enterprise$_{e}xtensions Documentation$

Release 2.4.0

Stephen R. Taylor

May 20, 2022

CONTENTS:

CHAPTER

ONE

ENTERPRISE_EXTENSIONS

A set of extension codes, utilities, and scripts for the enterprise PTA analysis framework.

- Free software: MIT license
- Documentation: https://enterprise-extensions.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install enterprise_extensions, run this command in your terminal:

```
$ pip install enterprise_extensions
```

This is the preferred method to install enterprise_extensions, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for enterprise_extensions can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/stevertaylor/enterprise_extensions
```

Or download the tarball:

```
$ curl  -OL https://github.com/stevertaylor/enterprise_extensions/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use enterprise_extensions in a project:

```
import enterprise_extensions
```

CHAPTER

FOUR

ENTERPRISE_EXTENSIONS

4.1 enterprise_extensions package

4.1.1 Subpackages

enterprise_extensions.chromatic package

Submodules

enterprise_extensions.chromatic.chromatic module

enterprise_extensions.chromatic.chromatic.**chrom_dual_exp_cusp**(*toas*, *freqs*, *t0=54000*, *sign_param=- 1.0*, *log10_Amp_1=- 7*, *log10_tau_pre_1=1.7*, *log10_tau_post_1=1.7*, *log10_Amp_2=- 7*, *log10_tau_pre_2=1.7*, *log10_tau_post_2=1.7*, *symmetric=False*, *idx1=2*, *idx2=4*)

Chromatic exponential-cusp delay term in TOAs.

Parameters

- **t0** – time of exponential minimum [MJD]
- **tau_pre** – 1/e time of exponential before peak [s]
- **tau_post** – 1/e time of exponential after peak[s]
- **symmetric** – whether or not tau_pre = tau_post
- **log10_Amp** – amplitude of cusp
- **sign_param** – sign of waveform
- **idx** – index of chromatic dependence

Return wf delay time-series [s]

enterprise_extensions.chromatic.chromatic.**chrom_exp_cusp**(*toas*, *freqs*, *log10_Amp=- 7*, *sign_param=- 1.0*, *t0=54000*, *log10_tau_pre=1.7*, *log10_tau_post=1.7*, *symmetric=False*, *idx=2*)

Chromatic exponential-cusp delay term in TOAs.

Parameters

- **t0** – time of exponential minimum [MJD]
- **tau_pre** – 1/e time of exponential before peak [s]
- **tau_post** – 1/e time of exponential after peak[s]
- **symmetric** – whether or not tau_pre = tau_post
- **log10_Amp** – amplitude of cusp
- **sign_param** – sign of waveform
- **idx** – index of chromatic dependence

Return wf delay time-series [s]

enterprise_extensions.chromatic.chromatic.**chrom_exp_decay**(*toas*, *freqs*, *log10_Amp=- 7*, *sign_param=- 1.0*, *t0=54000*, *log10_tau=1.7*, *idx=2*)

Chromatic exponential-dip delay term in TOAs.

Parameters

- **t0** – time of exponential minimum [MJD]
- **tau** – 1/e time of exponential [s]
- **log10_Amp** – amplitude of dip
- **sign_param** – sign of waveform
- **idx** – index of chromatic dependence

Return wf delay time-series [s]

enterprise_extensions.chromatic.chromatic.**chrom_yearly_sinusoid**(*toas*, *freqs*, *log10_Amp=- 7*, *phase=0*, *idx=2*)

Chromatic annual sinusoid.

Parameters

- **log10_Amp** – amplitude of sinusoid
- **phase** – initial phase of sinusoid
- **idx** – index of chromatic dependence

Return wf delay time-series [s]

enterprise_extensions.chromatic.chromatic.**chromatic_quad_basis**(*toas*, *freqs*, *idx=4*)

Basis for chromatic quadratic function.

Parameters **idx** – index of chromatic dependence

Return ret normalized quadratic basis matrix [Ntoa, 3]

enterprise_extensions.chromatic.chromatic.**chromatic_quad_prior**(*toas*)

Prior for chromatic quadratic function.

Return prior prior-range for quadratic coefficients

enterprise_extensions.chromatic.chromatic.**dm_annual_signal**(*idx=2*, *name='dm_s1yr'*)

Returns chromatic annual signal (i.e. TOA advance):

Parameters

- **idx** – index of radio frequency dependence (i.e. DM is 2). If this is set to 'vary' then the index will vary from 1 - 6
- **name** – Name of signal

Return dm1yr chromatic annual waveform.

enterprise_extensions.chromatic.chromatic.**dm_dual_exp_cusp**(*tmin*, *tmax*, *idx1=2*, *idx2=4*, *sign='negative'*, *symmetric=False*, *name='dual_dm_cusp'*)

Returns chromatic exponential cusp (i.e. TOA advance):

Parameters

- **tmax** (*tmin,*) – search window for exponential cusp time.
- **idx** – index of radio frequency dependence (i.e. DM is 2). If this is set to 'vary' then the index will vary from 1 - 6
- **sign** – set sign of dip: 'positive', 'negative', or 'vary'
- **name** – Name of signal

Return dmexp chromatic exponential dip waveform.

enterprise_extensions.chromatic.chromatic.**dm_exponential_cusp**(*tmin*, *tmax*, *idx=2*, *sign='negative'*, *symmetric=False*, *name='dm_cusp'*)

Returns chromatic exponential cusp (i.e. TOA advance):

Parameters

- **tmax** (*tmin,*) – search window for exponential cusp time.
- **idx** – index of radio frequency dependence (i.e. DM is 2). If this is set to 'vary' then the index will vary from 1 - 6
- **sign** – set sign of dip: 'positive', 'negative', or 'vary'
- **name** – Name of signal

Return dmexp chromatic exponential dip waveform.

enterprise_extensions.chromatic.chromatic.**dm_exponential_dip**(*tmin*, *tmax*, *idx=2*, *sign='negative'*, *name='dmexp'*)

Returns chromatic exponential dip (i.e. TOA advance):

Parameters

- **tmax** (*tmin,*) – search window for exponential dip time.
- **idx** – index of radio frequency dependence (i.e. DM is 2). If this is set to 'vary' then the index will vary from 1 - 6
- **sign** – set sign of dip: 'positive', 'negative', or 'vary'
- **name** – Name of signal

Return dmexp chromatic exponential dip waveform.

enterprise_extensions.chromatic.chromatic.**dmx_delay**(*toas*, *freqs*, *dmx_ids*, ***kwargs*)

Delay in DMX model of DM variations.

