronous coordinate system defined in Paradise, et al (2021), with the north pole centered on the substellar point, or 'syncfourier', meaning Fourier coefficients computed along the dipolar meridians in the synchronous coordinate system (e.g. the substellar-antistellar-polar meridian, which is 0 degrees, or the substellar-evening-antistellar-morning equatorial meridian, which is 90 degrees). Because this will get assigned to the original latitude array, that will become -90 degrees for the polar meridian, and 0 degrees for the equatorial meridian, identical to the typical equatorial coordinate system.
- **zonal** (*bool, optional*) – For grid modes ("grid" and "synchronous"), compute and output zonal means
- **physfilter** (*bool, optional*) – Whether or not a physics filter should be used when transforming spectral variables to Fourier or grid domains

- **substellarlon** (*float, optional*) – If mode='synchronous', the longitude of the substellar point in equatorial coordinates, in degrees
- **radius** (*float, optional*) – Planet radius in Earth radii
- **gravity** (*float, optional*) – Surface gravity in m/s^2.
- **gascon** (*float, optional*) – Specific gas constant for dry gas (R$_d$) in J/kg/K.
- **logfile** (*str or None, optional*) – If None, log diagnostics will get printed to standard output. Otherwise, the log file to which diagnostic output should be written.

Returns Dictionary of extracted variables

Return type dict

exoplasim.pyburn.**csv**(*rdataset*, *filename='most_output.tar.gz'*, *logfile=None*, *extracompression=False*)

Write a dataset to CSV/TXT-type output, optionally compressed.

If a tarball format (e.g. *.tar or *.tar.gz) is used, output files will be packed into a tarball. gzip (.gz), bzip2 (.bz2), and lzma (.xz) compression types are supported. If a tarball format is not used, then accepted file extensions are .csv, .txt, or .gz. All three will produce a directory named following the filename pattern, with one file per variable in the directory. If the .gz extension is used, NumPy will compress each output file using gzip compression.

Files will only contain 2D variable information, so the first N-1 dimensions will be flattened. The original variable shape is included in the file header (prepended with a # character) as the first items in a comma-separated list, with the first non-dimension item given as the 'lll' placeholder. On reading variables from these files, they should be reshaped according to these dimensions. This is true even in tarballs (which contain CSV files).

Parameters

- **rdataset** (*dict*) – A dictionary of outputs as generated from *pyburn.dataset()*
- **filename** (*str, optional*) – Path to the output file that should be written. This will be parsed to determine output type.
- **logfile** (*str or None, optional*) – If None, log diagnostics will get printed to standard output. Otherwise, the log file to which diagnostic output should be written.
- **extracompression** (*bool, optional*) – If True, then component files in tarball outputs will be compressed individually with gzip, instead of being plain-text CSV files.

Returns If non-tarball output was used, a tuple containing a list of paths to output files, and a string giving the name of the output directory. If tarball output was used, a relative path to the tarball.

Return type tuple or str

exoplasim.pyburn.**dataset**(*filename*, *variablecodes*, *mode='grid'*, *zonal=False*, *substellarlon=180.0*, *physfilter=False*, *radius=1.0*, *gravity=9.80665*, *gascon=287.0*, *logfile=None*)

Read a raw output file, and construct a dataset.

Parameters

- **filename** (*str*) – Path to the raw output file
- **variablecodes** (*array-like*) – list of variables to include. Can be the integer variable codes from the burn7 postprocessor conventions (as either strings or integers), or the short variable name strings (e.g. 'rlut'), or a combination of the two.

- **mode** (*str, optional*) – Horizontal output mode. Can be 'grid', meaning the Gaussian latitude-longitude grid used in ExoPlaSim, 'spectral', meaning spherical harmonics, 'fourier', meaning Fourier coefficients and latitudes, 'synchronous', meaning a Gaussian latitude-longitude grid in the synchronous coordinate system defined in Paradise, et al (2021), with the north pole centered on the substellar point, or 'syncfourier', meaning Fourier coefficients computed along the dipolar meridians in the synchronous coordinate system (e.g. the substellar-antistellar-polar meridian, which is 0 degrees, or the substellar-evening-antistellar-morning equatorial meridian, which is 90 degrees). Because this will get assigned to the original latitude array, that will become -90 degrees for the polar meridian, and 0 degrees for the equatorial meridian, identical to the typical equatorial coordinate system.
- **zonal** (*bool, optional*) – For grid modes ("grid" and "synchronous"), compute and output zonal means
- **substellarlon** (*float, optional*) – If mode='synchronous', the longitude of the substellar point in equatorial coordinates, in degrees
- **physfilter** (*bool, optional*) – Whether or not a physics filter should be used when transforming spectral variables to Fourier or grid domains
- **radius** (*float, optional*) – Planet radius in Earth radii
- **gravity** (*float, optional*) – Surface gravity in m/s^2.
- **gascon** (*float, optional*) – Specific gas constant for dry gas (R$_d$) in J/kg/K.
- **logfile** (*str or None, optional*) – If None, log diagnostics will get printed to standard output. Otherwise, the log file to which diagnostic output should be written.

Returns Dictionary of extracted variables

Return type dict

exoplasim.pyburn.**hdf5**(*rdataset*, *filename='most_output.hdf5'*, *append=False*, *logfile=None*)

Write a dataset to HDF5 output.

Note: HDF5 files are opened in append mode. This means that this format can be used to create a single output dataset for an entire simulation.

HDF5 files here are generated with gzip compression at level 9, with chunk rearrangement and Fletcher32 checksum data protection.

