rss_ringoccs Documentation

Release v1.1

Dick French, Sophia Flury, Jolene Fong, Ryan Maquire, Glenn Ste

Aug 28, 2019

Contents:

1 rss_ringoccs package **1**

1.1 Subpackages 2

1.1.1 rss_ringoccs.calibration package 2

1.1.1.1 Submodules 2

1.1.2 rss_ringoccs.diffrec package 14

1.1.2.1 Submodules 15

1.1.3 rss_ringoccs.occgeo package 28

1.1.3.1 Submodules 28

1.1.4 rss_ringoccs.rsr_reader package 39

1.1.4.1 Submodules 40

1.1.5 rss_ringoccs.scatter package 41

1.1.5.1 Submodules 41

1.1.6 rss_ringoccs.tools package 43

1.1.6.1 Submodules 43

2 Indices and tables **51**

Bibliography **53**

Python Module Index **55**

Index **57**

CHAPTER 1

rss_ringoccs package

This program is part of the rss_ringoccs repository hosted at https://github.com/NASA-Planetary-Science/rss_ringoccs and developed with the financial support of NASA's Cassini Mission to Saturn.

Purpose: Provide tools for analysis of ring occultation experiments, particularly those pertaining to the Cassini Radio Science Experiment, based on methods from [MTR1986] and [CRSUG2018]. This software package contains methods for reading and extracting RSR data, computing occultation geometry, calibrating RSR data, and performing diffraction reconstruction for calibrated data at different resolutions. Also included are tools for writing and reading the data products output by the software.

Dependencies:

1. numpy
2. spiceypy
3. scipy
4. time
5. sys
6. subprocess
7. pdb
8. pandas

9. matplotlib
10. os
11. platform

References:

1.1 Subpackages

1.1.1 rss_ringoccs.calibration package

Purpose: Provide tools for calibrating raw radio science data by phase-correcting the measured complex signal and fitting the free-space power, as discussed in [MTR1986] and [CRSUG2018]. Final calibrated signal can be used to compute the diffraction-limited profile of the occultation.

Dependencies:

1. numpy
2. scipy
3. sys
4. pdb
5. matplotlib

1.1.1.1 Submodules

rss_ringoccs.calibration.calc_f_sky_recon module

Purpose Calculate sky frequency from the reconstructed event kernels. copied from Nicole Rappaport's `predicts` program in Fortran.

References This is a pythonized version of Nicole Rappaport's PREDICTS program, which predicts sky frequencies by computing the Doppler shift due to motion of the spacecraft relative to the observer (i.e., the receiving station).

Dependencies

1. numpy
2. spiceypy

`rss_ringoccs.calibration.calc_f_sky_recon.`**`calc_f_sky_recon`**(*f_spm*, *rsr_inst*, *sc_name*, *f_uso*, *kernels*)

Calculates sky frequency at given times.

Arguments

f_spm (***np.ndarray***) SPM values to evaluate sky frequency at

rsr_inst (***object***) Instance of RSRReader class

sc_name (***str***) Name of spacecraft to get sky frequency for. In our case, this should always be 'Cassini'

f_uso (***float***) USO sky frequency for the event and the right band

kernels (*list*) String list of full path name to set of kernels

Returns

RF (*np.ndarray*) Reconstructed sky frequency computed from spacecraft telemetry and oscillator frequency

`rss_ringoccs.calibration.calc_f_sky_recon.`**`derlt`**(*sc_code*, *etsc*, *rs_code*, *et*)

Arguments

sc_code (*int*) Spacecraft NAIF ID

etsc (*float*) Epoch (in ephemeris seconds past J2000 TDB) at which the signal arrives at the receiver station

rs_code (*int*) Receiving station NAIF ID

et (*float*) Ephemeris time

Returns

DLTDT2 (*float*)

`rss_ringoccs.calibration.calc_f_sky_recon.`**`derpt`**(*et*, *code*)

Arguments

et (*float*) Ephemeris time

code (*int*) NAIF ID

Returns

B (*float*)

rss_ringoccs.calibration.calc_freq_offset module

Purpose Class for computing the frequency corresponding to the maximum power in the FFT power spectrum

Dependencies

1. numpy

`class` `rss_ringoccs.calibration.calc_freq_offset.`**`calc_freq_offset`**(*rsr_inst*, *spm_min*, *spm_max*, *dt_freq=2.0*)

Bases: `object`

Purpose Calls functions to sample raw signal at regular intervals using a window of width `dt_freq`

Arguments

rsr_inst (*object*) Object instance of the RSRReader class

spm_min (*float*) Minimum observed event time for sampling

spm_max (*float) Maximum observed event time for sampling

Keyword Arguments

dt_freq (*float*) half the width of the FFT window

Attributes

spm_vals (*np.ndarray*) Observed event time at full sampling

IQ_m (*np.ndarray*) Uncorrected real and imaginary components of signal

dt (*float*) Raw time sampling from spm_vals

dt_freq (*float*) half the width of the FFT window

spm_min (*float*) Minimum time for sampling

spm_max (*float*) Maximum time for sampling

f_spm (*np.ndarray*) Observed event time for frequency offset

f_offset (*np.ndarray*) Frequency offset, or frequency at max power

rss_ringoccs.calibration.calc_tau_thresh module

Purpose Compute τ_{thresh} for use as a proxy for maximum reliable value of optical depth within the diffraction-limited or diffraction-reconstructed profile. This follows [MTR1986] Equations 24 and 26, which define, respectively, the power of the thermal noise

$$\hat{P}_N = \frac{\dot{\rho}_0}{\mathrm{SNR}_0 \Delta\rho_0}$$

and the threshold optical depth

$$\tau_{thresh} = -\sin(B)\ln\left(\frac{1}{2}C_\alpha \hat{P}_N\right)$$

Dependencies

1. numpy
2. matplotlib
3. scipy

class rss_ringoccs.calibration.calc_tau_thresh.**calc_tau_thresh**(*rsr_inst*, *geo_inst*, *cal_inst*, *res_km=1.0*, *Calpha=2.41*)

Bases: `object`

Purpose Compute threshold optical depth following

Arguments

rsr_inst (*object*) object instance of the RSRReader class

geo_inst (*object*) object instance of the Geometry class

cal_inst (*object*) object instance of the Calibration class

Keyword Arguments

res_km (*float*) Reconstruction resolution in km

Calpha (*float*) Constant for scaling Bandwidth/SNR ratio. Default is 2.41 for 70% confidence (see [MTR1986])

Attributes

snr (*np.ndarray*) Signal-to-noise ratio SNR0 over the entire occultation. This changes over the occultation because the signal power fluctuates.

tau_thresh (*np.ndarray*) threshold optical depth computed using [MTR1986]

spm_vals (*np.ndarray*) Observed event time array from cal_inst

rho_vals (*np.ndarray*) Ring intercept point array interpolated to spm_vals

find_noise(*spm*, *IQ*, *df*)

Locate the additive receiver noise within the data set. This is done by computing a spectrogram of the raw complex signal, filtering out the spacecraft signal, and averaging over the frequency and time domains.

Arguments

spm (*np.ndarray*) raw SPM in seconds

IQ (*np.ndarray*) measured complex signal

df (*float*) sampling frequency in Hz of the IQ

Returns

p_noise (*np.ndarray*) noise power

rss_ringoccs.calibration.calibration_class module

Purpose Class framework for performing the necessary calibration steps for the RSR data. This includes phase correction based on frequency offset of the spacecraft and normalization of received power with respect to the intrinsic spacecraft power.

Notes Can be computationally cumbersome, especially for chord occultations. May require up to 30 mins for 16 kHz RSR data files.

Dependencies:

1. numpy
2. scipy
3. sys

class rss_ringoccs.calibration.calibration_class.**Calibration**(*rsr_inst*, *geo_inst*, *pnf_order=3*, *dt_cal=1.0*, *verbose=False*, *write_file=True*, *interact=False*)

Bases: `object`

Purpose Define a class which, when instantiated, calls the submodules for performing each step of the calibration process by instantiating the classes `FreqOffsetFit` in the `freq_offset_fit.py` script and `Normalization` in the `power_normalization.py` script.

Arguments

rsr_inst (*object*) Instance of the RSRReader class

geo_inst (*object*) Instance of the Geometry class

Keyword Arguments

pnf_order (*float*) whole number specifying the polynomial order to use when fitting the freespace power. Default is 3.

dt_cal (*float*) Desired final spacing in SPM between data points. Default is 1 sec.

verbose (*bool*) If True, print intermediate steps and results. Default is False.

write_file (*bool*) If True, write output CAL .TAB and CAL .LBL files. Default is True.

interact (*bool*) If True, enables the interactive mode in the terminal for fitting the freespace power. Default is False.

Attributes

rev_info (*dict*) *dict* of information identifying the specific occultation: rsrfile, year, day of year, direction and type of occultation, spacecraft revolution number, and observation band

t_oet_spm_vals (*np.ndarray*) SPM values for observed event time t

f_sky_hz_vals (*np.ndarray*) sum of the reconstructed sky frequency values and the fit to frequency offset $\hat{f}(t)_{sky} = f_{dr}(t) + \hat{f}(t)_{offset}$ following Equation 19 in [CRSUG2018].

f_offset_fit_vals (*np.ndarray*) fit to frequency offset $\hat{f}(t)_{offset}$

p_free_vals (*np.ndarray*) fit to freespace power $\hat{P}_0(t)$

IQ_c (*np.ndarray*) phase-corrected spacecraft signal $I_c + iQ_c$

history (*dict*) information about the parameters, results, and computation of the calibration procedures

FORFIT_chi_squared (*float*) sum of the squared residual frequency offset fit such that $\chi^2 = \frac{1}{N-m}\sum((\hat{f}(t)_{offset} - f(t)_{offset})/\hat{f}(t)_{offset})^2$

FSPFIT_chi_squared (*float*) $\chi^2 = \frac{1}{N-m}\sum((\hat{P}_0(t) - P_0(t))/\hat{P}_0(t))^2$

`correct_IQ`(*spm_vals*, *IQ_m*, *f_spm*, *f_offset_fit*)

Purpose:

Apply frequency offset fit to raw measured signal using the signal frequencies calculated by `FreqOffsetFit`. First resamples the frequency offset fit to a 0.1 sec separation. Then, computes detrending function by integrating frequency offset fit to get phase detrending function ψ using Equation 18 from [CRSUG2018] where

$$\psi = \int^{t} \hat{f}(\tau)_{offset} \mathrm{d}\tau + \psi(t_0)$$

Finally, applies phase detrending correction to signal to raw signal such that

$$I_c + iQ_c = [I_m + iQ_m] \exp(-i\psi)$$

as discussed in [CRSUG2018] (see their Equation 17).

Arguments:

spm_vals (*np.ndarray*) raw SPM values

IQ_m (*np.ndarray*) raw complex signal measured by DSN

f_spm (*np.ndarray*) SPM sampled for frequency offset calculation in the `calc_freq_offset` class in the `calc_freq_offset.py` script.

f_offset_fit (*np.ndarray*) frequency of the spacecraft signal corresponding to `f_spm`

Returns:

IQ_c (*np.ndarray*) Frequency-corrected complex signal $I_c + iQ_c$ corresponding to `spm_vals`

rss_ringoccs.calibration.dlp_class module

Purpose Create a class whose attributes have all the necessary inputs for performing a Fresnel inversion usng DiffractionCorrection, given instances of the classes RSRReader, Geometry, and Calibration.

Dependencies

1. numpy
2. copy
3. scipy

class `rss_ringoccs.calibration.dlp_class.`**`DiffractionLimitedProfile`**(*rsr_inst, geo_inst, cal_inst, dr_km, verbose=False, write_file=True, profile_range=[65000.0, 150000.0]*)

Bases: `object`

Purpose Framework for an object class containing the diffraction-limited optical depth profile (DLP) and related attributes.

