psrqpy Documentation

Release 1.1.3

Matthew Pitkin

Oct 11, 2021

Contents

A Python tool for interacting with the ATNF pulsar catalogue

This package provides a way to directly query the ATNF Pulsar Catalogue[1] using Python. It does this by downloading and parsing the full catalogue database, which itself is cached and can be reused. It is primarily aimed at astronomers wanting access to the latest pulsar information via a script, rather than through the standard web interface.

Other functionality that it includes:

- it can produce a $P - \dot{P}$ *diagram* using the latest catalogue information.
- a function (`get_glitch_catalogue()`) to access the Jodrell Bank pulsar glitch catalogue.

[1] Manchester, Hobbs, Teoh & Hobbs, *AJ*, **129**, 1993-2006 (2005), arXiv:astro-ph/0412641

CHAPTER 1

Installation

This package can be installed using `pip` via `pip install psrqpy` or `conda` using `conda install -c conda-forge psrqpy`. Alternatively the source code can be obtained from github, and installed using:

```
python setup.py install
```

with `sudo` if wanted to install system wide, and with the `--user` flag if just installing for an individual user.

1.1 Requirements

The requirements for installing the code are:

- `requests`
- `bs4`
- `numpy`
- `astropy`
- `pandas`
- `scipy`

The `ads` module and `matplotlib` are optional requirements to get the full functionality.

CHAPTER 2

Examples

Downloading the full database can be simply achieved via

```
>>> from psrqpy import QueryATNF
>>> query = QueryATNF()
```

From this query the database can then be accessed as an `astropy.table.Table` via

```
>>> table = query.table
```

or as a `pandas.DataFrame` via

```
>>> df = query.pandas
```

You can also specifically limit the query to any combination of the pulsar parameters listed here.

A simple example of such a limited query is to get the frequency 'F0' for all pulsars in the catalogue. This could be done with

```
>>> from psrqpy import QueryATNF
>>> query = QueryATNF(params=['F0'])
```

where the parameter names are case insensitive. This will also automatically include the database uncertainty on `'F0'`, stored as a variable called `'F0_ERR'` (parameter uncertainties will always be stored using the upper-case version of the parameter name, with `_ERR` appended). Again, the table, now only containing `'F0'` and `'F0_ERR'`, can be accessed with

```
>>> table = query.table
```

Note that the full catalogue is still stored in the `psrqpy.QueryATNF` (as a `pandas.DataFrame`) and can accessed with

```
>>> catalogue = query.catalogue
```

Other parameters could be selected using the same `query` object with, e.g.,

```
>>> query.query_params = ['F1', 'RAJ']
>>> print(query.table)
        F1_ERR              RAJ       RAJ_ERR           F1
```

(continues on next page)

(continued from previous page)

```
        1 / s2                                      1 / s2
---------------------- ------------ ------- ----------------------
                 5e-18  00:02:58.17    0.02           -4.48354e-13
2.4933208594475155e-17   00:06:04.8     0.2 -4.357078201884523e-15
                 5e-16   00:07:01.7     0.2              -3.612e-12
                    --     00:11:34   114.0                     --
                 9e-20  00:14:17.75    0.04            -3.6669e-16
                 4e-20            0      --           -1.22783e-15
                   ...          ...     ...                    ...
                 8e-19  23:39:38.74    0.01            -1.6952e-15
                    --     23:40:45     7.0                     --
                    --        23:43      --                     --
                 3e-20            2      --             -9.765e-16
                    --        23:52      --                     --
                    --        23:54     7.0                     --
                 9e-20 23:54:04.724   0.004          -1.821923e-14
Length = 2659 rows
```

The number of pulsars can easily be accessed, e.g.,

```
>>> numstring = 'Version {} of the ATNF catalogue contains {} pulsars'
>>> print(numstring.format(query.get_version, query.num_pulsars))
Version 1.59 of the ATNF catalogue contains 2659 pulsars
```

2.1 More complex queries

Setting conditions

You can set logical conditions on the parameters that you query. Let's say you want all pulsars with rotation frequencies between 100 and 200 Hz, then you could do the following:

```
>>> from psrqpy import QueryATNF
>>> query = QueryATNF(condition='F0 > 100 && F0 < 200')
>>> print(query.num_pulsars)
82
```

If you also wanted pulsars with this condition, but also in globular clusters, you could do

```
>>> query = QueryATNF(condition='F0 > 100 && F0 < 200', assoc='GC')
>>> print(len(query))
33
```

This is equivalent to having `condition='F0 > 100 && F0 < 200 && assoc(GC)'`.

Save a query

You can save a query as a pickled object for later use, e.g., if using a previous `query` we had done:

```
>>> query.save('atnfquery.pkl')
```

Then we could reload this with

```
>>> oldquery = QueryATNF(loadquery='atnfquery.pkl')
```

Query specific pulsars

We might just want to get information on certain pulsars, such as the Crab pulsar (J0534+2200) and J0537-6910, then we could get their sky positions with:

```
>>> from psrqpy import QueryATNF
>>> query = QueryATNF(params=['JNAME', 'RAJ', 'DECJ'], psrs=['J0534+2200', 'J0537-
↪6910'])
>>> print(query.table)
RAJ_ERR     RAJ          DECJ      DECJ_ERR   JNAME
------- ------------ ------------ -------- ----------
  0.005 05:34:31.973 +22:00:52.06     0.06 J0534+2200
   0.11 05:37:47.416 -69:10:19.88      0.6 J0537-6910
```

You can also access these pulsars using the *psrqpy.pulsar.Pulsars* class. This will create a dictionary of *psrqpy.pulsar.Pulsar* objects keyed on the pulsar names. The attributes of the *Pulsar* objects are the parameters that have been retrieved by the query. But, the *Pulsar* objects themselves can query the ATNF Pulsar Catalogue if you request a parameter that they don't already contain. E.g., so first lets get the *psrqpy.pulsar.Pulsars*:

```
>>> psrs = query.get_pulsars()
>>> for psr in psrs:
...     print(psr)
J0534+2200
J0537-6910
```

```
>>> print(psrs['J0534+2200'].keys()) # show attributes of the psr class
['DECJ', 'RAJ', 'DECJ_ERR', 'RAJ_ERR', 'JNAME']
```

What if we want the frequency of J0534+2200? Well, we just have to do

```
>>> crab = psrs['J0534+2200']
>>> print(crab.F0)
29.946923
```

We can also get the whole ephemeris for the Crab with

```
>>> print(query.get_ephemeris('J0534+2200'))
NAME       J0534+2200
JNAME      J0534+2200
BNAME      B0531+21
PSRJ       J0534+2200
PSRB       B0531+21
RAJ        05:34:31.973              0.005000000000000
DECJ       +22:00:52.06              0.060000000000000
PMRA       -14.699999999999999       0.800000000000000
PMDEC      2                         0.800000000000000
POSEPOCH   40706
ELONG      84.097631599851169
ELAT       -1.294467050350203
PMELONG    -14.597441126565251
PMELAT     2.646641750683564
GL         184.557559483180171
GB         -5.784269849609095
RAJD       83.633220833333311
DECJD      22.014461111111110
TYPE       HE[cdt69,fhm+69,hjm+70]
PML        -9.558486649099681
PMB        -11.345718707027030
DIST       2
DIST_DM    1.310000000000000
...
```

Note: This style of ephemeris is not completely equivalent to the pulsar ephemerides returned by the ATNF Pulsar Catalogue.