Parameters

- **rdataset** (*dict*) – A dictionary of outputs as generated from *pyburn.dataset()*
- **filename** (*str, optional*) – Path to the output file that should be written.
- **append** (*bool, optional*) – Whether or not the file should be opened in append mode, or overwritten (default).
- **logfile** (*str or None, optional*) – If None, log diagnostics will get printed to standard output. Otherwise, the log file to which diagnostic output should be written.

Returns An HDF5 object corresponding to the file that has been written.

Return type object

exoplasim.pyburn.**netcdf**(*rdataset*, *filename='most_output.nc'*, *append=False*, *logfile=None*)

Write a dataset to a netCDF file.

Parameters

- **rdataset** (*dict*) – A dictionary of outputs as generated from *pyburn.dataset()*

- **filename** (*str, optional*) – Path to the output file that should be written.
- **append** (*bool, optional*) – Whether the file should be opened in "append" mode, or overwritten (default).
- **logfile** (*str or None, optional*) – If None, log diagnostics will get printed to standard output. Otherwise, the log file to which diagnostic output should be written.

Returns A netCDF object corresponding to the file that has been written.

Return type object

exoplasim.pyburn.**npsavez**(*rdataset*, *filename='most_output.npz'*, *logfile=None*)

Write a dataset to a NumPy compressed .npz file.

Two output files will be created: filename as specified (e.g. most_output.npz), which contains the data variables, and a metadata file (e.g. most_output_metadata.npz), which contains the metadata headers associated with each variable.

Parameters

- **rdataset** (*dict*) – A dictionary of outputs as generated from *pyburn.dataset()*
- **filename** (*str, optional*) – Path to the output file that should be written.
- **logfile** (*str or None, optional*) – If None, log diagnostics will get printed to standard output. Otherwise, the log file to which diagnostic output should be written.

Returns A 2-item tuple containing (variables, meta), each of which is a dictionary with variable names as keys.

Return type tuple

exoplasim.pyburn.**postprocess**(*rawfile*, *outfile*, *logfile=None*, *namelist=None*, *variables=None*, *mode='grid'*, *zonal=False*, *substellarlon=180.0*, *physfilter=False*, *timeaverage=True*, *stdev=False*, *times=12*, *interpolatetimes=True*, *radius=1.0*, *gravity=9.80665*, *gascon=287.0*, *mars=False*)

Convert a raw output file into a postprocessed formatted file.

Output format is determined by the file extension of outfile. Current supported formats are NetCDF (*.nc), HDF5 (*.hdf5, *.he5, *.h5), numpy's `np.savez_compressed` format (*.npz), and CSV format. If NumPy's single-array .npy extension is used, .npz will be substituted–this is a compressed ZIP archive containing .npy files. Additionally, the CSV output format can be used in compressed form either individually by using the .gz file extension, or collectively via tarballs (compressed or uncompressed).

If a tarball format (e.g. *.tar or *.tar.gz) is used, output files will be packed into a tarball. gzip (.gz), bzip2 (.bz2), and lzma (.xz) compression types are supported. If a tarball format is not used, then accepted file extensions are .csv, .txt, or .gz. All three will produce a directory named following the filename pattern, with one file per variable in the directory. If the .gz extension is used, NumPy will compress each output file using gzip compression.

CSV-type files will only contain 2D variable information, so the first N-1 dimensions will be flattened. The original variable shape is included in the file header (prepended with a # character) as the first items in a comma-separated list, with the first non-dimension item given as the '|||' placeholder. On reading variables from these files, they should be reshaped according to these dimensions. This is true even in tarballs (which contain CSV files).

A T21 model output with 10 vertical levels, 12 output times, all supported variables in grid mode,and no standard deviation computation will have the following sizes for each format:

Format	Size
netCDF	12.8 MiB
HDF5	17.2 MiB
NumPy (default)	19.3 MiB
tar.xz	33.6 MiB
tar.bz2	36.8 MiB
gzipped	45.9 MiB
uncompressed	160.2 MiB

Using the NetCDF (.nc) format requires the netCDF4 python package.

Using the HDF5 format (.h5, .hdf5, .he5) requires the h5py python package.

Parameters

- **rawfile** (*str*) – Path to the raw output file
- **outfile** (*str*) – Path to the destination output file. The file extension determines the format. Currently, netCDF (*.nc). numpy compressed (*.npz), HDF5 (*.hdf5, *.he5, *.h5), or CSV-type (*.csv, *.txt, *.gz, *.tar, *.tar.gz, *.tar.bz2, *.tar.xz) are supported. If a format (such as npz) that requires that metadata be placed in a separate file is chosen, a second file with a '_metadata' suffix will be created.
- **append** (*bool, optional*) – If True, and outfile already exists, then append to outfile rather than overwriting it. Currently only supported for netCDF and HDF5 formats. Support for other formats coming soon.
- **logfile** (*str or None, optional*) – If None, log diagnostics will get printed to standard output. Otherwise, the log file to which diagnostic output should be written.
- **namelist** (*str, optional*) – Path to a burn7 postprocessor namelist file. If not given, then `variables` must be set.
- **variables** (*list or dict, optional*) – If a list is given, a list of either variable keycodes (integers or strings), or the abbreviated variable name (e.g. 'ts' for surface temperature). If a dict is given, each item in the dictionary should have the keycode or variable name as the key, and the desired horizontal mode and additional options for that variable as a sub-dict. Each member of the subdict should be passable as **kwargs to *advancedDataset()*. If None, then `namelist` must be set.
- **mode** (*str, optional*) – Horizontal output mode, if modes are not specified for individual variables. Options are 'grid', meaning the Gaussian latitude-longitude grid used in ExoPlaSim, 'spectral', meaning spherical harmonics, 'fourier', meaning Fourier coefficients and latitudes, 'synchronous', meaning a Gaussian latitude-longitude grid in the synchronous coordinate system defined in Paradise, et al (2021), with the north pole centered on the substellar point, or 'syncfourier', meaning Fourier coefficients computed along the dipolar meridians in the synchronous coordinate system (e.g. the substellar-antistellar-polar meridian, which is 0 degrees, or the substellar-evening-antistellar-morning equatorial meridian, which is 90 degrees). Because this will get assigned to the original latitude array, that will become -90 degrees for the polar meridian, and 0 degrees for the equatorial meridian, identical to the typical equatorial coordinate system.
- **zonal** (*bool, optional*) – Whether zonal means should be computed for applicable variables.
- **substellarlon** (*float, optional*) – Longitude of the substellar point. Only relevant if a synchronous coordinate output mode is chosen.