Arguments

rsr_inst (*object*) Instance of RSRReader class

geo_inst (*object*) Instance of Geometry class

cal_inst (*object*) Instance of Calibration class

dr_km (*float*) radial sampling rate $\Delta\rho$ for the DLP in kilometers. DLP radial *resolution* is the Nyquist radial sampling, i.e., twice the input value of *dr_km*, meaning that this will affect the minimum resolution of the diffraction-reconstructed profile. Value for *dr_km* can range from 0.05 km to 0.75 km for the reconstruction resolutions supported by *rss_ringoccs*. PDS sampling rate is 0.25 km, which gives a DLP resolution of 0.5 km.

Keyword Arguments

verbose (*bool*) When True, turns on verbose output. Default is False.

write_file (*bool*) When True, writes processing results to file. Default is True.

profile_range (*list*) 1x2 list specifying the radial limits in km of on the occultation. Default is [65000,150000].

Attributes

dr_km (*float*) raw DLP sampling rate

raw_tau_threshold_vals (*np.ndarray*) threshold optical depth

rho_km_vals (*np.ndarray*) Ring-intercept points in km

t_oet_spm_vals (*np.ndarray*) Observed event times in seconds past midnight

p_norm_vals (*np.ndarray*) Normalized diffraction-limited power

phase_rad_vals (*np.ndarray*) Phase of diffraction-limited signal, in radians

B_rad_vals (*np.ndarray*) Ring opening angle in radians

D_km_vals (*np.ndarray*) Ring intercept point to spacecraft distance in km

F_km_vals (*np.ndarray*) Fresnel scale in km

f_sky_hz_vals (*np.ndarray*) Sky frequency in Hz

phi_rad_vals (*np.ndarray*) Observed ring azimuth

t_ret_spm_vals (*np.ndarray*) Ring event time in seconds past midnight

t_set_spm_vals (*np.ndarray*) Spacecraft event time in seconds past midnight

phi_rl_rad_vals (*np.ndarray*) Ring longitude in radians

rho_dot_kms_vals (*np.ndarray*) Ring intercept radial velocity in km/s

rho_corr_pole_km_vals (*np.ndarray*) Radius correction due to improved pole. This is populated with a placeholder of zeros

rho_corr_timing_km_vals (*np.ndarray*) Radius correction due to timing offset. This is populated with a placeholder of zeros

tau_vals (*np.ndarray*) Diffraction-limited optical depth

history (*dict*) Processing history with all inputs necessary to rerun pipeline to obtain identical output

rev_info (*dict*) *dict* containing rev- and rsr-specific info

Note:

1. All *np.ndarray* attributes are sampled at dr_km radial spacing.

classmethod create_dlps(*rsr_inst, geo_inst, cal_inst, dr_km, verbose=False, write_file=False, profile_range=[65000.0, 150000.0]*)

Create ingress and egress instances of DiffractionLimitedProfile.

Arguments

rsr_inst (*object*) Instance of RSRReader class

geo_inst (*object*) Instance of Geometry class

cal_inst (*object*) Instance of Calibration class

dr_km (*float*) radial sampling rate $\Delta\rho$ for the DLP in kilometers. DLP radial *resolution* is the Nyquist radial sampling, i.e., twice the input value of *dr_km*, meaning that this will affect the minimum resolution of the diffraction-reconstructed profile. Value for *dr_km* can range from 0.05 km to 0.75 km for the reconstruction resolutions supported by *rss_ringoccs*. PDS sampling rate is 0.25 km, which gives a DLP resolution of 0.5 km.

Keyword Arguments

verbose (*bool*) When True, turns on verbose output. Default is False.

write_file (*bool*) When True, writes processing results to file. Default is True.

profile_range (*list*) 1x2 list specifying the radial limits in km of on the occultation. Default is [65000,150000].

rss_ringoccs.calibration.freq_offset_fit module

Purpose: Compute a fit to the frequency offset using offset frequencies calculated from raw data, sigma-clipping frequencies contaminated by rings, and fitting with a polynomial of order determined by an iterative F-test.

class `rss_ringoccs.calibration.freq_offset_fit.`**`FreqOffsetFit`**(*rsr_inst*, *geo_inst*, *f_uso_x=8427222034.3405*, *verbose=False*, *write_file=False*)

Bases: `object`

Purpose Obtains $f(t)_{offset}$ from `calc_freq_offset`, $f(t)_{dr}$ from `calc_f_sky_recon`. Computes a polynomial fit $\hat{f}(t)_{offset}$ of F-test specified order to sigma-clipped frequency offset. Final sky frequency $\hat{f}(t)_{sky}$ is calculated by summing the polynomial fit $\hat{f}(t)_{offset}$ with the reconstructed sky frequency $f(t)_{dr}$.

Arguments

rsr_inst (*object*) object instance of the RSRReader class

geo_inst (*object*) object instance of the Geometry class

Keyword Arguments

f_uso_x (*float*) frequency in Hz of the X-band ultra-stable oscilator onboard the Cassini spacecraft. Default is 8427222034.3405 Hz.

verbose (*bool*) when True, enables verbose output mode

Attributes:

f_offset_fit (*np.ndarray*) fit to frequency offset :math:‘hat{f}(t)_{offset}

f_spm (*np.ndarray*) SPM at which the offset frequency was sampled

f_sky_recon (*np.ndarray*) reconstructed sky frequency $f(t)_{dr}$

f_offset_fit (*np.ndarray*) fit to the frequency offset math:*hat{f}(t)_{offset}* evaluated at `f_spm`

chi_squared (*float*) sum of the squared residual difference between the frequency offset and the frequency offset fit normalized by the fit value (Pearson's χ^2) such that $\chi^2 = \frac{1}{N-m}\sum((\hat{f}(t)_{offset} - f(t)_{offset})/\hat{f}(t)_{offset})^2$ for N data and m free parameters (i.e., the polynomial order plus one).

`calc_poly_order`(*f_spm_cl*, *f_offset_cl*, *verbose=False*)

Use a variant of the F-test to determine the best order polynomial to use to fit the frequency offset.

Arguments

f_spm_cl (*np.ndarray*) SPM sampled by `calc_freq_offset` and clipped by the initial boolean mask.

f_offset_cl (*np.ndarray*) carrier frequency offset from center of band

`create_mask`(*f_spm*, *f_rho*, *f_offset*)

Creates a Boolean mask array which excludes data based on the following critera:

1. ring or planetary occultation in region prevents accurate estimation of the offset frequency
2. offset frequencies fall more than 5-sigma beyond the median offset frequency
3. offset frequencies vary by more than 0.25 Hz relative to neighboring offset frequencies

4. adjacent data all excluded by previous requirements (excludes noise which by happenstance satisfies the above criteria)

Arguments

f_spm (*np.ndarray*) SPM sampled by `calc_freq_offset` when calculating the offset frequencies for the occultation

f_rho (*np.ndarray*) ring intercept radius of the spacecraft signal resampled to match f_spm

f_offset (*np.ndarray*) frequency offset

Returns

fsr_mask (*np.ndarray*) Array of booleons, with True for reliable frequency offset.

`fit_freq_offset` (*f_spm*, *f_rho*, *f_offset*, *verbose=False*)
Fit a polynomial to frequency offset.

Arguments

f_spm (*np.ndarray*) SPM sampled by `calc_freq_offset` when calculating the offset frequencies for the occultation

f_rho (*np.ndarray*) ring intercept radius of the spacecraft signal resampled to match f_spm

f_offset (*np.ndarray*) carrier frequency offset from center of band

Keyword Arguments

verbose (*bool*) If True, print processing steps

Returns

f_offset_fit (*np.ndarray*) fit to the frequency offset math:*hat{f}(t)_{offset}* evaluated at `f_spm`

chi2 (*float*) sum of the squared residual difference between frequency offset and frequency offset fit normalized by the fit value (Pearson's χ^2) such that $\chi^2 = \frac{1}{N-m}\sum((\hat{f}(t)_{offset} - f(t)_{offset})/\hat{f}(t)_{offset})^2$ for N data and m free parameters (i.e., the polynomial order plus one).

`plotFORFit` (*spm*, *f_offset*, *fit*, *mask*, *spm_min*, *spm_max*, *occ_min*, *occ_max*)
Plot results of the automated frequency offset fit and save plot to a file. File name will match the .LBL and .TAB nomenclature.

Arguments

spm (*np.ndarray*) SPM sampled by `calc_freq_offset` when calculating the offset frequencies for the occultation

f_offset (*np.ndarray*) frequency offset

fit (*np.ndarray*) polynomial fit to the frequency offset

mask (*np.ndarray*) boolean array used to mask frequency offset for the polynomial fitting

spm_min (*float*) start of occultation in SPM

spm_max (*float*) end of occultation in SPM

rss_ringoccs.calibration.power_normalization module

Purpose Normalize frequency-corrected power using a polynomial fit of specified order.

Dependencies

1. numpy
2. scipy
3. matplotlib

`class rss_ringoccs.calibration.power_normalization.`**`Normalization`**(*spm_raw*, *IQ_c*, *geo_inst*, *order=3*, *fittype='poly'*, *interact=False*, *verbose=False*, *write_file=False*)

Bases: `object`

Purpose Finds freespace power based on ring geometry, locations of gaps computed by the Geometry class, and signal power relative to median power within the gaps. Fits the freespace power with a polynomial. If desired, this fitting process can be interactive. A plot will be saved following rss_ringoccs nomenclature to visualize the fit results.

Arguments

spm_raw (*np.ndarray*) SPM as sampled in the raw data

IQ_c (*np.ndarray*) frequency-corrected signal corresponding to spm_raw

geo_inst (*np.ndarray*) instance of the Geometry class, used to estimate the freespace regions within and surrounding the ring system, accounting for Saturn occultation and the Keplerian geometry of the rings

rho_km_vals (*np.ndarray*) radial intercept poin of occultation in km

Keyword Arguments

verbose (*bool*) when True, outputs information to command line about the freespace power fitting progress and results. All results will be output to the *CAL*.LBL file regardless of this keyword. Default is False.

order (*float*) a whole number specifying the order of the polynomial fit to the freespace power. Default is 3.

interact (*bool*) If True, allows user to interactively adjust fit to the freespace power. Default is False.

fittype (*str*) Type of fit to freespace regions. Default is a polynomial fit.

`create_mask`(*spm*, *gaps_spm*, *pc*)

Purpose: Set mask and gaps attribute.

Arguments:

spm (*np.ndarray*) SPM in seconds of the downsampled signal

rho (*np.ndarray*) occultation intercept radius of the downsampled signal

gaps_spm (*list*) location of freespace regions as predicted by the gap-finding routines `get_freespace.py` using tabulated orbital parameters.

pc (*np.ndarray*) phase-corrected downsampled power where $P_c = |I_c^2 + iQ_c^2|$. Sets attributes

mask (*np.ndarray*) array of booleans wherein True designates data corresponding to freespace power and False data corresponding to occultation events

gaps (*list*) an Nx2 list of lower and upper radial limits in km for each of N gaps designated

`downsample_IQ`(*spm_raw*, *IQ_c_raw*, *dt_down=0.5*, *verbose=False*)

Purpose: Downsample complex signal to specified time spacing to avoid diffraction pattern affecting freespace power fit

Arguments:

spm_raw (*np.ndarray*) raw SPM values

IQ_c_raw (*np.nparray*) $I_c + iQ_c$ sampled at the raw SPM rate

Keyword Arguments:

dt_down (*float*) Time spacing to downsample to

verbose (*bool*) If True, prints downsampled results

Returns:

spm_vals_down (*np.ndarray*) SPM values after downsampling

rho_km_vals_down (*np.ndarray*) Rho values after downsampling

p_obs_down (*np.ndarray*) Observed power after downsampling

`extract_list_from_str`(*gaps_str*)

Purpose: Extract an Nx2 list from the string of user input freespace regions.

Arguments:

gaps_str (*str*) string containing the user input freespace regions.

Returns:

gaps (*list*) an Nx2 list of floats indicating the lower and upper limits to the user-specific freespace regions.

`fit_check`(*spm_down*, *p_obs_down*, *freespace_spm*, *order*)

Purpose: Allows user to update the freespace regions and fit order during the freespace power fitting step. This is done by prompting the user for input in the command line and displaying the results of their input for the polynomial fit to the freespace power. Only called if the Normalization keyword `interact` is set to True.