Query pulsars within a circular boundary

We can query the catalogue to only return pulsars within a circular boundary defined by a central right ascension and declination, and with a given radius (in degrees).

```
>>> from psrqpy import QueryATNF
>>> # set the boundary circle centre (RAJ then DECJ) and radius
>>> c = ['12:34:56.7', '-02:54:12.3', 10.]
>>> query = QueryATNF(params=['JNAME', 'RAJ', 'DECJ'], circular_boundary=c)
>>> print(query.table)
  JNAME          RAJ       DECJ_ERR       DECJ       RAJ_ERR
---------- ------------- -------- -------------- -------
J1257-1027 12:57:04.7686       -- -10:27:05.7817      --
J1312+0051          13:12       --          +00:51      --
```

The circle's coordinates can also be define as, e.g.:

```
>>> c = ['12h34m56.7s', '-02d54m12.3s', 10.]
```

Return a reference

We can make use of the ADS module to return links to references for pulsars/pulsar parameters. For example we could get the reference for the orbital period of J0737-3039A with

```
>>> from psrqpy import QueryATNF
>>> query = QueryATNF(params='PB', psrs='J0737-3039A', include_refs=True,␣
↪adsref=True)
>>> print(query.parse_ref(query.table['PB_REF'])[0])
(" Kramer, M., Stairs, I. H., Manchester, R. N., McLaughlin, M. A., Lyne, A. G.,␣
↪Ferdman, R. D., Burgay, M., Lorimer, D. R., Possenti, A., D'Amico, N.,␣
↪Sarkissian, J. M., Hobbs, G. B., Reynolds, J. E., Freire, P. C. C. & Camilo, F.,␣
↪2006. Tests of General Relativity from Timing the Double Pulsar. Science, 314,␣
↪97-102. ", 'https://ui.adsabs.harvard.edu/#abs/2006Sci...314...97K/')
```

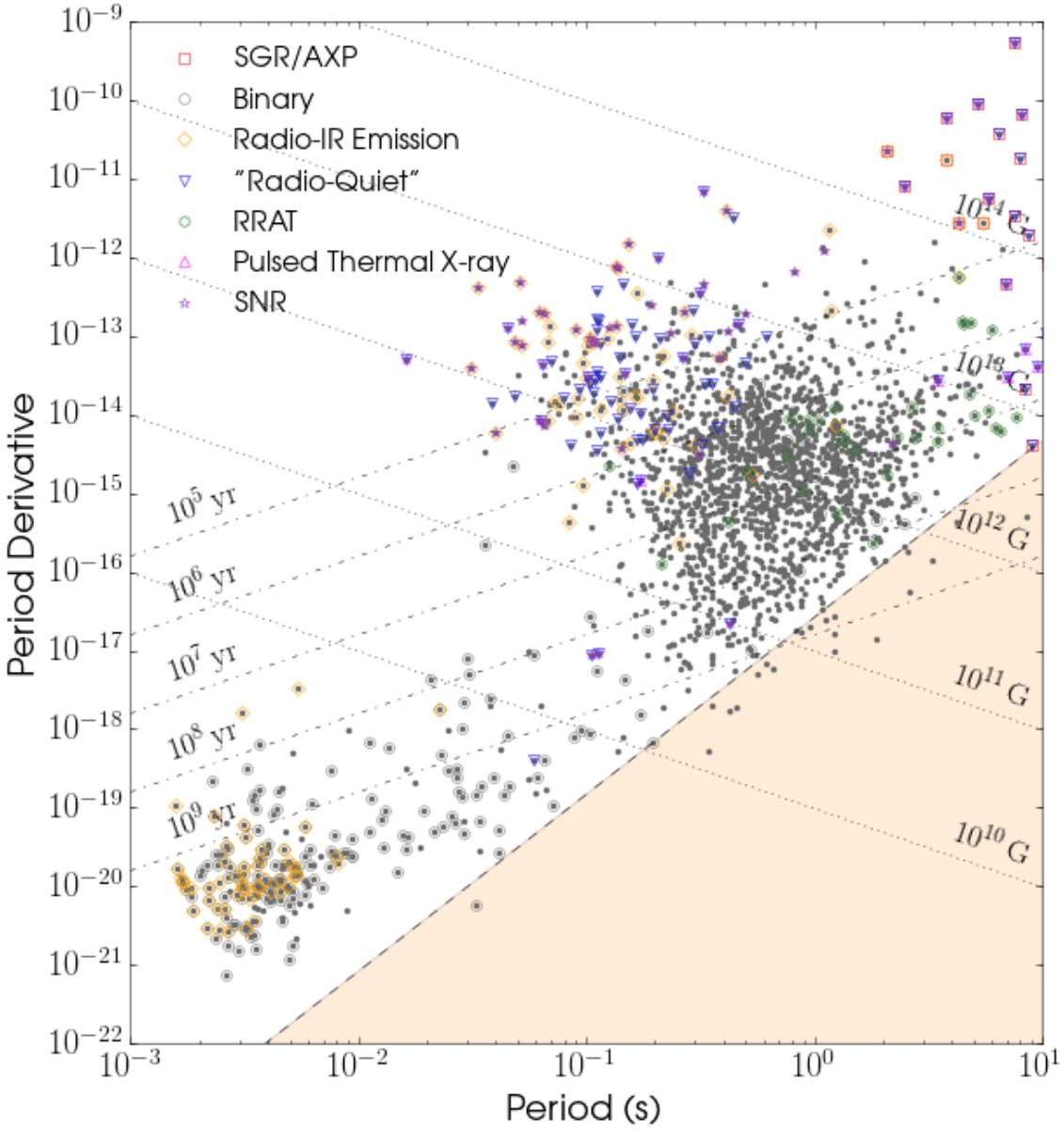
Note: To use this feature you need to have an API key from NASA ADS labs. Getting this is described here.

Make a P-Pdot diagram

You can generate a *lovely* period vs. period derivative diagram based on the latest catalogue information using the `ppdot()` function in just three lines of code:

```
>>> from psrqpy import QueryATNF
>>> query = QueryATNF(params=['P0', 'P1', 'ASSOC', 'BINARY', 'TYPE', 'P1_I'])
>>> query.ppdot(showSNRs=True, showtypes='all')
```

where this shows all pulsar types and pulsars in supernova remnants, to give


SGR/AXP
Binary
Radio-IR Emission
"Radio-Quiet"
RRAT
Pulsed Thermal X-ray
SNR
10^{14} G
10^{13} G
10^{12} G
10^{11} G
10^{10} G
10^{5} yr
10^{6} yr
10^{7} yr
10^{8} yr
10^{9} yr
Period Derivative
Period (s)
10^{-9}
10^{-10}
10^{-11}
10^{-12}
10^{-13}
10^{-14}
10^{-15}
10^{-16}
10^{-17}
10^{-18}
10^{-19}
10^{-20}
10^{-21}
10^{-22}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}

CHAPTER 3

Additional catalogues

In addition to returning a querying the ATNF Pulsar Catalogue, psrqpy can also download and parse:

- the Jodrell Bank pulsar glitch table
- Paolo Freire's table of pulsars in globular clusters (note that the downloaded table does not include accompanying notes or references, and binary parameters that are listed as upper/lower limits within the table are just returned as equalities.)
- Duncan Lorimer & Elizabeth Ferrara's table of galactic millisecond pulsars (accessed via the JSON-ified version by Ujjwal Panda)

3.1 Examples

Accessing the glitch table can be achieved with the *get_glitch_catalogue()* function using:

```
>>> from psrqpy.utils import get_glitch_catalogue
>>> glitches = get_glitch_catalogue()
```

The glitches for a single pulsar can be returned with, e.g.:

```
>>> crabglitches = get_glitch_catalogue(psr="J0534+2200")
```

Accessing the globular cluster table can be achieved with the *get_gc_catalogue()* function using:

```
>>> from psrqpy.utils import get_gc_catalogue
>>> gctable = get_gc_catalogue()
```