- **physfilter** (*bool, optional*) – Whether or not a physics filter should be used in spectral transforms.
- **times** (*int or array-like or None, optional*) – Either the number of timestamps by which to divide the output, or a list of times given as a fraction of the output file duration (which enables e.g. a higher frequency of outputs during periapse of an eccentric orbit, when insolation is changing more rapidly). If None, the timestamps in the raw output will be written directly to file.
- **timeaverage** (*bool, optional*) – Whether or not timestamps in the output file should be averaged to produce the requested number of output timestamps. Timestamps for averaged outputs will correspond to the middle of the averaged time period.
- **stdev** (*bool, optional*) – Whether or not standard deviations should be computed. If timeaverage is True, this will be the standard deviation over the averaged time period; if False, then it will be the standard deviation over the whole duration of the output file
- **interpolatetimes** (*bool, optional*) – If true, then if the times requested don't correspond to existing timestamps, outputs will be linearly interpolated to those times. If false, then nearest-neighbor interpolation will be used.
- **radius** (*float, optional*) – Planet radius in Earth radii
- **gravity** (*float, optional*) – Surface gravity in m/s^2.
- **gascon** (*float, optional*) – Specific gas constant for dry gas (R$_d$) in J/kg/K.
- **mars** (*bool, optional*) – If True, use Mars constants

exoplasim.pyburn.**readallvariables**(*fbuffer*)

Extract all variables and their headers from a file byte buffer.

Doing this and then only keeping the codes you want may be faster than extracting variables one by one, because it only needs to seek through the file one time.

Parameters **fbuffer** (*bytes*) – Binary bytes read from a file opened with `mode='rb'` and read with `file.read()`.

Returns A dictionary containing all variable headers (by variable code), and a dictionary containing all variables, again by variable code.

Return type dict, dict

exoplasim.pyburn.**readfile**(*filename*)

Extract all variables from a raw plasim output file and refactor them into the right shapes

This routine will only produce what it is in the file; it will not compute derived variables.

Parameters **filename** (*str*) – Path to the output file to read

Returns Dictionary of model variables, indexed by numerical code

Return type dict

exoplasim.pyburn.**readrecord**(*fbuffer*, *n*, *en*, *ml*, *mf*)

Read a Fortran record from the buffer, starting at index n, and return the header, data, and updated n.

Parameters

- **fbuffer** (*bytes*) – Binary bytes read from a file opened with `mode='rb'` and read with `file.read()`.
- **n** (*int*) – The index of the word at which to start, in bytes. A 32-bit word has length 4, so the current position in words would be 4*n assuming 4-byte words, or 8*n if 64 bits and 8-byte words.

- **en** (*str*) – Endianness, denoted by ">" or "<"
- **ml** (*int*) – Length of a record marker
- **mf** (*str*) – Format of the record marker ('i' or 'l')

Returns A tuple containing first the header, then the data contained in the record, and finally the new position in the buffer in bytes.

Return type array-like, array-like, int

exoplasim.pyburn.**readvariablecode**(*fbuffer*, *kcode*, *en*, *ml*, *mf*)

Seek through a binary output buffer and extract all records associated with a variable code.

Note, assembling a variable list piece by piece in this way may be slower than reading **all** variables at once, because it requires seeking all the way through the buffer multiple times for each variable. This will likely only be faster if you only need a small number of variables.

Parameters

- **fbuffer** (*bytes*) – Binary bytes read from a file opened with `mode='rb'` and read with `file.read()`.
- **kcode** (*int*) – The integer code associated with the variable. For possible codes, refer to the ``Postprocessor Variable Codes. <postprocessor.html#postprocessor-variable-codes>`_
- **en** (*str*) – Endianness, denoted by ">" or "<"
- **ml** (*int*) – Length of a record marker
- **mf** (*str*) – Format of the record marker ('i' or 'l')

Returns A tuple containing first the header, then the variable data, as one concatenated 1D variable.

Return type array-like, array-like

exoplasim.pyburn.**refactorvariable**(*variable*, *header*, *nlev=10*)

Given a 1D data array extracted from a file with *readrecord*, reshape it into its appropriate dimensions.

Parameters

- **variable** (*array-like*) – Data array extracted from an output file using *readrecord*. Can also be the product of a concatenated file assembled with `readvariable`.
- **header** (*array-like*) – The header array extracted from the record associated with `variable`. This header contains dimensional information.
- **nlev** (*int, optional*) – The number of vertical levels in the variable. If 1, vertical levels will not be a dimension in the output variable.

Returns A numpy array with dimensions inferred from the header.