Arguments:

gaps_str (*str*) string containing the user input freespace regions.

Returns:

gaps (*list*) an Nx2 list of floats indicating the lower and upper limits to the user-specific freespace regions.

`fit_freespace_power`(*spm*, *power*, *order=3*, *fittype='poly'*)

Arguments:

spm (***np.ndarray***) downsampled SPM

power (***np.ndarray***) absolute square of downsampled phase-corrected signal

Keyword Arguments:

order (***float***) order of the fit, whole number between 1 and 5. Default order is 3.

type (***str***) type of fit to use, default is 'poly'. Options are

- 'poly' a single polynomial
- 'spline' an unsmoothed spline fit

Returns

fit (***np.ndarray***) best fit to freespace power

`hfit_med`(*p_obs_down*)

`plot_power_profile`(*spm*, *pow*, *gaps*, *order*, *save=False*)

Purpose: Plot results of the freespace power for total profile and each individual freespace region and either show plot in a GUI or save it to a file. File name will match the *.LBL and *.TAB nomenclature.

Arguments:

spm (***np.ndarray***) SPM sampled by `calc_freq_offset` when calculating the offset frequencies for the occultation

pow (***np.ndarray***) residual sky frequency

gaps (***np.ndarray***) gap edges used to select freespace power

order (***float***) order of polynomial fit to the residual sky frequency

Keyword Arguments:

save (***bool***) If True, saves plots to file. Otherwise, plot is shown in GUI. Default is False.

rss_ringoccs.calibration.resample_IQ module

Purpose Resample I and Q from uniformly spaced time to uniformly spaced radius. This is set up to downsample from the raw resolution data.

`rss_ringoccs.calibration.resample_IQ.pre_resample`(*rho_km*, *vec*, *freq*)

Set vector sampling to be uniform with respect to radius at a spacing comparable to that of raw resolution. For ingress occultations, this step implicitly reverses the radius scale when interpolating.

Arguments

rho_km (***np.ndarray***) radius in kilometers

vec (***np.ndarray***) a single vector component I or Q of the complex signal

freq (***float***) radial sampling frequency

Returns

rho_grid (***np.ndarray***) Radii at uniform spacing at which the signal component is resampled

vec_grid (***np.ndarray***) Signal resampled with respect to radius with a uniform spacing

p (***float***) upsampling rate to be used by scipy.signal.resample_poly. This will always be unity because no upsampling is done.

q (***float***) downsampling rate to be used by scipy.signal.resample_poly. This depends on the uniform radial sampling rate at which *rho_grid* and *vec_grid* are sampled.

`rss_ringoccs.calibration.resample_IQ.`**`resample_IQ`**(*rho_km*, *IQ_c*, *dr_desired*, *verbose=False*)

Resample I and Q to uniformly spaced radius. Based off of Matlab's `resample` function

Arguments

rho_km (***np.ndarray***) Set of ring intercept point values at initial resolution before resampling

IQ_c (***np.ndarray***) Frequency-corrected complex signal at initial resolution before resampling

dr_desired (***float***) Desired final radial sample spacing

verbose (***bool***) Testing variable to print out the first few resampled results

Returns

rho_km_desired (***np.ndarray***) array of ring radius at final desired spacing

IQ_c_desired (***np.ndarray***) Frequency-corrected complex signal at final desired spacing

1.1.2 rss_ringoccs.diffrec package

Subpackage Name: diffrec

Purpose: Provide functions and classes that aid in the process of Diffraction Correction / Fresnel Inversion. Additional functions for the purpose of forward modelling of reconstructed data and diffraction modelling are included. Special mathematical functions and solutions to common problems in diffraction theory are also included.

Sub-Modules:

advanced_tools: This submodule is good for modeling the geometry of a given occultation, and for comparing your results to the results obtained by others (Ex. Results on the PDS). This submodule contains the following:

compare_tau: Class Used for running diffraction correction on a given set of diffracted data and then comparing the outcome to a given set of reconstructed data.

find_optimal_resolution: Class Given a set of data and a reconstruction, this class will run diffraction correction over a set of resolutions. The output contains the L_2 and L_infinity difference of the two reconstructions as a function of resolution.

delta_impulse_diffraction: Class Given a set of geometry data, this class create modeled data of the solution of diffraction through a Dirac-Delta impulse function. Reconstruction is then perform on the mock-data. This tool is good for modeling the problem and determining resolution constraints based on the geometry available.

diffraction_correction: This is the main sub-module in the entire subpackage. Given a set of diffracted data and a requested resolution (in kilometers), diffraction corrections will be performed to produce a diffraction corrected profile. This submodule comtains the following:

DiffractionCorrection: Class Given a requested resolution and an instance of the NormDiff class (See Calibration subpackage), this produces a diffraction corrected profile.

Special Functions: fresnel_sin. . . . . The Fresnel sine integral. fresnel_cos. . . . . The Fresnel cosine integral. sq_well_solve. . . . . . .Diffraction pattern through square well. compute_norm_eq. . . . .Computes the normalized equivalent width. resolution_inverse..Computes the inverse of the function

y = x/(exp(-x)+x-1)

fresnel_scale.......Compute the Fresnel scale.

window_functions: rect................Rectangular window. coss................Squared cossine window. kb20................Kaiser-Bessel 2.0 window. kb25................Kaiser-Bessel 2.5 window. kb35................Kaiser-Bessel 3.5 window. kbmd20.............Modified Kaiser-Bessel 2.0 window. kbmd25.............Modified Kaiser-Bessel 2.5 window. get_range_actual....Given an array of numbers (usually the

radial range of the data), a range request, and a window width, compute the allowed range of processing.

1.1.2.1 Submodules

rss_ringoccs.diffrec.advanced_tools module

class `rss_ringoccs.diffrec.advanced_tools.`**`CompareTau`**(*geo*, *cal*, *dlp*, *tau*, *res*, *rng='all'*, *wtype='kbmd20'*, *fwd=False*, *bfac=True*, *sigma=2e-13*, *verbose=False*, *norm=True*, *psitype='fresnel4'*, *res_factor=0.75*)

Bases: `object`

class `rss_ringoccs.diffrec.advanced_tools.`**`FindOptimalResolution`**(*geo, cal, dlp, tau, sres, dres, nres, norm=True, bfac=True, sigma=2e-13, psitype='fresnel4', rng='all', wlst=['kbmd20'], res_factor=0.75, verbose=True*)

Bases: `object`

class `rss_ringoccs.diffrec.advanced_tools.`**`ModelFromGEO`**(*geo*, *lambda_km*, *res*, *rho*, *width=100*, *dx_km_desired=0.25*, *occ='other'*, *wtype='kb25'*, *norm=True*, *bfac=True*, *verbose=True*, *psitype='fresnel'*, *use_fresnel=False*, *eccentricity=0.0*, *periapse=0.0*, *use_deprecate=False*, *res_factor=0.75*, *rng='all'*, *model='squarewell'*, *echo=False*, *rho_shift=0.0*)

Bases: `object`

rss_ringoccs.diffrec.diffraction_correction module

Purpose: Provide the DiffractionCorrection class for performing the necessary mathematics to correct for the diffraction effects that are obtained during occultation observations of planetary rings using radio waves.

Dependencies:

1. numpy
2. scipy
3. rss_ringoccs

class `rss_ringoccs.diffrec.diffraction_correction.`**`DiffractionCorrection`**(*DLP*, *res*, *rng='all'*, *wtype='kbmd20'*, *fwd=False*, *norm=True*, *verbose=False*, *bfac=True*, *sigma=2e-13*, *psitype='fresnel4'*, *write_file=False*, *res_factor=0.75*, *eccentricity=0.0*, *periapse=0.0*)

Bases: `object`

Purpose: Perform diffraction correction for a ring occultation on a data set that is a near radially symmetric function of the ring radius, or ring intercept point (RIP).

Arguments:

DLP (***object***) The data set, usually an instance of the DiffractionLimitedProfile class from the rss_ringoccs Calibration subpackage. This instance MUST contain the following attributes and have the same names.

rho_km_vals: Ring Radius (km)
phi_rad_vals: Ring Azimuth Angle (Radians)
p_norm_vals: Normalized Power
phase_rad_vals: Phase (Radians)
B_rad_vals: Elevation Angle (Radians)
D_km_vals: RIP-Distance (km)
f_sky_hz_vals: Sky Frequency (Hertz)
rho_dot_kms_vals: RIP-velocity (km/s)
history: History dictionary

res (*float* or *int*) The requested resolution for processing (km). This must be a positive real number.

Keywords:

rng (*list* or *str*) The requested range for diffraction correction. Preferred input is rng = [a,b]. Arrays are allowed and the range will be set as:

rng = [MIN(array), MAX(array)]

Finally, certain strings containing a few of the regions of interests within the rings of Saturn are allowed. Permissable strings are:

'all'	[1.0, 400000.0]
'cringripples'	[77690.0, 77760.0]
'encke'	[132900.0, 134200.0]
'enckegap'	[132900.0, 134200.0]
'janusepimetheus'	[96200.0, 96800.0]
'maxwell'	[87410.0, 87610.0]
'maxwellringlet'	[87410.0, 87610.0]
'titan'	[77870.0, 77930.0]
'titanringlet'	[77870.0, 77930.0]
'huygens'	[117650.0, 117950.0]
'huygensringlet'	[117650.0, 117950.0]

Strings are neither case nor space sensitive. For other planets use rng = [a,b]. Default value is set to 'all' which processes [1, 400000] Values MUST be set in kilometers.

wtype (*str) The requested tapering function for diffraction correction. A string with several allowed inputs:

'rect'	Rectangular Window.
'coss'	Squared Cosine Window.
'kb20'	Kaiser-Bessel 2.0 Window.
'kb25'	Kaiser-Bessel 2.5 Window.
'kb35'	Kaiser-Bessel 3.5 Window.
'kbmd20'	Modified kb20 Window.
'kbmd25'	Modified kb25 Window.

The variable is neither case nor space sensitive. Default window is set to 'kb25'. See window_functions submodule for further documentation.

fwd (*bool*) A Boolean for determining whether or not forward modelling will be computed. This is good starting point for deciding if the diffraction correction is physically significant or valid. If the reconstruction is good, the forward model should reproduce the p_norm_vals attribute from the input DLP instance. Default is set to False.

norm (*bool*) A Boolean for determining whether or not the reconstructed complex transmittance is normalize by the window width. This normalization is the complex transmittance that is computed by using free space divided by the complex transmittance that is computed using free space weighted by the selected tapering function. Default is True.

bfac (*bool*) A Boolean for determining whether or not the 'b' factor in the window width computation is used. This is equivalent to setting the Allen Deviation for the spacecraft to a positive value or to zero. If set to False, the Allen Deviation is assumed to be zero. If set to True the Allen Deviation is set to 2e-13, or whichever number you wish to specify in the sigma keyword (See below). Default is True.

sigma (*float*) The Allen deviation for the spacecraft. If the bfac keyword (See above) is set to False, this is ignored. If bfac is set to True, and sigma is NOT specified, then sigma=2e-13 will be used, which is the Allen deviation for Cassini with 1 second integration time. For spacecraft other than Cassini, you should provide the Allen deviation yourself. Default is sigma=2e-13

psitype (*str*) A string for determining what approximation to the geometrical 'psi' function is used. Several strings are allowed:

'full' No Approximation is applied.

'MTR2' Second Order Series from MTR86.

'MTR3' Third Order Series from MTR86.

'MTR4' Fourth Order Series from MTR86.

'Fresnel' Standard Fresnel approximation.

The variable is neither case nor space sensitive. Default is set to 'full'.

verbose (*bool*) A Boolean for determining if various pieces of information are printed to the screen or not. Default is False.

