hera_sim Documentation

HERA-Team

Jan 23, 2019

Contents

1 Contents 3

2 Indices and tables 11

hera_sim is a simple simulator that generates instrumental effects and applies them to visibilities.

CHAPTER 1

Contents

1.1 Installation

1.1.1 Requirements

Requires:

- *numpy*
- *scipy*
- *aipy*
- *hera_cal* (which requires *h5py*)
- *pyuvdata*

Then, at the command line, navigate to the *hera_sim* repo/directory, and:

```
pip install .
```

If developing, from the top-level directory do:

```
pip install -e .
```

1.2 Tutorials and FAQs

The following introductory tutorial will help you get started with *hera_sim*:

1.2.1 Tour of hera_sim

This notebook briefly introduces some of the effects that can be modeled with `hera_sim`.

[ ]:
```
%matplotlib notebook
import aipy, uvtools
import numpy as np
import pylab as plt
```

[5]:
```
from hera_sim import foregrounds, noise, sigchain, rfi
```

[6]:
```
fqs = np.linspace(.1,.2,1024,endpoint=False)
lsts = np.linspace(0,2*np.pi,10000, endpoint=False)
times = lsts / (2*np.pi) * aipy.const.sidereal_day
bl_len_ns = 30.
```

Foregrounds

Diffuse Foregrounds

[7]:
```
Tsky_mdl = noise.HERA_Tsky_mdl['xx']
vis_fg_diffuse = foregrounds.diffuse_foreground(Tsky_mdl, lsts, fqs, bl_len_ns)
```

[8]:
```
MX, DRNG = 2.5, 3
plt.figure()
plt.subplot(211); uvtools.plot.waterfall(vis_fg_diffuse, mode='log', mx=MX,␣
↪drng=DRNG); plt.colorbar(); plt.ylim(0,4000)
plt.subplot(212); uvtools.plot.waterfall(vis_fg_diffuse, mode='phs'); plt.colorbar();␣
↪plt.ylim(0,4000)
plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Point-Source Foregrounds

[9]:
```
vis_fg_pntsrc = foregrounds.pntsrc_foreground(lsts, fqs, bl_len_ns, nsrcs=200)
```

[10]:
```
plt.figure()
plt.subplot(211); uvtools.plot.waterfall(vis_fg_pntsrc, mode='log', mx=MX, drng=DRNG);
↪ plt.colorbar()#; plt.ylim(0,4000)
plt.subplot(212); uvtools.plot.waterfall(vis_fg_pntsrc, mode='phs'); plt.colorbar();␣
↪plt.ylim(0,4000)
plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Diffuse and Point-Source Foregrounds

[11]:
```
vis_fg = vis_fg_diffuse + vis_fg_pntsrc
```

[12]:
```
plt.figure()
plt.subplot(211); uvtools.plot.waterfall(vis_fg, mode='log', mx=MX, drng=DRNG); plt.
↪colorbar(); plt.ylim(0,4000)
plt.subplot(212); uvtools.plot.waterfall(vis_fg, mode='phs'); plt.colorbar(); plt.
↪ylim(0,4000)
plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Noise

[13]:
```
tsky = noise.resample_Tsky(fqs,lsts,Tsky_mdl=noise.HERA_Tsky_mdl['xx'])
t_rx = 150.
nos_jy = noise.sky_noise_jy(tsky + t_rx, fqs, lsts)
```

[14]:
```
plt.figure()
plt.subplot(211); uvtools.plot.waterfall(nos_jy, mode='log', mx=MX, drng=DRNG); plt.
↪colorbar()#; plt.ylim(0,4000)
plt.subplot(212); uvtools.plot.waterfall(nos_jy, mode='phs'); plt.colorbar()#; plt.
↪ylim(0,4000)
plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[16]:
```
vis_fg_nos = vis_fg + nos_jy
```