Return type numpy.ndarray

1.3.5 exoplasim.randomcontinents module

```
usage: randomcontinents.py [-h] [-z] [-c CONTINENTS] [-f LANDFRACTION]
                           [-n NAME] [-m MAXZ] [--nlats NLATS]
                           [-l HEMISPHERELONGITUDE] [-o]

Randomly generate continents up to a specified land-fraction. Topography
optional.

optional arguments:
  -h, --help            show this help message and exit
  -z, --topo            Generate topographical geopotential map
  -c CONTINENTS, --continents CONTINENTS
                        Number of continental cratons
  -f LANDFRACTION, --landfraction LANDFRACTION
                        Land fraction
  -n NAME, --name NAME  Assign a name for the planet
  -m MAXZ, --maxz MAXZ  Maximum elevation in km assuming Earth gravity
  --nlats NLATS         Number of latitudes (evenly-spaced)--will also set
                        longitudes (twice as many).
  -l HEMISPHERELONGITUDE, --hemispherelongitude HEMISPHERELONGITUDE
                        Confine land to a hemisphere centered on a given
                        longitude
  -o, --orthographic    Plot orthographic projections centered on
                        hemispherelongitude
```

exoplasim.randomcontinents.**generate**(*name='Alderaan'*, *continents=7*, *landfraction=0.29*, *maxz=10.0*, *nlats=32*, *hemispherelongitude=nan*, *ntopo=False*, *orthographic=False*)

Randomly generate continents up to specified land fraction. Topography optional.

Generates name_surf_0172.sra, the land mask file, and (if requested) name_surf_0129.sra, the topography file.

Parameters

- **name** (*str, optional*) – Name for the planet; will be used in filenames.
- **continents** (*int, optional*) – Number of initial continental cratons. Note that due to craton collisions, this may not be the number of final landmasses.
- **landfraction** (*float, optional*) – Target land fraction (may deviate slightly).
- **maxz** (*float, optional*) – Maximum surface elevation under Earth gravity (non-Earth gravity will change the final elevation)
- **nlats** (*int, optional*) – Number of latitudes. If set to False, T21 Gaussian latitudes will be used. Longitudes are 2*nlats.
- **hemispherelongitude** (*float, optional*) – If finite, confine land to a hemisphere centered on this longitude.
- **topo** (*bool, optional*) – If True, compute topography.
- **orthorgraphic** (*bool, optional*) – If True, plot orthographic projections centered on hemispherelongitude.

Returns Longitude, Latitude, land-sea mask, and if requested, surface geopotential (topography)

Return type np.ndarray(2*nlat), np.ndarray(nlat), np.ndarray(nlat,2*nlat)[, np.ndarray(nlat,2*nlat)]

exoplasim.randomcontinents.**main**()
Command-line tool to randomly generate continents up to specified land fraction. Topography optional.

Do not invoke as an imported function; must run directly.

Options

-z,–topo Generate topographical geopotential map

-c,–continents Number of continental cratons

-f,–landfraction Land fraction

-n,–name Assign a name for the planet

-m,–maxz Maximum elevation in km assuming Earth gravity

--nlats Number of latitudes (evenly-spaced)–will also set longitudes (twice as many). If unset, PlaSim latitudes and longitudes will be used (T21 resolution)"

-l,–hemispherelongitude Confine land to a hemisphere centered on a given longitude

-o,–orthographic Plot orthographic projections centered on hemispherelongitude

name_surf_0172.sra Land mask SRA file

name_surf_0129.sra (optional) Topography geopotential SRA file (if requested)

exoplasim.randomcontinents.**writePGM**(*name*, *heightfield*)
Write a lat-lon field to a .pgm image file (usually topo field)

Parameters

- **name** (*str*) – The name with which to label this map
- **heightfield** (*numpy.ndarray*) – The 2-D map to write to file.

exoplasim.randomcontinents.**writeSRA**(*name*, *kcode*, *field*, *NLAT*, *NLON*)
Write a lat-lon field to a formatted .sra file

Parameters

- **name** (*str*) – The name with which to label this map
- **kcode** (*int*) – The integer map code for specifying what kind of boundary file this is (see the PlaSim documentation for more details)
- **field** (*numpy.ndarray*) – The map to write to file. Should have the dimensions (NLAT,NLON).
- **NLAT** (*int*) – The number of latitudes
- **NLON** (*int*) – The number of longitudes

1.3.6 exoplasim.makestellarspec module

exoplasim.makestellarspec.**main**()
Convert spectral files to exoplasim-compliant formats, including resampling to the necessary resolutions.

Must give as input a spectrum file generated by the Phoenix stellar spectrum web simulator, https://phoenix.ens-lyon.fr/simulator-jsf22-26. **Do not use as an imported function; only call directly as a command-line program.**

Usage

```
>>> python makestellarspec.py spectrum.txt mystarsname
```

Yields

- *mystarsname.dat* – 965 wavelengths
- *mystarsname_hr.dat* – 2048 wavelengths

exoplasim.makestellarspec.**readspec**(*sfile*, *cgs=False*)
Read a Phoenix stellar spectrum and return wavelengths, fluxes, and units

Takes as input a spectrum file generated by the Phoenix stellar spectrum web simulator, https://phoenix.ens-lyon.fr/simulator-jsf22-26.

Parameters

- **sfile** (*str*) – Path to the spectrum file
- **cgs** (*bool, optional*) – Whether or not we should try to use CGS units (irrelevant for exoplasim)

Returns Returns wavelengths, fluxes, and the units

Return type numpy.ndarray, numpy.ndarray, str

exoplasim.makestellarspec.**writedat**(*wvls*, *fluxes*, *name*)
Write wavelengths and fluxes to a .dat file ExoPlaSim can read.

Parameters

- **wvls** (*array-like*) – An array-like object containing a monotonic list of wavelengths going from short to long
- **fluxes** (*array-like*) – An array-like object containing a flux for each wavelength in wvls.
- **name** (*str*) – A name to use when generating output files.

