time_domain_astronomy_sandbox Documentation

Release 0.0.1a

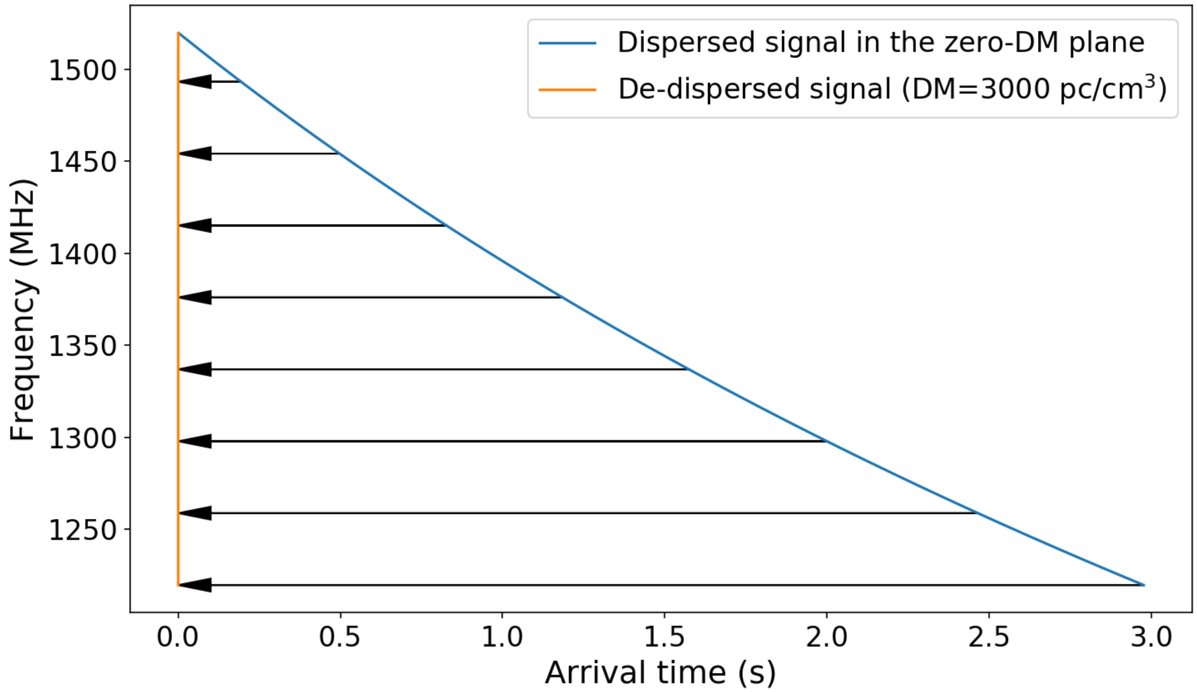
D. Vohl

Apr 01, 2022

Contents:

This consists of a series of classes to simulate time-domain astronomy data products.

Try it on on .



Classes currently includes:

- `Backend`: properties describing an observatory backend,
- `Pulse`: a broadband dispersed pulse,
- `Observation`: an observation data product generated for a given `Backend`,
- `RFIm`: radio frequency interference mitigation functions,
- `SNR`: signal-to-noise functions,
- `Plotting`: plotting functions.

CHAPTER 1

Getting the code

```
git clone https://github.com/macrocosme/time_domain_astronomy_sandbox.git
cd time_domain_astronomy_sandbox/
pip[3] install -r requirements.txt
```

CHAPTER 2

Requirements

```
numpy>=1.17.0
matplotlib>=2.1.2
ipywidgets>=7.4.1
```

CHAPTER 3

Getting started

Instanciate a Backend using your preferred properties and use as argument to instanciate an Observation. You can then add Pulses via add_dispersed_pulse (Pulse()) and add_rfi.

You can test it for yourself by clicking on the file "Usage example.ipynb" on to launch a jupyter notebook (it takes a few second to launch the server).