The pulsars with a single globular cluster, e.g., 47 Tuc, can the be returned with:

```
>>> tucpulsars = gctable.cluster_pulsars("47 Tuc")
```

Accessing the MSP table can be achieved with the *get_msp_catalogue()* function using:

```
>>> from psrqpy.utils import get_msp_catalogue
>>> msps = get_msp_catalogue()
```

CHAPTER 4

Differences with the ATNF Pulsar Catalogue

There are differences between some of the values returned by psrqpy and those calculated by the `psrcat` software used to generation the ATNF Pulsar Catalogue results. These are listed below:

- The cartesian Galactic coordinates returned by `psrqpy.QueryATNF` (`XX`, `YY`, and `ZZ`) *do not* match those returned by the ATNF Pulsar Catalogue and the `psrcat` software. The values returned by psrqpy are defined using the conventions in the `astropy.coordinates.Galactocentric` class. This uses a Galactic centre distance of 8.3 kpc compared to 8.5 kpc in `psrcat` and is rotated 90 degrees anticlockwise compared to `psrcat`.
- The Galactic coordinate proper motions returned by `psrqpy.QueryATNF` (`PML` and `PMB`) *do not* match those returned by the ATNF Pulsar Catalogue and the `psrcat` software. The values returned by psrqpy purely convert the observed proper motions in right ascension and declination (or elliptic longitude and latitude) into equivalent values in the Galactic coordinate system (via the `astropy.coordinates.Galactic` class). However, the values returned by the ATNF Pulsar Catalogue and the `psrcat` software are in the Galactic cooridinate system, but additionally have the local solar system velocity and Galactic rotation of the pulsar removed from them as described in Section 3 of[2].

[2] Harrison, Lyne & Anderson, *MNRAS*, **261**, 113-124 (1993)

CHAPTER 5

Development and Support

Code development is done via the package's GitHib repository. Any contributions can be made via a fork and pull request model from that repository, and must adhere to the *MIT license*. Any problems with the code or support requests can be submitted via the repository's Issue tracker.

CHAPTER 6

API interface

6.1 Querying

The classes defined here are for querying the ATNF pulsar catalogue and viewing the resulting information.

class QueryATNF(*params=None*, *condition=None*, *psrtype=None*, *assoc=None*, *bincomp=None*, *exactmatch=False*, *sort_attr='jname'*, *sort_order='asc'*, *psrs=None*, *include_errs=True*, *include_refs=False*, *adsref=False*, *loadfromfile=None*, *loadquery=None*, *loadfromdb=None*, *cache=True*, *checkupdate=False*, *circular_boundary=None*, *coord1=None*, *coord2=None*, *radius=0.0*, *frompandas=None*, *fromtable=None*)

Bases: `object`

A class to generate a query of the ATNF pulsar catalogue. By default, this class will download and cache the latest version of the catalogue database file. The catalogue can be queried for specificpulsar parameters and for specific named pulsars. Conditions on the parameter can be specified. The results will be stored as a `pandas.DataFrame`, but can also be accessed as an `astropy.table.Table`.

Parameters

- **params** (str, `list`) – a list of strings with the pulsar parameters to query. The parameter names are case insensitive. If this is not given, then all parameters will be returned by default.
- **condition** (*str*) – a string with logical conditions for the returned parameters. The allowed format of the condition string is given here. Defaults to None.
- **psrtype** (str, `list`) – a list of strings, or single string, of conditions on the type of pulsars to return (logical AND will be used for any listed types). Defaults to None.
- **assoc** (`list`, str) – a condition on the associations of pulsars to return (logical AND will be used for any listed associations). Defaults to None.
- **bincomp** (str, `list`) – a list of strings, or single string, of conditions on the binary companion types of pulsars to return (logical AND will be used for any listed associations). Defaults to None.
- **exactmatch** (*bool*) – a boolean stating whether associations and types given as the condition should be an exact match. Defaults to False.
- **sort_attr** (*str*) – the (case insensitive) parameter name on which with sort the returned pulsars. Defaults to `JName`.

- **sort_ord** (*str*) – the order of the sorting, can be either `asc` or `desc`. Defaults to ascending.
- **psrs** (`list`) – a list of pulsar names for which to get the requested parameters. Defaults to None.
- **circular_boundary** (`list`, tuple) – a list containing three entries defining the centre (in right ascension and declination), and radius of a circle in which to search for and return pulsars. The first entry is the centre point right ascension as a string in format 'hh:mm:ss', the second entry is the centre point declination as a string in format 'dd:mm:ss', and the final entry is the circle's radius in degrees. This condition will only be applied if viewing the results as an `astropy.table.Table`. Alternatively, *coord1*, *coord2*, and *radius* can be used.
- **coord1** (*str*) – a string containing a right ascension in the format ('hh:mm:ss') that centres a circular boundary in which to search for pulsars (requires coord2 and radius to be set).
- **coord2** (*str*) – a string containing a declination in the format ('dd:mm:ss') that centres a circular boundary in which to search for pulsars (requires coord1 and radius to be set).
- **radius** (*float*) – the radius (in degrees) of a circular boundary in which to search for pulsars (requires coord1 and coord2 to be set).
- **include_errs** (*bool*) – Set if wanting parameter errors to be returned. Defaults to True.
- **include_refs** (*bool*) – Set if wanting to include references tags in the output tables. Defaults to False.
- **adsref** (*bool*) – Set if wanting to use an `ads.search.SearchQuery` to get reference information. Defaults to False.
- **loadfromdb** (*str*) – Load a pulsar database file from a given path rather than using the ATNF Pulsar Catalogue database. Defaults to None.
- **loadquery** (*str*) – load an instance of *QueryATNF* from the given file, rather than performing a new query. This was *loadfromfile* in earlier versions, which still works but has been deprecated. Defaults to None.
- **cache** (*bool*) – Cache the catalogue database file for future use. This is ignored if *loadfromdb* is given. Defaults to True.
- **checkupdate** (*bool*) – If True then check whether a cached catalogue file has an update available, and re-download if there is an update. Defaults to False.
- **frompandas** (`pandas.DataFrame`) – create a new `psrqpy.QueryATNF` object from an existing `pandas.DataFrame`.
- **fromtable** (`astropy.table.Table`) – create a new `psrqpy.QueryATNF` object from an existing `astropy.table.Table`.

as_array()

Returns the output table as an array.

Return type `ndarray`

catalogue
Return the entire stored `DataFrame` catalogue without any sorting or conditions applied.

catalogue_len
The length of the entire catalogue, i.e., the number of pulsars it contains. This should be the same as `catalogue_nrows`.

catalogue_ncols
The number of columns in the entire catalogue, i.e. the number of parameters it contains.

catalogue_nrows
The number of rows in the entire catalogue, i.e. the number of pulsars it contains.

catalogue_shape
The shape of the entire catalogue table as a tuple containing the number of rows and the number of columns.

catalogue_table
Return the full catalogue as a `astropy.table.Table` without any query conditions applied.

Note: in this returned table any references will not be converted into actual reference strings, but will still be the ATNF Pulsar Catalogue tags.

columns
Return the table column names.

condition
Return the string of logical conditions applied to the pulsars.

dataframe
Return the query table as a `pandas.DataFrame`.

define_dist()
Set the *DIST* and *DIST1* parameters using other values.

define_galactic()
Calculate the galactic longitude, latitude and position.

Note: The cartesian galactic coordinates returned by this function *do not* match those returned by the ATNF Pulsar Catalogue and the `psrcat` software. They are defined using the conventions in the `astropy.coordinates.Galactocentric` class. This uses a Galactic centre distance of 8.3 kpc compared to 8.5 kpc in `psrcat` and rotated 90 degrees anticlockwise compared to `psrcat`.