Attributes:

bfac (*bool*) Boolean for bfac (See keywords).

dathist (*dict*) History from DLP instance.

dx_km (*float*) Radial spacing for the data points (km).

f_sky_hz_vals (*np.ndarray*) Recieved frequency from the spacecraft (Hz).

finish (*int*) Final point that was reconstructed.

fwd (*bool*) Boolean for fwd (See keywords).

history (*dict*) History for the DiffractionCorrection class. This contains system info and user info, including what operating system was used, username, hostname, computer name, and the inputs provided.

lambda_sky_km_vals (*np.ndarray*) Wavelength of recieved signal from spacecraft (km).

mu_vals (*np.ndarray*) The sine of the ring opening angle (Unitless).

n_used (*int*) Number of points that were reconstructed.

norm (*bool*) Boolean for norm (See keywords).

norm_eq (*float*) Normalized equivalent width computed from window that was used during reconstruction. See the window_functions submodule for more information.

p_norm_fwd_vals (*np.ndarray*) Normalized power computer from the forward modelling of the reconstructed data. This will be a None type variable unless fwd=True is set. If the reconstruction went well, this should mimic the raw data, p_norm_vals.

p_norm_vals (*np.ndarray*) Normalized power from the diffracted signal. This is the square of the absolute value of the recieved complex transmittance.

phase_fwd_vals (*np.ndarray*) Phase computed from the forward model of the reconstructed data. This will be a None type variable unless fwd=True is set. If the reconstruction went well, this should mimic phase_rad_vals. This variable is in radians.

phase_rad_vals (*np.ndarray*) Phase from the diffracted signal (Radians).

phase_vals (*np.ndarray*) Reconstructed phase (Radians).

phi_rad_vals (*np.ndarray*) Ring azimuth angle of the ring intercept (Radians).

phi_rl_rad_vals (*np.ndarray*) Ring longitude angle. This will be a None type unless it was provided in the DLP class. Otherwise, this variable is in radians.

power_vals (*np.ndarray*) Normalized reconstructed power.

psitype (*str*) String for psitype (See keywords).

raw_tau_threshold_vals (*np.ndarray*) Threshold optical depth for the diffracted data. This will be a None type unless provided for in the DLP class.

res (*float*) Requested resolution (See arguments). In kilometers.

rho_corr_pole_km_vals (*np.ndarray*) Radial corrections from the Planet's pole. This will be a None type variable unless provided in the DLP class. Otherwise, this is in kilometers.

rho_corr_timing_km_vals (*np.ndarray*) Radial corrections from timing offsets. This will be a None type variable unless provided in the DLP class. Otherwise, this is in kilometers.

rho_dot_kms_vals (*np.ndarray*) Time derivative of the ring intercept point (km/s).

rho_km_vals (*np.ndarray*) Ring-intercept-point (RIP) in kilometers.

rng (*list*) Range that was used for reconstruction, taking into the range that was requested by the user. The actual range takes into account limits in the available data and limits in the required window sizes.

rngreq (*str* or *list*) Requested range (See keywords).

sigma (*float*) Requested Allen deviation (See keywords).

start (*int*) First point that was reconstructed.

t_oet_spm_vals (*np.ndarray*) Time the signal is measured on Earth. This is a None type unless provided for in the DLP class.

t_ret_spm_vals (*np.ndarray*) Time the signal passes through the diffracting medium. This is a None type unless provided for in the DLP class.

t_set_spm_vals (*np.ndarray*) Time the signal is emitted from the spacecraft. This is a None type unless provided in the DLP class.

tau_threshold_vals (*np.ndarray*) Threshold optical depth of the reconstructed data.

tau_vals (*np.ndarray*) Optical depth of the reconstructed data.

verbose (*bool*) Boolean for Verbose (See keywords).

w_km_vals (*np.ndarray*) Window width as a function of radius (km).

wtype (*str*) String for wtype (See keywords).

rss_ringoccs.diffrec.special_functions module

rss_ringoccs.diffrec.special_functions.**compute_norm_eq**(*w_func*, *error_check=True*)

Purpose: Compute normalized equivalenth width of a given function.

Arguments:

w_func (*np.ndarray*) Function with which to compute the normalized equivalent width of.

Outputs:

normeq (*float*) The normalized equivalent width of w_func.

Notes: The normalized equivalent width is effectively computed using Riemann sums to approximate integrals. Therefore large dx values (Spacing between points in w_func) will result in an inaccurate normeq. One should keep this in mind during calculations.

Examples: Compute the Kaiser-Bessel 2.5 window of width 30 and spacing 0.1, and then use compute_norm_eq to compute the normalized equivalent width:

```
>>> from rss_ringoccs import diffrec as dc
>>> w = dc.window_functions.kb25(30, 0.1)
>>> normeq = dc.special_functions.compute_norm_eq(w)
>>> print(normeq)
1.6573619266424229
```

In contrast, the actual value is 1.6519208. Compute the normalized equivalent width for the squared cosine window of width 10 and spacing 0.25.

```
>>> from rss_ringoccs import diffrec as dc
>>> w = dc.window_functions.coss(10, 0.25)
>>> normeq = dc.special_functions.compute_norm_eq(w)
>>> print(normeq)
1.5375000000000003
```

The normalized equivalent width of the squared cosine function can be computed exactly using standard methods from a calculus course. It's value is exactly 1.5 If we use a smaller dx when computing w, we get a better approximation. Use width 10 and spacing 0.001.

```
>>> from rss_ringoccs import diffrec as dc
>>> w = dc.window_functions.coss(10, 0.001)
>>> normeq = dc.special_functions.compute_norm_eq(w)
>>> print(normeq)
1.50015
```

rss_ringoccs.diffrec.special_functions.**d2psi**(*kD*, *r*, *r0*, *phi*, *phi0*, *B*, *D*)

Purpose: Compute $\mathrm{d}^2\psi/\mathrm{d}\phi^2$

Arguments:

kD (*float*) Wavenumber, unitless.

r (*float*) Radius of reconstructed point, in kilometers.

r0 (*np.ndarray*) Radius of region within window, in kilometers.

phi (*np.ndarray*) Root values of $\mathrm{d}\psi/\mathrm{d}\phi$, radians.

phi0 (*np.ndarray*) Ring azimuth angle corresponding to r0, radians.

B (*float*) Ring opening angle, in radians.

D (*float*) Spacecraft-RIP distance, in kilometers.

Outputs:

dpsi (*np.ndarray*) Second partial derivative of ψ with respect to ϕ.

`rss_ringoccs.diffrec.special_functions.`**`double_slit_diffraction`**(*x*, *z*, *a*, *d*)

Purpose: Compute diffraction through a double slit for the variable x with a distance z from the slit and slit parameter a and a distance d between the slits. This assumes Fraunhofer diffraction.

Variables:

x A real or complex argument, or numpy array.

z (*float*) The perpendicular distance from the slit plane to the observer.

a (*float*) The slit parameter. This is a unitless paramter defined as the ratio between the slit width and the wavelength of the incoming signal.

d (*float*) The distance between slits.

Outputs:

f Single slit diffraction pattern.

`rss_ringoccs.diffrec.special_functions.`**`dpsi`**(*kD*, *r*, *r0*, *phi*, *phi0*, *B*, *D*)

Purpose: Compute $\mathrm{d}\psi/\mathrm{d}\phi$

Arguments:

kD (*float*) Wavenumber, unitless.

r (*float*) Radius of reconstructed point, in kilometers.

r0 (*np.ndarray*) Radius of region within window, in kilometers.

phi (*np.ndarray*) Root values of $\mathrm{d}\psi/\mathrm{d}\phi$, radians.

phi0 (*np.ndarray*) Ring azimuth angle corresponding to r0, radians.

B (*float*) Ring opening angle, in radians.

D (*float*) Spacecraft-RIP distance, in kilometers.

Outputs:

dpsi (*array*) Partial derivative of ψ with respect to ϕ.

`rss_ringoccs.diffrec.special_functions.`**`dpsi_ellipse`**(*kD*, *r*, *r0*, *phi*, *phi0*, *B*, *D*, *ecc*, *peri*)

Purpose: Compute $\mathrm{d}\psi/\mathrm{d}\phi$

Arguments:

kD (*float*) Wavenumber, unitless.

r (*float*) Radius of reconstructed point, in kilometers.

r0 (*np.ndarray*) Radius of region within window, in kilometers.

phi (*np.ndarray*) Root values of $\mathrm{d}\psi/\mathrm{d}\phi$, radians.

phi0 (*np.ndarray*) Ring azimuth angle corresponding to r0, radians.

B (*float*) Ring opening angle, in radians.

D (*float*) Spacecraft-RIP distance, in kilometers.

Outputs:

dpsi (*array*) Partial derivative of ψ with respect to ϕ.

`rss_ringoccs.diffrec.special_functions.`**`fresnel_cos`**(*x*)

Purpose: Compute the Fresnel cosine function.

Arguments:

x (***np.ndarray*** **or** ***float***) A real or complex number, or numpy array.

Outputs:

f_cos (***np.ndarray*** **or** ***float***) The fresnel cosine integral of x.

Notes:

1. The Fresnel Cosine integral is the solution to the equation $\mathrm{d}y/\mathrm{d}x = \cos(\frac{\pi}{2}x^2)$, $y(0) = 0$. In other words, $y = \int_{t=0}^{x}\cos(\frac{\pi}{2}t^2)\mathrm{d}t$
2. The Fresnel Cosine and Sine integrals are computed by using the scipy.special Error Function. The Error Function, usually denoted Erf(x), is the solution to $\mathrm{d}y/\mathrm{d}x = \frac{2}{\sqrt{\pi}}\exp(-x^2)$, $y(0) = 0$. That is: $y = \frac{2}{\sqrt{\pi}}\int_{t=0}^{x}\exp(-t^2)\mathrm{d}t$. Using Euler's Formula for exponentials allows one to use this to solve for the Fresnel Cosine integral.
3. The Fresnel Cosine integral is used for the solution of diffraction through a square well. Because of this it is useful for forward modeling problems in radiative transfer and diffraction.

Examples: Compute and plot the Fresnel Cosine integral.

```
>>> import rss_ringoccs.diffcorr.special_functions as sf
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> x = np.array(range(0,10001))*0.01 - 50.0
>>> y = sf.fresnel_cos(x)
>>> plt.show(plt.plot(x,y))
```

`rss_ringoccs.diffrec.special_functions.`**`fresnel_scale`**(*Lambda*, *d*, *phi*, *b*, *deg=False*)

Purpose: Compute the Fresnel Scale from λ, D, ϕ, and B.

Arguments:

Lambda (***np.ndarray*** **or** ***float***) Wavelength of the incoming signal.

d (***np.ndarray*** **or** ***float***) RIP-Spacecraft Distance.

phi (***np.ndarray*** **or** ***float***) Ring azimuth angle.

b (***np.ndarray*** **or** ***float***) Ring opening angle.

Keywords:

deg (***bool***) Set True if ϕ or B are in degrees. Default is radians.

Output:

fres (***np.ndarray*** **or** ***float***) The Fresnel scale.

Note: λ and D must be in the same units. The output (Fresnel scale) will have the same units as λ and d. In addition, B and ϕ must also have the same units. If B and ϕ are in degrees, make sure to set deg=True. Default is radians.

`rss_ringoccs.diffrec.special_functions.`**`fresnel_sin`**(*x*)

Purpose: Compute the Fresnel sine function.

Variables:

x (***np.ndarray* or *float***) The independent variable.

Outputs:

f_sin (***np.ndarray* or *float***) The fresnel sine integral of x.

Notes:

1. The Fresnel sine integral is the solution to the equation $\mathrm{d}y/\mathrm{d}x = \sin(\frac{\pi}{2}x^2)$, $y(0) = 0$. In other words, $y = \int_{t=0}^{x}\sin(\frac{\pi}{2}t^2)dt$
2. The Fresnel Cossine and Sine integrals are computed by using the scipy.special Error Function. The Error Function, usually denoted Erf(x), is the solution to $\mathrm{d}y/\mathrm{d}x = \frac{2}{\sqrt{\pi}}\exp(-x^2)$, $y(0) = 0$. That is: $y = \frac{2}{\sqrt{\pi}}\int_{t=0}^{x}\exp(-t^2)dt$. Using Euler's Formula for exponentials allows one to use this to solve for the Fresnel Sine integral.
3. The Fresnel sine integral is used for the solution of diffraction through a square well. Because of this is is useful for forward modeling problems in radiative transfer and diffraction.

