

---

# **Powderday Documentation**

*Release 0.0*

**Desika Narayanan**

**Aug 15, 2017**



---

# Contents

---

<b>1</b>	<b>Getting Started</b>	<b>3</b>
1.1	Overview of Requirements . . . . .	3
1.2	Installation . . . . .	4
1.3	Troubleshooting your Installation . . . . .	7
<b>2</b>	<b>Quickstart</b>	<b>9</b>
2.1	Gadget/Gizmo . . . . .	9
2.2	Gasoline/Changa . . . . .	10
<b>3</b>	<b>Detailed Description of Parameters</b>	<b>11</b>
3.1	parameters_master . . . . .	11
3.2	parameters_model . . . . .	14
<b>4</b>	<b>Detailed Algorithm Description</b>	<b>15</b>
<b>5</b>	<b>Convenience Scripts</b>	<b>17</b>
<b>6</b>	<b>Referencing</b>	<b>19</b>
<b>7</b>	<b>Troubleshooting and Known Issues</b>	<b>21</b>
7.1	1. yt compilation complains with something like: “Error compiling Cython file” . . . . .	21
7.2	2. Python-fsps compilation returns errors along the lines of . . . . .	21
7.3	3. When running pd via a SLURM scheduler, you get the error when importing fsps . . . . .	21
7.4	4. If in powderday you start getting memory errors on the Pool process call that look like this: . . . . .	22
7.5	5. Assertion Error in SED generation related to zmet being out of range . . . . .	22
<b>8</b>	<b>Getting Help</b>	<b>25</b>
<b>9</b>	<b>Indices and tables</b>	<b>27</b>





`powderday` is a dust radiative transfer package designed to interface with galaxy formation simulations in order to produce realistic spectral energy distributions and images. The code utilizes `'fsps` <<https://code.google.com/p/fsps/source/checkout>>' (and its python hooks, `python-fsps`) for stellar SEDs, and `Hyperion` for dust radiative transfer. Threaded throughout is `yt`.

The code has two principle aims - to be flexible (and extremely modular), and to be easy to use. The package is written in python, though makes use of fortran, C and cython.

As of now, the code works principally for Gadget, Gizmo and other variants. Support for Gasoline, RAMSES, ART and Enzo are planned however. The current timeline is for TIPSy front ends to be built over Spring 2015, with AMR support to follow

Finally, this code benefits from the contributions, either directly to this software, or indirectly by contributions to dependency software, by many astrophysicists, including Matt Turk, Tom Robitaille, Robert Thompson, Charlie Conroy, Dan Forman-Mackey and Phil Hopkins.

---

**Important: Disclaimer:**

While the developers have made every effort to ensure that the code is bug-free, bugs invariably find their way in. The developers are not responsible for incorrect results, arising either from inherent bugs in the code, or mistakes in usage. Any questions regarding the code or usage should be sent to the mailing list.

---

Contents:



### Overview of Requirements

- **python2.x**
  - numpy (any version except 1.10.\*)
  - scipy
  - astropy
  - Atpy
  - h5py
  - ipdb
- **compilers**
  - gcc
  - gfortran
- **Additional Packages (with Instructions Below)**
  - mercurial <<http://mercurial.selenic.com/>>
  - git <<http://git-scm.com/>>
  - powderday <<http://bitbucket.org/desika/powderday>>
  - yt <<http://yt-project.org>>
  - FSFS <<https://code.google.com/p/fsfs/source/checkout>>
  - python-fsps <<http://dan.iel.fm/python-fsps/current/>>
  - Hyperion <<http://www.hyperion-rt.org/>>
  - Hyperion Dust Files <<http://docs.hyperion-rt.org/en/stable/dust/dust.html>>

## Installation

### All in One Installer

The first stop is to try the all-in-one installer for powderday. This is available at the main download page <<https://bitbucket.org/desika/powderday/downloads>> from the Bitbucket repository. This is a bash script that will append relevant path names to your .bashrc, as well as attempt to go package-by-package and install everything. The plus side is that if it works, it's super easy. The downside is that if a package installation fails, you might not catch it because the bash script will go on to the next package. This installer assume gnu C and Fortran compilers.