The Galactic coordinate proper motions returned by this function *do not* match those returned by the ATNF Pulsar Catalogue and the `psrcat` software. The values returned here convert the observed proper motions in right ascension and declination (or elliptic longitude and latitude) into equivalent values in the Galactic coordinate system (via the `astropy.coordinates.Galactic` class). However, the values returned by the ATNF Pulsar Catalogue and the `psrcat` software are in the Galactic cooridinate system, but additionally have the local solar system velocity and Galactic rotation of the pulsar removed from them as described in Section 3 of Harrison, Lyne & Anderson (1993).

---

derived_age()
Calculate the characteristic age in years (see *characteristic_age()*, with an assumed braking index of n=3).

derived_age_i()
Calculate the characteristic age (in years), dervied from period and intrinsic period derivative.

derived_b_lc()
Calculate the magnetic field strength at the light cylinder.

derived_binary()
Calculate derived binary system parameters.

derived_bsurf()
Calculate the surface magnetic field strength (see *B_field()*).

derived_bsurf_i()
Calculate the surface magnetic field strength, dervied from period and intrinsic period derivative.

derived_ecliptic()
Calculate the ecliptic coordinates, and proper motions, from the right ascension and declination if they are not already given. The ecliptic used here is the astropy's BarycentricMeanEcliptic (note that this does not include nutation unlike the BarycentricTrueEcliptic for which a bug fix was added in astropy 3.2), which may not exactly match that used in *psrcat*.

derived_edot()
Calculate the spin-down luminosity.

derived_edot_i()
Calculate the spin-down luminosity, dervied from period and intrinsic period derivative.

derived_edotd2()
Calculate the spin-down luminosity flux at the Sun.

derived_equatorial()
Calculate equatorial coordinates if only ecliptic coordinates are given. Unlike *psrcat* this function does not currently convert errors on ecliptic coordinates into equavalent errors on equatorial coordinates.

derived_f0()
Calculate the frequency from the period in cases where frequency is not given.

derived_f1()
Calculate the frequency derivative from the period derivative in cases where frequency derivative is not given.

derived_fb0()
Calculate orbital frequency from orbital period.

derived_fb1()
Calculate the orbital frequency derivative from the binary orbital period derivative.

derived_flux()
Calculate spectral index between 400 and 1400 MHz and radio flux at 400 and 1400 MHz.

derived_gw_h0_spindown_limit()
Calculate the limit on the gravitational-wave emission amplitude at Earth assuming all rotational kinetic energy is lost via gravitational-waves generated from an l=m=2 mass quadrupole (i.e., a braking index of n=5). This uses the intrinsic spin-down values rather than the observed spin-downs.

derived_p0()
Calculate the period from the frequency in cases where period is not given.

derived_p1()
Calculate the period derivative from the frequency derivative in cases where period derivative is not given.

derived_p1_i()
Calculate the intrinsic period derivative.

derived_pb()
Calculate binary orbital period from orbital frequency.

derived_pbdot()
Calculate binary orbital period derivative from orbital frequency derivative.

derived_pmtot()
Calculate the total proper motion and error.

derived_vtrans()
Calculate the transverse velocity.

empty
Return True if the `pandas.DataFrame` containing the catalogue is empty.

exactmatch
Return the boolean stating whether certain conditions should apply an exact match.

get_catalogue(*path_to_db=None*, *cache=True*, *update=False*, *overwrite=True*)
Call the *psrqpy.utils.get_catalogue()* function to download the ATNF Pulsar Catalogue, or load a given catalogue path.

Parameters

- **path_to_db** (*str*) – if the path to a local version of the database file is given then that will be read in rather than attempting to download the file (defaults to None).
- **cache** (*bool*) – cache the downloaded ATNF Pulsar Catalogue file. Defaults to True. This is ignored if *path_to_db* is given.
- **update** (*bool*) – if True the ATNF Pulsar Catalogue will be re-downloaded and cached if there has been a change compared to the currently cached version. This is ignored if *path_to_db* is given.
- **overwrite** (*bool*) – if True the returned catalogue will overwrite the catalogue currently contained within the `QueryATNF` class. If False then a new `QueryATNF` copy of the catalogue will be returned.

Returns a table containing the catalogue.

Return type `psrqpy.QueryATNF`

get_ephemeris(*psr*, *precision=15*, *selected=False*)

Return the table row for a particular pulsar and output it as an ephemeris-style string output.

Parameters

- **psr** (*str*) – The name of a pulsar to return.
- **precision** (*int*) – The precision (number of decimal places) at which to output numbers. Defaults to 15.
- **selected** (*bool*) – If True only output the parameters specified by `query_params()`, otherwise output all parameters. Defaults to False.

Returns an ephemeris

Return type str

get_pulsar(*psr*, *selected=False*)

Return the table row for a particular pulsar for all the catalogue parameters.

Parameters

- **psr** (*str*) – The name of a pulsar to return.
- **selected** (*bool*) – If True then output return a table row containing parameters specified by `query_params()`, otherwise return all parameters. Defaults to False.

Returns a table row

Return type `astropy.table.Table`

get_pulsars()

Returns the queried pulsars returned as a *Pulsars* object, which is a dictionary of *Pulsar* objects.

Return type *psrqpy.pulsar.Pulsars*

get_references(*useads=False*, *cache=True*)

Get a dictionary of short reference tags keyed to the full reference string. If requested also get a dictionary of reference tags keyed to NASA ADS URLs for the given reference. This uses the function *psrqpy.utils.get_references()*.

Parameters

- **useads** (*bool*) – Set this to True to get the NASA ADS reference URLs. Defaults to False.
- **cache** (*bool*) – The flag sets whether or not to use a pre-cached database of references. Defaults to True.

get_version

Return a string with the ATNF version number, or None if not found.

Returns the ATNF version number.

Return type str

gw_ellipticity()
Return the $l = m = 2$ mass quadrupoles based on the spin-down limits for the pulsars in the catalogue using *h0_to_ellipticity()*.

Returns

a table containing PSRJ names and ε values as a column called ELL.

Return type astropy.table.Table

gw_mass_quadrupole()
Return the $l = m = 2$ mass quadrupoles based on the spin-down limits for the pulsars in the catalogue using *h0_to_q22()*.

Returns

a table containing PSRJ names and Q_{22} values as a column called Q22.

Return type astropy.table.Table

include_errs
Return a boolean stating whether errors are to be included.

load(*fname*)
Load a previously saved pickle of this class.

Parameters **fname** (*str*) – the filename of the pickled object

num_pulsars
Return the number of pulsars found in with query

pandas
Return the query table as a pandas.DataFrame.

parse_assoc()
Parse default string representing source associations, extracting (first) value and reference. Multiple values and references currently not supported.

parse_bincomp()
Parse default string representing source companion type, extracting (first) value and reference. Multiple values and references currently not supported.

parse_conditions(*psrtype=None*, *assoc=None*, *bincomp=None*)
Parse a string of conditions, i.e., logical statements with which to apply to a catalogue query, e.g., `condition = 'f0 > 2.5 && assoc(GC)'`, so that they are in the format required for the query URL.

Parameters

- **psrtype** (*list, str*) – a list of strings, or single string, of conditions on the type of pulsars to return (logical AND will be used for any listed types)
- **assoc** (*list, str*) – a list of strings, or single string, of conditions on the associations of pulsars to return (logical AND will be used for any listed associations)
- **bincomp** (*list, str*) – a list of strings, or single string, of conditions on the binary companion types of pulsars to return (logical AND will be used for any listed associations)
- **exactmatch** (*bool*) – a boolean stating whether assciations and types given as the condition should be an exact match

Returns a string with the format required for use in QUERY_URL

Return type str

parse_ref(*refs*, *useads=False*)
This function takes a short format reference string from the ATNF Pulsar Catalogue and returns the full format reference. It can also return a NASA ADS URL if requested and present.