Parameters

- **dmx_ids** – dictionary of DMX data for each pulsar from parfile

- **kwargs** – dictionary of enterprise DMX parameters

Return wf DMX signal

enterprise_extensions.chromatic.chromatic.**dmx_signal**(*dmx_data*, *name='dmx_signal'*)

Returns DMX signal:

Parameters

- **dmx_data** – dictionary of DMX data for each pulsar from parfile.
- **name** – Name of signal.

Return dmx_sig dmx signal waveform.

enterprise_extensions.chromatic.solar_wind module

enterprise_extensions.frequentist package

Submodules

enterprise_extensions.frequentist.F_statistic module

class enterprise_extensions.frequentist.F_statistic.**FpStat**(*psrs*, *params=None*, *psrTerm=True*, *bayesephem=True*, *pta=None*, *tnequad=False*)

Bases: object

Class for the Fp-statistic.

Parameters

- **psrs** – List of *enterprise* Pulsar instances.
- **noisedict** – Dictionary of white noise parameter values. Default=None
- **psrTerm** – Include the pulsar term in the CW signal model. Default=True
- **bayesephem** – Include BayesEphem model. Default=True

compute_Fp(*fgw*)

Computes the Fp-statistic.

Parameters **fgw** – GW frequency

Returns fstat: value of the Fp-statistic at the given frequency

compute_fap(*fgw*)

Compute false alarm rate for Fp-Statistic. We calculate the log of the FAP and then exponentiate it in order to avoid numerical precision problems

Parameters **fgw** – GW frequency

Returns False alarm probability as defined in Eq (64) of Ellis, Seiemens, Creighton (2012)

get_Nmats()

Makes the Nmatrix used in the fstatistic

enterprise_extensions.frequentist.F_statistic.**innerProduct_rr**(*x*, *y*, *Nmat*, *Tmat*, *Sigma*, *TNx=None*, *TNy=None*)

Compute inner product using rank-reduced approximations for red noise/jitter Compute: x^T N^{-1} y - x^T N^{-1} T Sigma^{-1} T^T N^{-1} y

Parameters

- **x** – vector timeseries 1
- **y** – vector timeseries 2
- **Nmat** – white noise matrix
- **Tmat** – Modified design matrix including red noise/jitter
- **Sigma** – Sigma matrix (varphi^{-1} + T^T N^{-1} T)
- **TNx** – T^T N^{-1} x precomputed
- **TNy** – T^T N^{-1} y precomputed

Returns inner product (x|y)

enterprise_extensions.frequentist.F_statistic.**make_Nmat**(*phiinv*, *TNT*, *Nvec*, *T*)

enterprise_extensions.frequentist.Fe_statistic module

class enterprise_extensions.frequentist.Fe_statistic.**FeStat**(*psrs*, *params=None*)

Bases: object

Class for the Fe-statistic.

Parameters

- **psrs** – List of *enterprise* Pulsar instances.
- **params** – Dictionary of noise parameters.

compute_Fe(*f0*, *gw_skyloc*, *brave=False*, *maximized_parameters=False*)

Computes the Fe-statistic (see Ellis, Siemens, Creighton 2012).

Parameters

- **f0** – GW frequency
- **gw_skyloc** – 2x{number of sky locations} array containing [theta, phi] for each queried sky location, where theta=pi/2-DEC, phi=RA, for singlge sky location use gw_skyloc= np.array([[theta,],[phi,]])
- **brave** – Skip sanity checks in linalg for speedup if True.
- **maximized_parameters** – Calculate maximized extrinsic parameters if True.

Returns fstat: value of the Fe-statistic

If maximized_parameters=True also returns inc_max: Maximized value of inclination psi_max: Maximized value of polarization angle phase0_max: Maximized value of initial fhase h_max: Maximized value of amplitude

get_Nmats()

Makes the Nmatrix used in the fstatistic

enterprise_extensions.frequentist.Fe_statistic.**innerProduct_rr**(*x*, *y*, *Nmat*, *Tmat*, *Sigma*, *TNx=None*, *TNy=None*, *brave=False*)

Compute inner product using rank-reduced approximations for red noise/jitter Compute: x^T N^{-1} y - x^T N^{-1} T Sigma^{-1} T^T N^{-1} y

Parameters

- **x** – vector timeseries 1
- **y** – vector timeseries 2
- **Nmat** – white noise matrix
- **Tmat** – Modified design matrix including red noise/jitter
- **Sigma** – Sigma matrix (varphi^{-1} + T^T N^{-1} T)
- **TNx** – T^T N^{-1} x precomputed
- **TNy** – T^T N^{-1} y precomputed

Returns inner product (x|y)

enterprise_extensions.frequentist.Fe_statistic.**make_Nmat**(*phiinv*, *TNT*, *Nvec*, *T*)

enterprise_extensions.frequentist.optimal_statistic module

4.1.2 Submodules

4.1.3 enterprise_extensions.blocks module

enterprise_extensions.blocks.**bwm_block**(*Tmin*, *Tmax*, *amp_prior='log-uniform'*, *skyloc=None*, *logmin=-18*, *logmax=-11*, *name='bwm'*)