CHAPTER 4

Usage examples

1. Load classes:

```
from time_domain_astronomy_sandbox.backend import Backend
from time_domain_astronomy_sandbox.observation import Observation
from time_domain_astronomy_sandbox.pulse import Pulse
from time_domain_astronomy_sandbox.plotting import plot_multi_images, plot_multi_1D
from time_domain_astronomy_sandbox.rfim import RFIm
```

2. Plot a dispersed pulse interactively (using *ipywidgets*)

```
def interactive_pulse_arts():
    """Plot interactive dispersed pulse using ASTRON's ARTS backend."""
    pulse = Pulse(Backend())
    pulse.plot_delay_v_frequency_interactive(xscale='linear')

interactive_pulse_arts()
```

3. Plot low- and high-DM broadband dispersed pulses, narrowband periodic pulses, and run RFI cleaning.

```
def pulse_and_rfi__cleaned():
  obs = Observation(Backend(), length=1.024/1.5)
  raw = obs.window.copy()
  obs.add_dispersed_pulse(dm=500, width=0.006, pulse_t0=0.04, snr=15)
  frb = obs.window.copy()
  obs.add_dispersed_pulse(dm=1, width=0.006, pulse_t0=0.23, snr=125)
  obs.add_dispersed_pulse(dm=10, width=0.001, pulse_t0=0.33, snr=125)

  for t_start, t_step, t_width, f1, f2 in [
      [0., 0.01, 0.003, 350, 360],
      [0.1, 0.008, 0.005, 700, 715],
  ]:
      obs.add_rfi(
          t_start=t_start,
          t_stop=t_start+0.3,
```

(continues on next page)

(continued from previous page)

```
        t_step=t_step,
        t_width=t_width,

        f_start=f1,
        f_stop=f2,

        snr=125,
    )

xstep = 1100
ystep = 500

rc('font', size=16)
rc('axes', labelsize=18)

original = copy.deepcopy(obs)

plot_multi_images(
    (
        raw,
        frb,
        original.window,
    ),

    labels=(
        'Noise (gaussian)',
        'Noise + Faint FRB',
        'Noise + Faint FRB + Strong RFI',
    ),

    direction='vertical',

    xticks=obs.time_indices[::xstep],
    xtick_labels=["%.2f" % t for t in obs.times[::xstep]],

    yticks=obs.backend.freq_indices[::ystep],
    ytick_labels=["%.0f" % f for f in obs.backend.frequencies[::ystep]],

    xfig_size=12,
    yfig_size=7.4,
    spectrum=False,
    colorbar=True,
    savefig=True,
    fig_name='noise_pulses_rfi',
    ext='pdf'
)

del raw

o_tc = RFIm().tdsc_amber(obs.window.copy())
o_fc = RFIm().fdsc_amber(obs.window.copy(), bin_size=Backend().n_channels,␣
↪threshold=3.25)
plot_multi_images(
    (
        o_tc,
        o_fc,
        obs.frequency_cleaning(obs.time_cleaning(), keep_state=True, bin_
↪size=Backend().n_channels, threshold=3.25),
```

(continues on next page)

(continued from previous page)

```
    ),

    labels=(
        'RFI mitigation (time)',
        'RFI mitigation (freq.)',
        'RFI mitigation (time and freq.)',
    ),

    direction='vertical',

    xticks=obs.time_indices[::xstep],
    xtick_labels=["%.2f" % t for t in obs.times[::xstep]],

    yticks=obs.backend.freq_indices[::ystep],
    ytick_labels=["%.0f" % f for f in obs.backend.frequencies[::ystep]],

    xfig_size=12,
    yfig_size=7.4,
    spectrum=False,
    colorbar=True,
    savefig=True,
    fig_name='rficlean',
    ext='pdf'
)

plot_multi_images(
    (
        original.dedisperse(dm=500),
        obs.dedisperse(dm=500),
    ),

    labels=(
        'Dedispersed input (DM=500 pc/cm^3)',
        'Dedispersed w/ RFI mitigation (time and freq., DM=500 pc/cm^3)',
    ),

    direction='vertical',

    xticks=obs.time_indices[::xstep],
    xtick_labels=["%.2f" % t for t in obs.times[::xstep]],

    yticks=obs.backend.freq_indices[::ystep],
    ytick_labels=["%.0f" % f for f in obs.backend.frequencies[::ystep]],

    xfig_size=12,
    yfig_size=9.4,

    loc=1,

    detection_threshold=8.,

    spectrum=True,
    colorbar=False,
    savefig=True,
    fig_name='input_dedispersed',
    ext='pdf'
)
```

(continues on next page)

(continued from previous page)

```
pulse_and_rfi__cleaned()
```

CHAPTER 5

Comments and issues

Comments and issues can be posted by opening an on github.