Something that has worked for a number of users is to download the all-in-one installer, and then copy and paste the instructions for each package one at a time. This can alert the user to package failures.

### Manual Installation

What follows is a self-contained installation manual, though for problematic installs of any of the sub packages, it's definitely best to visit the main docs on the main software site (which are always linked below in each subsection).

#### python

powderday currently only works with python 2.x. The code was developed on, and principally tested with python 2.7.

#### mercurial

You'll need this to clone powderday using mercurial (hg). If you don't want to install mercurial, then first install yt (before powderday) - yt ships with its own hg which you can optionally use.

#### powderday

Simply clone the latest and greatest from the repo:

```
>hg clone https://desika@bitbucket.org/desika/powderday
```

And that's it! Once it's cloned, there's no subsidiary installation commands.

#### yt

powderday has yt threaded throughout, and thus needs the software to be installed to function. If you already have yt installed, be sure that it's yt 3.x (i.e at least the 'stable' branch).

If you don't have yt already installed, it's super easy! There are many ways to do this (as directed on the yt project website. We require the standalone installer script for the Development version. This will ensure that bugs caught in or bleeding-edge features of yt are included.

#### fsps

fsps can be checked out with:



```
> git clone https://github.com/cconroy20/fsps
```

and directions to the installation are in the [Manual](#)

It's likely going to be necessary downstream when installing `python-fsps` to have the `-fPIC` flags set in `fsps` when making. So, in the Makefile of `fsps`, set:

```
>F90FLAGS = -O -cpp -fPIC
```

Additionally, at this time `powderday` doesn't work with the default MIST Isochrones. To fix this, you'll need to edit `sps_vars.f90` in `fsps` to look like:

```
!-----set the isochrone library-----!
#define MIST 0
!Padova models circa 2008
#define PADOVA 1
#define PARSEC 0
#define BASTI 0
#define GENEVA 0
```

Finally, the `SPS_HOME` variable must be set in your environment to point to the `FSPS/src` directory. For example, if your environment is `bash`, in your `.bashrc` set something along the lines of:

```
>export SPS_HOME=/Users/desika/fsps/
```

## python-fsps

`powderday` depends on python hooks for `fsps` written by Daniel Foreman-Mackey and others called `python-fsps`. There are a few ways to install it. Perhaps the easiest is via a pip installer:

```
>pip install fsps
```

Though you could also install the development version:

```
>git clone https://github.com/dfm/python-fsps.git
>cd python-fsps
>python setup.py install
```

You can test the installation by opening python and typing:

```
>import fsps
```

## Hyperion

`Hyperion` is the main work horse of `powderday`. The full directions for installation are well-described on the main `Hyperion` website. Here, we summarize the installation which should get most users through without any real difficulty.

1. First download the tarball and unpack it.:

```
>tar -xzvf hyperion.xxx
>cd hyperion.xxx
```

2. Install the fortran dependencies:

```
>cd deps/fortran
>python install.py <prefix>
```

where <prefix> is where you want the libraries to be installed. To avoid conflicts with other packages, I usually install somewhere like:

```
>python install.py /usr/local/hyperion
```

as suggested by the [Hyperion](#) docs. Ensure that the following commands return something sensible:

```
>which mpif90
>which h5fc
```

if not, your path probably needs to include wherever the <prefix> directory pointed to.

3. Install any remaining python dependencies. These are listed [here](#)
4. Install [Hyperion](#) itself. To do this:

```
>cd hyperion.xxx
>python setup.py install
```

or:

```
>python setup.py install --user
```

if you don't have root access. At this point:

```
>import hyperion
```

from within python should work, and typing:

```
>hyperion
```

at the command line should return something along the lines of:

```
>usage: hyperion [-h] [-f] [-m n_cores] input output
>hyperion: error: too few arguments
```

if not, check the the path that is near one of the last lines of the setup.py installation (that is something associated with the number 755) and make sure it's in your path. It's most likely to be a python binaries directory.

