turbo$_{s}eti$

May 25, 2022

Contents:

CHAPTER 1

De-Doppler Search

1.1 turboSETI Command Main Program

Main program module for executable turboSETI

turbo_seti.find_doppler.seti_event.**exec_proc**(*args*)
Interface to FindDoppler class, called by main().

Parameters **args** (*dict*) –

turbo_seti.find_doppler.seti_event.**main**(*args=None*)
This is the entry-point to turboSETI.

Parameters **args** (*dict*) –

1.2 Find Doppler

1.2.1 turbo_seti doppler search module

This module is deeply dependent on classes and functions in data_handler.py.

Main class: FindDoppler

Independent functions: search_coarse_channel - for a given coarse channel, doppler search. load_the_data - loads everything needed by search_coarse_channel. populate_tree - populate "tree_findoppler" used by several functions. hitsearch - Searches for hits at given drift rate. tophitsearch - Searches for hits with largest SNR within 2*tsteps fine frequency channels.

class turbo_seti.find_doppler.find_doppler.**FindDoppler**(*datafile*, *max_drift=10.0*, *min_drift=1e-05*, *snr=25.0*, *out_dir='./'*, *coarse_chans=None*, *obs_info=None*, *flagging=False*, *n_coarse_chan=None*, *kernels=None*, *gpu_backend=False*, *gpu_id=0*, *precision=1*, *append_output=False*, *log_level_int=20*, *blank_dc=True*)

Initializes FindDoppler object.

Parameters

- **datafile** (*string*) – Input filename (.h5 or .fil)
- **max_drift** (*float*) – Maximum drift rate in Hz/second.
- **min_drift** (*float*) – Minimum drift rate in Hz/second.
- **snr** (*float*) – Minimum Signal to Noise Ratio (SNR) - A ratio bigger than 1 to 1 has more signal than noise.
- **out_dir** (*string*) – Directory where output files should be placed. By default this is the current working directory.
- **coarse_chans** (*list(int)*) – The input comma-separated list of coarse channels to analyze, if any. By default, all coarse channels will be searched. Use this to search only specified channels, e.g. [7,12] will search channels 7 and 12 only.
- **obs_info** (*dict*) – Used to hold information found on file, including info about pulsars, RFI, and SEFD.
- **flagging** (*bool*) – Flags the edges of the PFF for BL data (with 3Hz res per channel)? (True/False) Anybody - please improve this cryptic description.
- **n_coarse_chan** (*int*) – Number of coarse channels in the file. If None (default), blimpy will make this determination (undesirable, in general).
- **kernels** (Kernels, *optional*) – Pre-configured class of Kernels.
- **gpu_backend** (*bool, optional*) – Use GPU accelerated Kernels? (True/False)
- **gpu_id** (*int*) – If gpu_backend=True, then this is the GPU device to use. Default is 0.
- **precision** (*int {2: float64, 1: float32}, optional*) – Floating point precision for the GPU. The default is 1 (recommended).
- **append_output** (*bool, optional*) – Append output DAT & LOG files? (True/False) Default is False. DEPRECATED.
- **log_level_int** (*int, optional*) – Python logging threshold level (INFO, DEBUG, or WARNING) Default is logging.INFO.
- **blank_dc** (*bool, optional*) – Remove the DC spike? (True/False) Default is True (recommended).

last_logwriter(*arg_path*, *arg_text*)

Write the last LogWriter entry

Parameters

- **arg_path** (*str*) – Path of log for the final log entries.
- **arg_text** (*str*) – Text message to include at end of the log file.

Returns

Return type None.

search (*n_partitions=1*, *progress_bar='n'*)
Top level search routine.

Parameters

- **n_partitions** (*int*) – Number of Dask partitions (processes) to use in parallel. Defaults to single-partition (process).
- **progress_bar** (*str {'y', 'n'}, optional*) – Enable command-line progress bar.

Returns

Return type None.

Notes

self.data_handle.cchan_list [the list of coarse channel objects for searching,] created by self.data_handle = DATAHandle() during __init__() execution.

If using dask (n_partitions > 1): * Launch multiple drift searches in parallel. * Each search works on a single coarse channel object. * n_partitions governs the maximum number of partitions to run in parallel. Else, the searches are done in sequence of the coarse channel objects.

It is not recommended to mix dask partitions with GPU mode as this could cause GPU queuing.

turbo_seti.find_doppler.find_doppler.**hitsearch** (*fd*, *spectrum*, *specstart*, *specend*, *snr_thresh*, *drift_rate*, *header*, *tdwidth*, *max_val*, *the_median*, *the_stddev*)
Searches for hits that exceed the given SNR threshold.

Note that the "max" arrays share the index values as any given spectrum. They represent maximums with respect to the frequency columns in the range (0, FFT length).

Let S be the subspectrum given by spectrum[specstart:specend]. Set hit-counter to 0. For each element of S,

Subtract the given median and divide that result by the given standard deviation, giving the new element value.

if the element value > snr_thresh then Increment hit-counter If element value > current max SNR using the common index then

Set the current max SNR at the common index = this element. Set the current max drift rate at the common index = drift rate of this element.

Increment the grand total of hits by the hit-counter.

Parameters

- **fd** (FindDoppler) – Instance of FindDoppler class.
- **spectrum** (*ndarray*) – Array of data values along the frequency axis of length = FFT length.
- **specstart** (*int*) – First index to search for hit in spectrum.

- **specend** (*int*) – Last index to search for hit in spectrum.
- **snr_thresh** (*float*) – Minimum signal to noise ratio for candidacy.
- **drift_rate** (*float*) – Drift rate at which we are searching for hits.
- **header** (*dict*) – Header in fits header format. See data_handler.py's DATAH5 class header.
- **tdwidth** (*int*) – FFT Length = # fine channels / # coarse channels.
- **max_val** (max_vals) – Object to be filled with max values from this search and then returned. Length of each subarray = FFT length.

turbo_seti.find_doppler.find_doppler.**load_the_data**(*cchan_dict*, *precision*)

Load the DATAH5 object, spectra matrix, and the associated drift indexes.