CHAPTER 6

Contribution

If you want to contribute to the project, you can contact me or simply do a pull request on github.

CHAPTER 7

License

This project is licensed under the terms of the GNU GPL v3+ license.

7.1 backend

class time_domain_astronomy_sandbox.backend.**Backend**(*n_channels: int = 1536*, *channel_bandwidth: float = 0.1953125*, *fmin: float = 1219.700927734375*, *sampling_time: float = 8.192e-05*, *samples_per_second: int = 12500*)

Defaults are currently ARTS observing properties.

7.2 pulse

class time_domain_astronomy_sandbox.pulse.**Pulse**(*backend: time_domain_astronomy_sandbox.backend.Backend = <time_domain_astronomy_sandbox.backend.Backend object>*, *width: int = 10*)

delays(*dm*)

Create array of delays for each backend frequency channel.

Parameters **dm** (*int*) – Value for dispersion measure of the pulse

Returns **delays** – Array of delays (in second)

Return type Numpy.array

plot_delay_v_frequency(*dm*, *xscale='linear'*, *savefig=False*, *ext='png'*)

Plot pulse's delay vs frequency.

Parameters

- **dm** (*int*) – Value for dispersion measure of the pulse
- **xscale** (*str*) – matplotlib's xscale option (default: 'linear')
- **savefig** (*bool*) – save figure to disk (default: False)
- **ext** (*'str'*) – figure's file extention (default: png)

plot_delay_v_frequency_interactive(*xscale='linear'*, *dm_min=0*, *dm_max=5000*, *dm_step=5*, *dm_init=0*, *savefig=False*, *ext='png'*)

Plot pulse's delay vs frequency interactively with ipywidgets.

Parameters

- **xscale** (*str*) – matplotlib's xscale option
- **dm_min** (*int*) – minimum dm for interactive widget (default: 0)
- **dm_max** (*int*) – maximum dm for interactive widget (default: 5000)
- **dm_step** (*int*) – increment step dm for interactive widget (default: 0)
- **dm_init** (*int*) – initial dm for interactive widget (default: 5000)
- **savefig** (*bool*) – save figure to disk (default: False)
- **ext** (*'str'*) – figure's file extention

plot_signal_dispersed_dedispersed(*dm*, *step=200*, *xscale='linear'*, *savefig=False*, *ext='png'*, *dpi=150*)

Plot pulse's delay vs frequency.

Parameters

- **dm** (*int*) – Value for dispersion measure of the pulse
- **xscale** (*str*) – matplotlib's xscale option (default: 'linear')
- **savefig** (*bool*) – save figure to disk (default: False)
- **ext** (*'str'*) – figure's file extention (default: png)

7.3 observation

class time_domain_astronomy_sandbox.observation.**Observation**(*backend: time_domain_astronomy_sandbox.backend.B*
length: int = 1, *t0: float = 0.0*)

Observation class.

dedisperse(*dm*, *window=[]*)

Dedisperse an observation window for a given dispersion measure (DM).

Parameters

- **dm** (*int*) – Dispersion measure to use for dedispersion
- **window** (*(list | Numpy.array)*) – An observation window (to clean a specific instance of window). If empty, cleans self.window

Returns **dedispersed_window** – The dedispersed window.

Return type Numpy.array

frequency_cleaning(*window=[]*, *n_iter=1*, *bin_size=32*, *threshold=2.75*, *symetric=False*, *keep_state=False*)

RFI mitigation (cleaning) in frequency domain.