You then have to install the Fortran Binaries:

```
>./configure --prefix=prefix
>make
>make install
```

where the prefix is wherever you installed the Fortran libraries before. Make sure this works by typing at the command line:

```
>hyperion_sph
```

which should return something like:

```
>Usage: hyperion_sph [-f] input_file output_file

.. _Hyperion_dust:
```

## Hyperion Dust Files

Unless you've written your own dust files, you will likely want to use the pre-compiled dust files developed by Tom Robitaille (though don't ship with `Hyperion` due to their size). To install these download them here: <http://docs.hyperion-rt.org/en/stable/dust/dust.html>. Then to install:

```
>tar xvzf hyperion-dust-xxx.tar.gz
>cd hyperion-dust-0.1.0
>python setup.py build_dust
```

If you want to use the PAH model in `powderday`, you'll additionally need these files in the same dust directory. To download, click on the link, then click 'raw' on the right side of each page.

1. <https://github.com/hyperion-rt/paper-galaxy-rt-model/blob/master/dust/big.hdf5>
2. <https://github.com/hyperion-rt/paper-galaxy-rt-model/blob/master/dust/vsg.hdf5>
3. <https://github.com/hyperion-rt/paper-galaxy-rt-model/blob/master/dust/usg.hdf5>

Please note the caveat that the PAH files are generated using some approximations described in Robitaille et al., and we encourage the user of these PAH files to read this paper, especially section 3.4.2.

## Troubleshooting your Installation

### python-fsps installation issues

#### 1. python-fsps can't find f2py

`f2py` is a numpy package that is sometimes named `f2py2.7` by numpy. At the same time, `python-fsps` expects it to be called `f2py` (as it sometimes is; for example in Anaconda). So, you might need to locate `f2py` (it ships with `yt`, so if you for example use the `yt` python) you need to link the following files:

```
>cd /Users/desika/yt-x86_64/bin
>ln -s f2py2.7 f2py
```

and:

```
>cd /Users/desika/yt-x86_64/lib/python2.7/site-packages
>ln -s numpy/f2py/ f2py
```

This should hopefully fix it.

#### 2. Issues with 'f2py' in the python-fsps installation:

Numpy has made some changes to `f2py` in the 1.10.x version of numpy. The easiest fix is to use a non 1.10.\* version of numpy (thanks to Ben Johnson for finding this).

3. `python-fsps` has mysterious installation failures. Often this has to do with a bad `FSPS` compilation. Even if it seems like `FSPS` has compiled, it may not actually execute properly if the correct compilers aren't set in the `MakeFile`. Thanks to Ena Choi for pointing this one out.



## CHAPTER 2

---

### Quickstart

---

In the examples subdirectory of the `powderday` root directory are some example snapshots for different hydro codes supported thusfar. This will likely change over time as the code evolves and parameter files change. Also, eventually the examples will migrate to the `agora` project snapshots.

For each example file there should be two parameter files that will be reasonable for the associated snapshot, though you'll need to edit the hard linked directories that specify where (e.g.) dust files are and output should go. To run, type (in the `powderday`) source directory:

```
>python pd_front_end.py <example directory> <parameters_master_file>
<parameters_model_file>
```

Note - the `.py` extensions on the parameter files need to be left off. So, as an example, you might type the following to run a gadget example snapshot:

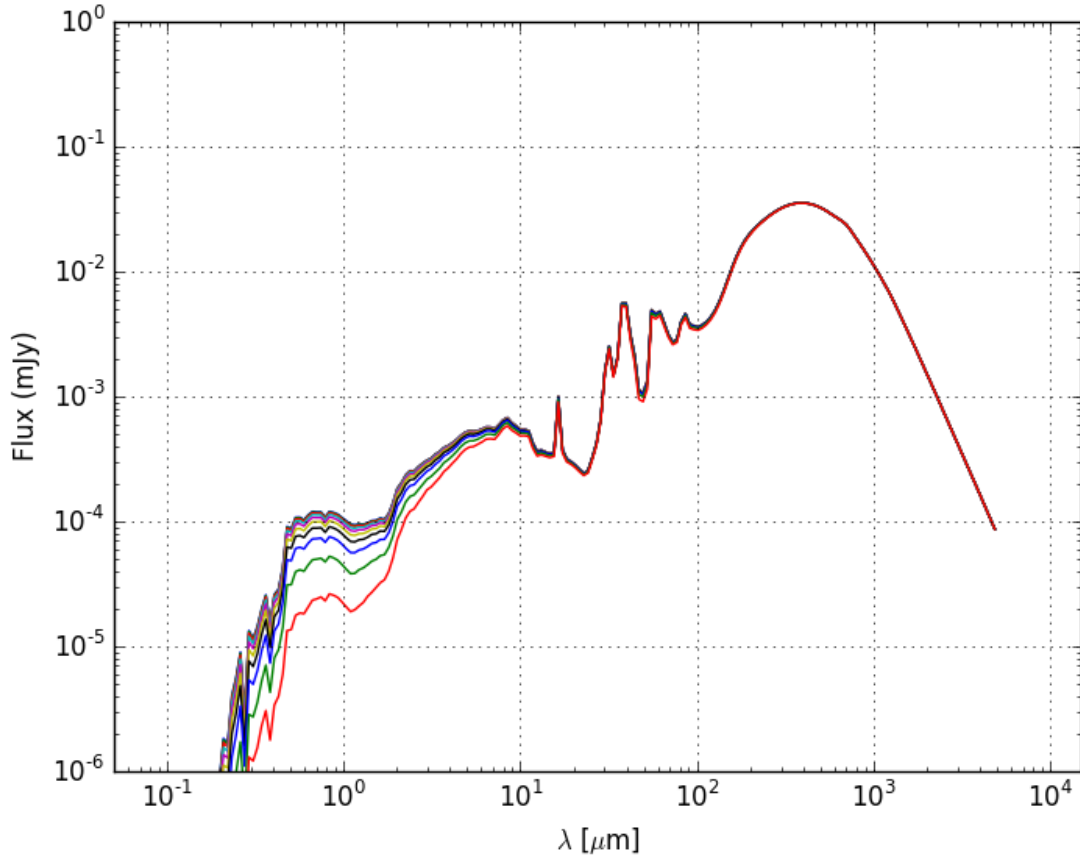
```
>python pd_front_end.py examples/gadget parameters_master parameters_model
```

### Gadget/Gizmo

The snapshot is an idealized disk galaxy that can be downloaded here (13 MB download):

[https://www.dropbox.com/s/vy6qu43056q5ii5/snapshot\\_200.hdf5?dl=0](https://www.dropbox.com/s/vy6qu43056q5ii5/snapshot_200.hdf5?dl=0)

The SED (placed at  $z=4$  with a Planck13 cosmology) looks like:



and an example plotting code can be found in the convenience subdirectory of the `powderday` root directory.

## Gasoline/Changa

---

## Detailed Description of Parameters

---

There are two parameters files - `parameters_master` and `parameters_model`. The distinction between where a parameter goes is somewhat arbitrary, but is based on the following. `Parameters_master` tends to contain parameters that will likely be shared amongst all snapshots in a given galaxy run, while `parameters_model` are parameters that might change from run to run (like the snapshot name or the galaxy center).

The parameters files are (somewhat non-traditionally) written as python code files because this enables you to embed little snippets in them that may be particular to your galaxy run. For example, for gadget snapshot naming conventions, it can be useful to have a snippet along the lines of:

```
Gadget_snap_num = 20
if Gadget_snap_num < 10:
    snapnum_str = '00'+str(Gadget_snap_num)
elif Gadget_snap_num >= 10 and Gadget_snap_num <100:
    snapnum_str = '0'+str(Gadget_snap_num)
else:
    snapnum_str = str(Gadget_snap_num)

Gadget_snap_name = 'snapshot_'+snapnum_str+'.hdf5'
```

That puts the correct number of 0's in front of the snapshot name.

## parameters\_master

### Resolution Keywords

**oref** Over Refinement of the Octree - 1 means each data holding cell (a False) gets split into 8 one more time. Very heavy on the memory. Default is 0.

**n\_ref** Threshold number of particles to refine over. When `nparticles > n_ref` the octree refines further. Default is 64.

**zoom\_box\_len** Side length to zoom in on. Is +/- zoom\_box\_len from the center. Units are proper kpc. So, a grid centered on [0,0,0] with zoom\_box\_len = 200 would extend from [-200,200] kpc in physical units at the redshift of the simulation.