Parameters

- **refs** (*str, array_like*) – a single short reference string, or an array of reference strings.
- **useads** (*bool*) – Set whether or not to also return a NASA ADS reference URL if present.

Returns a single full reference string, or an array of full reference strings. If NASA ADS URLs are requested, each return value will be a tuple containing the reference string and URL.

Return type array_like

parse_type()
Parse default string representing source type, extracting (first) value and reference. Multiple values and references currently not supported.

parse_types()
Parse information in 'ASSOC', 'TYPE', and 'BINCOMP', as described in http://www.atnf.csiro.au/research/pulsar/psrcat/psrcat_help.html#psr_types.

ppdot(*intrinsicpdot=False*, *excludeGCs=False*, *showtypes=[]*, *showGCs=False*, *showSNRs=False*, *markertypes={}*, *deathline=True*, *deathmodel='Ip'*, *filldeath=True*, *filldeathtype={}*, *showtau=True*, *brakingidx=3*, *tau=None*, *showB=True*, *Bfield=None*, *pdotlims=None*, *periodlims=None*, *usecondition=True*, *usepsrs=True*, *rcparams={}*)
Draw a lovely period vs period derivative diagram.

Parameters

- **intrinsicpdot** (*bool*) – use the intrinsic period derivative corrected for the Shklovskii effect rather than the observed value. Defaults to False.
- **excludeGCs** (*bool*) – exclude globular cluster pulsars as their period derivatives can be contaminated by intra-cluster accelerations. Defaults to False.
- **showtypes** (*list, str*) – a list of pulsar types to highlight with markers in the plot. These can contain any of the following: `BINARY`, `HE`, `NRAD`, `RRAT`, `XINS`, `AXP` or `SGR`, or `ALL` to show all types. Default to showing no types.
- **showGCs** (*bool*) – show markers to denote the pulsars in globular clusters. Defaults to False.
- **showSNRs** (*bool*) – show markers to denote the pulsars with supernova remnants associated with them. Defaults to False.
- **markertypes** (*dict*) – a dictionary of marker styles and colors keyed to the pulsar types above
- **deathline** (*bool*) – draw the pulsar death line. Defaults to True.
- **deathmodel** (*str*) – the type of death line to draw based on the models in *psrqpy.utils.death_line()*. Defaults to `'Ip'`.
- **filldeath** (*bool*) – set whether to fill the pulsar graveyard under the death line. Defaults to True.
- **filldeathtype** (*dict*) – a dictionary of keyword arguments for the fill style of the pulsar graveyard.
- **showtau** (*bool*) – show lines for a selection of characteritic ages. Defaults to True, and shows lines for 10^5 through to 10^9 yrs with steps in powers of 10.
- **brakingidx** (*int*) – a braking index to use for the calculation of the characteristic age lines. Defaults to 3 for magnetic dipole radiation.

- **tau** (*list*) – a list of characteristic ages to show on the plot.
- **showB** (*bool*) – show lines of constant magnetic field strength. Defaults to True, and shows lines for 10^{10} through to 10^{14} gauss with steps in powers of 10.
- **Bfield** (*list*) – a list of magnetic field strengths to plot.
- **periodlims** (*array_like*) – the [min, max] period limits to plot with
- **pdotlims** (*array_like*) – the [min, max] pdot limits to plot with
- **usecondition** (*bool*) – if True create the P-Pdot diagram only with pulsars that conform to the original query condition values. Defaults to True.
- **usepsrs** (*bool*) – if True create the P-Pdot diagram only with pulsars specified in the original query. Defaults to True.
- **rcparams** (*dict*) – a dictionary of `matplotlib.rcParams` setup parameters for the plot.

Returns the figure object

Return type `matplotlib.figure.Figure`

psrs
Return the name(s) of particular pulsars asked for in the query.

query_params
Return the parameters required for the query.

query_table(*query_params=None*, *usecondition=True*, *usepsrs=True*, *useseparation=True*)
Return an `astropy.table.Table` from the query with new parameters or conditions if given.

Parameters

- **query_params** (*str, list*) – a parameter, or list of parameters, to return from the query. If this is None then all parameters are returned.
- **usecondition** (*bool, str*) – If True then the condition parsed to the `psrqpy.QueryATNF`: class will be used when returning the table. If False no condition will be applied to the returned table. If a string is given then that will be the assumed condition string.
- **usepsrs** (*bool, str*) – If True then the list of pulsars parsed to the `psrqpy.QueryATNF`: class will be used when returning the table. If False then all pulsars in the catalogue will be returned. If a string, or list of strings, is given then that will be assumed to be a set of pulsar names to return.
- **useseparation** (*bool*) – If True and a set of sky coordinates and radius around which to return pulsars was set in the `psrqpy.QueryATNF`: class then only pulsars within the given radius of the sky position will be returned. Otherwise all pulsars will be returned.

Returns a table of the pulsar data returned by the query.

Return type `astropy.table.Table`

save(*fname*)
Output the *QueryATNF* instance to a pickle file for future loading.

Parameters **fname** (*str*) – the filename to output the pickle to

set_derived()
Compute any derived parameters and add them to the class.

These calculations are based on those in the *readCatalogue.c* and *defineParameters.c* files from the *PSRCAT* code.

sort(*sort_attr='JNAME'*, *sort_order='asc'*, *inplace=False*)
Sort the generated catalogue `DataFrame` on a given attribute and in either ascending or descending order.

Parameters

- **sort_attr** (*str*) – The parameter on which to perform the sorting of the query output. Defaults to 'JNAME'.
- **sort_order** (*str*) – Set to 'asc' to sort the parameter values in ascending order, or 'desc' to sort in descending order. Defaults to ascending.
- **inplace** (*bool*) – If True, and sorting the class' internal `DataFrame`, then the sorting will be done in place without returning a copy of the table, otherwise a sorted copy of the table will be returned.

Returns a table containing the sorted catalogue.

Return type `DataFrame`

table
Return a `astropy.table.Table` based on the query.

update(*column*, *name=None*, *overwrite=False*)
Update a column in the internal `pandas.DataFrame` table using `pandas.DataFrame.update()`. If the column does not exist, it will be added to the table.

Parameters

- **column** (`pandas.Series`) – a named column of values.
- **name** (*str*) – the name of the column (required if *column* is not a `pandas.Series`, or to overwrite the current column name)
- **overwrite** (*bool*) – set whether to overwrite non-NA values or not if the column already exists. Defaults to False, so non-NA values will not be overwritten.

6.2 Pulsars

The classes defined here are hold information on an individual pulsar or an interable list of pulsars.

class Pulsar(*psrname*, *query=None*, ***kwargs*)
Bases: `object`

An object to hold a single pulsar. The class requires a pulsar name. The pulsar parameters are set as attributes of the class.

Parameters

- **psrname** (*str*) – a string containing a pulsar name
- **query** (`psrqpy.QueryATNF`) – a query

Additional keyword arguments are any of the valid queriable pulsar parameters.

items()
Return a list of the class attribute values.

keys()
Return a list of the class attribute names for allowed pulsar parameters.

name
Return the pulsar name

class Pulsars
Bases: `object`

Class to contain multiple *Pulsar* objects.

add_pulsar (*psr*)
Add a pulsar into the object.

Parameters **psr** (*Pulsar*, *Pulsars*) – a *Pulsar* object, or *Pulsars* object

pop (*psrname*)
Remove a pulsar from the object and return the removed pulsar.

Parameters **psrname** (*str*) – a string with the name of a pulsar

remove_pulsar (*psrname*)
Remove a pulsar from the object. Only do one at a time.