Parameters

- **window** (*(list | Numpy.array)*) – An observation window (to clean a specific instance of window). If empty, cleans self.window
- **n_iter** (*int*) – Number of cleaning iteration
- **keep_state** (*bool*) – Save result of cleaning to self.window

Returns **self.window** – The cleaned window.

Return type Numpy.array

time_cleaning(*window=[]*, *n_iter=1*, *threshold=3.25*, *symetric=False*, *keep_state=False*)

RFI mitigation (cleaning) in time domain.

Parameters

- **window** (*(list | Numpy.array)*) – An observation window (to clean a specific instance of window). If empty, cleans self.window
- **n_iter** (*int*) – Number of cleaning iteration
- **keep_state** (*bool*) – Save result of cleaning to self.window

Returns **self.window** – The cleaned window.

Return type Numpy.array

7.4 rfim

class time_domain_astronomy_sandbox.rfim.**RFIm**

RFIm class. A class for radio interference mitigation.

fdsc(*data*, *bin_size=32*, *threshold=2.75*)

Frequency domain sigma cut.

(Modified code from https://github.com/liamconnor/arts-analysis/blob/master/triggers.py)

Parameters

- **data** (*Numpy.Array*) – 2D Array
- **bin_size** (*int*) – Size of averaging bin Size
- **threshold** (*float*) – Threshold to use for sigma cut inequality

fdsc_amber(*data*, *bin_size=32*, *threshold=2.75*, *n_iter=1*, *symmetric=False*)

Frequency domain sigma cut.

Parameters

- **data** (*Numpy.Array*) – 2D Array
- **bin_size** (*int*) – Size of averaging bin Size
- **threshold** (*float*) – Threshold to use for sigma cut inequality
- **n_iter** (*int*) – Number of cleaning iteration
- **symmetric** (*bool*) – Filter equally or not on both side of the distribution

fdsc_old(*data*, *bin_size=32*, *threshold=2.75*, *n_iter=1*)
Frequency domain sigma cut.

Parameters

- **data** (*Numpy.Array*) – 2D Array
- **bin_size** (*int*) – Size of averaging bin Size
- **threshold** (*float*) – Threshold to use for sigma cut inequality
- **n_iter** (*int*) – Number of cleaning iteration

tdsc(*data*, *threshold=3.25*, *n_iter=1*)
Time domain sigma cut.

(Modified code from https://github.com/liamconnor/arts-analysis/blob/master/triggers.py)

Parameters

- **data** (*Numpy.Array*) – 2D Array
- **threshold** (*float*) – Threshold to use for sigma cut inequality
- **n_iter** (*int*) – Number of cleaning iteration

tdsc_amber(*data*, *threshold=3.25*, *n_iter=1*, *symmetric=False*)
Time domain sigma cut as implemented in AA-ALERT RFIm.

Parameters

- **data** (*Numpy.Array*) – 2D Array
- **threshold** (*float*) – Threshold to use for sigma cut inequality
- **n_iter** (*int*) – Number of cleaning iteration
- **symmetric** (*bool*) – Filter equally or not on both side of the distribution

tdsc_per_channel(*data*, *threshold=3.25*, *n_iter=1*)
Time domain sigma cut.

(Code from https://github.com/liamconnor/arts-analysis/blob/master/triggers.py)

Parameters

- **data** (*Numpy.Array*) – 2D Array
- **threshold** (*float*) – Threshold to use for sigma cut inequality
- **n_iter** (*int*) – Number of cleaning iteration

7.5 SNR

class time_domain_astronomy_sandbox.snr.**SNR**
SNR class. A class for signal-to-noise computation.

simple_snr(*a*, *axis=0*)
Compute signal-to-noise ratio

Parameters

- **a** (*list or numpy array*) – Array of data
- **axis** (*int*) – Axis onto which compute SNR

Returns **vals** – Values (SNR per bin)

Return type array of float

7.6 plotting

Plotting methods.