Returns deterministic GW burst with memory model: 1. Burst event parameterized by time, sky location, polarization angle, and amplitude

Parameters

- **Tmin** – Min time to search, probably first TOA (MJD).
- **Tmax** – Max time to search, probably last TOA (MJD).
- **amp_prior** – Prior on log10_A. Default if "log-uniform". Use "uniform" for upper limits.
- **skyloc** – Fixed sky location of BWM signal search as [cos(theta), phi]. Search over sky location if `None` given.
- **logmin** – log of minimum BWM amplitude for prior (log10)
- **logmax** – log of maximum BWM amplitude for prior (log10)
- **name** – Name of BWM signal.

enterprise_extensions.blocks.**bwm_sglpsr_block**(*Tmin*, *Tmax*, *amp_prior='log-uniform'*, *logmin=-17*, *logmax=-12*, *name='ramp'*, *fixed_sign=None*)

enterprise_extensions.blocks.**chromatic_noise_block**(*gp_kernel='nondiag'*, *psd='powerlaw'*, *nondiag_kernel='periodic'*, *prior='log-uniform'*, *dt=15*, *df=200*, *idx=4*, *include_quadratic=False*, *Tspan=None*, *name='chrom'*, *components=30*, *coefficients=False*)

Returns GP chromatic noise model :

1. Chromatic modeled with user defined PSD with 30 sampling frequencies. Available PSDs are ['powerlaw', 'turnover' 'spectrum']

Parameters

- **gp_kernel** – Whether to use a diagonal kernel for the GP. ['diag','nondiag']
- **nondiag_kernel** – Which nondiagonal kernel to use for the GP. ['periodic','sq_exp','periodic_rfband','sq_exp_rfband']
- **psd** – PSD to use for common red noise signal. Available options are ['powerlaw', 'turnover' 'spectrum']
- **prior** – What type of prior to use for amplitudes. ['log-uniform','uniform']
- **dt** – time-scale for linear interpolation basis (days)
- **df** – frequency-scale for linear interpolation basis (MHz)
- **idx** – Index of radio frequency dependence (i.e. DM is 2). Any float will work.
- **include_quadratic** – Whether to include a quadratic fit.
- **name** – Name of signal
- **Tspan** – Tspan from which to calculate frequencies for PSD-based GPs.
- **components** – Number of frequencies to use in 'diag' GPs.
- **coefficients** – Whether to keep coefficients of the GP.

enterprise_extensions.blocks.**common_red_noise_block**(*psd='powerlaw'*, *prior='log-uniform'*, *Tspan=None*, *components=30*, *combine=True*, *log10_A_val=None*, *gamma_val=None*, *delta_val=None*, *logmin=None*, *logmax=None*, *orf=None*, *orf_ifreq=0*, *leg_lmax=5*, *name='gw'*, *coefficients=False*, *pshift=False*, *pseed=None*)

Returns common red noise model:

1. Red noise modeled with user defined PSD with 30 sampling frequencies. Available PSDs are ['powerlaw', 'turnover' 'spectrum']

Parameters

- **psd** – PSD to use for common red noise signal. Available options are ['powerlaw', 'turnover' 'spectrum', 'broken_powerlaw']
- **prior** – Prior on log10_A. Default if "log-uniform". Use "uniform" for upper limits.
- **Tspan** – Sets frequency sampling f_i = i / Tspan. Default will use overall time span for individual pulsar.
- **log10_A_val** – Value of log10_A parameter for fixed amplitude analyses.
- **gamma_val** – Value of spectral index for power-law and turnover models. By default spectral index is varied of range [0,7]

- **delta_val** – Value of spectral index for high frequencies in broken power-law and turnover models. By default spectral index is varied in range [0,7]. :param logmin: Specify the lower bound of the prior on the amplitude for all psd but 'spectrum'. If psd=='spectrum', then this specifies the lower prior on log10_rho_gw
- **logmax** – Specify the lower bound of the prior on the amplitude for all psd but 'spectrum'. If psd=='spectrum', then this specifies the lower prior on log10_rho_gw
- **orf** – String representing which overlap reduction function to use. By default we do not use any spatial correlations. Permitted values are ['hd', 'dipole', 'monopole'].
- **orf_ifreq** – Frequency bin at which to start the Hellings & Downs function with numbering beginning at 0. Currently only works with freq_hd orf.
- **leg_lmax** – Maximum multipole of a Legendre polynomial series representation of the overlap reduction function [default=5]
- **pshift** – Option to use a random phase shift in design matrix. For testing the null hypothesis.
- **pseed** – Option to provide a seed for the random phase shift.
- **name** – Name of common red process

enterprise_extensions.blocks.**dm_noise_block**(*gp_kernel='diag'*, *psd='powerlaw'*, *nondiag_kernel='periodic'*, *prior='log-uniform'*, *dt=15*, *df=200*, *Tspan=None*, *components=30*, *gamma_val=None*, *coefficients=False*)

Returns DM noise model:

1. DM noise modeled as a power-law with 30 sampling frequencies

Parameters

- **psd** – PSD function [e.g. powerlaw (default), spectrum, tprocess]
- **prior** – Prior on log10_A. Default if "log-uniform". Use "uniform" for upper limits.
- **dt** – time-scale for linear interpolation basis (days)
- **df** – frequency-scale for linear interpolation basis (MHz)
- **Tspan** – Sets frequency sampling f_i = i / Tspan. Default will use overall time span for indivicual pulsar.
- **components** – Number of frequencies in sampling of DM-variations.
- **gamma_val** – If given, this is the fixed slope of the power-law for powerlaw, turnover, or tprocess DM-variations

enterprise_extensions.blocks.**red_noise_block**(*psd='powerlaw'*, *prior='log-uniform'*, *Tspan=None*, *components=30*, *gamma_val=None*, *coefficients=False*, *select=None*, *modes=None*, *wgts=None*, *combine=True*, *break_flat=False*, *break_flat_fq=None*, *logmin=None*, *logmax=None*, *dropout=False*, *k_threshold=0.5*)

Returns red noise model: Red noise modeled as a power-law with 30 sampling frequencies

Parameters

- **psd** – PSD function [e.g. powerlaw (default), turnover, spectrum, tprocess]
- **prior** – Prior on log10_A. Default if "log-uniform". Use "uniform" for upper limits.