**bbox\_lim** Initial bounding box of grid for SPH simulations (+/- bbox\_lim). Units are kpc. This must encompass all of the particles in a simulation currently. This just has to be a big number, but you want to be careful of making *too* large as precision limitations only allow for up to 20 levels of refinement.

## Parallelization

**n\_processes** Number of MPI processes to run the radiative transfer on. Note, the stellar population synthesis will only run on as many processors as are on a core since its parallelization is pool.map (not MPI)

## RT Information

For all photon counts, a decent rule of thumb is 10-100x the number of grid cells that you have, though of course you should check the convergence properties of your simulation.

**n\_photons\_initial** Number of photons to use in main iterations (for the whole grid) for specific energy and temperature calculations.

**n\_photons\_imaging** Number of photons to use for the SED/image calculation

**n\_photons\_raytracing\_sources** If raytracing is set (which is the default hard-coded into the code), number of raytracing photons to use for source emission.

**n\_photons\_raytracing\_dust** Similar to n\_photons\_raytracing\_sources but for dust emission.

## Dust Information

**dustdir** String. Path to where your dust files are. String format - (e.g. '/home/desika/hyperion-dust-0.1.0/dust\_files/')

**dustfile** String. Name of your main dust file. String format - (e.g. 'd03\_3.1\_6.0\_A.hdf5')

**PAH** Boolean - True means use model for PAHs, False means don't.

**dusttometals\_ratio** Dust mass to metals mass ratio

**enforce\_energy\_range** Boolean. False ensures energy conservation. But the emissivities may not be strictly correct if the energy in a cell is out of range of the emissivities. True modifies the energy in the simulation, but ensures that the emissivities are consistent with the energy. See: <[http://docs.hyperion-rt.org/en/latest/api/hyperion.model.Model.html?highlight=enforce#hyperion.model.Model.set\\_enforce\\_energy\\_range](http://docs.hyperion-rt.org/en/latest/api/hyperion.model.Model.html?highlight=enforce#hyperion.model.Model.set_enforce_energy_range)>

## Hydro Code Units

Currently these are actually not used in `powderday`). They remain in the parameters file as a placeholder though as we may need them as an over-ride if we find some HDF5 files don't contain this information.

**unit\_mass** Mass code units for galaxy simulation. Units: Msun/h

**unit\_length** Length code unit for galaxy simulation. Units: kpc/h



**unit\_age** Stellar age units. Units: Gyr/h

**unit\_velocity** Velocity code unit for galaxy simulation. Units: cm/s

## Stellar SEDs Info

**Force\_Binning** Boolean. True means force binning of the stellar SEDs (in bins of age and metallicity). False means don't. False results in an exact solution since the stellar SEDs are individually represented (as opposed to broken up into bins). This said, this can be very slow to run, and extremely hard on the memory.

**COSMOFLAG** Boolean. True means this is a cosmological simulation, False means idealized galaxy simulation.

**imf\_type** IMF parameter for stellar pops calculations.

0. Salpeter
1. Chabrier
2. Kroupa
3. Van Dokkum
4. Dave

Though note options 3 and 4 are currently not supported.

**pagb** Weight given to post AGB stars. 1 is the default.

**CF\_on** Boolean. If set to True, then enables the Charlot & Fall birthcloud models for all stars with age younger than `birth_cloud_clearing_age`.

**birth\_cloud\_clearing\_age** Stars with age < `birth_cloud_clearing_age` have Charlot & Fall birthclouds (if `CF_on == True`). Meaningless if `CF_on == False`. Units: Gyr.

**Z\_init** Forced metallicity increase in the newstar particles. Useful for idealized galaxy simulations where the stars can form out of pristine gas. Units are absolute (so 0.02 = Solar). Setting to 0 (default) means that you use the stellar metallicities as they come in the simulation (i.e. for Cosmological simulations).

**disk\_stars\_age** Age in Gyr of disk stars for idealized simulations. Meaningless for cosmological simulations. Note, if this is  $\leq 7$ , then these will live in Charlot & Fall birthclouds (if `CF_on = True`).

Note, for Gadget simulations, stars are divided into newstars, disk stars and bulge stars. For Topsy outputs, the stars initialized with the simulation are auto-detected by their nonsensical ages, and assigned as disk stars. So, if there are stars initialized with your Topsy simulation, assign their ages (and metallicities below) as disk stars.