[17]:
```
plt.figure()
plt.subplot(211); uvtools.plot.waterfall(vis_fg_nos, mode='log', mx=MX, drng=DRNG);␣
↪plt.colorbar(); plt.ylim(0,4000)
plt.subplot(212); uvtools.plot.waterfall(vis_fg_nos, mode='phs'); plt.colorbar(); plt.
↪ylim(0,4000)
plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

RFI

[18]:
```
rfi1 = rfi.rfi_stations(fqs, lsts)
rfi2 = rfi.rfi_impulse(fqs, lsts, chance=.02)
rfi3 = rfi.rfi_scatter(fqs, lsts, chance=.001)
rfi_all = rfi1 + rfi2 + rfi3
```

[19]:
```
plt.figure()
plt.subplot(211); uvtools.plot.waterfall(rfi_all, mode='log', mx=MX, drng=DRNG); plt.
↪colorbar(); plt.ylim(0,4000)
plt.subplot(212); uvtools.plot.waterfall(rfi_all, mode='phs'); plt.colorbar(); plt.
↪ylim(0,4000)
plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

```
/home/steven/miniconda3/envs/hera_sim/lib/python2.7/site-packages/uvtools/plot.py:13:␣
↪RuntimeWarning: divide by zero encountered in log10
  data = np.log10(data)
```

[21]:
```
vis_fg_nos_rfi = vis_fg_nos + rfi_all
```

[22]:
```
plt.figure()
plt.subplot(211); uvtools.plot.waterfall(vis_fg_nos_rfi, mode='log', mx=MX,␣
↪drng=DRNG); plt.colorbar(); plt.ylim(0,4000)
plt.subplot(212); uvtools.plot.waterfall(vis_fg_nos_rfi, mode='phs'); plt.colorbar();␣
↪plt.ylim(0,4000)
plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Gains

[23]:
```
g = sigchain.gen_gains(fqs, [1,2,3])
plt.figure()
for i in g: plt.plot(fqs, np.abs(g[i]), label=str(i))
plt.legend(); plt.show()
gainscale = np.average([np.median(np.abs(g[i])) for i in g])
MXG = MX + np.log10(gainscale)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

[24]:
```
vis_total = sigchain.apply_gains(vis_fg_nos_rfi, g, (1,2))
plt.figure()
plt.subplot(211); uvtools.plot.waterfall(vis_total, mode='log', mx=MXG, drng=DRNG);␣
↪plt.colorbar(); plt.ylim(0,4000)
plt.subplot(212); uvtools.plot.waterfall(vis_total, mode='phs'); plt.colorbar(); plt.
↪ylim(0,4000)
plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

Crosstalk

[25]:
```
xtalk = sigchain.gen_xtalk(fqs)
vis_xtalk = sigchain.apply_xtalk(vis_fg_nos_rfi, xtalk)
vis_xtalk = sigchain.apply_gains(vis_xtalk, g, (1,2))
plt.figure()
plt.subplot(211); uvtools.plot.waterfall(vis_xtalk, mode='log', mx=MXG, drng=DRNG);␣
↪plt.colorbar(); plt.ylim(0,4000)
plt.subplot(212); uvtools.plot.waterfall(vis_xtalk, mode='phs'); plt.colorbar(); plt.
↪ylim(0,4000)
plt.show()
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.HTML object>
```

1.3 API Reference

1.3.1 hera_sim

hera_sim.vis
hera_sim.antpos
hera_sim.eor
hera_sim.foregrounds
hera_sim.io
hera_sim.noise
hera_sim.reflections
hera_sim.rfi
hera_sim.sigchain
hera_sim.utils

1.4 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

1.4.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

1.4.2 Documentation improvements

hera_sim could always use more documentation, whether as part of the official py21cmmc docs, in docstrings, or even on the web in blog posts, articles, and such.

1.4.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/HERA-Team/hera_sim/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

1.4.4 Development

To set up *hera_sim* for local development:

1. Fork hera_sim (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/hera_sim.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

1.5 Developing *hera_sim*

hera_sim broadly follows the best-practices laid out in XXX.

Todo: where is that best-practices doc?

All docstrings should be written in Google docstring format.

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

1.6 AUTHORS

- HERA-Team - https://github.com/HERA-Team

1.7 Changelog

CHAPTER 2

Indices and tables

- genindex
- modindex
- search