- **Tspan** – Sets frequency sampling f_i = i / Tspan. Default will use overall time span for indivicual pulsar.
- **components** – Number of frequencies in sampling of red noise
- **gamma_val** – If given, this is the fixed slope of the power-law for powerlaw, turnover, or tprocess red noise
- **coefficients** – include latent coefficients in GP model?
- **dropout** – Use a dropout analysis for intrinsic red noise models. Currently only supports power law option.
- **k_threshold** – Threshold for dropout analysis.

enterprise_extensions.blocks.**white_noise_block**(*vary=False*, *inc_ecorr=False*, *gp_ecorr=False*, *efac1=False*, *select='backend'*, *tnequad=False*, *name=None*)

Returns the white noise block of the model:

1. EFAC per backend/receiver system
2. EQUAD per backend/receiver system
3. ECORR per backend/receiver system

Parameters

- **vary** – If set to true we vary these parameters with uniform priors. Otherwise they are set to constants with values to be set later.
- **inc_ecorr** – include ECORR, needed for NANOGrav channelized TOAs
- **gp_ecorr** – whether to use the Gaussian process model for ECORR
- **efac1** – use a strong prior on EFAC = Normal(mu=1, stdev=0.1)
- **tnequad** – Whether to use the TempoNest definition of EQUAD. Defaults to False to follow Tempo, Tempo2 and Pint definition.

4.1.4 enterprise_extensions.deterministic module

enterprise_extensions.deterministic.**CWSignal**(*cw_wf*, *ecc=False*, *psrTerm=False*, *name='cw'*)

enterprise_extensions.deterministic.**bwm_delay**(*toas*, *pos*, *log10_h=- 14.0*, *cos_gwtheta=0.0*, *gwphi=0.0*, *gwpol=0.0*, *t0=55000*, *antenna_pattern_fn=None*)

Function that calculates the earth-term gravitational-wave burst-with-memory signal, as described in: Seto et al, van haasteren and Levin, phsirkov et al, Cordes and Jenet. This version uses the F+/Fx polarization modes, as verified with the Continuous Wave and Anisotropy papers.

Parameters

- **toas** – Time-of-arrival measurements [s]
- **pos** – Unit vector from Earth to pulsar
- **log10_h** – log10 of GW strain
- **cos_gwtheta** – Cosine of GW polar angle
- **gwphi** – GW azimuthal polar angle [rad]
- **gwpol** – GW polarization angle

- **t0** – Burst central time [day]
- **antenna_pattern_fn** – User defined function that takes *pos*, *gwtheta*, *gwphi* as arguments and returns (fplus, fcross)

Returns the waveform as induced timing residuals (seconds)

enterprise_extensions.deterministic.**bwm_sglpsr_delay**(*toas*, *sign*, *log10_A=- 15*, *t0=55000*)

Function that calculates the earth-term gravitational-wave burst-with-memory signal for an optimally oriented source in a single pulsar

Parameters

- **toas** – Time-of-arrival measurements [s]
- **log10_A** – log10 of the amplitude of the ramp (delta_f/f)
- **t0** – Burst central time [day]

Returns the waveform as induced timing residuals (seconds)

enterprise_extensions.deterministic.**compute_eccentric_residuals**(*toas*, *theta*, *phi*, *cos_gwtheta*, *gwphi*, *log10_mc*, *log10_dist*, *log10_h*, *log10_F*, *cos_inc*, *psi*, *gamma0*, *e0*, *l0*, *q*, *nmax=400*, *pdist=1.0*, *pphase=None*, *pgam=None*, *psrTerm=False*, *tref=0*, *check=False*)

Simulate GW from eccentric SMBHB. Waveform models from Taylor et al. (2015) and Barack and Cutler (2004). WARNING: This residual waveform is only accurate if the GW frequency is not significantly evolving over the observation time of the pulsar.

Parameters

- **toa** – pulsar observation times
- **theta** – polar coordinate of pulsar
- **phi** – azimuthal coordinate of pulsar
- **gwtheta** – Polar angle of GW source in celestial coords [radians]
- **gwphi** – Azimuthal angle of GW source in celestial coords [radians]
- **log10_mc** – Base-10 lof of chirp mass of SMBMB [solar masses]
- **log10_dist** – Base-10 uminosity distance to SMBMB [Mpc]
- **log10_F** – base-10 orbital frequency of SMBHB [Hz]
- **inc** – Inclination of GW source [radians]
- **psi** – Polarization of GW source [radians]
- **gamma0** – Initial angle of periastron [radians]
- **e0** – Initial eccentricity of SMBHB
- **l0** – Initial mean anomoly [radians]
- **q** – Mass ratio of SMBHB
- **nmax** – Number of harmonics to use in waveform decomposition
- **pdist** – Pulsar distance [kpc]
- **pphase** – Pulsar phase [rad]

- **pgam** – Pulsar angle of periastron [rad]
- **psrTerm** – Option to include pulsar term [boolean]
- **tref** – Fidicuial time at which initial parameters are referenced [s]
- **check** – Check if frequency evolves significantly over obs. time

Returns Vector of induced residuals

enterprise_extensions.deterministic.**cw_block_circ**(*amp_prior='log-uniform'*, *dist_prior=None*, *skyloc=None*, *log10_fgw=None*, *psrTerm=False*, *tref=0*, *name='cw'*)

Returns deterministic, cirular orbit continuous GW model:

Parameters

- **amp_prior** – Prior on log10_h. Default is "log-uniform." Use "uniform" for upper limits, or "None" to search over log10_dist instead.
- **dist_prior** – Prior on log10_dist. Default is "None," meaning that the search is over log10_h instead of log10_dist. Use "log-uniform" to search over log10_h with a log-uniform prior.
- **skyloc** – Fixed sky location of CW signal search as [cos(theta), phi]. Search over sky location if `None` given.
- **log10_fgw** – Fixed log10 GW frequency of CW signal search. Search over GW frequency if `None` given.
- **ecc** – Fixed log10 distance to SMBHB search. Search over distance or strain if `None` given.
- **psrTerm** – Boolean for whether to include the pulsar term. Default is False.
- **name** – Name of CW signal.

enterprise_extensions.deterministic.**cw_block_ecc**(*amp_prior='log-uniform'*, *skyloc=None*, *log10_F=None*, *ecc=None*, *psrTerm=False*, *tref=0*, *name='cw'*)

Returns deterministic, eccentric orbit continuous GW model:

Parameters

- **amp_prior** – Prior on log10_h and log10_Mc/log10_dL. Default is "log-uniform" with log10_Mc and log10_dL searched over. Use "uniform" for upper limits, log10_h searched over.
- **skyloc** – Fixed sky location of CW signal search as [cos(theta), phi]. Search over sky location if `None` given.
- **log10_F** – Fixed log-10 orbital frequency of CW signal search. Search over orbital frequency if `None` given.
- **ecc** – Fixed eccentricity of SMBHB search. Search over eccentricity if `None` given.
- **psrTerm** – Boolean for whether to include the pulsar term. Default is False.
- **name** – Name of CW signal.

enterprise_extensions.deterministic.**cw_delay**(*toas*, *pos*, *pdist*, *cos_gwtheta=0*, *gwphi=0*, *cos_inc=0*, *log10_mc=9*, *log10_fgw=- 8*, *log10_dist=None*, *log10_h=None*, *phase0=0*, *psi=0*, *psrTerm=False*, *p_dist=1*, *p_phase=None*, *evolve=False*, *phase_approx=False*, *check=False*, *tref=0*)

Function to create GW incuced residuals from a SMBMB as defined in Ellis et. al 2012,2013.

Parameters

- **toas** – Pular toas in seconds
- **pos** – Unit vector from the Earth to the pulsar
- **pdist** – Pulsar distance (mean and uncertainty) [kpc]
- **cos_gwtheta** – Cosine of Polar angle of GW source in celestial coords [radians]
- **gwphi** – Azimuthal angle of GW source in celestial coords [radians]
- **cos_inc** – cosine of Inclination of GW source [radians]
- **log10_mc** – log10 of Chirp mass of SMBMB [solar masses]
- **log10_fgw** – log10 of Frequency of GW (twice the orbital frequency) [Hz]
- **log10_dist** – log10 of Luminosity distance to SMBMB [Mpc], used to compute strain, if not None
- **log10_h** – log10 of GW strain, used to compute distance, if not None
- **phase0** – Initial GW phase of source [radians]
- **psi** – Polarization angle of GW source [radians]
- **psrTerm** – Option to include pulsar term [boolean]
- **p_dist** – Pulsar distance parameter
- **p_phase** – Use pulsar phase to determine distance [radian]
- **evolve** – Option to include/exclude full evolution [boolean]
- **phase_approx** – Option to include/exclude phase evolution across observation time [boolean]
- **check** – Check if frequency evolves significantly over obs. time [boolean]
- **tref** – Reference time for phase and frequency [s]

Returns Vector of induced residuals

enterprise_extensions.deterministic.**fdm_block**(*Tmin*, *Tmax*, *amp_prior='log-uniform'*, *name='fdm'*, *amp_lower=- 18*, *amp_upper=- 11*, *freq_lower=- 9*, *freq_upper=- 7*, *use_fixed_freq=False*, *fixed_freq=- 8*)

Returns deterministic fuzzy dark matter model:

1. **FDM parameterized by frequency, phase,** and amplitude (mass and DM energy density).

Parameters

- **Tmin** – Min time to search, probably first TOA (MJD).
- **Tmax** – Max time to search, probably last TOA (MJD).
- **amp_prior** – Prior on log10_A.
- **logmin** – log of minimum FDM amplitude for prior (log10)
- **logmax** – log of maximum FDM amplitude for prior (log10)
- **name** – Name of FDM signal.

- **freq_lower** (*amp_upper, amp_lower, freq_upper,*) – The log-space bounds on the amplitude and frequency priors.
- **use_fixed_freq** – Whether to do a fixed-frequency run and not search over the frequency.
- **fixed_freq** – The frequency value to do a fixed-frequency run with.

enterprise_extensions.deterministic.**fdm_delay**(*toas*, *log10_A*, *log10_f*, *phase_e*, *phase_p*)

Function that calculates the earth-term gravitational-wave fuzzy dark matter signal, as described in: Kato et al. (2020).

Parameters

- **toas** – Time-of-arrival measurements [s]
- **log10_A** – log10 of GW strain
- **log10_f** – log10 of GW frequency
- **phase_e** – The Earth-term phase of the GW
- **phase_p** – The Pulsar-term phase of the GW

Returns the waveform as induced timing residuals (seconds)

enterprise_extensions.deterministic.**generalized_gwpol_psd**(*f*, *log10_A_tt=- 15*, *log10_A_st=- 15*, *log10_A_vl=- 15*, *log10_A_sl=- 15*, *kappa=3.3333333333333335*, *p_dist=1.0*)

PSD for a generalized mixture of scalar+vector dipole radiation and tensorial quadrupole radiation from SMBHBs.