Yields *name.dat* – The provided spectrum in a format ExoPlaSim can read.

1.3.7 exoplasim.pRT module

exoplasim.pRT.**basicclouds**(*pressure*, *temperature*, *cloudwater*)
A basic cloud parameterization using T-dependent particle size distribution.

This could be replaced with a different (better) cloud particle parameterization, but it should have the same call signature and return the same thing. This parameterization is borrowed from Edwards, et al (2007, doi:10.1016/j.atmosres.2006.03.002).

Parameters

- **pressure** (*numpy.ndarray*) – Pressure array for a column of the model
- **temperature** (*numpy.ndarray*) – Air temperatures for a column [K]
- **cloudwater** (*numpy.ndarray*) – Cloud water as a mass fraction [kg/kg] – this is condensed water suspended in the cloud.

Returns Dictionary of keyword arguments for setting clouds with an empirical particle size distribution.

Return type dict

exoplasim.pRT.**image**(*output*, *imagetimes*, *gases_vmr*, *obsv_coords*, *gascon=287.0*, *gravity=9.80665*, *Tstar=5778.0*, *Rstar=1.0*, *orbdistances=1.0*, *h2o_lines='HITEMP'*, *num_cpus=4*, *cloudfunc=None*, *smooth=True*, *smoothweight=0.5*, *filldry=0.0*, *stellarspec=None*, *ozone=False*, *stepsperyear=11520.0*, *logfile=None*, *debug=False*, *orennayar=True*, *sigma=None*, *allforest=False*, *baremountainz=50000.0*, *colorspace='sRGB'*, *gamma=True*, *consistency=True*, *vegpowerlaw=1.0*)

Compute reflection+emission spectra for snapshot output

This routine computes the reflection+emission spectrum for the planet at each indicated time.

Note that deciding what the observer coordinates ought to be may not be a trivial operation. Simply setting them to always be the same is fine for a 1:1 synchronously-rotating planet, where the insolation pattern never changes. But for an Earth-like rotator, you will need to be mindful of rotation rate and the local time when snapshots are written. Perhaps you would like to see how things look as the local time changes, as a geosynchronous satellite might observe, or maybe you'd like to only observe in secondary eclipse or in quadrature, and so the observer-facing coordinates may not be the same each time.

Parameters

- **output** (*ExoPlaSim snapshot output*) – Preferably opened with *exoplasim.gcmt.load()*.
- **imagetimes** (*list(int)*) – List of time indices at which the image should be computed.
- **gases_vmr** (*dict*) – Dictionary of gas species volume mixing ratios for the atmosphere
- **obsv_coords** (*numpy.ndarray (3D)*) – List of observer (lat,lon) coordinates for each observing time. First axis is time, second axis is for each observer; the third axis is for lat and lon. Should have shape (time,observers,lat-lon). These are the surface coordinates that are directly facing the observer.
- **gascon** (*float, optional*) – Specific gas constant
- **gravity** (*float, optional*) – Surface gravity in SI units
- **Tstar** (*float, optional*) – Effective temperature of the parent star [K]
- **Rstar** (*float, optional*) – Radius of the parent star in solar radii
- **orbdistances** (*float or numpy.ndarray, optional*) – Distance between planet and star in AU
- **h2o_lines** (*{'HITEMP','EXOMOL'}, optional*) – Either 'HITEMP' or 'EXOMOL'–the line list from which H2O absorption should be sourced
- **num_cpus** (*int, optional*) – The number of CPUs to use

- **cloudfunc** (*function, optional*) – A routine which takes pressure, temperature, and cloud water content as arguments, and returns keyword arguments to be unpacked into calc_flux_transm. If not specified, *basicclouds* will be used.
- **smooth** (*bool, optional*) – Whether or not to smooth humidity and cloud columns. As of Nov 12, 2021, it is recommended that you use smooth=True for well-behaved spectra. This is a conservative smoothing operation, meaning the water and cloud column mass should be conserved–what this does is move some water from the water-rich layers into the layers directly above and below.
- **smoothweight** (*float, optional*) – The fraction of the water in a layer that should be retained during smoothing. A higher value means the smoothing is less severe. 0.95 is probably the upper limit for well-behaved spectra.
- **filldry** (*float, optional*) – If nonzero, the floor value for water humidity when moist layers are present above dry layers. Columns will be adjusted in a mass-conserving manner with excess humidity accounted for in layers *above* the filled layer, such that total optical depth from TOA is maintained at the dry layer.
- **stellarspec** (*array-like (optional)*) – A stellar spectrum measured at the wavelengths in surfacespecs.wvl. If None, a blackbody will be used.
- **ozone** (*bool or dict, optional*) – True/False/dict. Whether or not forcing from stratospheric ozone should be included. If a dict is provided, it should contain the keys "height", "spread", "amount","varlat","varseason", and "seasonoffset", which correspond to the height in meters of peak O3 concentration, the width of the gaussian distribution in meters, the baseline column amount of ozone in cm-STP, the latitudinal amplitude, the magnitude of seasonal variation, and the time offset of the seasonal variation in fraction of a year. The three amounts are additive. To set a uniform, unvarying O3 distribution, ,place all the ozone in "amount", and set "varlat" and "varseason" to 0.
- **stepsperyear** (*int or float, optional*) – Number of timesteps per sidereal year. Only used for computing ozone seasonality.
- **orennayar** (*bool, optional*) – If True, compute true-colour intensity using Oren-Nayar scattering instead of Lambertian scattering. Most solar system bodies do not exhibit Lambertian scattering.
- **sigma** (*float, optional*) – If not None and *orennayar* is *True*, then this sets the roughness parameter for Oren-Nayar scattering. *sigma=0* is the limit of Lambertian scattering, while *sigma=0.97* is the limit for energy conservation. *sigma* is the standard deviation of the distribution of microfacet normal angles relative to the mean, normalized such that *sigma=1.0* would imply truly isotropic microfacet distribution. If sigma is None (default), then sigma is determined based on the surface type in a column and whether clouds are present, using 0.4 for ground, 0.1 for ocean, 0.9 for snow/ice, and 0.95 for clouds.
- **allforest** (*bool, optional*) – If True, force all land surface to be forested.
- **baremountainz** (*float, optional*) – If vegetation is present, the geopotential above which mountains become bare rock instead of eroded vegetative regolith. Functionally, this means gray rock instead of brown/tan ground.
- **colorspace** (*str or np.ndarray(3,3)*) – Color gamut to be used. For available built-in color gamuts, see colormatch.colorgamuts.
- **gamma** (*bool or float, optional*) – If True, use the piecewise gamma-function defined for sRGB; otherwise if a float, use rgb^(1/gamma). If None, gamma=1.0 is used.