Examples: Compute and plot the Fresnel Sine integral.

```
>>> import rss_ringoccs.diffcorr.special_functions as sf
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> x = np.array(range(0,10001))*0.01 - 50.0
>>> y = sf.fresnel_sin(x)
>>> plt.show(plt.plot(x,y))
```

`rss_ringoccs.diffrec.special_functions.`**`inverse_square_well_diffraction`**(*x*, *a*, *b*, *F*)

`rss_ringoccs.diffrec.special_functions.`**`psi`**(*kD*, *r*, *r0*, *phi*, *phi0*, *B*, *D*)

Purpose: Compute ψ (MTR Equation 4)

Arguments:

kD (***float***) Wavenumber, unitless.

r (***float***) Radius of reconstructed point, in kilometers.

r0 (***np.ndarray***) Radius of region within window, in kilometers.

phi (***np.ndarray***) Root values of $\mathrm{d}\psi/\mathrm{d}\phi$, radians.

phi0 (***np.ndarray***) Ring azimuth angle corresponding to r0, radians.

B (***float***) Ring opening angle, in radians.

D (***float***) Spacecraft-RIP distance, in kilometers.

Outputs:

psi (***np.ndarray***) Geometric Function from Fresnel Kernel.

`rss_ringoccs.diffrec.special_functions.`**`resolution_inverse`**(*x*)

Purpose: Compute the inverse of $y = x/(\exp(-x) + x - 1)$

Arguments:

x (***np.ndarray* or *float***) Independent variable

Outputs:

f (*np.ndarray* or *float*) The inverse of $x/(\exp(-x)+x-1)$

Dependencies:

1. numpy
2. scipy.special

Method: The inverse of $x/(\exp(-x)+x-1)$ is computed using the LambertW function. This function is the inverse of $y = x\exp(x)$. This is computed using the scipy.special subpackage using their lambertw function.

Warnings:

1. The real part of the argument must be greater than 1.
2. The scipy.special lambertw function is slightly inaccurate when it's argument is near $-1/e$. This argument is $z = \exp(x/(1-x)) * x/(1-x)$

Examples: Plot the function on the interval (1,2)

```
>>> import rss_ringoccs.diffcorr.special_functions as sf
>>> import numpy as np
>>> x = np.array(range(0,1001))*0.001+1.01
>>> y = sf.resolution_inverse(x)
>>> import matplotlib.pyplot as plt
>>> plt.show(plt.plot(x,y))
```

`rss_ringoccs.diffrec.special_functions.`**`savitzky_golay`**(*y*, *window_size*, *order*, *deriv=0*, *rate=1*)

Purpose: To smooth data with a Savitzky-Golay filter. This removes high frequency noise while maintaining many of the original features of the input data.

Arguments:

y (*np.ndarray*) The input "Noisy" data.

window_size (*int*) The length of the window. Must be an odd number.

order (*int*) The order of the polynomial used for filtering. Must be less then window_size - 1.

Keywords:

deriv (*int*) The order of the derivative what will be computed.

Output:

y_smooth (*np.ndarray*) The data smoothed by the Savitzky-Golay filter. This returns the nth derivative if the deriv keyword has been set.

Notes: The Savitzky-Golay is a type of low-pass filter, particularly suited for smoothing noisy data. The main idea behind this approach is to make for each point a least-square fit with a polynomial of high order over a odd-sized window centered at the point.

`rss_ringoccs.diffrec.special_functions.`**`single_slit_diffraction`**(*x*, *z*, *a*)

Purpose: Compute diffraction through a single slit for the variable x with a distance z from the slit and slit parameter a. This assume Fraunhofer diffraction.

Variables:

x A real or complex argument, or numpy array.

z (*float*) The perpendicular distance from the slit plane to the observer.

a (*float*) The slit parameter. This is a unitless paramter defined as the ratio between the slit width and the wavelength of the incoming signal.

Outputs:

f Single slit diffraction pattern.

`rss_ringoccs.diffrec.special_functions.`**`square_well_diffraction`**`(`x, a, b, F`)`

rss_ringoccs.diffrec.window_functions module

Purpose: Provide a suite of window functions and functions related to the normalized equivalent width of a given array.

Dependencies:

1. numpy
2. spicy

`rss_ringoccs.diffrec.window_functions.`**`coss`**`(`x, W`)`

Purpose: Compute the Cosine Squared Window Function.

Arguments:

x (*np.ndarray*) Real valued array used for the independent variable, or x-axis.

w_in (*float*) Width of the window. Positive float.

Outputs:

w_func (*np.ndarray*) Window function of width w_in evaluated along x.

`rss_ringoccs.diffrec.window_functions.`**`kb20`**`(`x, W`)`

Purpose: Compute the Kaiser-Bessel 2.0 Window Function.

Arguments:

x (*np.ndarray*) Real valued array used for the independent variable, or x-axis.

w_in (*float*) Width of the window. Positive float.

Outputs:

w_func (*np.ndarray*) Window function of width w_in evaluated along x.

`rss_ringoccs.diffrec.window_functions.`**`kb25`**`(`x, W`)`

Purpose: Compute the Kaiser-Bessel 2.5 Window Function.

Arguments:

x (*np.ndarray*) Real valued array used for the independent variable, or x-axis.

w_in (*float*) Width of the window. Positive float.

Outputs:

w_func (*np.ndarray*) Window function of width w_in evaluated along x.

`rss_ringoccs.diffrec.window_functions.`**`kb35`**`(`x, W`)`

Purpose: Compute the Kaiser-Bessel 3.5 Window Function.

Arguments:

x (*np.ndarray*) Real valued array used for the independent variable, or x-axis.

w_in (*float*) Width of the window. Positive float.

Outputs:

w_func (*np.ndarray*) Window function of width w_in evaluated along x.

`rss_ringoccs.diffrec.window_functions.`**`kbal`**(*x*, *W*, *alpha*)

Purpose: Create a Kaiser-Bessel window with a user specified alpha parameter.

Variables:

W (*float*) Window width.

dx (*float*) Width of one point.

al (*float*) The alpha parameter α_0.

Outputs:

w_func (*np.ndarray*) The Kaiser-Bessel alpha window of width w_in and spacing dx between points.

Notes:

1. The Kaiser-Bessel window is computed using the modified Bessel Function of the First Kind. It's value is $y = I_0(\alpha\sqrt{1-4x^2/w^2})/I_0(\alpha)$, where w is the window width.
2. We automatically multiply the alpha parameter by pi, so the kbal window function has an alpha value of $\alpha = \alpha_0\pi$
3. The endpoints of the Kaiser-Bessel function tend to zero faster than $(1+2\alpha)/\exp(\alpha)$

Warnings:

1. None of the Kaiser-Bessel windows evaluate to zero at their endpoints. The endpoints are $1/I_0(\alpha)$. For small values of alpha this can create Gibb's like effects in reconstruction do to the large discontinuity in the window. For alpha values beyond 2.5π this effect is negligible.

`rss_ringoccs.diffrec.window_functions.`**`kbmd20`**(*x*, *W*)

Purpose: Compute the Modified Kaiser-Bessel 2.0 Window Function.

Arguments:

x (*np.ndarray*) Real valued array used for the independent variable, or x-axis.

w_in (*float*) Width of the window. Positive float.

Outputs:

w_func (*np.ndarray*) Window function of width w_in evaluated along x.

`rss_ringoccs.diffrec.window_functions.`**`kbmd25`**(*x*, *W*)

Purpose: Compute the Modified Kaiser-Bessel 2.5 Window Function.

Arguments:

x (*np.ndarray*) Real valued array used for the independent variable, or x-axis.

w_in (*float*) Width of the window. Positive float.

Outputs:

w_func (*np.ndarray*) Window function of width w_in evaluated along x.

`rss_ringoccs.diffrec.window_functions.`**`kbmd35`**(*x*, *W*)

Purpose: Compute the Modified Kaiser-Bessel 3.5 Window Function.

Arguments:

x (*np.ndarray*) Real valued array used for the independent variable, or x-axis.

w_in (*float*) Width of the window. Positive float.

Outputs:

w_func (*np.ndarray*) Window function of width w_in evaluated along x.

`rss_ringoccs.diffrec.window_functions.`**`kbmdal`**(*x*, *W*, *al*)

Purpose: Create a modifed Kaiser-Bessel window with a user specified alpha parameter.

Variables:

W (*float*) Window width.

dx (*float*) Width of one point.

al (*float*) The alpha parameter α_0.

Outputs:

w_func (*np.ndarray*) The Kaiser-Bessel alpha window of width and w_in spacing dx between points.

Notes:

1. The Kaiser-Bessel window is computed using the modified Bessel Function of the First Kind. It's value is $y = I_0(\alpha\sqrt{1-4x^2/w^2})/I_0(\alpha)$, where w is the window width.
2. We automatically multiply the alpha parameter by pi, so the kbal window function has an alpha value of $\alpha = \alpha_0\pi$
3. The endpoints of the Kaiser-Bessel function tend to zero faster than $(1+2\alpha))/\exp(\alpha)$

Warnings:

1. None of the Kaiser-Bessel windows evaluate to zero at their endpoints. The endpoints are $1/I_0(\alpha)$. For small values of alpha this can create Gibb's like effects in reconstruction due to the large discontinuity in the window. For alpha values beyond 2.5π this effect is negligible.

`rss_ringoccs.diffrec.window_functions.`**`normalize`**(*dx*, *ker*, *f_scale*)

Purpose: Compute the window normalization

Arguments:

ker (*np.ndarray*) The Fresnel Kernel.

dx (*float*) The spacing between points in the window. This is equivalent to the sample spacing. This value is in kilometers.

f_scale (*np.ndarray*) The Fresnel Scale in kilometers.

Outputs:

norm_fact (*float*) The normalization of the input Fresnel Kernel.

`rss_ringoccs.diffrec.window_functions.`**`rect`**(*x*, *W*)

Purpose: Compute the rectangular window function.

Arguments:

x (*np.ndarray*) Real valued array used for the independent variable, or x-axis.

w_in (*float*) Width of the window. Positive float.

Outputs:

w_func (*np.ndarray*) Window function of width w_in evaluated along x.

`rss_ringoccs.diffrec.window_functions.`**`window_width`**(*res*, *normeq*, *fsky*, *fres*, *rho_dot*, *sigma*, *bfac=True*, *Return_P=False*)

Purpose: Compute the window width as a function of ring radius. This is given from MTR86 Equations 19, 32, and 33.

Variables:

res (*float*) The requested resolution.

normeq (*float*) The normalized equivalent width. Unitless.

fsky (*float* **or** *np.ndarray*) The sky frequency.

fres (*float* **or** *np.ndarray*) The Fresnel scale.

rdot (*float*) **or** (*np.ndarray*) The time derivative of the ring radius.

Output:

w_vals (*np.ndarray*) The window width as a function of ring radius.

1.1.3 rss_ringoccs.occgeo package

Purpose Calculate occultation geometry parameters listed within CORSS_8001 v2 *GEO.TAB and *GEO.LBL files, as well as geometrical quantities needed for calibrating RSS ring data.

1.1.3.1 Submodules

rss_ringoccs.occgeo.calc_occ_geometry module

Purpose Functions for calculating occultation geometry parameters listed in *GEO.LBL file from Archived_Cassini_RSS_RingOccs_2018/ and other useful geometry information, such as free-space regions and planetary occultation times.

Dependencies

1. spiceypy
2. numpy
3. scipy

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_B_deg`**(*et_vals*, *spacecraft*, *dsn*, *nhat_p*, *kernels=None*, *ref='J2000'*)

This calculates ring opening angle, or the observed ring elevation, as the complement to the angle made by the planet pole vector and the spacecraft to DSN vector

Arguments

et_vals (*np.ndarray*) Array of earth-received times in ephemeris seconds.

dsn (*str*) DSN observing station ID – must be compatible with NAIF.

nhat_p (*np.ndarray*) 1x3 array unit vector in planet pole direction.