4.1.5 enterprise_extensions.dropout module

enterprise_extensions.dropout.**Dropout_PhysicalEphemerisSignal**(*frame_drift_rate=enterprise.signals.parameter.Unifor*
d_jupiter_mass=enterprise.signals.parameter.Normal
d_saturn_mass=enterprise.signals.parameter.Normal,
d_uranus_mass=enterprise.signals.parameter.Normal
d_neptune_mass=enterprise.signals.parameter.Norma
jup_orb_elements=enterprise.signals.parameter.Unifo
sat_orb_elements=enterprise.signals.parameter.Unifo
inc_jupiter_orb=True,
inc_saturn_orb=False,
use_epoch_toas=True,
k_drop=enterprise.signals.parameter.Uniform,
k_threshold=0.5, *name=''*)

Class factory for dropout physical ephemeris model signal.

enterprise_extensions.dropout.**dropout_physical_ephem_delay**(*toas*, *planetssb*, *pos_t*, *frame_drift_rate=0*, *d_jupiter_mass=0*, *d_saturn_mass=0*, *d_uranus_mass=0*, *d_neptune_mass=0*, *jup_orb_elements=array([0.0, 0.0, 0.0, 0.0, 0.0, 0.0])*, *sat_orb_elements=array([0.0, 0.0, 0.0, 0.0, 0.0, 0.0])*, *inc_jupiter_orb=False*, *jup_orbelxyz=None*, *jup_mjd=None*, *inc_saturn_orb=False*, *sat_orbelxyz=None*, *sat_mjd=None*, *equatorial=True*, *k_drop=0.5*, *k_threshold=0.5*)

Dropout BayesEphem model. Switches BayesEphem on or off depending on whether k_drop exceeds k_threshold.

enterprise_extensions.dropout.**dropout_powerlaw**(*f*, *name*, *log10_A=- 16*, *gamma=5*, *dropout_psr='B1855+09'*, *k_drop=0.5*, *k_threshold=0.5*)

Dropout powerlaw for a stochastic process. Switches a stochastic process on or off in a single pulsar depending on whether k_drop exceeds k_threshold.

Parameters **dropout_psr** – Which pulsar to use a dropout switch on. The value 'all' will use the method on all pulsars.

4.1.6 enterprise_extensions.empirical_distr module

class enterprise_extensions.empirical_distr.**EmpiricalDistribution1D**(*param_name*, *samples*, *bins*)

Bases: object

Class used to define a 1D empirical distribution based on posterior from another MCMC.

Parameters

- **samples** – samples for hist
- **bins** – edges to use for hist (left and right) make sure bins cover whole prior!

draw()

logprob(*params*)

prob(*params*)

class enterprise_extensions.empirical_distr.**EmpiricalDistribution1DKDE**(*param_name*, *samples*, *minval=None*, *maxval=None*, *bandwidth=0.1*, *nbins=40*)

Bases: object

Minvals and maxvals should specify priors for these. Should make these required.

draw()

class enterprise_extensions.empirical_distr.**EmpiricalDistribution2D**(*param_names*, *samples*, *bins*)

Bases: object

Class used to define a 1D empirical distribution based on posterior from another MCMC.

Parameters

- **samples** – samples for hist
- **bins** – edges to use for hist (left and right) make sure bins cover whole prior!

draw()

logprob(*params*)

prob(*params*)

class enterprise_extensions.empirical_distr.**EmpiricalDistribution2DKDE**(*param_names*, *samples*, *minvals=None*, *maxvals=None*, *bandwidth=0.1*, *nbins=40*)

Bases: object

Minvals and maxvals should specify priors for these. Should make these required.

Parameters

- **param_names** – 2-element list of parameter names
- **samples** – samples, with dimension (2 x Nsamples)

Return distr list of empirical distributions

draw()

logprob(*params*)

prob(*params*)

enterprise_extensions.empirical_distr.**make_empirical_distributions**(*pta*, *paramlist*, *params*, *chain*, *burn=0*, *nbins=81*, *filename='distr.pkl'*, *return_distribution=True*, *save_dists=True*)

Utility function to construct empirical distributions.

Parameters

- **pta** – the pta object used to generate the posteriors
- **paramlist** – a list of parameter names, either single parameters or pairs of parameters
- **chain** – MCMC chain from a previous run
- **burn** – desired number of initial samples to discard
- **nbins** – number of bins to use for the empirical distributions

Return distr list of empirical distributions

enterprise_extensions.empirical_distr.**make_empirical_distributions_KDE**(*pta*, *paramlist*, *params*, *chain*, *burn=0*, *nbins=41*, *filename='distr.pkl'*, *bandwidth=0.1*, *return_distribution=True*, *save_dists=True*)

Utility function to construct empirical distributions.

Parameters

- **paramlist** – a list of parameter names, either single parameters or pairs of parameters
- **params** – list of all parameter names for the MCMC chain
- **chain** – MCMC chain from a previous run, has dimensions Nsamples x Nparams
- **burn** – desired number of initial samples to discard
- **nbins** – number of bins to use for the empirical distributions

Return distr list of empirical distributions

4.1.7 enterprise_extensions.gp_kernels module

enterprise_extensions.gp_kernels.**dmx_ridge_prior**(*avetoas*, *log10_sigma=- 7*)

DMX-like signal with Gaussian prior

enterprise_extensions.gp_kernels.**get_tf_quantization_matrix**(*toas*, *freqs*, *dt=2592000*, *df=None*, *dm=False*, *dm_idx=2*)

Quantization matrix in time and radio frequency to cut down on the kernel size.

enterprise_extensions.gp_kernels.**linear_interp_basis_dm**(*toas*, *freqs*, *dt=2592000*)

enterprise_extensions.gp_kernels.**linear_interp_basis_freq**(*freqs*, *df=64*)

Linear interpolation in radio frequency

enterprise_extensions.gp_kernels.**periodic_kernel**(*avetoas*, *log10_sigma=- 7*, *log10_ell=2*, *log10_gam_p=0*, *log10_p=0*)