- **consistency** (*bool, optional*) – If True, force surface albedo to match model output
- **vegpowerlaw** (*float, optional*) – Scale the apparent vegetation fraction by a power law. Setting this to 0.1, for example, will increase the area that appears partially-vegetated, while setting it to 1.0 leaves vegetation unchanged.

Returns pRT Atmosphere object, wavelength in microns, Numpy array with dimensions (ntimes,nlat,nlon,nfreq), where ntimes is the number of output times, and nfreq is the number of frequencies in the spectrum, longitudes, latitudes, and an array with dimensions (ntimes,nfreq) corresponding to disk-averaged spectra, with individual contributions weighted by visibility and projected area.

Return type Atmosphere, numpy.ndarray, numpy.ndarray, numpy.ndarray, numpy.ndarray

exoplasim.pRT.**makecolors**(*intensities*, *gamma=True*, *colorspace='sRGB'*)

Convert (x,y,Y) intensities to RGB values.

Uses CIE color-matching functions and a wide-gamut RGB colorspace to convert XYZ-coordinate color intensities to RGB intensities.

Parameters **intensities** (*array-like*) – Last index must have length 3–array of (x,y,Y) intensities

Returns RGB color values.

Return type array-like (N,3)

exoplasim.pRT.**makeintensities**(*wvl*, *fluxes*)

Convert spectrum to (x,y,Y) intensities.

Parameters

- **wvl** (*array-like*) – Wavelengths in microns
- **fluxes** (*array-like*) – Spectrum in fluxes (units are arbitrary)

Returns (x,y,Y) tuple

Return type (float,float,float)

exoplasim.pRT.**orennayarcorrection**(*intensity*, *lon*, *lat*, *sollon*, *sollat*, *zenith*, *observer*, *albedo*, *sigma*)

Correct scattering intensity from Lambertian to full Oren-Nayar.

Parameters

- **intensity** (*array-like or float*) – Intensity to correct
- **lon** (*array-like or float*) – Column(s) longitude in degrees
- **lat** (*array-like or float*) – Column(s) latitude in degrees
- **sollon** (*float*) – Substellar longitude
- **sollat** (*float*) – Substellar latitude
- **zenith** (*array-like or float*) – Solar zenith angle(s) in degrees
- **observer** (*tuple*) – (lat,lon) tuple of sub-observer coordinates
- **albedo** (*array-like or float*) – Scattering surface reflectivity (0–1)
- **sigma** (*array-like or float*) – Scattering surface roughness. 0.0 is Lambertian, 0.97 is the maximum energy-conserving roughness. 0.25-0.3 is appropriate for many planetary bodies.

Returns Corrected intensity of the same shape as the input intensity

Return type array-like

exoplasim.pRT.**orennayarcorrection_col**(*intensity*, *lon*, *lat*, *sollon*, *sollat*, *zenith*, *observer*, *albedo*, *sigma*)

Correct scattering intensity from Lambertian to full Oren-Nayar.

Parameters

- **intensity** (*array-like or float*) – Intensity to correct
- **lon** (*array-like or float*) – Column(s) longitude in degrees
- **lat** (*array-like or float*) – Column(s) latitude in degrees
- **sollon** (*float*) – Substellar longitude
- **sollat** (*float*) – Substellar latitude
- **zenith** (*array-like or float*) – Solar zenith angle(s) in degrees
- **observer** (*tuple*) – (lat,lon) tuple of sub-observer coordinates
- **albedo** (*array-like or float*) – Scattering surface reflectivity (0–1)
- **sigma** (*array-like or float*) – Scattering surface roughness. 0.0 is Lambertian, 0.97 is the maximum energy-conserving roughness. 0.25-0.3 is appropriate for many planetary bodies.

Returns Corrected intensity of the same shape as the input intensity

Return type array-like

exoplasim.pRT.**save**(*filename*, *dataset*, *logfile=None*, *extracompression=False*)

Save petitRADTRANS ExoPlaSim output to a file.

Output format is determined by the file extension in filename. Current supported formats are NetCDF (*.nc), HDF5 (*.hdf5, *.he5, *.h5), numpy's `np.savez_compressed` format (*.npz), and CSV format. If NumPy's single-array .npy extension is used, .npz will be substituted–this is a compressed ZIP archive containing .npy files. Additionally, the CSV output format can be used in compressed form either individually by using the .gz file extension, or collectively via tarballs (compressed or uncompressed).