Parameters

- **cchan_dict** (*dict*) – A single coarse channel object created by data_handler.py DATAHandle __split_h5.
- **precision** (*int {2: float64, 1: float32}*) – Floating point precision for the GPU.

Returns

- **datah5_obj** (*DATAH5 object (complex!)*)
- **spectra** (*numpy.ndarray*) – Spectra data array. Set by the data_handler.py load_data function.
- **drift_indexes** (*numpy.ndarray*) – Drift index matrix. Set by the data_handler.py load_data function.

class turbo_seti.find_doppler.find_doppler.**max_vals**

Class used to initialize some maximums.

turbo_seti.find_doppler.find_doppler.**populate_tree**(*fd*, *spectra*, *tree_findoppler*, *nframes*, *tdwidth*, *tsteps*, *fftlen*, *shoulder_size*, *roll=0*, *reverse=0*)

This script populates the findoppler tree with the spectra.

Parameters

- **fd** (*FindDoppler object*) – Instance of FindDoppler class.
- **spectra** (*ndarray*) – Spectra matrix.
- **tree_findoppler** (*ndarray*) – Tree to be populated with spectra.
- **nframes** (*int*) –
- **tdwidth** (*int*) –
- **tsteps** (*int*) –
- **fftlen** (*int*) – Length of fast fourier transform (fft) matrix.
- **shoulder_size** (*int*) – Size of shoulder region.
- **roll** (*int, optional*) – Used to calculate amount each entry to the spectra should be rolled (shifted).
- **reverse** (*int, optional*) – Used to determine which way spectra should be rolled (shifted).

Returns Spectra-populated version of the input tree_findoppler.

Return type ndarray

Notes

It creates two "shoulders" (each region of tsteps*(shoulder_size/2) in size) to avoid "edge" issues. It uses np.roll() for drift-rate blocks higher than 1.

turbo_seti.find_doppler.find_doppler.**search_coarse_channel**(*cchan_dict*, *fd*, *dataloader=None*, *logwriter=None*, *filewriter=None*)

Run a turboseti search on a single coarse channel.

Parameters

- **cchan_dict** (*dict*) – A single coarse channel object created by data_handler.py DATAHandle __split_h5. Contains the following fields: * filename : file path (common to all objects) * f_start : start frequency of coarse channel * f_stop : stop frequency of coarse channel * cchan_id : coarse channel number * n_coarse_chan : total number of coarse channels (common to all objects)
- **fd** (*FindDoppler object*) – Instance of the FindDoppler class.
- **logwriter** (LogWriter, *optional*) – A LogWriter to write log output into. If None, one will be created.
- **filewriter** (FileWriter, *optional*) – A FileWriter to use to write the dat file. If None, one will be created.

Returns Returns True if no exceptions occur (needed for dask).

Return type bool

Notes

This function is separate from the FindDoppler class to allow parallelization. This should not be called directly, but rather via the FindDoppler.search() routine. One exception: turboseti_search package.

turbo_seti.find_doppler.find_doppler.**tophitsearch**(*fd*, *tree_findoppler_original*, *max_val*, *tsteps*, *header*, *tdwidth*, *fftlen*, *max_drift*, *drift_rate_resolution*, *logwriter=None*, *filewriter=None*, *obs_info=None*)

This finds the hits with largest SNR within a nearby window of frequency channels. The window size is calculated so that we cannot report multiple overlapping hits.

Parameters

- **tree_findoppler_original** (*ndarray*) – Spectra-populated findoppler tree
- **max_val** (max_vals) – Contains max values from hitsearch
- **tsteps** (*int*) –
- **header** (*dict*) – Header in fits header format. Used to report tophit in filewriter. See *DATAH5*
- **tdwidth** (*int*) –
- **fftlen** (*int*) – Length of fast fourier transform (fft) matrix

- **max_drift** (*float*) – Maximum drift rate in Hz/second
- **drift_rate_resolution** (*float*) – The drift rate corresponding to drifting rightwards one bin in the whole observation
- **logwriter** (LogWriter, *optional*) – Logwriter to which we should write if we find a top hit.
- **filewriter** (FileWriter, *optional*) – Filewriter corresponding to file to which we should save the local maximum of tophit. See *report_tophit()*
- **obs_info** (*dict, optional*) –

Returns Same filewriter that was input.

Return type *FileWriter*

1.3 Data Handler

Filterbank data handler for the find_doppler.py functions.

class turbo_seti.find_doppler.data_handler.**DATAH5**(*filename*, *f_start=None*, *f_stop=None*, *t_start=None*, *t_stop=None*, *cchan_id=0*, *n_coarse_chan=None*, *kernels=None*, *gpu_backend=False*, *precision=1*, *gpu_id=0*)

This class is where the waterfall data is loaded, as well as the DATAH5 header info. Don't be surprised at the use of FITS header names! [?] It creates other attributes related to the dedoppler search (load_drift_indexes).

Parameters

- **filename** (*string*) – Name of file.
- **f_start** (*float*) – Start frequency in MHz.
- **f_stop** (*float*) – Stop frequency in MHz.
- **t_start** (*int*) – Start integration ID.
- **t_stop** (*int*) – Stop integration ID.
- **coarse_chan** (*int*) – Coarse channel ID.
- **n_coarse_chan** (*int*) – Total number of coarse channels.
- **kernels** (Kernels) – Pre-configured class of kernels.

close()

Closes file and sets the data attribute *.closed* to True. A closed object can no longer be used for I/O operations. *close()* may be called multiple times without error.

load_data()

Read the spectra and drift indices from file.

Returns **spectra, drift indices**

Return type ndarray, ndarray

load_drift_indexes()

The drift indices are read from a stored file so that there is no need to recalculate. This speed things up.