Quasi-periodic kernel for DM

enterprise_extensions.gp_kernels.**se_dm_kernel**(*avetoas*, *log10_sigma=- 7*, *log10_ell=2*)

Squared-exponential kernel for DM

enterprise_extensions.gp_kernels.**se_kernel**(*avefreqs*, *log10_sigma=- 7*, *log10_lam=3*)

Squared-exponential kernel for FD

enterprise_extensions.gp_kernels.**sf_kernel**(*labels*, *log10_sigma=- 7*, *log10_ell=2*, *log10_ell2=4*, *log10_alpha_wgt=0*)

The product of a squared-exponential time kernel and a rational-quadratic frequency kernel.

enterprise_extensions.gp_kernels.**tf_kernel**(*labels*, *log10_sigma=- 7*, *log10_ell=2*, *log10_gam_p=0*, *log10_p=0*, *log10_ell2=4*, *log10_alpha_wgt=0*)

The product of a quasi-periodic time kernel and a rational-quadratic frequency kernel.

4.1.8 enterprise_extensions.hypermodel module

4.1.9 enterprise_extensions.model_orfs module

enterprise_extensions.model_orfs.**anis_orf**(*pos1*, *pos2*, *params*, ***kwargs*)

Anisotropic GWB spatial correlation function.

enterprise_extensions.model_orfs.**bin_orf**(*pos1*, *pos2*, *params*)

Agnostic binned spatial correlation function. Bin edges are placed at edges and across angular separation space. Changing bin edges will require manual intervention to create new function.

Param params inter-pulsar correlation bin amplitudes.

Author: S. R. Taylor (2020)

enterprise_extensions.model_orfs.**dipole_orf**(*pos1*, *pos2*)

Dipole spatial correlation function.

enterprise_extensions.model_orfs.**freq_hd**(*pos1*, *pos2*, *params*)

Frequency-dependent Hellings & Downs spatial correlation function. Implemented as a model that only enforces H&D inter-pulsar correlations after a certain number of frequencies in the spectrum. The first set of frequencies are uncorrelated.

Param params params[0] is the number of components in the stochastic process. params[1] is the frequency at which to start the H&D inter-pulsar correlations (indexing from 0).

Reference: Taylor et al. (2017), https://arxiv.org/abs/1606.09180 Author: S. R. Taylor (2020)

enterprise_extensions.model_orfs.**generalized_gwpol_psd**(*f*, *log10_A_tt=- 15*, *log10_A_st=- 15*, *alpha_tt=- 0.6666666666666666*, *alpha_alt=- 1*, *log10_A_vl=- 15*, *log10_A_sl=- 15*, *kappa=0*, *p_dist=1.0*)

General powerlaw spectrum allowing for existence of all possible modes of gravity as predicted by a general metric spacetime theory and generated by a binary system. The SL and VL modes' powerlaw relations are not normalized.

Param f A list of considered frequencies

Param log10_A_tt Amplitude of the tensor transverse mode

Param log10_A_st Amplitude of the scalar transverse mode

Param log10_A_vl Amplitude of the vector longitudinal mode

Param log10_A_sl Amplitude of the scalar longitudinal mode

Param kappa Relative amplitude of dipole radiation over quadrupolar radiation

Param p_dist Pulsar distance in kpc

Param alpha_tt spectral index of the TT mode.

Param alpha_alt spectral index of the non-Einsteinian modes.

Reference: Cornish et al. (2017), https://arxiv.org/abs/1712.07132 Author: S. R. Taylor, N. Laal (2020)

enterprise_extensions.model_orfs.**gt_orf**(*pos1*, *pos2*, *tau*)

General Transverse (GT) Correlations. This ORF is used to detect the relative significance of all possible correlation patterns induced by the most general family of transverse gravitational waves.

Param tau tau = 1 results in ST correlations while tau = -1 results in HD correlations.

Author: N. Laal (2020)

enterprise_extensions.model_orfs.**gw_dipole_orf**(*pos1*, *pos2*)

GW-dipole Correlations. Author: N. Laal (2020)

enterprise_extensions.model_orfs.**gw_monopole_orf**(*pos1*, *pos2*)

GW-monopole Correlations. This phenomenological correlation pattern can be used in Bayesian runs as the simplest type of correlations. Author: N. Laal (2020)

enterprise_extensions.model_orfs.**hd_orf**(*pos1*, *pos2*)

Hellings & Downs spatial correlation function.

enterprise_extensions.model_orfs.**legendre_orf**(*pos1*, *pos2*, *params*)

Legendre polynomial spatial correlation function. Assumes process normalization such that autocorrelation signature is 1. A separate function is needed to use a "split likelihood" model with this Legendre process decoupled from the autocorrelation signature ("zero_diag_legendre_orf").

Param params Legendre polynomial amplitudes describing the Legendre series approximation to the inter-pulsar correlation signature. H&D coefficients are a_0=0, a_1=0, a_2=0.3125, a_3=0.0875, ...

Reference: Gair et al. (2014), https://arxiv.org/abs/1406.4664 Author: S. R. Taylor (2020)

enterprise_extensions.model_orfs.**monopole_orf**(*pos1*, *pos2*)

Monopole spatial correlation function.

enterprise_extensions.model_orfs.**param_hd_orf**(*pos1*, *pos2*, *a=1.5*, *b=- 0.25*, *c=0.5*)

Pre-factor parametrized Hellings & Downs spatial correlation function.

Param a, b, c: coefficients of H&D-like curve [default=1.5,-0.25,0.5].

Reference: Taylor, Gair, Lentati (2013), https://arxiv.org/abs/1210.6014 Author: S. R. Taylor (2020)

enterprise_extensions.model_orfs.**spline_orf**(*pos1*, *pos2*, *params*)

Agnostic spline-interpolated spatial correlation function. Spline knots are placed at edges, zeros, and minimum of H&D curve. Changing locations will require manual intervention to create new function.

