ares Documentation

Release 0.1

Jordan Mirocha

September 10, 2015

Contents

The Accelerated Reionization Era Simulations (*ares*) code was designed to rapidly generate models for the global 21-cm signal. It can also be used as a 1-D radiative transfer code, stand-alone non-equilibrium chemistry solver, or global radiation background calculator.

A few papers on how it works:

- 1-D radiative transfer: Mirocha et al. (2012).
- Uniform backgrounds & global 21-cm signal: Mirocha (2014).

This documentation is very much a work in progress. Feel free to email me if you find gaps or errors.

CHAPTER 1

Quick-Start

To make sure everything is working, a quick test is to generate a realization of the global 21-cm signal using all default parameter values:

```
import ares

sim = ares.simulations.Global21cm()
sim.run()

anl = ares.analysis.Global21cm(sim)
ax = anl.GlobalSignature()
```

See `example_21cm_simple` in Examples for a more thorough introduction to this type of calculation.

CHAPTER 2

Contents

2.1 Installation

ares depends on:

- numpy
- scipy
- matplotlib

and optionally:

- python-progressbar
- hmf (halo mass function calculator written by Stephen Murray)
- mpi4py
- h5py

If you have mercurial installed, you can clone *ares* and its entire revision history via:

```
hg clone https://bitbucket.org/mirochaj/ares ares
cd ares
python setup.py install
```

If you do not have mercurial installed, and would rather just grab a tarball of the most recent version, select the Download repository option on bitbucket.

Once you've got the code, you'll need to set an environment variable which points to the *ares* install directory, e.g. (in bash)

```
export ARES=/users/<yourusername>/ares
```

A few lookup tables will be downloaded to `$ARES/input` automatically.

2.1.1 Help

If you encounter problems with installation or running simple scripts, first check the Troubleshooting page in the documentation to see if you're dealing with a common problem. If you don't find your problem listed there, please let me know!

2.2 Examples

See below.

2.2.1 Basics

- **Reionization & Global 21-cm Signal**
  - `example_21cm_simple`
  - `example_21cm_multipop`
- **1-D Radiative Transfer**
  - `example_rt06_1`
  - `example_rt06_2`
- **Uniform Radiation Backgrounds**
  - `example_cuvb`
  - `example_cxrb`

2.2.2 Advanced

Coming soon. For now, peruse `ares/tests/dTb` for lots of different 21-cm examples and `ares/tests/rt1d` for a few more 1-D radiative transfer problems.

2.2.3 Under the hood

- `example_stellar_pop`
- `example_bh_pop`
- `example_igm`
- `example_chem`

2.2.4 Surveying Parameter Space

- `example_grid_I`
- `example_grid_II`
- `example_grid_analysis`
- `example_mcmc_I`
- `example_mcmc_analysis`

2.2.5 Post-Processing Cosmological Simulations

Stay tuned.

2.3 Parameters

We use keyword arguments to pass parameters around to various *ares* routines. A complete listing of parameters and their default values can be found in `ares.util.SetDefaultParameterValues.py`.

Here, we'll provide a brief description of each parameter.

- `params_grid`
- `params_physics`
- `params_sources`
- `params_populations`
- `params_spectrum`
- `params_inference`
- `params_hmf`
- `params_control`
- `params_cosmology`

2.3.1 Custom Defaults

To adapt the defaults to your liking *without* modifying the source code (all defaults set in `ares.util.SetDefaultParameterValues.py`), open the file:

```
$HOME/.ares/defaults.py
```

which by default contains nothing:

```
pf = {}
```

To craft your own set of defaults, simply add elements to the `pf` dictionary. For example, if you want to use a default star-formation efficiency of 5% rather than 10%, open `$HOME/.ares/defaults.py` and do:

```
pf = {'fstar': 0.05}
```

That's it! Elements of `pf` will override the defaults listed in `ares.util.SetDefaultParameterValues.py` at run-time.

Alternatively, within a python script you can modify defaults by doing

```
import ares
ares.rcParams['fstar'] = 0.05
```

This is similar to how things work in matplotlib (with the `matplotlibrc` file and `matplotlib.rcParams` variable).

2.3.2 Custom Axis-Labels

You can do the analogous thing for axis labels (all defaults set in `ares.util.Aesthetics.py`). Open the file:

```
$HOME/.ares/labels.py
```

which by default contains nothing:

```
pf = {}
```

If you wanted to change the default axis label for the 21-cm brightness temperature, from δT_b (mK) to T_b, you would do:

```
pf = {'dTb': r'$T_b$'}
```

This change will automatically propagate to all built-in analysis routines.