Returns **drift_indexes**

Return type ndarray

class turbo_seti.find_doppler.data_handler.**DATAHandle**(*filename=None*, *out_dir='./'*, *n_coarse_chan=None*, *coarse_chans=None*, *kernels=None*, *gpu_backend=False*, *precision=1*, *gpu_id=0*)

Class to setup input file for further processing of data. Handles conversion to h5 (from fil), extraction of coarse channel info, waterfall info, and file size checking.

Parameters

- **filename** (*str*) – Name of file (.h5 or .fil).
- **out_dir** (*str*) – Directory where output files should be saved.
- **n_coarse_chan** (*int*) – Number of coarse channels.
- **coarse_chans** (*list or None*) – List of course channels.
- **kernels** (*Kernels, optional*) – Pre-configured class of Kernels.
- **gpu_backend** (*bool, optional*) – Use GPU accelerated Kernels?
- **precision** (*int {2: float64, 1: float32}, optional*) – Floating point precision. Default: 1.
- **gpu_id** (*int*) – If gpu_backend=True, then this is the device ID to use.

get_info()

Get the header of the file.

Returns **header** – Header of the blimpy file.

Return type dict

1.4 File Writers

class turbo_seti.find_doppler.file_writers.**FileWriter**(*filename*, *header*)

Used to write information to turboSETI output files.

Initializes FileWriter object and writes its header.

Parameters

- **filename** (*str*) – Name of file on which we would like to perform operations.
- **header** (*dict*) – Information to be written to header of file filename.

report_header(*header*)

Write header information per given obs.

Parameters **header** (*dict*) – Information to be written to file header.

report_tophit(*max_val*, *ind*, *ind_tuple*, *tdwidth*, *fftlen*, *header*, *total_n_candi*, *obs_info=None*)

This function looks into the top hit in a region, basically finds the local maximum and saves that.

Parameters

- **max_val** (*findopp*) –
- **ind** (*int*) – Index at which top hit is located in max_val's maxdrift and maxsnr.

- **ind_tuple** (*tuple(int, int) (lbound, ubound)*) –
- **tdwidth** (*int*) –
- **fftlen** (*int*) – Length of the fast fourier transform matrix.
- **header** (*dict*) – Contains info on coarse channel to be written to file.
- **total_n_candi** (*int*) –
- **obs_info** (*dict, optional*) – Used to hold info found on file, including info about pulsars, RFI, and SEFD.

Returns

Return type FileWriter object that called this function.

class turbo_seti.find_doppler.file_writers.**GeneralWriter**(*filename=''*, *mode='a'*)

Wrapper class for file operations.

Initializes GeneralWriter object. Opens given file with given mode, sets new object's filehandle to the file object, sets the new object's filename to the file's name, then closes the file.

Parameters

- **filename** (*str*) – Name of file on which we would like to perform operations.
- **mode** (*str {'a', 'r', 'w', 'x'}, optional*) – Mode which we want to use to open file, same modes as the built-in python built-in open function: read (*r*), append (*a*), write (*w*), or create (*x*).

close()

Closes file object if it is open.

is_open()

Checks if file is open.

Returns True if file is open, False otherwise.

Return type boolean

open(*mode='a'*)

Opens the file with the inputted mode, then closes it. Does not actually leave the file opened, only used for changing mode.

Parameters **mode** (*str {'a', 'r', 'w', 'x'}, optional*) – Mode which we want to assign to this file, same modes as the built-in python built-in open function: read (*r*), append (*a*), write (*w*), or create (*x*).

writable()

Checks if file is open, and if it is, checks that mode is either write or append.

Returns True if file is open and writeable, False otherwise.

Return type boolean

write(*info_str*, *mode='a'*)

Sets file mode to a writeable mode and opens it if it is not already open in a writeable mode, writes info_str to it, and then closes it. If the file was not previously open when this is called, the file is closed after writing in order to maintain the state the filewriter was in before.

Parameters

- **info_str** (*str*) – Data to be written to file.

- **mode** (*str {'a', 'w'}, optional*) – Mode for file. If it is not a writeable mode, it will be set to a writeable mode.

class turbo_seti.find_doppler.file_writers.**LogWriter**(*filename=''*, *mode='a'*)
Used to write data to log.

Initializes GeneralWriter object. Opens given file with given mode, sets new object's filehandle to the file object, sets the new object's filename to the file's name, then closes the file.

Parameters

- **filename** (*str*) – Name of file on which we would like to perform operations.
- **mode** (*str {'a', 'r', 'w', 'x'}, optional*) – Mode which we want to use to open file, same modes as the built-in python built-in open function: read (*r*), append (*a*), write (*w*), or create (*x*).

info(*info_str*)
Writes info_str to file.

Parameters **info_str** (*str*) – String to be written to file.

1.5 Kernels

1.5.1 Hitsearch

This kernel implements a GPU accelerated version of the *hitsearch()* method written as a RAW CUDA kernel.

1.5.2 De-Doppler

This kernel implements a slightly modified version of the Taylor Tree algorithm published by J.H. Taylor in 1974.

1. This GPU implementation is based on Cupy array library accelerated with CUDA and ROCm.
2. This CPU implementation is based on Numba Just-In-Time compilation.

turbo_seti.find_doppler.kernels._taylor_tree._core_numba.**flt**
This is a function to Taylor-tree-sum a data stream. It assumes that the arrangement of data stream is, all points in first spectra, all points in second spectra, etc. Data are summed across time.

Parameters

- **outbuf** (*array_like*) – Input data array, replaced by dedispersed data at the output.
- **nchn** (*int*) – Number of timesteps in the data.