`time_domain_astronomy_sandbox.plotting.`**`plot_image`**(*data, xticks=[], xtick_labels=[], yticks=[], ytick_labels=[], ncols=1, nrows=1, xfig_size=10, yfig_size=5*)

Plot spectrum.

Parameters

- **data** (*Numpy.Array*) –
- **xticks** (*list*) – List of ticks for x axis
- **xticks_labels** (*list*) – List of tick labels for x axis
- **yticks** (*list*) – List of ticks for y axis
- **yticks_labels** (*list*) – List of tick labels for y axis
- **ncols** (*int*) – Number of column for matplotlib.pyplot.subplots
- **nrows** (*int*) – Number of rows for matplotlib.pyplot.subplots
- **xfig_size** (*int*) – Figure size in x
- **yfig_size** (*int*) – Figure size in y

`time_domain_astronomy_sandbox.plotting.`**`plot_multi_1D`**(*data_arr, labels=[], xticks=[], xtick_labels=[], yticks=[], ytick_labels=[], direction='horizontal', xfig_size=10, yfig_size=5, loc=4, detection_threshold=None, savefig=False, fig_name='multi-1D', ext='png', dpi=150*)

Plot multiple spectrum.

Parameters

- **data** (*list (Numpy.Array)*) – list of data arrays
- **xticks** (*list*) – List of ticks for x axis
- **xticks_labels** (*list*) – List of tick labels for x axis
- **yticks** (*list*) – List of ticks for y axis
- **yticks_labels** (*list*) – List of tick labels for y axis
- **direction** (*str*) – General direction onto which append subplots (default: 'horizontal')
- **xfig_size** (*int*) – Figure size in x (default: 10)
- **yfig_size** (*int*) – Figure size in y (default: 5)
- **savefig** (*bool*) – Save figure (default: False)

- **fig_name** (*str*) – Figure name (default: 'multi-images')
- **ext** (*str*) – File extension (default 'png')

time_domain_astronomy_sandbox.plotting.**plot_multi_images**(*data_arr*, *labels=[]*, *xticks=[]*, *xtick_labels=[]*, *yticks=[]*, *ytick_labels=[]*, *direction='horizontal'*, *xfig_size=10*, *yfig_size=5*, *loc=4*, *spectrum=False*, *detection_threshold=None*, *colorbar=False*, *savefig=False*, *fig_name='multi-images'*, *ext='png'*, *dpi=150*)

Plot images.

Parameters

- **data** (*list (Numpy.Array)*) – list of data arrays
- **xticks** (*list*) – List of ticks for x axis
- **xticks_labels** (*list*) – List of tick labels for x axis
- **yticks** (*list*) – List of ticks for y axis
- **yticks_labels** (*list*) – List of tick labels for y axis
- **direction** (*str*) – General direction onto which append subplots (default: 'horizontal')
- **xfig_size** (*int*) – Figure size in x (default: 10)
- **yfig_size** (*int*) – Figure size in y (default: 5)
- **savefig** (*bool*) – Save figure (default: False)
- **fig_name** (*str*) – Figure name (default: 'multi-images')
- **ext** (*str*) – File extension (default 'png')

time_domain_astronomy_sandbox.plotting.**plot_spectrum**(*data*, *ncols=1*, *nrows=1*)

Plot spectrum.

Parameters

- **data** (*Numpy.Array*) –
- **ncols** (*int*) – Number of column for matplotlib.pyplot.subplots
- **nrows** (*int*) – Number of rows for matplotlib.pyplot.subplots

time_domain_astronomy_sandbox.plotting.**set_multi_axes**(*ax*, *direction*, *xticks*, *xtick_labels*, *yticks*, *ytick_labels*, *spectrum=False*, *dual=False*)

Set axes ticks and tick labels

Parameters

- **ax** (*matplotlib.axes.Axes*) – Array of axes
- **direction** (*str*) – General direction onto which append subplots

- **xticks** (*list*) – List of ticks for x axis
- **xticks_labels** (*list*) – List of tick labels for x axis
- **yticks** (*list*) – List of ticks for y axis
- **yticks_labels** (*list*) – List of tick labels for y axis

Python Module Index

t

Index