**bulge\_stars\_age** As `disk_stars_age` but for bulge stars.

**disk\_stars\_metals** Metallicity of disk stars in FSPS metallicity units. See last page of FSPS manual for numbers. (e.g. 20 = Solar for Padova + BaSeL tracks). Meaningless for cosmological simulations.

**bulge\_stars\_metals** As `disk_stars_metals` but for bulge stars.

**N\_STELLAR\_AGE\_BINS** Number of bins to bin the stellar ages in (boundaries are the oldest and youngest star particles; linear bins in  $\log(\text{age})$ ).

**N\_MASS\_BINS** Meaningless parameter; place holder for future code additions.

**metallicity\_legend** String. Location of the metallicity maps in FSPS for the stellar libraries you use. Currently Padova2007 is the default (hard coded into `powderday`), so this should point to something like: `"/Users/desika/fsps/ISOCHRONES/Padova/Padova2007/zlegend_basel.dat"`

## Images and SED Parameters

**NTHETA** Number of polar angles to view galaxy at

## GRID INFORMATION

**MANUAL\_CENTERING** Boolean. False means the simulation automatically centers on the cell with the highest gas density peak. True means you center on `x_cent,y_cent,z_cent` as given in `parameters_model`.

## DEBUGGING

You should probably never touch any of these.

## parameters\_model

**Gadget\_snap\_name** String - currently the snapshot name of your galaxy run. (Naming will change as other front ends built).

**hydro\_dir** Location of snapshots

**PD\_output\_dir** String - location of where `powderday` output files should go.

**Auto\_TF\_file** String - name of the TF logical file to be written (doesn't need a path - will go into `PD_output_dir`)

**Auto\_dustdens\_file** String - name of the dust density ascii file to be written (doesn't need a path - will go into `PD_output_dir`)

**inputfile** String - name of the input HDF5 (rtin) file for `powderday` to write before radiative transfer begins.

**outputfile** String - name of the output HDF5 (rtout) file after radiative transfer

**x\_cent** Location in grid coordinates of the x-coordinate of the center of your galaxy. Only pertinent if `MANUAL_CENTERING==True`. Otherwise ignored by `powderday`.

**y\_cent** As `x_cent` but for the y-coordinate

**z\_cent** As `x_cent` but for the z-coordinate.

## CHAPTER 4

---

### Detailed Algorithm Description

---



---

## Convenience Scripts

---

The convenience scripts tend to live (where else) in the convenience subdirectory of the `powderday` root directory. There's a brief description of them here.

**sed\_plot** Enter a redshift and rtout file name (in the editable header) and it will plot out an SED for all viewing angles in Jy. You'll want to manually monkey with the range, etc. This is mostly just serves as an example.



## CHAPTER 6

---

### Referencing

---

If you use `powderday`, we would definitely appreciate a hat tip toward the following papers that describe software that we depend on. We suggest either the following paragraph, or some variant.

This paper makes use of the `powderday` radiative transfer software, which is built on `yt` (Turk et al. 2011), `Hyperion` (Robitaille 2011), and `FSPS` (Conroy & Gunn, 2010).

References:

1. `yt`: Turk et al. 2011
2. `Hyperion` : Robitaille, 2011
3. `fsps`: Conroy & Gunn, 2010





---

## Troubleshooting and Known Issues

---

### 1. yt compilation complains with something like: “Error compiling Cython file”

Answer - try running:

```
pip install -U Cython
```

### 2. Python-fsps compilation returns errors along the lines of

Error:

```
compiling Fortran sources
Fortran f77 compiler: /usr/bin/gfortran -Wall -ffixed-form -fno-second-underscore -
↳fPIC -O3 -funroll-loops
Fortran f90 compiler: /usr/bin/gfortran -fPIC -fPIC -O3 -funroll-loops
```

Try re-compiling with the flag:

```
-fPIC
```

### 3. When running pd via a SLURM scheduler, you get the error when importing fsps

Error:

```
build/bdist.linux-x86_64/egg/fsps/__init__.py in <module>()
ImportError: Your FSPS version does not seem to be under git version control. FSPS is
↳available on github at https://github.com/cconroy20/fsps and should be cloned from
↳there
```

Comment out the lines in `python-fsps/fsps/__init__.py` surrounding the checking of githashes. (h/t to [Ena Choi](#) for uncovering this one).