References

- R. Ramachandran, 07-Nov-97, nfra. – Original algorithm.
- A. Siemion, 2011 – float/64 bit addressing (C-code)
- H. Chen, 2014 – python version
- E. Enriquez + P.Schellart, 2016 – cython version
- L. Cruz, 2020 – numba version

class turbo_seti.find_doppler.kernels.**Kernels**(*gpu_backend=False*, *precision=2*, *gpu_id=0*)

Dynamically loads the right modules according to parameters.

Parameters

- **gpu_backend** (*bool, optional*) – Enable GPU acceleration.
- **precision** (*int {2: float64, 1: float32}, optional*) – Floating point precision.

get_spectrum(*tt_output*, *tsteps*, *tdwidth*, *drift_index*)

The different Taylor tree kernels have a slightly different output. Both of them you can think of indexed by [row index][frequency], although it is reshaped as a 1-dimensional array. In the GPU version, the row index is the same as the "drift index". 0 is the least drift, 1 is the next least drift, et cetera. In the CPU version, the row index is bit-reversed from this. This method lets the caller get data for a particular drift without knowing how the rows are ordered. There's a good chance that one or both of these is suboptimal; please update this comment if you change the underlying algorithm.

static has_gpu()

Check if the system has the modules needed for the GPU acceleration.

Note: Modules are listed on *requirements_gpu.txt*.

Returns **has_gpu** – True if the system has GPU capabilities.

Return type bool

1.6 Helper Functions

turbo_seti.find_doppler.helper_functions.**FlipX**(*outbuf*, *xdim*, *ydim*, *xp=None*)

This function takes in an array of values and iteratively flips ydim chunks of values of length xdim.

Parameters

- **outbuf** (*ndarray*) – An array with shape like (int, 1)
- **xdim** (*int*) – Size of segments to be flipped.
- **ydim** (*int*) – Amount of segments of size xdim to be flipped.
- **xp** (*Numpy or Cupy, optional*) – Math module to be used. If *None*, Numpy will be used.

Examples

If you have an array [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and enter it with xdim = 5 and ydim = 2, the array will be modified to become [5, 4, 3, 2, 1, 10, 9, 8, 7, 6]. Note that if you wish for the whole array to be modified in this way, xdim * ydim should equal the length of the array. If ydim * xdim is greater than the length of the array, this function will error.

turbo_seti.find_doppler.helper_functions.**bitrev**(*inval*, *nbits*)

This function bit-reverses the given value "inval" with the number of bits, "nbits".

Parameters

- **inval** (*int*) – Number to be bit-reversed.

- **nbits** (*int*) – The length of inval in bits. If user only wants the bit-reverse of a certain amount of bits of inval, nbits is the amount of bits to be reversed counting from the least significant (rightmost) bit. Any bits beyond this length will not be reversed and will be truncated from the result.

Returns The bit-reverse of inval. If there are more significant bits beyond nbits, they are truncated.

Return type int

References

- R. Ramachandran, 10-Nov-97, nfra. – Original C implementation.
- H. Chen, 2014 – Python version.
- R. Elkins (texadactyl), 2020 – Speedup.

turbo_seti.find_doppler.helper_functions.**chan_freq**(*header*, *fine_channel*, *tdwidth*, *ref_frame*)

Find channel frequency. Note issue #98.

Parameters

- **header** –
- **fine_channel** –
- **tdwidth** –
- **ref_frame** –

Returns **chanfreq**

Return type float

turbo_seti.find_doppler.helper_functions.**comp_stats**(*np_arr*, *xp=None*)

Compute median and stddev of floating point vector array in a fast way, discarding outliers.

Parameters

- **np_arr** (*ndarray*) – Floating point vector array.
- **xp** (*Numpy or Cupy, optional*) – Math module to be used. If *None*, Numpy will be used.

Returns **the_median, the_stddev** – Median and standard deviation of input array with outliers removed.

Return type numpy.float32, numpy.float32

1.7 Merge DAT and LOG Files

Source file for merge_dats_logs()

turbo_seti.find_doppler.merge_dats_logs.**merge_dats_logs**(*arg_h5: str*, *arg_dir: str*, *arg_type: str*, *cleanup='n'*)

Merge multiple DAT (or LOG) files.

Parameters

- **arg_h5** (*str*) – HDF5 file used by *search()* to produce the DAT and LOG files.

- **arg_dir** (*str*) – Directory holding multiple DAT and LOG files after FindDoppler.search() which ran with more than 1 partition.
- **arg_type** (*str*) – File extension of interest ('dat' or 'log').

CHAPTER 2

De-Doppler Analysis

In this code, the following terminology is used:

- Hit: Single strong narrowband signal in an observation.
- Event: Strong narrowband signal that is associated with multiple hits across ON observations.

Note: This code works for .dat files that were produced by seti_event.py after turboSETI version 0.8.2, and blimpy version 1.1.7 (~mid 2019). The drift rates *before* that version were recorded with the incorrect sign and thus the drift rate sign would need to be flipped in the make_table function.

2.1 Authors

- Version 2.0 - Sofia Sheikh (ssheikhmsa@gmail.com) and Karen Perez (kip2105@columbia.edu)
- Version 1.0 - Emilio Enriquez (jeenriquez@gmail.com)

2.2 plotSETI Command Main Program

Main program module for executable plotSETI. Facilitates the automation of 2 large functions:

find_event_pipline() plot_event_pipline()

turbo_seti.find_event.run_pipelines.**clean_event_stuff**(*path_out_dir*)

Clean up the output directory of old artifacts.

Parameters **path_out_dir** (*str*) – Output path of directory holding old artifacts.

Returns

Return type None.

turbo_seti.find_event.run_pipelines.**count_text_lines**(*path_list_file*)
Count the list of text lines in a file.

Parameters **path_list_file** (*str*) – Path of file containing a list of text lines..

Returns Count of text lines.

Return type int

turbo_seti.find_event.run_pipelines.**execute_pipelines**(*args*)
Interface to the pipeline functions, called by main().