Parameters **psrname** (*str*) – a string with the name of a pulsar

6.3 Setup constants

This submodule sets up common constants for use, such as the allowed pulsar parameters and various URLs used for queries.

ADS_URL = 'https://ui.adsabs.harvard.edu/#abs/{}/'
The URL for the NASA ADS.

ATNF_BASE_URL = 'http://www.atnf.csiro.au/people/pulsar/psrcat/'
The ATNF pulsar catalogue base URL.

ATNF_TARBALL = 'http://www.atnf.csiro.au/people/pulsar/psrcat/downloads/psrcat_pkg.tar.g
The name of the tarball containing the entire catalogue database.

GC_URL = 'http://www.naic.edu/~pfreire/GCpsr.txt'
Paolo Freire's globular cluster pulsar table URL

GLITCH_URL = 'http://www.jb.man.ac.uk/pulsar/glitches/gTable.html'
The Jodrell Bank glitch catalogue table URL.

MSP_URL = 'http://astro.phys.wvu.edu/GalacticMSPs/GalacticMSPs.txt'
Dunc Lorimer's MSP table URL

PSR_ALL = {'A1': {'err': True, 'ref': True, 'units': 's'}, 'A12DOT': {'err': True,
A dictionary of allowed pulsars parameters (e.g., name, position, distance...)

Each parameter name key gives a dictionary containing the keys:

- `ref` (bool) - True if the parameter has an associated reference with it
- `err` (bool) - True if the parameter has an associated error value
- `units` (str) - a string with the parameters units that can be parsed by `Unit`

The allowed parameters and their units are given here.

PSR_ASSOC_TYPE = ['AXP', 'EXGAL', 'GC', 'GRS', 'OPT', 'PWN', 'SNR', 'XRS']
Other sources/objects associated with the pulsar.

PSR_BINARY_TYPE = ['MS', 'NS', 'CO', 'He', 'UL']
Binary companion types for use in `bincomp()` when setting logical conditions.

PSR_TYPE = ['AXP', 'BINARY', 'HE', 'NRAD', 'RADIO', 'RRAT', 'XINS']
Allowed pulsar types for use in `type()` when setting logical conditions.

6.4 Utility functions

A selection of useful functions used by the module.

B_field(*period*, *pdot*)

Function defining the polar magnetic field strength at the surface of the pulsar in gauss (Equation 5.12 of Lyne & Graham-Smith, Pulsar Astronmy, 2nd edition) with

$$B = 3.2 \times 10^{19} \sqrt{P\dot{P}}$$

NaNs are returned for any negative period derivates, or NaN imput values.

Parameters

- **period** (*float*) – a pulsar period (s)
- **pdot** (*float*) – a period derivative

Returns the magnetic field strength in gauss.

Return type float

B_field_pdot(*period*, *Bfield=10000000000.0*)

Function to get the period derivative from a given pulsar period and magnetic field strength using

$$\dot{P} = \frac{1}{P}\left(\frac{B}{3.2 \times 10^{19}}\right)^2$$

Parameters

- **period** (list, ndarray) – a list of period values
- **Bfield** (*float*) – the polar magnetic field strength (Defaults to 10^{10} G)

Returns an array of period derivatives

Return type numpy.ndarray

age_pdot(*period*, *tau=1000000.0*, *braking_idx=3.0*)

Function returning the period derivative for a pulsar with a given period and characteristic age, using

$$\dot{P} = \frac{P}{\tau(n-1)}$$

Parameters

- **period** (list, numpy.ndarray) – the pulsar period in seconds
- **tau** (*float*) – the characteristic age in years
- **braking_idx** (*float*) – the pulsar braking index (defaults to $n = 3$)

Returns an array of period derivatives.

Return type numpy.ndarray

characteristic_age(*period*, *pdot*, *braking_idx=3.0*)

Function defining the characteristic age of a pulsar. Returns the characteristic age in years using

$$\tau = \frac{P}{\dot{P}(n-1)}$$

NaNs are returned for any negative period derivates, or NaN imput values.

Parameters

- **period** (*float, array_like*) – the pulsar period in seconds
- **pdot** (*float, array_like*) – the pulsar period derivative
- **braking_idx** (*float*) – the pulsar braking index (defaults to $n = 3$)

Returns the characteristic age in years

Return type float

check_update()
Check if the ATNF Pulsar Catalogue has been updated compared to the version in the cache.

Returns True if the cache can be updated.

Return type bool

condition(*table*, *expression*, *exactMatch=False*)
Apply a logical expression to a table of values.

Parameters

- **table** (astropy.table.Table or pandas.DataFrame) – a table of pulsar data
- **expression** (str, ndarray) – a string containing a set of logical conditions with respect to pulsar parameter names (also containing conditions allowed when accessing the ATNF Pulsar Catalogue), or a boolean array of the same length as the table.
- **exactMatch** (*bool*) – set to true to exactly match some string comparison expressions, e.g., if asking for *'ASSOC(SNR)'* and *exactMatch* is True then only pulsar with an association that is just *'SNR'* will be returned, whereas if it is False then there could be multiple associations including *'SNR'*.

Returns the table of values conforming to the input condition. Depending on the type of input table the returned table will either be a astropy.table.Table or pandas.DataFrame.

Return type astropy.table.Table or pandas.DataFrame

Example

Some examples of this might are:

1. finding all pulsars with frequencies greater than 100 Hz

```
>>> newtable = condition(psrtable, 'F0 > 100')
```

2. finding all pulsars with frequencies greater than 50 Hz and period derivatives less than 1e-15 s/s.

```
>>> newtable = condition(psrtable, '(F0 > 50) & (P1 < 1e-15)')
```

3. finding all pulsars in binary systems

```
>>> newtable = condition(psrtable, 'TYPE(BINARY)')
```

4. parsing a boolean array equivalent to the first example

```
>>> newtable = condition(psrtable, psrtable['F0'] > 100)
```

death_line(*logP*, *linemodel='Ip'*, *rho6=1.0*)
The pulsar death line. Returns the base-10 logarithm of the period derivative for the given values of the period.

Parameters

- **logP** (list, ndarray) – the base-10 log values of period.
- **linemodel** (*str*) – a string with one of the above model names. Defaults to 'Ip'.
- **rho6** (*float*) – the value of the ρ_6 parameter from [ZHM] . Defaults to 1 is, which is equivalent to 10^6 cm.

Returns a vector of period derivative values

Return type numpy.ndarray

Note: The death line models can be:

- 'I' - Equation 3 of [ZHM]
- 'Ip' - Equation 4 of [ZHM]
- 'II' - Equation 5 of [ZHM]
- 'IIp' - Equation 6 of [ZHM]
- 'III' - Equation 8 of [ZHM]
- 'IIIp' - Equation 9 of [ZHM]
- 'IV' - Equation 10 of [ZHM]
- 'IVp' - Equation 11 of [ZHM]

---

ellipticity_to_q22 (*ellipticity*, *Izz=1e+38*)

Convert a fiducial ellipticity ε of the pulsar to the equivalent $l = m = 2$ mass quadrupole Q_{22} using (see Equation A3 of [LVC]):

$$Q_{22} = \varepsilon I_{zz} \sqrt{\frac{15}{8\pi}}$$

Parameters

- **ellipticity** (list, ndarray) – a list of ellipticity values.
- **Izz** (*float*) – the moment of inertia of the principle axis (defaults to 1e38 kg m^2).

Returns an array of Q_{22} values.

Return type numpy.ndarray

fdot_to_pdot (*fdot*, *period=None*, *frequency=None*)

Convert frequency derivatives to period derivatives. Either periods or frequencies must be supplied.

$$\dot{P} = -\frac{\dot{f}}{f^2}$$

Parameters

- **fdot** (list, ndarray) – a list of frequency derivative values.
- **period** (list, ndarray) – a list of periods (seconds)
- **frequency** (list, ndarray) – a list of frequencies (Hz)

Returns an array of period derivatives.