#### 4. If in powderday you start getting memory errors on the Pool process call that look like this:

Error:

```
calculating the SEDs for 42 bins
Traceback (most recent call last):
File "/ufrc/narayanan/desika.narayanan/pd/pd_front_end.py", line 139, in <module>
    m=add_binned_seds(df_nu,stars_list,diskstars_list,bulgestars_list,m)
File "/ufrc/narayanan/desika.narayanan/pd/source_creation.py", line 244, in add_
↳binned_seds
    #of star particles that go in every [wz,wa,wm]
File "/ufrc/narayanan/desika.narayanan/pd/SED_gen.py", line 225, in allstars_sed_gen
    p = Pool(processes = cfg.par.n_processes)
File "/ufrc/narayanan/desika.narayanan/miniconda2/lib/python2.7/multiprocessing/
↳init__.py", line 232, in Pool
    return Pool(processes, initializer, initargs, maxtasksperchild)
File "/ufrc/narayanan/desika.narayanan/miniconda2/lib/python2.7/multiprocessing/pool.
↳py", line 159, in __init__
    self._repopulate_pool()
File "/ufrc/narayanan/desika.narayanan/miniconda2/lib/python2.7/multiprocessing/pool.
↳py", line 223, in _repopulate_pool
    w.start()
File "/ufrc/narayanan/desika.narayanan/miniconda2/lib/python2.7/multiprocessing/
↳process.py", line 130, in start
    self._popen = Popen(self)
File "/ufrc/narayanan/desika.narayanan/miniconda2/lib/python2.7/multiprocessing/
↳forking.py", line 121, in __init__
    self.pid = os.fork()
OSError: [Errno 12] Cannot allocate memory
```

Try reducing the number of `n_processes` in the `parameters_master.py` file

#### 5. Assertion Error in SED generation related to `zmet` being out of range

Error:

```
assigning stars to SED bins
Running SPS for Binned SEDs
calculating the SEDs for 105 bins
Traceback (most recent call last):
File "pd_front_end.py", line 134, in <module>
m=add_binned_seds(df_nu,stars_list,diskstars_list,bulgestars_list,m)
File "source_creation.py", line 300, in add_binned_seds
    binned_stellar_nu,binned_stellar_fnu_has_stellar_mass,disk_fnu,bulge_fnu = sg.
↳allstars_sed_gen(sed_bins_list_has_stellar_mass,diskstars_list,bulgestars_list)
File "SED_gen.py", line 216, in allstars_sed_gen
```

```
    add_neb_emission = cfg.par.add_neb_emission, add_agb_dust_model=cfg.par.add_agb_
↪dust_model)
File "build/bdist.linux-x86_64/egg/fsps/fsps.py", line 468, in __init__
    self.params[k] = kwargs.pop(k, v)
File "build/bdist.linux-x86_64/egg/fsps/fsps.py", line 1093, in __setitem__
    self.check_params()
File "build/bdist.linux-x86_64/egg/fsps/fsps.py", line 1067, in check_params
    "zmet={0} out of range [1, {1}]" .format(self._params["zmet"], NZ)
AssertionError: zmet=20 out of range [1, 12]
```

Recompile fsps with the libraries set to Padova (and not MIST) in sps\_vars.f90. Also recompile python-fsps



## CHAPTER 8

---

### Getting Help

---

powderday has an email list serve on google groups. We encourage you to visit the group at <http://groups.google.com/group/powderday> and post any questions you have. You can also search the email archives there.

You can also directly email the group at [codename] AT googlegroups.com. For the benefit of future archive readers, if you could please include a descriptive subject line of the issue you're having, that would very helpful.

If a run crashes, it can oftentimes be very useful to post your parameters files as well as include which hash you're working off of. You can recover the hash by typing:

```
>hg id
```

And similarly, if you want a neat way to create a link out of your parameters files, you can type:

```
>yt pastebin parameters_master.py  
>yt pastebin parmaeters_model.py
```

and two separate links will be created with the parameters files.



## CHAPTER 9

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`