Parameters **args** (*dict*) –

turbo_seti.find_event.run_pipelines.**main**(*args=None*)
This is the entry point to the plotSETI executable.

Parameters **args** (*dict*) –

turbo_seti.find_event.run_pipelines.**make_lists**(*path_h5_dir*, *path_h5_list*, *path_dat_dir*, *path_dat_list*)
Create a list of .h5 files and a list of .dat files.

Parameters

- **path_h5_dir** (*str*) – Directory where the h5 files reside.
- **path_h5_list** (*str*) – Path of output list of h5 files.
- **path_dat_dir** (*str*) – Directory where the dat files reside.
- **path_dat_list** (*str*) – Path of output list of dat files.

Returns Number in cadence : Success. 0 : Failure.

Return type int

2.3 Find Event Pipeline

Front-facing script to find drifting, narrowband events in a set of generalized cadences of ON-OFF radio SETI observations.

The main function contained in this file is *find_event_pipeline()* calls find_events from find_events.py to read a list of turboSETI .dat files. It then finds events within this group of files.

class turbo_seti.find_event.find_event_pipeline.**PathRecord**(*path_dat*, *tstart*, *source_name*, *fch1*, *foff*, *nchans*)
Definition of a DAT record

turbo_seti.find_event.find_event_pipeline.**close_enough**(*x*, *y*)
Make sure that x and y are close enough to be considered roughly equal.

turbo_seti.find_event.find_event_pipeline.**find_event_pipeline**(*dat_file_list_str*, *h5_file_list_str=None*, *check_zero_drift=False*, *filter_threshold=3*, *on_off_first='ON'*, *number_in_cadence=6*, *on_source_complex_cadence=False*, *saving=True*, *csv_name=None*, *user_validation=False*, *sortby_tstart=True*, *SNR_cut=None*, *min_drift_rate=None*, *max_drift_rate=None*)

Find event pipeline.

Parameters

- **dat_file_list_str** (*str*) – The string name of a plaintext file ending in .lst that contains the filenames of .dat files, each on a new line, that were created with seti_event.py. The .lst should contain a set of cadences (ON observations alternating with OFF observations). The cadence can be of any length, given that the ON source is every other file. This includes Breakthrough Listen standard ABACAD as well as OFF first cadences like BACADA. Minimum cadence length is 2, maximum cadence length is unspecified (currently tested up to 6). Example: ABACAD|ABACAD|ABACAD
- **h5_file_list_str** (*str | None*) – The string name of a plaintext file ending in .lst that contains the filenames of .h5 files, each on a new line, that were created with seti_event.py. The .lst should contain a set of cadences (ON observations alternating with OFF observations). The cadence can be of any length, given that the ON source is every other file. This includes Breakthrough Listen standard ABACAD as well as OFF first cadences like BACADA. Minimum cadence length is 2, maximum cadence length is unspecified (currently tested up to 6).
- **check_zero_drift** (*bool*) – A True/False flag that tells the program whether to include hits that have a drift rate of 0 Hz/s. Earth- based RFI tends to have no drift rate, while signals from the sky are expected to have non-zero drift rates.
- **filter_threshold** (*int, default is 3*) – Specification for how strict the hit filtering will be. There are 3 different levels of filtering, specified by the integers 1, 2, and 3. * Filter_threshold = 1 applies the following parameter checks:

  check_zero_drift SNR_cut min_drift_rate max_drift_rate

  However, Filter_threshold = 1 applies no ON-OFF check. * Filter_threshold = 2 returns hits that passed level 1 AND that are in at least one ON table but no OFF tables. * Filter_threshold = 3 returns events that passed level 2 AND that are present in *ALL* ON tables.
- **on_off_first** (*str {'ON', 'OFF'}*) – Tells the code whether the .dat sequence starts with the ON or the OFF observation. Valid entries are 'ON' and 'OFF' only. Default is 'ON'.
- **number_in_cadence** (*int*) – The number of files in a single ON-OFF cadence. Default is 6 for ABACAD.
- **on_source_complex_cadence** (*bool*) – If using a complex cadence (i.e. ons and offs not alternating), this variable should be the string target name used in the .dat filenames.

The code will then determine which files in your dat_file_list_str cadence are ons and which are offs.

- **saving** (*bool*) – A True/False flag that tells the program whether to save the output array as a .csv.
- **user_validation** (*bool*) – A True/False flag that, when set to True, asks if the user wishes to continue with their input parameters (and requires a 'y' or 'n' typed as confirmation) before beginning to run the program. Recommended when first learning the program, not recommended for automated scripts.
- **sortby_tstart** (*bool*) – If True, the input file list is sorted by header.tstart.
- **SNR_cut** (*None (default value) or float value > 0*) – If None, then all SNR values from the dedoppler results in the dat files are accepted as-is. Otherwise, the specified value is the threshold SNR below which hits will be discarded.
- **min_drift_rate** (*None (default value) or float value > 0*) – If None, then all drift rate values from the dedoppler results in the dat files are accepted as-is. Otherwise, the specified value is the threshold drift rate below which hits will be discarded.
- **max_drift_rate** (*None (default value) or float value > 0*) – If None, then all drift rate values from the dedoppler results in the dat files are accepted as-is. Otherwise, the specified value is the threshold drift rate above which hits will be discarded.

Returns

- a Pandas dataframe with all the events that were found.
- None, if no events were found.

Return type Either

Notes

The HDF5 file is ASSUMED(!!) to have the same name as .dat files.

Examples

```
>>> import find_event_pipeline;
>>> find_event_pipeline.find_event_pipeline(dat_file_list_str,
...                                         SNR_cut=10,
...                                         min_drift_rate=0.1,
...                                         max_drift_rate=4,
...                                         check_zero_drift=False,
...                                         filter_threshold=3,
...                                         on_off_first='ON',
...                                         number_in_cadence=6,
...                                         on_source_complex_cadence=False,
...                                         saving=True,
...                                         user_validation=False)
```

turbo_seti.find_event.find_event_pipeline.**get_file_header**(*filepath_h5*)

Extract and return the target's source name from the DAT file path.