Return type numpy.ndarray

get_catalogue (*path_to_db=None*, *cache=True*, *update=False*, *pandas=False*)

This function will attempt to download and cache the entire ATNF Pulsar Catalogue database tarball, or read in database file from a provided path. The database will be converted into an astropy.table.Table or pandas.DataFrame. This was originally based on the method in the ATNF.ipynb notebook by Joshua Tan (@astrophysically).

Parameters

- **path_to_db** (*str*) – if the path to a local version of the database file is given then that will be read in rather than attempting to download the file (defaults to None).
- **cache** (*bool*) – cache the downloaded ATNF Pulsar Catalogue file. Defaults to True. This is ignored if *path_to_db* is given.

- **update** (*bool*) – if True the ATNF Pulsar Catalogue will be re-downloaded and cached if there has been a change compared to the currently cached version. This is ignored if *path_to_db* is given.
- **pandas** (*bool*) – if True the catalogue will be returned as a `pandas.DataFrame` rather than the default of an `Table`.

Returns a table containing the entire catalogue.

Return type `Table` or `DataFrame`

get_gc_catalogue()

Download and parse Paolo Freire's table of pulsars in globular clusters from http://www.naic.edu/~pfreire/GCpsr.txt. This will be returned as a `astropy.table.Table`, but with some additional methods to extract information for individual clusters. If the webpage returned an error `None` is returned.

The additional methods of the returned `Table` are:

- `clusters`: by default this returns a list if the common names of the globular clusters in the table, or using the `ngc=True` argument will return the NGC names of all clusters.
- `cluster_pulsars`: returns a sub-table only containing pulsars within a supplied globular cluster name.

get_glitch_catalogue(*psr=None*)

Return a `Table` containing the Jodrell Bank pulsar glitch catalogue. If using data from the glitch catalogue then please cite Espinoza et al. (2011) and the URL http://www.jb.man.ac.uk/pulsar/glitches.html.

The output table will contain the following columns:

- *NAME*: the pulsars common name
- *JNAME*: the pulsar name based on J2000 coordinates
- *Glitch number*: the number of the glitch for a particular pulsar in chronological order
- *MJD*: the time of the glitch in Modified Julian Days
- *MJD_ERR*: the uncertainty on the glitch time in days
- *DeltaF/F*: the fractional frequency change
- *DeltaF/F_ERR*: the uncertainty on the fractional frequency change
- *DeltaF1/F1*: the fractional frequency derivative change
- *DeltaF1/F1_ERR*: the uncertainty on the fractional frequency derivative change
- *Reference*: the glitch publication reference

Parameters **psr** (*str*) – if a pulsar name is given then only the glitches for that pulsar are returned, otherwise all glitches are returned.

Returns a table containing the entire glitch catalogue.

Return type `Table`

Example

An example of using this to extract the glitches for the Crab Pulsar would be:

```
>>> import psrqpy
>>> gtable = psrqpy.get_glitch_catalogue(psr='J0534+2200')
>>> print("There have been {} glitches observed from the Crab pulsar".
↪format(len(gtable)))
27
```

get_msp_catalogue()
Download and parse Dunc Lorimer's catalogue of galactic millisecond pulsars. This makes use of code from the now deleted galmsps respository that scraped the database into JSON every month, contributed by @astrogewgaw before deletion via PR #88.

Returns a table the MSPs.

Return type Table

get_references(*useads=False*, *cache=True*, *updaterefcache=False*, *bibtex=False*, *showfails=False*)
Return a dictionary of paper reference in the ATNF catalogue. The keys are the ref strings given in the ATNF catalogue.

Note: The way that the ATNF references are stored has changed, so if you downloaded the catalogue with a version of psrqpy before v1.0.8 you may need to run this function with `updaterefcache=True` to allow references to work. You may also want to update the ATNF catalogue tarball with:

```
>>> import psrqpy
>>> psrqpy.QueryATNF(checkupdate=True)
```

Parameters

- **useads** (*bool*) – boolean to set whether to use the python mod:*ads* module to get the NASA ADS URL for the references.
- **cache** (*bool*) – use cached, or cache, the reference bundled with the catalogue tarball.
- **updaterefcache** (*bool*) – update the cached references.
- **bibtex** (*bool*) – if using ADS return the bibtex for the reference along with the ADS URL.
- **showfails** (*bool*) – if outputting NASA ADS references set this flag to True to output the reference tags of references that fail to be found (mainly for debugging purposes).

Returns a dictionary of references.

Return type dict

gw_h0_spindown_limit(*frequency*, *fdot*, *distance*, *Izz=1e+38*)
The gravitational-wave amplutitude at Earth assuming all rotational energy loss is radiated via emission from an $l = m = 2$ mass quadrupole mode, using (see Equation A7 of [LVC])

$$h_0^{\rm sd} = \frac{1}{d}\left(\frac{5}{2}\frac{GI_{zz}}{c^3}\frac{|\dot{f}|}{f}\right)^{1/2}$$

Parameters

- **frequency** (list, ndarray) – a list of frequencies (Hz).
- **fdot** (list, ndarray) – a list of frequency derivatives (Hz/s).
- **distance** (list, ndarray) – a list of distances (kpc).
- **Izz** (*float*) – the moment of inertia along the principle axis (defaults to 1e38 kg m^2).

Returns an array of spin-down limits.

Return type numpy.ndarray

gw_luminosity(*h0*, *frequency*, *distance*)
The gravitational-wave luminosity as derived from a given gravitatonal-wave amplitude measured at Earth, assuming all rotational energy loss is radiated via emission from an $l = m = 2$ mass quadrupole mode, using (see Equation A5 of [LVC])

$$\dot{E}_{\rm gw} = \frac{8\pi^2 c^3}{5G} f^2 h_0^2 d^2$$

Parameters

- **h0** (list, ndarray) – a list of gravitational-wave amplitudes.
- **frequency** (list, ndarray) – a list of frequencies (Hz).
- **distance** (list, ndarray) – a list of distances (kpc).

Returns an array of luminosities.

Return type numpy.ndarray

h0_to_ellipticity (*h0*, *frequency*, *distance*, *Izz=1e+38*)

Convert a gravitational-wave amplitude h_0 at Earth to an equivalent fiducial ellipticity ε of the pulsar using (see Equation A4 of [LVC]):

$$\varepsilon = h_0 \left(\frac{c^4 d}{16\pi^2 G f^2} \right) \sqrt{\frac{15}{8\pi}}$$

Parameters

- **h0** (list, ndarray) – a list of gravitational-wave amplitudes.
- **frequency** (list, ndarray) – a list of frequencies (Hz).
- **distance** (list, ndarray) – a list of distances (kpc).
- **Izz** (*float*) – the moment of inertia of the principle axis (defaults to 1e38 kg m^2).

Returns an array of Q_{22} values.

Return type numpy.ndarray

h0_to_q22 (*h0*, *frequency*, *distance*)

Convert a gravitational-wave amplitude h_0 at Earth to an equivalent $l = m = 2$ mass quadrupole Q_{22} of the pulsar using (see Equations A3 and A4 of [LVC]):

$$Q_{22} = h_0 \left(\frac{c^4 d}{16\pi^2 G f^2} \right) \sqrt{\frac{15}{8\pi}}$$

Parameters

- **h0** (list, ndarray) – a list of gravitational-wave amplitudes.
- **frequency** (list, ndarray) – a list of frequencies (Hz).
- **distance** (list, ndarray) – a list of distances (kpc).

Returns an array of Q_{22} values.