If a tarball format (e.g. *.tar or *.tar.gz) is used, output files will be packed into a tarball. gzip (.gz), bzip2 (.bz2), and lzma (.xz) compression types are supported. If a tarball format is not used, then accepted file extensions are .csv, .txt, or .gz. All three will produce a directory named following the filename pattern, with one file per variable in the directory. If the .gz extension is used, NumPy will compress each output file using gzip compression.

CSV-type files will only contain 2D variable information, so the first N-1 dimensions will be flattened. The original variable shape is included in the file header (prepended with a # character) as the first items in a comma-separated list, with the first non-dimension item given as the 'III' placeholder. On reading variables from these files, they should be reshaped according to these dimensions. This is true even in tarballs (which contain CSV files).

A T21 model output with 10 vertical levels, 12 output times, all supported variables in grid mode,and no standard deviation computation will have the following sizes for each format:

Format	Size
netCDF	12.8 MiB
HDF5	17.2 MiB
NumPy (default)	19.3 MiB
tar.xz	33.6 MiB
tar.bz2	36.8 MiB
gzipped	45.9 MiB
uncompressed	160.2 MiB

Using the NetCDF (.nc) format requires the netCDF4 python package.

Using the HDF5 format (.h5, .hdf5, .he5) requires the h5py python package.

Parameters

- **filename** (*str*) – Path to the destination output file. The file extension determines the format. Currently, netCDF (*.nc). numpy compressed (*.npz), HDF5 (*.hdf5, *.he5, *.h5), or CSV-type (*.csv, *.txt, *.gz, *.tar, *.tar.gz, *.tar.bz2, *.tar.xz) are supported. If a format (such as npz) that requires that metadata be placed in a separate file is chosen, a second file with a '_metadata' suffix will be created.
- **dataset** (*dict*) – A dictionary containing the fields that should be written to output.
- **logfile** (*str or None, optional*) – If None, log diagnostics will get printed to standard output. Otherwise, the log file to which diagnostic output should be written.

Returns Open cross-format dataset object

Return type gcmt._Dataset object

exoplasim.pRT.**transit**(*output*, *transittimes*, *gases_vmr*, *gascon=287.0*, *gravity=9.80665*, *rplanet=6371.0*, *h2o_lines='HITEMP'*, *num_cpus=4*, *cloudfunc=None*, *smooth=False*, *smoothweight=0.95*, *ozone=False*, *stepsperyear=11520.0*, *logfile=None*)

Compute transmission spectra for snapshot output

This routine computes the transmission spectrum for each atmospheric column along the terminator, for each time in transittimes.

Note: This routine does not currently include emission from atmospheric layers.

Parameters

- **output** (*ExoPlaSim snapshot output*) – Preferably opened with *exoplasim.gcmt.load()*.
- **transittimes** (*list(int)*) – List of time indices at which the transit should be computed.
- **gases_vmr** (*dict*) – Dictionary of gas species volume mixing ratios for the atmosphere
- **gascon** (*float, optional*) – Specific gas constant
- **gravity** (*float, optional*) – Surface gravity in SI units
- **rplanet** (*float, optional*) – Planet radius in km
- **h2o_lines** (*{'HITEMP','EXOMOL'}, optional*) – Either 'HITEMP' or 'EXOMOL'–the line list from which H2O absorption should be sourced
- **num_cpus** (*int, optional*) – The number of CPUs to use

- **cloudfunc** (*function, optional*) – A routine which takes pressure, temperature, and cloud water content as arguments, and returns keyword arguments to be unpacked into calc_flux_transm. If not specified, *basicclouds* will be used.
- **smooth** (*bool, optional*) – Whether or not to smooth humidity and cloud columns. As of Nov 12, 2021, it is recommended that you use smooth=True for well-behaved spectra. This is a conservative smoothing operation, meaning the water and cloud column mass should be conserved–what this does is move some water from the water-rich layers into the layers directly above and below.
- **smoothweight** (*float, optional*) – The fraction of the water in a layer that should be retained during smoothing. A higher value means the smoothing is less severe. 0.95 is probably the upper limit for well-behaved spectra.

Returns pRT Atmosphere object, Wavelength in microns, array of all transit columns with shape (ntimes,nterm,nfreq), where nterm is the number of terminator columns (time-varying), and nfreq is the number of frequencies in the spectrum, array of lon-lat coordinates for each transit specturm, array of spatial weights for each column with the shape (ntimes,nterm) (for averaging), and the spatially-averaged transit spectrum, with shape (ntimes,nfreq). Transit radius is in km.

Return type Atmosphere,numpy.ndarray,numpy.ndarray,numpy.darray,numpy.ndarray

Created by Adiv Paradise

This API was written with Python 3 in mind, but should work with Python 2 and outdated versions of NumPy.

1.4 Requirements

- Python (including development libraries, e.g. python-dev or python3.9-dev on Ubuntu–if using anaconda, these should already be included in your installation)
- numpy
- scipy (only needed for additional utilities, postprocessor)
- matplotlib (only needed for additional utilities)
- GNU C (gcc/g++) and Fortran (gfortran) compilers (development headers must be present)
- (optionally) Other compilers whose use you prefer for the model itself
- (optionally) MPI libraries for those compilers

1.4.1 Compatibility

- Linux (tested on Ubuntu 18.04, CentOS 6.10): **Yes**
- Google Colaboratory: Yes (note that OpenMPI support on Colaboratory is limited due to automatic root privileges; look up how to run OpenMPI executables with root permissions and note that this is not recommended)
- Windows 10: Yes, via Windows Subsystem for Linux
- Mac OS X: Yes, requires Xcode and developer tools, and OpenMPI support requires that Fortran-compatible libraries be built. Tested on Mac OS X Catalina and Big Sur (with MacPorts, GCC10, OpenMPI, and Anaconda3), Apple M1 compatibility has not been tested.