Parameters **dat_path** (*str*) – Full or relative path name of the DAT file

Returns **header**

Return type Waterfall header object

2.4 Find Event

Backend script to find drifting, narrowband events in a generalized cadence of radio SETI observations (any number of ons, any number of offs, any pattern - streamlined for alternating on-off sequences).

The main function contained in this file is `find_events()` uses the other helper functions in this file (described below) to read a list of turboSETI .dat files. It then finds events within this group of files.

turbo_seti.find_event.find_event.**calc_freq_range**(*hit*, *delta_t=0.0*, *max_dr=True*, *follow=False*)

Calculates a range of frequencies where RFI in an off-source could be related to a hit in an on-source, given a freq and drift_rate.

Parameters

- **hit** (*dict*) –
- **delta_t** (*float, optional*) –
- **max_dr** (*bool, optional*) –
- **follow** (*bool, optional*) –

Returns [low_bound, high_bound]

Return type list

turbo_seti.find_event.find_event.**end_search**(*t0*)

Ends the search when there are no candidates left, or when the filter level matches the user-specified level.

Parameters **t0** (*time*) –

turbo_seti.find_event.find_event.**find_events**(*dat_file_list*, *check_zero_drift=False*, *filter_threshold=3*, *on_off_first='ON'*, *complex_cadence=False*, *SNR_cut=None*, *min_drift_rate=None*, *max_drift_rate=None*)

Reads a list of turboSETI .dat files.

Parameters

- **dat_file_list** (*list*) – A Python list of .dat files with ON observations of a single target alternating with OFF observations. This cadence can be of any length, given that the ON source is every other file. This includes Breakthrough Listen standard ABACAD as well as OFF first cadences like BACADA. Minimum cadence length is 2, maximum cadence length is unspecified (currently tested up to 6).
- **check_zero_drift** (*bool, optional*) – A True/False flag that tells the program whether to include hits that have a drift rate of 0 Hz/s. Earth- based RFI tends to have no drift rate, while signals from the sky are expected to have non-zero drift rates. Default is False.
- **filter_threshold** (*int, default is 3*) – Specification for how strict the hit filtering will be. There are 3 different levels of filtering, specified by the integers 1, 2, and 3. * Filter_threshold = 1 applies the following parameter checks:

  check_zero_drift SNR_cut min_drift_rate max_drift_rate

  However, Filter_threshold = 1 applies no ON-OFF check. * Filter_threshold = 2 returns hits that passed level 1 AND that are in at least one ON table but no OFF tables. * Filter_threshold = 3 returns events that passed level 2 AND that are present in *ALL* ON tables.

- **on_off_first** (*str {'ON', 'OFF}, optional*) – Tells the code whether the .dat sequence starts with the ON or the OFF observation. Valid entries are 'ON' and 'OFF' only.
- **complex_cadence** (*bool, optional*) – A Python list of 1s and 0s corresponding to which files in the file_sublist are on-sources and which are off_sources for complex (i.e. non alternating) cadences.
- **SNR_cut** (*None (default value) or float value > 0*) – If None, then all SNR values from the dedoppler results in the dat files are accepted as-is. Otherwise, the specified value is the threshold SNR below which hits will be discarded.
- **min_drift_rate** (*None (default value) or float value > 0*) – If None, then all drift rate values from the dedoppler results in the dat files are accepted as-is. Otherwise, the specified value is the threshold drift rate below which hits will be discarded.
- **max_drift_rate** (*None (default value) or float value > 0*) – If None, then all drift rate values from the dedoppler results in the dat files are accepted as-is. Otherwise, the specified value is the threshold drift rate above which hits will be discarded.

Examples

It is highly recommended that users interact with this program via the front-facing find_event_pipeline.py script. See the usage of that file in its own documentation.

If you would like to run find_events without calling find_event_pipeline.py, the usage is as follows:

```
>>> find_event.find_events(file_sublist, SNR_cut=10, check_zero_drift=False,
...                        filter_threshold=3, on_off_first='ON', complex_
↪cadence=False)
```

Notes

It calls other functions to find events within this group of files. Filter_threshold allows the return of a table of events with hits at different levels of filtering. Filter_threshold = [1,2,3] means:

1) Hits above an SNR cut witout AB check
2) Hits that are only in some As and no Bs
3) Hits that are only in all As and no Bs

turbo_seti.find_event.find_event.**follow_event**(*hit*, *on_table*, *get_count=True*)

Follows a given hit to the next observation of the same target and looks for hits which could be part of the same event.

Parameters

- **hit** (*dict*) –
- **on_table** (*dict*) –
- **get_count** (*bool*) –

Returns new_on_table or count

Return type dict or int

turbo_seti.find_event.find_event.**not_yet_seen**(*mylist*, *argument*)

Search a list to see if argument is already there.

Parameters

- **mylist** (*list*) – List of things that have been already seen.
- **argument** (*int*) – An integer to add to list if not alreay seen.

Returns True :: Not yet seen so the argument was added. False :: Already seen.

Return type bool

turbo_seti.find_event.find_event.**read_dat**(*filename*)

Read a turboseti .dat file.

Parameters **filename** (*str*) – Name of .dat file to open.

Returns **df_data** – Pandas dataframe of hits.