Keyword Arguments

kernels (*str* or *list*) List of NAIF kernels, including path.

ref (*str*) Reference frame to be used in spiceypy calls. Default is 'J2000'

Returns

B_deg_vals (*np.ndarray*) Array of ring opening angle in degrees.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_B_eff_deg`**(*B_deg*, *phi_ora_deg*)

This calculates the effective ring opening angle in degrees.

Arguments

B_deg (*float* or *np.ndarray*) Ring opening angle in degrees.

phi_ora_deg (*float* or *np.ndarray*) Observed ring azimuth in degrees.

Returns

B_eff_deg (*float* or *np.ndarray*) Effective ring opening angle in degrees.

Notes

1. Reference: [GRESH86] Eq. 16

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_D_km`**(*t1*, *t2*)

This calculates the light distance between two input times, in km.

Args

t1 (*np.ndarray*) Array of time in seconds.

t2 (*np.ndarray*) Array of time in seconds.

Returns

D_km_vals (*np.ndarray*) Array of light distance in km.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_F_km`**(*D_km_vals*, *f_sky_hz_vals*, *B_deg_vals*, *phi_ora_deg_vals*)

This calculates the Fresnel scale using Eq. 6 of [MTR1986].

Arguments

D_km_vals (*np.ndarray*) Array of spacecraft to ring intercept point distances in km.

f_sky_hz_vals (*np.ndarray*) Array of downlink sinusoidal signal frequency at the front-end of observing dsn station, in Hz.

B_deg_vals (*np.ndarray*) Array of ring opening angle in degrees.

phi_ora_deg_vals (*np.ndarray*) Array of observed ring azimuth in degrees.

Returns

F_km_vals (*np.ndarray*) Array of Fresnel scale in km.

Notes

1. diffcorr uses an independently-calculated Fresnel scale
2. Reference: [MTR1986] Equation 6

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_beta`**(*B_deg*, *phi_ora_deg*)

This calculates the optical depth enhancement factor.

Arguments

B_deg (*float* or *np.ndarray*) Ring opening angle in degrees.

phi_ora_deg (*float* or *np.ndarray*) Observed ring azimuth in degrees.

Returns

beta (*np.ndarray*) Optical depth enhancement factor.

rss_ringoccs.occgeo.calc_occ_geometry.**calc_elevation_deg**(*et_vals*, *target*, *obs*, *kernels=None*)

Calculate the elevation of a target above the horizon for a given observer.

Arguments

et_vals (*np.ndarray*) Array of observed event times in ET sec.

target (*str*) Target name – must be compatible with NAIF. This will typically be spacecraft or planet name.

obs (*str*) Observer name – must be compatible with NAIF. This will typically be observing dsn station name. Observer is assumed to be on Earth.

kernels (*str* or *list*) List of NAIF kernels, including path.

Returns

elev_deg_vals (*np.ndarray*) Array of elevation angles in degrees.

rss_ringoccs.occgeo.calc_occ_geometry.**calc_impact_radius_km**(*R_sc_km_vals*, *et_vals*, *spacecraft*, *dsn*, *nhat_p*, *ref='J2000'*, *kernels=None*)

This calculates the closest approach of the spacecraft signal to the planet defined as a sphere.

Arguments

R_sc_km_vals (*list*) List of 3-element arrays of spacecraft position vector in planetocentric frame at input et_vals.

et_vals (*np.ndarray*) Array of Earth-received times in ephemeris seconds.

spacecraft (*str*) Spacecraft name

dsn (*str*) DSN observing station ID

nhat_p (*np.ndarray*) 1x3 array unit vector in planet pole direction.

Keyword Arguments

kernels (*list*) List of NAIF kernels, including path.

ref (*str*) Reference frame to be used in spiceypy calls. Default is 'J2000'

Returns

R_imp_km_vals (*np.ndarray*) Array of impact radius in km.

rss_ringoccs.occgeo.calc_occ_geometry.**calc_phi_deg**(*et_vals*, *rho_vec_km_vals*, *spacecraft*, *dsn*, *nhat_p*, *ref='J2000'*, *kernels=None*)

This calculates observed ring azimuth and ring longitude.

Arguments

et_vals (*np.ndarray*) Array of earth-received time in ET seconds.

rho_vec_km_vals (*np.ndarray*) Nx3 array of ring intercept position vectors in km.

spacecraft (*str*) Name of spacecraft

dsn (*str*) DSN observing station ID

nhat_p (*np.ndarray*) 1x3 array unit vector in planet pole direction.

Keyword Arguments

kernels (*str* or *list*) List of NAIF kernels, including path

ref (*str*) Reference frame to be used in spiceypy calls. Default is 'J2000'

Returns

phi_rl_deg_vals (*np.ndarray*) Array of inertial longitude in degrees.

phi_ora_deg_vals (*np.ndarray*) Array of observed ring azimuth in degrees.

Notes:

1. phi_ora_deg differs from the [MTR1986] definition by 180 degrees.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_rho_km`**(*et_vals*, *planet*, *spacecraft*, *dsn*, *kernels=None*, *ring_frame=None*)

Calculate the distance between Saturn center to ring intercept point.

Arguments

et_vals (*np.ndarray*) Array of observed event times in ET sec.

planet (*str*) Planet name

spacecraft (*str*) Spacecraft name

dsn (*str*) DSN observing station ID

Keyword Arguments

kernels (*str* or *list*) List of NAIF kernels, including path.

ring_frame (*str*) Ring plane frame. Default is the equatorial frame, (e.g. 'IAU_SATURN')

Returns

rho_km_vals (*np.ndarray*) Array of ring intercept points in km.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_rho_vec_km`**(*et_vals*, *planet*, *spacecraft*, *dsn*, *ref='J2000'*, *kernels=None*, *verbose=False*, *ring_frame=None*)

This calculates the position vector of the ring intercept point from the planet center in J2000 frame.

Arguments

et_vals (*np.ndarray*) Array of earth-received times in ET sec

planet (*str*) Name of planet

spacecraft (*str*) Name of spacecraft

dsn (*str*) DSN observing station ID

Keyword Arguments

kernels (*str* or *list*) Path to NAIF kernels

verbose (*bool*) Option for printing processing steps

ring_frame (*str*) Ring plane frame. Default is the equatorial frame, (e.g. 'IAU_SATURN')

ref (*str*) Reference frame to be used in spiceypy calls. Default is 'J2000'

Returns

rho_vec_km_vals (*list*) List of 3xN np.ndarrays of the planet center to ring intercept point position vector in J2000 frame

t_ret_et_vals (*np.ndarray*) Array of ring event times in ET seconds.

References

1. Ring intercept point calculation using a dynamical frame. See [NAIF] page 19.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_rip_velocity`**(*rho_km_vals*, *phi_rl_deg_vals*, *dt*)

This calculates the ring intercept point radial and azimuthal velocity.

Arguments

rho_km_vals (*np.ndarray*) Array of ring intercept points in km.

phi_rl_deg_vals (*np.ndarray*) Array of ring longitudes in degrees.

dt (*float*) Constant time spacing between points.

Returns

rho_dot_kms_vals (*np.ndarray*) Array of ring intercept radial velocities in km/s.

phi_rl_dot_kms_vals (*np.ndarray*) Array of ring intercept azimuthal velocties in km/s.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_sc_state`**(*et_vals*, *spacecraft*, *planet*, *dsn*, *nhat_p*, *ref='J2000'*, *kernels=None*)

This calculates spacecraft state vector in a planetocentric frame.

Arguments

et_vals (*np.ndarray*) Array of spacecraft event times in ET seconds.

spacecraft (*str*) Spacecraft name

planet (*str*) Planet name

dsn (*str*) Deep Space Network observing station ID

nhat_p (*np.ndarray*) 1x3 array unit vector in planet pole direction.

Keyword Arguments

kernels (*str* or *list*) Path to NAIF kernel(s)

ref (*str*) Reference frame to be used in spiceypy calls. Default is 'J2000'

Returns

R_sc_km_vals (*list*) List of Nx3 np.ndarrays of spacecraft position vector in km in planetocentric frame

R_sc_dot_kms_vals (*list*) List of Nx3 np.ndarrays of spacecraft velocity vector in km/s.

Notes

1. Saturn planetocentric frame is defined by x-axis parallel to projection of spacecraft-to-Earth line-of-sight, z-axis in direction of Saturn's pole.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`calc_set_et`**(*et_vals*, *spacecraft*, *dsn*, *kernels=None*)

This calculates the time at which photons left the spacecraft, in ET sec.

Arguments

et_vals (*np.ndarray*) Array of earth-received times in ET seconds.

spacecraft (*str*)

dsn (*str*) Deep Space Network observing station ID

Keyword Arguments

kernels (*str* or *list) Path to NAIF kernels

Returns

t_set_et_vals (*np.ndarray*) Array of spacecraft event times in ET sec.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`find_gaps`**(*t_ret_spm_vals*, *year*, *doy*, *rho_km_vals*, *phi_rl_deg_vals*, *niter=100*, *tolerance=0.001*, *t0=2454467.0*, *gaps_file='../tables/gap_orbital_elements.txt'*, *kernels=None*)

Find regions of free-space power (gaps) in the ring system.

Arguments

t_ret_spm_vals (*np.ndarray*) Ring event times in SPM

year (*str*) Reference year for seconds past midnight

doy (*str*) Reference day of year for seconds past midnight

rho_km_vals (*np.ndarray*) Ring intercept points in km

phi_rl_deg_vals (*np.ndarray*) Inertial ring longitude in deg.

Keyword Arguments

niter (*int*) Maximum number of iterations

tolerance (*float*) Minimum difference between new and old guess for converging on a solution

t0 (*float*) Reference epoch UTC 2008 January 1 12:00:00 for values in gaps_file, in Julian date

gaps_file (*str*) Path to text file with orbital elements of Saturn ring features

kernels (*str* or *list*) Path to NAIF kernels

Returns

gap_bounds (*list*) List of 1x2 lists of gap boundaries in km

Notes

1. Reference: [NICH14]
2. Given the default "gaps_file" keyword argument, this script must be run in a directory one level below the top-level rss_ringoccs directory.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`get_freespace`**(*t_ret_spm_vals*, *year*, *doy*, *rho_km_vals*, *phi_rl_deg_vals*, *t_oet_spm_vals*, *atmos_occ_spm_vals*, *kernels=None*, *split_ind=None*)

Return list of gap boundaries (inner and outer edge) in distance from center of Saturn and in seconds past midnight.

Arguments

t_ret_spm_vals (*np.ndarray*) Ring event times in SPM

year (*str*) Reference year for seconds past midnight

doy (*str*) Reference day of year for seconds past midnight

rho_km_vals (*np.ndarray*) Ring intercept points in km

phi_rl_deg_vals (*np.ndarray*) Inertial ring longitude in deg.

t_oet_spm_vals (*np.ndarray*) Observed event times in SPM

atmos_occ_spm_vals (*np.ndarray*) SPM times of when spacecraft signal is blocked by planet atmosphere

Keyword Arguments

kernels (*str* or *list*) Path to NAIF kernels

split_ind (*int*) Index of when a chord event switches from ingress to egress

Returns

gaps_km (*list*) List of 1x2 lists of gap boundaries in km

gaps_spm (*list*) List of 1x2 lists of gap boundaries in SPM

`rss_ringoccs.occgeo.calc_occ_geometry.`**`get_freespace_km`**(*ret_spm*, *year*, *doy*, *rho_km*, *phi_rl_deg*)

Get all free-space regions, in and outside ring system.

Arguments

ret_spm (*np.ndarray*) Ring event times in SPM

year (*str*) Reference year

:doy (*str*) Reference day of year :rho_km (*np.ndarray*): Ring intercept points in km :phi_rl_deg (*np.ndarray*): Inertial ring longitudes in deg

Returns

freespace_km (*list*) List of free-space boundaries in km

`rss_ringoccs.occgeo.calc_occ_geometry.`**`get_planet_occ_times`**(*et_vals*, *obs*, *planet*, *spacecraft*, *height_above=500.0*, *kernels=None*)

Return times when the spacecraft-to-observer ray is blocked by planet.