Param params spline knot amplitudes.

Reference: Taylor, Gair, Lentati (2013), https://arxiv.org/abs/1210.6014 Author: S. R. Taylor (2020)

enterprise_extensions.model_orfs.**st_orf**(*pos1*, *pos2*)

Scalar tensor correlations as induced by the breathing polarization mode of gravity. Author: N. Laal (2020)

enterprise_extensions.model_orfs.**zero_diag_bin_orf**(*pos1*, *pos2*, *params*)

Agnostic binned spatial correlation function. To be used in a "split likelihood" model with an additional common uncorrelated red process. The latter is necessary to regularize the overall Phi covariance matrix.

Param params inter-pulsar correlation bin amplitudes.

Author: S. R. Taylor (2020)

enterprise_extensions.model_orfs.**zero_diag_hd**(*pos1*, *pos2*)

Off-diagonal Hellings & Downs spatial correlation function. To be used in a "split likelihood" model with an additional common uncorrelated red process. The latter is necessary to regularize the overall Phi covariance matrix.

Author: S. R. Taylor (2020)

enterprise_extensions.model_orfs.**zero_diag_legendre_orf**(*pos1*, *pos2*, *params*)

Legendre polynomial spatial correlation function. To be used in a "split likelihood" model with an additional common uncorrelated red process. The latter is necessary to regularize the overall Phi covariance matrix.

Param params Legendre polynomial amplitudes describing the Legendre series approximation to the inter-pulsar correlation signature. H&D coefficients are a_0=0, a_1=0, a_2=0.3125, a_3=0.0875, ...

Reference: Gair et al. (2014), https://arxiv.org/abs/1406.4664 Author: S. R. Taylor (2020)

4.1.10 enterprise_extensions.model_utils module

4.1.11 enterprise_extensions.models module

4.1.12 enterprise_extensions.sampler module

4.1.13 enterprise_extensions.sky_scrambles module

enterprise_extensions.sky_scrambles.**compute_match**(*orf1*, *orf1_mag*, *orf2*, *orf2_mag*)

Computes the match between two different ORFs.

enterprise_extensions.sky_scrambles.**compute_orf**(*ptheta*, *pphi*)

Computes the ORF coefficient. Takes different input than utils.hd_orf().

Parameters

- **ptheta** – Array of values of pulsar positions theta
- **pphi** – Array of values of pulsar positions phi

Returns orf: ORF for the given positions orf_mag: Magnitude of the ORF

enterprise_extensions.sky_scrambles.**get_scrambles**(*psrs*, *N=500*, *Nmax=10000*, *thresh=0.1*, *filename='sky_scrambles.npz'*, *resume=False*)

Get sky scramble ORFs and matches.

Parameters

- **psrs** – List of pulsar objects
- **N** – Number of desired sky scrambles
- **Nmax** – Maximum number of tries to get independent scrambles
- **thresh** – Threshold value for match statistic.
- **filename** – Name of the file where the sky scrambles should be saved. Sky scrambles should be saved in *npz* file.
- **resume** – Whether to resume from an earlier run.

enterprise_extensions.sky_scrambles.**make_true_orf**(*psrs*)

Computes the ORF by looping over pulsar pairs

4.1.14 enterprise_extensions.timing module

enterprise_extensions.timing.**timing_block**(*tmparam_list=['RAJ', 'DECJ', 'F0', 'F1', 'PMRA', 'PMDEC', 'PX']*)

Returns the timing model block of the model :param tmparam_list: a list of parameters to vary in the model

enterprise_extensions.timing.**tm_delay**(*residuals*, *t2pulsar*, *tmparams_orig*, *tmparams*, *which='all'*)

Compute difference in residuals due to perturbed timing model.

Parameters

- **residuals** – original pulsar residuals from Pulsar object
- **t2pulsar** – libstempo pulsar object
- **tmparams_orig** – dictionary of TM parameter tuples, (val, err)
- **tmparams** – new timing model parameters, rescaled to be in sigmas
- **which** – option to have all or only named TM parameters varied

Returns difference between new and old residuals in seconds

CHAPTER

FIVE

CONTRIBUTING

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

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/stevertaylor/enterprise_extensions/issues.

If you are reporting a bug, please include:

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

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/stevertaylor/enterprise_extensions/issues.

If you are proposing a feature:

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

5.2 Get Started!

Ready to contribute? Here's how to set up *enterprise_extensions* for local development.

1. Fork the *enterprise_extensions* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/enterprise_extensions.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv enterprise_extensions
$ cd enterprise_extensions/
$ python setup.py develop
```

4. Create a branch for local development:

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

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 enterprise_extensions tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

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

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

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/stevertaylor/enterprise_extensions/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_enterprise_extensions
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Stephen R. Taylor <stephen.r.taylor@vanderbilt.edu>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

2.4.0 (2022-02-10) Use Timing Package (Tempo,Tempo2,Pint) definition of EQUAD. Enterprise has broken backwards compatibility, and here we use the *tnequad* flag to switch on the old definition.

2.3.4 (2021-11-02) Fix phase shift seed caching issue.

2.3.3 (2021-10-04) Fix bug in release build by adding ACE text file to MANIFEST.in.

2.3.2 (2021-10-04) Fix bug in HyperModel when using save_runtime_info.

2.3.1 (2021-09-30) Fix bugs associated with recent function additions. Added linting and mild PEP8 rules. Also removed older Python functionality which is no longer supported.

2.3.0 (2021-09-15)* Functionality added for NANOGrav 15yr dataset analyses. Outlier analysis software moved into separate package.

2.2.0 (2021-08-10) Version with outlier analysis.

0.9.1 (2021-05-06) 0.9.0 (2019-09-20) ——————

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

e

INDEX

E

F

G

H

I

L

M

P

R

S

T

W

Z