Return type dict

2.5 Plot DAT

turbo_seti.find_event.plot_dat.**make_plot**(*dat*, *fil*, *f_start*, *f_stop*, *t0*, *candidate=None*, *check_zero_drift=False*, *alpha=1*, *color='black'*)

Parameters

- **dat** (*str*) – The .dat file containing information about the hits detected.
- **fil** (*str*) – Filterbank or h5 file corresponding to the .dat file.
- **f_start** (*float*) – Start frequency, in MHz.
- **f_stop** (*float*) – Stop frequency, in MHz.
- **t0** (*float*) – Start time of the candate event in mjd units.
- **candidate** (*dict, optional*) – A single row from a pandas dataframe containing information about one of the candidate signals detected. Contains information about the candidate signal to be plotted. The necessary data includes the start and stop frequencies, the drift rate, and the source name. The dataframe the candiate comes from is generated in plot_all_hit_and_candidates above as *candidate_event_dataframe*. The default is None.
- **check_zero_drift** (*bool, optional*) – A True/False flag that tells the program whether to include hits that have a drift rate of 0 Hz/s. Earth- based RFI tends to have no drift rate, while signals from the sky are expected to have non-zero drift rates. The default is False.
- **alpha** (*float, optional*) – The opacity of the overlayed hit plot. This should be between 0 and 1, with 0 being invisible, and 1 being the default opacity. This is passed into matplotlib.pyplot function.
- **color** (*str, optional*) – Allows for the specification of the color of the overlayed hits. The default is 'black'.

turbo_seti.find_event.plot_dat.**plot_dat**(*dat_list_string*, *fils_list_string*, *candidate_event_table_string*, *outdir=None*, *check_zero_drift=False*, *alpha=1*, *color='black'*, *window=None*)

Makes a plot similar to the one produced by plot_candidate_events, but also includes the hits detected, in addition to the candidate signal.

Calls *plot_hit_candidate* and *make_plot*

Parameters

- **dat_list_string** (*str*) – List of .dat files in the cadence.
- **fils_list_string** (*str*) – List of filterbank or .h5 files in the cadence.
- **candidate_event_table_string** (*str*) – The string name of a .csv file that contains the list of events at a given filter level, created as output from find_event_pipeline.py.
- **outdir** (*str, optional*) – Path to the directory where the plots will be saved to. The default is None, which will result in the plots being saved to the directory where the .dat file are located.
- **check_zero_drift** (*bool, optional*) – A True/False flag that tells the program whether to include hits that have a drift rate of 0 Hz/s. Earth- based RFI tends to have no drift rate, while signals from the sky are expected to have non-zero drift rates. The default is False.
- **outdir** – Path to the directory where the plots will be saved to. The default is None, which will result in the plots being saved to the directory the .dat file are located.
- **alpha** (*float, optional*) – The opacity of the overlayed hit plot. This should be between 0 and 1, with 0 being invisible, and 1 being the default opacity. This is passed into matplotlib.pyplot function.
- **color** (*str, optional*) – Allows for the specification of the color of the overlayed hits. The default is 'black'.
- **window** (*tuple, optional*) – Sets the start and stop frequencies of the plot, in MHz. The input takes the form of a tuple: (start, stop). And assumes that the start is less than the stop. If given, the resulting plot will range exactly between the start/stop frequencies. The default is None, which will result in a plot of the entire range of hits detected.

turbo_seti.find_event.plot_dat.**plot_hit_candidate**(*dat_file_list*, *fil_file_list*, *source_name_list*, *all_hits_frame*, *candidate=None*, *check_zero_drift=False*, *outdir=None*, *alpha=1*, *color='black'*, *window=None*)

Parameters

- **dat_file_list** (*list*) – A Python list that contains a series of strings corresponding to the filenames of .dat files, each on a new line, that corresponds to the .dat files created when running turboSETI candidate search on the .h5 or .fil files below
- **fil_file_list** (*list*) – A Python list that contains a series of strings corresponding to the filenames of .dat files, each on a new line, that corresponds to the cadence used to create the .csv file used for event_csv_string.
- **source_name_list** (*list*) – A Python list that contains a series of strings corresponding to the source names of the cadence in chronological (descending through the plot pannels) cadence.
- **all_hits_frame** (*dict*) – A pandas dataframe contining information about all the hits detected. The necessary data includes the start and stop frequencies, the drift rate, and the source name. This dataframe is generated in plot_all_hit_and_candidates above.
- **candidate** (*dict, optional*) – A single row from a pandas dataframe containing information about one of the candidate signals detected. Contains information about the candidate signal to be plotted. The necessary data includes the start and stop frequencies, the drift rate, and the source name. The dataframe the candiate comes from is generated in plot_all_hit_and_candidates above as *candidate_event_dataframe*. The default is None.

- **check_zero_drift** (*bool, optional*) – A True/False flag that tells the program whether to include hits that have a drift rate of 0 Hz/s. Earth- based RFI tends to have no drift rate, while signals from the sky are expected to have non-zero drift rates. The default is False.
- **outdir** (*str, optional*) – Path to the directory where the plots will be saved to. The default is None, which will result in the plots being saved to the directory the .dat file are located.
- **alpha** (*float, optional*) – The opacity of the overlayed hit plot. This should be between 0 and 1, with 0 being invisible, and 1 being the default opacity. This is passed into matplotlib.pyplot function.
- **color** (*str, optional*) – Allows for the specification of the color of the overlayed hits. The default is 'black'.
- **window** (*tuple, optional*) – Sets the start and stop frequencies of the plot, in MHz. The input takes the form of a tuple: (start, stop). And assumes that the start is less than the stop. The resulting plot will range exactly between the start/stop frequencies. The default is None, which will result in a plot of the entire range of hits detected.