Return type numpy.ndarray

label_line (*ax*, *line*, *label*, *color='k'*, *fs=14*, *frachoffset=0.1*)

Add an annotation to the given line with appropriate placement and rotation.

Based on code from "How to rotate matplotlib annotation to match a line?" and this answer.

Parameters

- **ax** (matplotlib.axes.Axes) – Axes on which the label should be added.
- **line** (matplotlib.lines.Line2D) – Line which is being labeled.
- **label** (*str*) – Text which should be drawn as the label.
- **color** (*str*) – a color string for the label text. Defaults to 'k'
- **fs** (*int*) – the font size for the label text. Defaults to 14.
- **frachoffset** (*float*) – a number between 0 and 1 giving the fractional offset of the label text along the x-axis. Defaults to 0.1, i.e., 10%.

Returns an object containing the label information

Return type matplotlib.text.Text

pdot_to_fdot (*pdot*, *period=None*, *frequency=None*)
Convert period derivatives to frequency derivatives. Either periods or frequencies must be supplied.

$$\dot{f} = -\frac{\dot{P}}{P^2}$$

Parameters

- **pdot** (list, ndarray) – a list of period derivative values.
- **period** (list, ndarray) – a list of periods (seconds)
- **frequency** (list, ndarray) – a list of frequencies (Hz)

Returns an array of frequency derivatives.

Return type numpy.ndarray

q22_to_ellipticity (*q22*, *Izz=1e+38*)
Convert a $l = m = 2$ mass quadrupole Q_{22} to an equivalent fiducial ellipticity ε of the pulsar using (see Equation A3 of [LVC]):

$$\varepsilon = \frac{Q_{22}}{I_{zz}}\sqrt{\frac{8\pi}{15}}$$

Parameters

- **q22** (list, ndarray) – a list of mass quadrupole values (kg m^2).
- **Izz** (*float*) – the moment of inertia of the principle axis (defaults to 1e38 kg m^2).

Returns an array of ε values.

Return type numpy.ndarray

6.5 Notable changes between versions

6.5.1 [1.1.3] 2021-10-11

- Move scraping of Galactic MSP catalogue into PSRQPy. See #88.

6.5.2 [1.1.2] 2021-06-22

- Fix parsing of the galactic MSP table after changes to the upstream format. See #85.
- Remove explicit use of certain NumPy built-in types following their deprecation. See #83.

6.5.3 [1.1.1] 2021-03-19

- Add function to download and parse the galactic MSP table. See #80.
- Add functions to calculate derived gravitational wave parameters. See #79.

6.5.4 [1.1.0] 2021-03-18

The major change for this release is that it no longer supports Python 2.7, and only supports Python versions greater than 3.5.

- Add a function to download and parse the globular cluster pulsar table. See #77.

6.5.5 [1.0.11] 2020-10-19

- Allow `psrqpy` query to work using ATNF catalogue v1.64, which contains a position typo. See #74.
- More fixes to enable parsing of more references using NASA ADS and fixes changes that occurred with v1.64. See #70.

6.5.6 [1.0.10] 2020-10-08

- Allow the `Pulsar` objects to access parameter references. See #71.

6.5.7 [1.0.9] 2020-06-08

- A minor fix to get ADS references to work. See #66.

6.5.8 [1.0.8] 2020-06-05

Changes for this release:

- The way references are stored in the ATNF pulsar catalogue has changed, so the `get_references` function has been changed to work with this new format. This breaks that function for older cached catalogue, but a warning is provided. See #64.
- The astropy Galactocentric coordinates have been set to use default values from the pre-v4.0 release. See #65.

6.5.9 [1.0.7] 2020-03-29

Changes for this release:

- Fix an issue with Tables in Astropy v4.0 (see #61).

6.5.10 [1.0.6] 2020-03-19

Changes for this release:

- Adds position errors on the right ascension and declination when converted into degrees (see #60).

6.5.11 [1.0.5] 2019-09-04

Changes for this release:

- The version 1.61 of the ATNF Pulsar Catalogue containing the declinations with the unicode “” character in place of an ascii minus sign “-” has been fixed upstream, so the code fix to account for this has been removed (it was also causing issues with the Python 2 version of the code).

6.5.12 [1.0.4] 2019-09-04

Changes for this release:

- Update the P-Pdot diagram, so that the user can specify which pulsars to include, either via the pulsars passed to the original query request (which is the default option) or by directly passing names to the P-Pdot function. If no particular pulsars were requested then all pulsars will be included.

- Version 1.61 of the ATNF Pulsar Catalogue contains some declinations with the unicode “” character in place of an ascii minus sign “-”. This release switched the unicode “” to the “-”, so that the values can be parsed to float numbers. This version of the ATNF Pulsar Catalogue also contains a couple of negative parallax values, so this release does not allow these to be used for calculating distances.

6.5.13 [1.0.3] 2019-07-29

Changes for this release:

- Updates to the `BarycentricTrueEcliptic` in astropy v3.2 caused test to fail (due to “correcting” bugs and including nutation). This version updates to use `BarycentricMeanEcliptic`, for which the tests no-longer fail. For earlier versions of astropy the former function is still used.
- The `version` attribute of the catalogue table is now set if loading a query from a previously read table (see #53).

6.5.14 [1.0.0] 2018-11-15

This release involves major changes to the API.

- PSRQpy now downloads the full ATNF Pulsar Catalogue database and stores it internally.
- PSRQpy will no longer generate queries of the ATNF Pulsar Catalogue via the web interface.
- Derived parameters are calculated internally rather than being values returned be the catalogue web interface.
- Full string paper references are no longer included in the output table, but can be returned using the `parse_ref` method of the `QueryATNF` class.

CHAPTER 7

Test suite

There are tests supplied that cover many of the functions within PSRQpy. These can be run from the base directory of the repository (after installing the pytest and pytest-socket modules, e.g., with `pip`) by just calling:

```
pytest
```

These tests are not included in the `pip` installed version of the code.

7.1 Copyright and referencing for the catalogue

Regarding the use of the catalogue and software behind it, the following statements apply:

> PSRCAT is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. PSRCAT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> PSRCAT makes use of "evaluateExpression: A Simple Expression Evaluator". Copyright © 1996 - 1999 Parsifal Software, All Rights Reserved.
>
> The programs and databases remain the property of the Australia Telescope National Facility, CSIRO, and are covered by the CSIRO Legal Notice and Disclaimer.
>
> If you make use of information from the ATNF Pulsar Catalogue in a publication, we would appreciate acknowledgement by reference to the publication "*The ATNF Pulsar Catalogue*", R. N. Manchester, G. B. Hobbs, A. Teoh & M. Hobbs, Astronomical Journal, 129, 1993-2006 (2005) and by quoting the web address http://www.atnf.csiro.au/research/pulsar/psrcat for updated versions.

7.2 Copyright & license for psrqpy

This code is licensed under the MIT License.

If making use of this code to access the catalogue, or produce plots, I would be grateful if (as well as citing the ATNF pulsar catalogue paper and URL given above) you consider citing the JOSS paper for this software:

```
@article{psrqpy,
  author = {{Pitkin}, M.},
   title = "{psrqpy: a python interface for querying the ATNF pulsar catalogue}",
  volume = 3,
  number = 22,
   pages = 538,
   month = feb,
    year = 2018,
 journal = "{Journal of Open Source Software}",
     doi = {10.21105/joss.00538},
     url = {https://doi.org/10.21105/joss.00538}
}
```


7.3 References

Bibliography

[ZHM] Zhang, Harding & Muslimov, *ApJ*, **531**, L135-L138 (2000), arXiv:astro-ph/0001341

[LVC] Abbott et al, *ApJ*, **879**, 28 (2019), arXiv:1902.08507

Python Module Index

p

Index