Arguments

et_vals (*np.ndarray*) Array of observed event times in ET sec.

obs (*str*) Observer name

planet (*str*) Planet name

spacecraft (*str*) Spacecraft name

Keyword Arguments

height_above (*float*) Height in km to be added to planet radius to account for the atmosphere

kernels (*str* or *list*) Path to NAIF kernels

Returns

et_blocked_vals (*np.ndarray*) Array of observed event times in ET

Note:

1. This was made to be generalizable to different planets, but has been only tested with planet='Saturn'.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`get_pole`**(*et*, *planet*, *kernels=None*)

Calculate unit vector in pole direction from kernel constants.

Arguments

et (*float*) Ephemeris seconds past J2000

planet (*str*) Planet name

Keyword Arguments

kernels (*str* or *list*) Path to NAIF kernels

Returns

nhat_p (*np.ndarray*) 1x3 unit vector in pole direction.

Note:

1. Quadratic terms for pole direction are typically zero but are retained here for consistency with PCK file format definitions.

`rss_ringoccs.occgeo.calc_occ_geometry.`**`get_start_jd`**(*year*, *doy*)

Get the start of a day in Julian date times.

Arguments

year (*str*) Year

doy (*str*) Day of year

Returns

start_jd (*float*) Julian date time of the start of a day

`rss_ringoccs.occgeo.calc_occ_geometry.`**`rad_converge`**(*t_ret_spm_vals*, *rho_km_vals*, *phi_rl_deg_vals*, *semimajor*, *eccentricity*, *curlypi_0*, *curlypi_dot*, *niter=100*, *tolerance=0.001*)

Computes initial guess for radius of ring feature using the semimajor axis. Selects time and longitude closest to guess and computes true anomaly for a new radius guess. Continues this estimation method iteratively until difference between new and old radius guesses is less than some tolerance or maximum number of iterations is reached.

Arguments

t_ret_spm_vals (*np.ndarray*) Ring event times in SPM.

rho_km_vals (*np.ndarray*) Ring intercept points in km.

phi_rl_deg_vals (*np.ndarray*) Inertial ring longitude in deg.

semimajor (*float*) Semimajor axis of ring feature in km.

eccentricity (*float*) Eccentricity of ring feature.

*curlypi_0 (*float*): Longitude of periapse in degrees. *curlypi_dot (*float*): Apsidal precession rate in degrees/day.

Keyword Arguments

niter (*int*) Maximum number of iterations

tolerance (*float*) Minimum difference between new and old guess for converging on a solution

Returns

radius_new (*float*) Estimated radius of ring feature in km

Notes

1. Reference: [NICH14]

`rss_ringoccs.occgeo.calc_occ_geometry.`**`remove_blocked`**(*t_oet_spm_vals*, *atmos_occ_spm_vals*, *t_ret_spm_vals*, *phi_rl_deg_vals*, *rho_km_vals*)

Remove values that occur during times blocked by planet atmosphere.

Arguments

t_oet_spm_vals (*np.ndarray*) Observed event times in SPM

atmos_occ_spm_vals (*np.ndarray*) SPM times of when spacecraft signal is blocked by planet atmosphere

t_ret_spm_vals (*np.ndarray*) Ring event times in SPM

phi_rl_deg_vals (*np.ndarray*) Inertial ring longitude in deg.

rho_km_vals (*np.ndarray*) Ring intercept points in km

Returns

t_ret_spm_vals (*np.ndarray*) Ring event times in SPM, excluding atmospheric occultation times

t_oet_spm_vals (*np.ndarray*) Observed event times in SPM, excluding atmospheric occultation times

phi_rl_deg_vals (*np.ndarray*) Inertial ring longitude in deg, excluding atmospheric occultation times

rho_km_vals (*np.ndarray*) Ring intercept points in km, excluding atmospheric occultation times

`rss_ringoccs.occgeo.calc_occ_geometry.`**`split_chord_arr`**(*t_ret_spm_vals*, *t_oet_spm_vals*, *atmos_occ_spm_vals*, *phi_rl_deg_vals*, *rho_km_vals*, *ind*, *profdir*)

Return array of only ingress or egress portion of a chord occultation.

Arguments

t_ret_spm_vals (*np.ndarray*) Ring event times in SPM

t_oet_spm_vals (*np.ndarray*) Observed event times in SPM

atmos_occ_spm_vals (*np.ndarray*) SPM times of when spacecraft signal is blocked by planet atmosphere

phi_rl_deg_vals (*np.ndarray*) Inertial ring longitude in deg.

rho_km_vals (*np.ndarray*) Ring intercept points in km

ind (*int*) Index of where ingress switches to egress

profdir (*str*) Profile direction to return ('"INGRESS"' or '"EGRESS"')

Returns

t_ret_spm_vals (*np.ndarray*) Ring event times in SPM of 'profdir' portion of occultation

t_oet_spm_vals (*np.ndarray*) Observed event times in SPM of 'profdir' portion of occultation

phi_rl_deg_vals (*np.ndarray*) Inertial ring longitude in deg of 'profdir' portion of occultation

rho_km_vals (*np.ndarray*) Ring intercept points in km of 'profdir' portion of occultation

`rss_ringoccs.occgeo.calc_occ_geometry.`**`xform_j2k_to_pcf`**(*vec*, *et*, *spacecraft*, *dsn*, *nhat_p*, *ref='J2000'*, *kernels=None*)

Transform vector in J2000 frame to planet ring plane frame.

Arguments

vec (*np.ndarray*) 3-element vector in J2000 frame

et (*float*) ET in seconds corresponding to input vec

dsn (*str*) DSN observing station ID

nhat_p (*np.ndarray*) 1x3 array unit vector in planet pole direction.

Keyword Arguments

kernels (*str* or *list*) Path to NAIF kernels

Returns

out_vec (*np.ndarray*) 3-element vector in planet ring plane frame.

rss_ringoccs.occgeo.occgeo module

Purpose Calculate occultation geometry for RSS ring events.

Notes

1. kernels list must include:
   1) spacecraft ephemeris kernel
   2) planetary constants kernel
   3) leapseconds kernel
   4) planet and lunar ephemeris kernel
   5) earth stations kernel
   6) earth rotation and constants kernel
   7) topocentric frame kernel

Dependencies

1. scipy
2. numpy
3. spiceypy

class `rss_ringoccs.occgeo.occgeo.`**`Geometry`**(*rsr_inst*, *planet*, *spacecraft*, *kernels*, *pt_per_sec=1.0*, *ref='J2000'*, *ring_frame=None*, *nhat_p=None*, *verbose=False*, *write_file=True*)

Bases: `object`

Purpose This is an object that calculates occultation geometry needed for diffraction reconstruction as well as other relevant geometry parameters.

Arguments

rsr_inst (*class*) Instance of RSRReader class.

kernels (*str* or *list*) List of NAIF kernels, including path.

planet (*str*) Planet name

spacecraft (*str*) Spacecraft name

Keyword Arguments

pt_per_sec (*float*) Number of points calculated per second for all geometry calculations.

verbose (*bool*) Boolean for whether processing steps are printed.

write_file (*bool*) Boolean for whether output *GEO.TAB and *GEO.LBL files will be created.

ref (*str*) Reference frame to be used in spiceypy calls. Default is 'J2000'

ring_frame (*str*) Ring plane frame. Default is the equatorial frame, (e.g. 'IAU_SATURN')

nhat_p (*list*) Unit vector in pole direction, in rectangular coordinates. If None, it will be calculated using contents of the planetary constants kernel.

Attributes

t_oet_spm_vals (*np.ndarray*) Observed event time in seconds past midnight.

t_ret_spm_vals (*np.ndarray*) Ring event time in seconds past midnight.

t_set_spm_vals (*np.ndarray*) Spacecraft event time in seconds past midnight.

rho_km_vals (*np.ndarray*) Distance in km from the center of Saturn to ring intercept point.

phi_rl_deg_vals (*np.ndarray*) Ring longitude (inertial longitude) in degrees.

phi_ora_deg_vals (*np.ndarray*) Observed ring azimuth in degrees.

D_km_vals (*np.ndarray*) Spacecraft to ring intercept point distance in km.

B_deg_vals (*np.ndarray*) Ring opening angle in degrees.

rho_dot_kms_vals (*np.ndarray*) Ring intercept radial velocity in km/s.

phi_rl_dot_kms_vals (*np.ndarray*) Ring intercept azimuthal velocity in km/s.

F_km_vals (*np.ndarray*) Fresnel scale in km.

R_imp_km_vals (*np.ndarray*) Impact radius in km.

rx_km_vals (*np.ndarray*) x-component of spacecraft position in a planetocentric frame, in km.

ry_km_vals (*np.ndarray*) y-component of spacecraft position in a planetocentric frame, in km.

rz_km_vals (*np.ndarray*) z-component of spacecraft position in a planetocentric frame, in km.

vx_kms_vals (*np.ndarray*) x-component of spacecraft velocity in a planetocentric frame, in km/s.

vy_kms_vals (*np.ndarray*) y-component of spacecraft velocity in a planetocentric frame, in km/s

vz_kms_vals (*np.ndarray*) z-component of spacecraft velocity in a planetocentric frame, in km/s

elev_deg_vals (*np.ndarray*) Elevation angle in degrees.

kernels (*str* or *list*) List of NAIF kernels, including path.

rev_info (*dict*) RSR file specific info

history (*dict*) Dictionary of processing history.

naif_toolkit_version (*str*) NAIF toolkit version used (e.g., "V.N0066").

B_eff_deg_vals (*np.ndarray*) Effective ring opening angle in deg.

beta_vals (*np.ndarray*) Optical depth enhancement factor.

ionos_occ_spm_vals (*np.ndarray*) Array of seconds past midnight when the signal is occulted by the planet ionosphere, defined as 5000km above an ellipsoid with radii from cpck file.

atmos_occ_spm_vals (*np.ndarray*) Array of seconds past midnight when the signal is occulted by the planet atmosphere, defined as 500km above an ellipsoid with radii from cpck file.

freespace_spm (*list*) List of 2x1 lists of seconds past midnight values that define the inner and outer edge of a free-space gap in the ring system

freespace_km (*np.ndarray*) Array of 2x1 lists of km values that define the inner and outer edge of a free-space gap in the ring system

ul_rho_km_vals (*np.ndarray*) Uplink ring intercept points. This is only calculated for events after USO failure (after year 2010)

ul_phi_rl_deg_vals (*np.ndarray*) Ring longitude of the uplink ring intercept point. This is only calculated for events after USO failure (after year 2010)

ul_phi_ora_deg_vals (*np.ndarray*) Observed ring azimuth of the uplink ring intercept point. This is only calculated for events after USO failure (after year 2010)

add_info (*dict*) Additional information about changes to the data (e.g., removing points blocked by atmosphere, removing false ring intercept points from proximal orbits, etc.)

`get_chord_ind``()`

Return index of where radial velocity sign change occurs in a chord occultation.

Returns

ind (*int*) Index of where chord occultation goes from '"INGRESS"' to '"EGRESS"' or vice versa.

`get_profile_dir``()`

Return observed profile direction.

Returns

prof_dir (*str*) Profile direction as '"INGRESS"', '"EGRESS"', or '"BOTH"'.

`verify_chord``()`

Verify that an occultation with an increasing and decreasing radial velocity is actually a chord occultation.

Returns

prof_dir (*str*) Profile direction as '"INGRESS"', '"EGRESS"', or '"BOTH"'.

1.1.4 rss_ringoccs.rsr_reader package

Purpose:

Reads in raw RSR data and meta data from the file header. Stores pertinant information and raw data as attributes for future use. Provides method for predicting the offset frequency from spacecraft telemetry at the time of observation.