2.6 Plot Event Pipeline

Front-facing script to plot drifting, narrowband events in a set of generalized cadences of ON-OFF radio SETI observations.

class turbo_seti.find_event.plot_event_pipeline.**PathRecord**(*path_h5*, *tstart*, *source_name*)

Definition of an H5 path record

turbo_seti.find_event.plot_event_pipeline.**plot_event_pipeline**(*event_csv_string*, *fils_list_string*, *user_validation=False*, *offset=0*, *filter_spec=None*, *sortby_tstart=True*, *plot_dir=None*)

This function calls *plot_candidate_events()* to plot the events in an output .csv file generated by find_event_pipeline.py

Parameters

- **event_csv_string** (*str*) – The string name of a .csv file that contains the list of events at a given filter level, created as output from find_event_pipeline.py. The .csv should have a filename containing information about its parameters, for example "kepler1093b_0015_f2_snr10.csv" Remember that the file was created with some cadence (ex. ABACAD) and ensure that the cadence matches the order of the files in fils_list_string
- **fils_list_string** (*str*) – The string name of a plaintext file ending in .lst that contains the filenames of .fil files, each on a new line, that corresponds to the cadence used to create the .csv file used for event_csv_string.
- **user_validation** (*bool, optional*) – A True/False flag that, when set to True, asks if the user wishes to continue with their input parameters (and requires a 'y' or 'n' typed as confirmation) before beginning to run the program. Recommended when first learning the program, not recommended for automated scripts.

- **offset** (*int, optional*) – The amount that the overdrawn "best guess" line from the event parameters in the csv should be shifted from its original position to enhance readability. Can be set to 0 (default; draws line on top of estimated event) or 'auto' (shifts line to the left by an auto- calculated amount, with addition lines showing original position).
- **sortby_tstart** (*bool*) – If True, the input file list is sorted by header.tstart.

Examples

```
>>> import plot_event_pipeline;
... plot_event_pipeline.plot_event_pipeline(event_csv_string, fils_list_string,
...                                         user_validation=False, offset=0)
```

2.7 Plot Event

Backend script to plot drifting, narrowband events in a generalized cadence of ON-OFF radio SETI observations. The main function contained in this file is *plot_candidate_events()* uses the other helper functions in this file (described below) to plot events from a turboSETI event .csv file.

turbo_seti.find_event.plot_event.**make_waterfall_plots**(*fil_file_list*, *on_source_name*, *f_start*, *f_stop*, *drift_rate*, *f_mid*, *filter_level*, *source_name_list*, *offset=0*, *plot_dir=None*, ***kwargs*)

Makes waterfall plots of an event for an entire on-off cadence.

Parameters

- **fil_file_list** (*str*) – List of filterbank files in the cadence.
- **on_source_name** (*str*) – Name of the on_source target.
- **f_start** (*float*) – Start frequency, in MHz.
- **f_stop** (*float*) – Stop frequency, in MHz.
- **drift_rate** (*float*) – Drift rate in Hz/s.
- **f_mid** (*float*) – <iddle frequency of the event, in MHz.
- **filter_level** (*int*) – Filter level (1, 2, or 3) that produced the event.
- **source_name_list** (*list*) – List of source names in the cadence, in order.
- **bandwidth** (*int*) – Width of the plot, incorporating drift info.
- **kwargs** (*dict*) – Keyword args to be passed to matplotlib imshow().

Notes

Makes a series of waterfall plots, to be read from top to bottom, displaying a full cadence at the frequency of a recorded event from find_event. Calls *plot_waterfall()*

turbo_seti.find_event.plot_event.**overlay_drift**(*f_event*, *f_start*, *f_stop*, *drift_rate*, *t_duration*, *offset=0*, *alpha=1*, *color='#cc0000'*)

Creates a dashed red line at the recorded frequency and drift rate of the plotted event - can overlay the signal exactly or be offset by some amount (offset can be 0 or 'auto').

turbo_seti.find_event.plot_event.**plot_candidate_events**(*candidate_event_dataframe*, *fil_file_list*, *filter_level*, *source_name_list*, *offset=0*, *plot_dir=None*, ***kwargs*)

Calls *make_waterfall_plots()* on each event in the input .csv file.

Parameters

- **candidate_event_dataframe** (*dict*) – A pandas dataframe containing information about a candidate event. The necessary data includes the start and stop frequencies, the drift rate, and the source name. To determine the required variable names and formatting conventions, see the output of find_event_pipeline.
- **fil_file_list** (*list*) – A Python list that contains a series of strings corresponding to the filenames of .fil files, each on a new line, that corresponds to the cadence used to create the .csv file used for event_csv_string.
- **filter_level** (*int*) – A string indicating the filter level of the cadence used to generate the candidate_event_dataframe. Used only for output file naming, convention is "f1", "f2", or "f3". Descriptions for the three levels of filtering can be found in the documentation for find_event.py
- **source_name_list** (*list*) – A Python list that contains a series of strings corresponding to the source names of the cadence in chronological (descending through the plot panels) cadence.
- **offset** (*int, optional*) – The amount that the overdrawn "best guess" line from the event parameters in the csv should be shifted from its original position to enhance readability. Can be set to 0 (default; draws line on top of estimated event) or 'auto' (shifts line to the left by an auto-calculated amount, with addition lines showing original position).
- **kwargs** (*dict*) –

Examples

It is highly recommended that users interact with this program via the front-facing plot_event_pipeline.py script. See the usage of that file in its own documentation.

If you would like to run plot_candidate_events without calling plot_event_pipeline.py, the usage is as follows:

```
>>> plot_event.plot_candidate_events(candidate_event_dataframe, fil_file_list,
...                                  filter_level, source_name_list, offset=0)
```

turbo_seti.find_event.plot_event.**plot_waterfall**(*wf*, *source_name*, *f_start=None*, *f_stop=None*, ***kwargs*)

Plot waterfall of data in a .fil or .h5 file.

Parameters

- **wf** (*blimpy.Waterfall object*) – Waterfall object of an H5 or Filterbank file containing the dynamic spectrum data.
- **source_name** (*str*) – Name of the target.
- **f_start** (*float*) – Start frequency, in MHz.
- **f_stop** (*float*) – Stop frequency, in MHz.
- **kwargs** (*dict*) – Keyword args to be passed to matplotlib imshow().

Notes

Plot a single-panel waterfall plot (frequency vs. time vs. intensity) for one of the on or off observations in the cadence of interest, at the frequency of the expected event. Calls *overlay_drift()*

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

W