2.4 Field Listing

The most fundamental quantities associated with any calculation done in ares are the gas density, species fractions and the gas temperature.

2.4.1 Species Fractions

Our naming convention is to denote ions using their chemical symbol (in lower-case), followed by the ionization state, separated by an underscore. Rather than denoting the ionization state with roman numerals, we simply use integers. For example, neutral hydrogen is *h_1* and ionized hydrogen is *h_2*.

Here is a complete listing:

- Neutral hydrogen fraction: `'h_1'`
- Ionized hydrogen fraction: `'h_2'`
- Neutral helium fraction: `'he_1'`
- Singly-ionized helium fraction: `'he_2'`
- Doubly-ionized helium fraction: `'he_3'`
- Electron fraction: `'e'`
- Gas density (in g cm^{-3}): `'rho'`

These are the default elements in the `history` dictionary, which is an attribute of all `ares.simulations` classes.

We also generally keep track of the ionization and heating rate coefficients:

- Rate coefficient for photo-ionization `Gamma_h_1`.
- etc.

2.4.2 Two-Zone IGM Models

For calculations of the reionization history or global 21-cm signal, in which we use a two-zone IGM formalism, all quantities described in the previous sections keep their usual names with one important change: they now also have an *igm* or *cgm* prefix to signify which phase of the IGM they belong to. The *igm* phase is of course short for inter-galactic medium, while the *cgm* phase stands for the circum-galactic medium (really just meant to indicate gas near galaxies).

- Kinetic temperature, `igm_Tk`.
- HII region volume filling factor, `cgm_h_2`.
- Neutral fraction in the bulk IGM, `igm_h_1`.
- Heating rate in the IGM, `igm_heat_h_1`.

- Volume-averaged ionization rate, or rate of change in `cgm_h_2`, `cgm_Gamma_h_1`.

There are also new (passive) quantities, like the neutral hydrogen excitation (or "spin" temperature), the 21-cm brightness temperature, and rate coefficients that govern the time evolution of the 21-cm signal:

- 21-cm brightness temperature: `'igm_dTb'`.
- Spin temperature: `'igm_Ts'`.

Each of these are only associated with the IGM grid patch, since the other phase of the IGM is assumed to be fully ionized and thus dark at 21-cm wavelengths.

2.5 Troubleshooting

This page is an attempt to keep track of common errors and instructions for how to fix them.

2.5.1 Plots not showing up

If, when running some *ares* script (e.g., those in `$ARES/tests`) the program runs to completion without errors but does not produce a figure, it may be due to your matplotlib settings. Most test scripts use `draw` to ultimately produce the figure because it is non-blocking and thus allows you to continue tinkering with the output if you'd like. One of two things is going on:

- You invoked the script with the standard Python interpreter (i.e., **not** iPython). Try running it with iPython, which will spit you back into an interactive session once the script is done, and thus keep the plot window open.
- Alternatively, your default `matplotlib` settings may have caused this. Check out your `matplotlibrc` file (in `$HOME/.matplotlibrc`) and make sure `interactive : True`.

Future versions of *ares* may use blocking commands to ensure that plot windows don't disappear immediately. Email me if you have strong opinions about this.

2.5.2 `IOError: No such file or directory`

There are a few different places in the code that will attempt to read-in lookup tables of various sorts. If you get any error that suggests a required input file has not been found, you should:

- Make sure you have set the `$ARES` environment variable. See the Installation page for instructions.
- Make sure the required file is where it should be, i.e., nested under `$ARES/input`.

In the event that a required file is missing, something has gone wrong. Many lookup tables are downloaded automatically when you run the `setup.py` script, so the first thing you should do is re-run `python setup.py install`.

2.5.3 `LinAlgError: singular matrix`

This is an odd one, known to occur in `ares.physics.Hydrogen` when using `scipy.interpolate.interp1d` to compute the collisional coupling coefficients for spin-exchange.

We still aren't sure why this happens – it cannot always be reproduced, even by two users using the same version of *scipy*. A temporary hack is to use linear interpolation, instead of a spline, or to hack off data points at high temperatures in the lookup table. Working on a more satisfying solution...email me if you encounter this problem.

2.6 How to get involved

If *ares* lacks functionality you're interested in but seems like it could be adapted to suit your needs, fork *ares* on bitbucket, and/or shoot me an email and I can help you get started!

CHAPTER 3

Indices and tables

- genindex
- modindex
- search