1.5 Optional Requirements

- netCDF4 (for netCDF support)
- h5py (for HDF5 support)

1.6 New in 3.2:

- Experimental integration with petitRADTRANS to compute transit spectra and reflectance spectra, including maps and true-colour images (use at your own risk)
- Ability to specify general keplerian orbits, with high eccentricity, using a revamped orbit code for higher accuracy
- Orbital elements now included in standard output
- Numerous bugfixes

1.7 New in 3.0:

- ExoPlaSim no longer depends on X11 libraries for installation and compilation!
- Revamped postprocessor no longer depends on NetCDF-C libraries, and supports additional output formats (including netCDF, HDF5, NumPy archives, and archives of CSV files).
- GCC and gfortran support through GCC 10.
- Improved cross-platform compatibility
- Numerous bugfixes

1.8 Installation

```
pip install exoplasim
```

OR:

```
python setup.py install
```

If you know you will want to use NetCDF or HDF5 output formats, you can install their dependencies at install-time:

```
pip install exoplasim[HDF5]
```

OR:

```
pip install exoplasim[netCDF4]
```

OR:

```
pip install exoplasim[netCDF4,HDF5]
```

The first time you import the module and try to create a model after either installing or updating, ExoPlaSim will run a configuration script, write the install directory into its source code, and compile the `pyfft` library.

You may also configure and compile the model manually if you wish to not use the Python API, by entering the exoplasim/ directory and running first configure.sh, then compile.sh (compilation flags are shown by running `./compile.sh -h`).

1.9 Most Common Error Modes

There are 3 major ways in which ExoPlaSim can crash. One is related to installation, one is related to model compilation/configuration, and one is related to numerical stability.

If in the run folder, diagnostic files are produced that appear to have made it all the way to the end of the year (there is a summary tag giving time elapsed and that sort of thing), then the problem is likely with the postprocessor. It is likely that the error output will be informative; if it is not clear how to resolve, please let me (the developer) know.

If the postprocessor itself is *not* the problem, then it's likely you somehow passed incorrect output codes to the postprocessor. This is the most common scenario for postprocessor-related crashes. Check your inputs for any errors. In particular, note that climatology outputs are not available if storm climatology was not enabled.

If things crashed and burned immediately, it's likely a configuration problem. Check to make sure you aren't using a restart file from a run that used a different resolution, or stellar spectrum files that aren't formatted correctly (use the *`makestellarspec`* utility to format Phoenix spectra for ExoPlaSim), or boundary condition `.sra` files that aren't properly-formatted.

If things were fine until they weren't, then it's likely ExoPlaSim encountered a numerical instability of some kind. Some of these are physical (e.g. you ran a model at a thousand times Earth's insolation, and the oceans boiled, or the model was too cold and the physics broke), while some are not (something happened to violate the CFL condition for the given timestep, or an unphysical oscillation wasn't damped properly by the dynamical core and it grew exponentially). If this happens, either try a model configuration that is more physically reasonable, or if the problem appears not to have been physical, try reducing the timestep or increasing hyperdiffusion. Sometimes it also works to slightly adjust a model parameter such as surface pressure by a fraction of a percent or less–just enough to nudge the model out of whatever chaotic local minimum it ran into, but not enough to qualitatively change the resulting climate.

New in ExoPlaSim 3.0.0, there is a "crash-tolerant" run mode. With this mode enabled, a runtime crash will result in rewinding 10 years and resuming. This deals with many of the most frustrating problems related to numerical instability. However, due to the potential for infinite loops, this is only recommended for advanced users.

1.10 PlaSim Documentation

Original PlaSim documentation is available in the exoplasim/docs/ folder.

1.11 Usage

To use the ExoPlaSim Python API, you must import the module, create a Model or one of its subclasses, call its configure method and/or modify method, and then run it.

An IPython notebook is included with ExoPlaSim; which demonstrates basic usage. It can be found in the ExoPlaSim installation directory, or downloaded directly here.

Basic example::

```
import exoplasim as exo
mymodel = exo.Model(workdir="mymodel_testrun",modelname="mymodel",resolution="T21",
↪layers=10,ncpus=8)
mymodel.configure()
mymodel.exportcfg()
mymodel.run(years=100,crashifbroken=True)
mymodel.finalize("mymodel_output")
```

In this example, we initialize a model that will run in the directory "mymodel_testrun", and has the name "mymodel", which will be used to label output and error logs. The model has T21 resolution, or 32x64, 10 layers, and will run on 8 CPUs. By default, the compiler will use 8-byte precision. 4-byte may run slightly faster, but possibly at the cost of reduced stability. If there are machine-specific optimization flags you would like to use when compiling, you may specify them as a string to the optimization argument, e.g. `optimization='mavx'`. ExoPlaSim will check to see if an appropriate executable has already been created, and if not (or if flags indicating special compiler behavior such as debug=True or an optimization flag are set) it will compile one. We then configure the model with all the default parameter choices, which means we will get a model of Earth. We then export the model configurations to a `.cfg` file (named automatically after the model), which will allow the model configuration to be recreated exactly by other users. We run the model for 100 years, with error-handling enabled. Finally, we tell the model to clean up after itself. It will take the most recent output files and rename them after the model name we chose, and delete all the intermediate output and configuration files.

1.12 A Note on NetCDF and the (deprecated) Burn7 Postprocessor

As of ExoPlaSim 3.0.0, `burn7` is deprecated. It is only available via the `exoplasim-legacy` package.

PYTHON MODULE INDEX

e

INDEX

N

O

P

R

S

T

U

W

X