1.1.4.1 Submodules

rss_ringoccs.rsr_reader.rsr_reader module

Purpose Class to create an instance linked to an RSR file

Dependencies

1. multiprocessing
2. numpy
3. os
4. scipy
5. struct
6. sys
7. time

class `rss_ringoccs.rsr_reader.rsr_reader.`**`RSRReader`**(*rsr_file*, *decimate_16khz_to_1khz=True*, *verbose=False*)

Bases: `object`

Purpose

Reads the header of a raw RSR file when you first create an instance. Then reads the full RSR file to read in the raw measured complex signal $I + iQ$

Arguments

rsr_file (*str*) Full path name of a raw RSR file to read. RSR files can be downloaded using the shell script in the data" directory of the GitHub clone

Keyword Arguments

decimate_16khz_to_1khz (*bool*) Optional Boolean argument which, if set to True, decimates 16kHz files down to 1kHz sampling rate. Note that this is a sticky keyword - if you set it to True, it will be True for any subsequent calls from the instance until you explicitly set it to False. This keyword is linked to the private attribute __decimate_16khz_to_1khz

cpu_count (*int*) Number of cores to use when reading data in from file. Default is number of cores on your computer

verbose (*bool*) Optional boolean variable which, when set to True, prints the header attributes that were set

Attributes

rsr_file (*str*) Full path name of a raw RSR file to read

spm_vals (*np.ndarray*) Seconds Past Midnight array of times over entire rsr file

doy (*int*) Day of year of event

year (*int*) Year of event

dsn (*str*) Deep Space Network ID of the downlink station

band (*str*) Name of the wavelength of downlink transmission (S, X, or Ka)

ul_band (*str*) Name of the wavelength of uplink transmission

ul_dsn (*str*) Deep Space Network ID of uplink station

sample_rate_khz (*int*) Sample rate, in kHz, of transmission (1 or 16)

history (*dict*) Dictionary recording parameters of the run

Example

```
>>> # Import rss_ringoccs
>>> import rss_ringoccs as rss
>>> # Define instance and set header attributes, and read in raw data
>>> rsr_inst = rss.rsr_reader.RSRReader(rsr_file)
>>>  # Get predicted sky frequency at chosen SPM values f_spm
>>> f_spm_returned, f_sky_pred = rsr_inst.get_f_sky_pred(f_spm=f_spm)
```

Notes:

1. Setting `decimate_16khz_to_1khz=True` for a 1kHz file will be ignored
2. 16kHz files will take a few minutes to read and decimate

get_f_sky_pred(*f_spm=None*, *verbose=False*)
Calculate predicted sky frequency at user-defined times using polynomial coefficients in each SFDU.

Arguments

f_spm (*np.ndarray*) Array of SPM values to evaluate predicted sky frequency at. Default is at 1 second spacing over entire data set.

verbose (*bool*) Print the first few predicted sky frequency values if set to True

Returns

f_spm (*np.ndarray*) Array of SPM values that predicted sky frequency was evaluated at.

f_sky_pred (*np.ndarray*) Predicted sky frequency, calculated from the polynomial coefficients in the RSR file

1.1.5 rss_ringoccs.scatter package

Purpose: Compute the spectrogram of the incoherent signal based on properties of the measured signal like sample rate. Spectrogram is optionally stacked to improve SNR and output to a .TAB file named following the software's output file nomenclature.

1.1.5.1 Submodules

rss_ringoccs.scatter.spectro_reader module

`rss_ringoccs.scatter.spectro_reader.`**`read_spectro`**(*filename*)

rss_ringoccs.scatter.spectrogram module

`rss_ringoccs.scatter.spectrogram.`**`Scatter`**(*rsr_inst, geo_inst, cal_inst, rho_limits=[65000.0, 140000.0], stack=False, nstack=16, hires=False, numpts=None, nsegs=None*)

Purpose: Run spectrogram code and output results to file

Arguments:

> **rsr_inst (*obj*)** instance of RSR reader
>
> **geo_inst (*obj*)** instance of Geometry
>
> **cal_inst (*obj*)** instance of Calibration

Keyword Arguments:

> **rho_limits (*list*)** 2x1 list of radii boundaries in km over which to compute the spectrogram. Default is [65000,140000].
>
> **stack (*bool*)** specifies whether to stack the resulting spectrogram to improve scattered signal SNR. Default is True.
>
> **nstack (*int*)** number of spectrogram slices to stack in each bin. Only used if `stack` is set to True.
>
> **hires (*bool*)** specifying whether to compute spectrogram "manually" in a high resolution time sampling mode with a continuous Fourier transform (this circumvents the Gabor uncertainty limit posed by the discrete STFT). Default for `hires` is False. Note: this will take a substantial amount of time to compute and is not recommended.

`rss_ringoccs.scatter.spectrogram.`**`cont_stft`**(*time*, *signal*, *numpts=1000*, *nsegs=500*)

Purpose Compute the continuous STFT

Arguments:

> **time (*np.ndarray*)** one-dimensional array of times at which signal is sampled – sampling *MUST* be uniform
>
> **signal (*np.ndarray*)** the uniformly-sampled signal

Keyword arguments:

> **numpts (*int*)** number of points per STFT segment
>
> **nsegs (*int*)** number of segments in STFT

Notes:

1. Both `numpts` and `nsegs` MUST be smaller than the length of `time`
2. Continuous STFT will have dimensions `nsegs``x``numpts`

`rss_ringoccs.scatter.spectrogram.`**`spectro`**(*time*, *signal*, *stack=True*, *nstack=16*, *hires=False*, *numpts=None*, *nsegs=None*)

Purpose: Compute short-time Fourtier transform (STFT), i.e. the spectrogram, from a given signal.

Arguments:

> **time (*np.ndarray*)** array of times at which signal was measured
>
> **signal (*np.ndarray*)** array of signal values for which to compute the spectrogram, must match length of times

Keyword Arguments:

> **stack (*bool*)** boolean specifying whether to stack the spectrogram to improve signal-to-noise, default is True
>
> **nstack (*int*)** number of FFT segments to include in each bin when spectrogram is stacked. only used if `stack` is True. Default is 16.

hires (*bool*) boolean specifying whether to use a continuous Fourier transform to circumvent Gabor uncertainty when computing the STFT.

numpts (*int*) number of points to use in each STFT segment. Only used if `hires` is true, default 1000.

nsegs (*int*) number of segments to use in total STFT. Only used if `hires` is True, default is length of `time` / `nperseg`

Notes:

1. The continuous Fourier transform specified by `hires` is computationally expensive and will take substatially longer to run than the discrete STFT bound by the Gabor limit.
2. `numpts` * `nsegs` is not to exceed the length of `time`.

`rss_ringoccs.scatter.spectrogram.`**`stack_spec`**(*time*, *Sxx*, *N=16*)

Purpose: Stack spectrogram slices to improve SNR of the incoherent signal.

Arguments:

time (*np.ndarray*) Jx1 array of times at which the spectrogram was computed

Sxx (*np.ndarray*) IxJ array of spectrogram power values

Keyword Arguments:

N (*int*) number of FFT segments to include in each bin, must be less than J, default is 16

1.1.6 rss_ringoccs.tools package

Purpose: Provides multiple miscellaneous tools for data input/output, reconstructing objects, and converting information formats. Some are used by multiple object classes within *rss_ringoccs* while others are standalone scripts. See the User's Guide for details on which scripts the user might call directly.

Most relevant to the user is the *ExtractCSVData*, which serves a critical role in starting the QuickLook rendition of the processing pipeline.

1.1.6.1 Submodules

rss_ringoccs.tools.CSV_tools module

Purpose Provide tools for reading in .TAB and .CSV files and converting the data into a usable instance of the DLP class.

Dependencies

1. pandas
2. numpy
3. scipy

class `rss_ringoccs.tools.CSV_tools.`**`ExtractCSVData`**(*geo*, *cal*, *dlp*, *tau=None*, *verbose=True*, *use_deprecate=False*)

Bases: `object`

Purpose: Read three csv files (Geo, Cal, and DLP) and return an instance containing all necessary attributes to run diffraction correction. This instance can be fed directly into the DiffractionCorrection class.

Variables:

geo (*str*) A string that contains the location of the requested Geo file. Ex: geo = “/path/to/geo.CSV”

cal (*str*) A string that contains the location of the requested Cal file. Ex: cal = “/path/to/cal.CSV”

dlp (*str*)**:** A string that contains the location of the requested dlp file. Ex: dlp = “/path/to/dlp.CSV”

Keywords:

tau (*str*) A string that contains the location of the requested Tau file. If not set, variables from the tau file will have NoneType. Ex: tau = “/path/to/tau.CSV”

verbose (*bool*) A Boolean for specifying if various status updates will be printed to the command line.

Attributes:

B_rad_vals The ring opening angle of the ring plane with respect to the line of sight from Earth to the spacecraft.

D_km_vals The distance from the spacecraft to the ring-intercept point.

f_sky_hz_vals The sky frequency of the incoming signal.

p_norm_vals The normalized diffracted power.

phase_rad_vals The diffracted phase, in radians.

phase_vals The reconstructed phase contained in the tau file. If tau is not set, this will be a NoneType variable. Units are in radians.

phi_rad_vals The observed ring azimuth angle, in radians.

phi_rl_rad_vals The observed ring longitude angle, in radians.

power_vals The reconstructed power contained in the tau file. If tau is not set, this will be a NoneType variable. Power is normalized to one in free space regions.

raw_tau_threshold_vals The threshold optical depth corresponding to the diffracted optical depth profile.

rho_corr_pole_km_vals Corrections for the ring-intercept point computed by taking into account Saturn’s pole direction.

rho_corr_timing_km_vals Timing offset corrections to the ring-intercept point.

rho_dot_kms_vals The rate of change of the ring-intercept point as a function of time. That is, drho/dt.

rho_km_vals The ring intercept point, in kilometers.

t_oet_spm_vals Observed event time, the time the signal is recieved on Earth, computed in Seconds Past Midnight.

t_ret_spm_vals Ring event time, the time the signal crosses the rings, computed in Seconds Past Midnight.

t_set_spm_vals Spacecraft Event Time, the time the signal was transmitted from the spacecraft, computed in Seconds Past Midnight.

tau_rho The ring-intercept point corresponding to the values in the tau file. If tau is not set, this will be a NoneType variable. Units are in kilometers.

tau_vals The normalized optical depth contained in the tau file. If tau is not set, this will be a NoneType variable.

`rss_ringoccs.tools.CSV_tools.`**`get_cal`**(*cal*, *verbose=True*)

Purpose: To extract a pandas DataFrame from a given CAL.TAB or CAL.CSV file.

Arguments:

cal (*str*) A string containing the location of the requested cal file. Ex: cal = "/path/to/cal.CSV" File must contain the following columns, in the following order: | spm_vals | f_sky_pred_vals | f_sky_resid_fit_vals | p_free_vals

Keywords:

verbose (*bool*) A Boolean for printing out auxiliary information to the command line.

`rss_ringoccs.tools.CSV_tools.`**`get_dlp`**(*dlp*, *verbose=True*, *use_deprecate=False*)

Purpose: To extract a pandas DataFrame from a given DLP.TAB or DLP.CSV file.

Arguments:

dlp (*str*) A string containing the location of the requested dlp file. Ex: dlp = "/path/to/dlp.CSV" File must contain the following columns, in the following order: | rho_km_vals | rho_corr_pole_km_vals | rho_corr_timing_km_vals | phi_rl_deg_vals | phi_ora_deg_vals | p_norm_vals | raw_tau_vals | phase_deg_vals | raw_tau_threshold_vals | t_oet_spm_vals | t_ret_spm_vals | t_set_spm_vals | B_deg_vals

Keywords:

verbose (*bool*) A Boolean for printing out auxiliary information to the command line.

`rss_ringoccs.tools.CSV_tools.`**`get_geo`**(*geo*, *verbose=True*, *use_deprecate=False*)
To extract a pandas DataFrame from a given GEO.TAB or GEO.CSV file.

Arguments

geo (*str*) A string containing the location of the requested geo file. Ex: geo = "/path/to/geo.CSV" File must contain the following columns, in the following order: | t_oet_spm_vals | t_ret_spm_vals | t_set_spm_vals | rho_km_vals | phi_rl_deg_vals | phi_ora_deg_vals | B_deg_vals | D_km_vals | rho_dot_kms_vals | phi_rl_dot_kms_vals | F_km_vals | R_imp_km_vals | rx_km_vals | ry_km_vals | rz_km_vals | vx_kms_vals | vy_kms_vals | vz_kms_vals | obs_spacecract_lat_deg_vals

Keywords

verbose (*bool*) A Boolean for printing out auxiliary information to the command line.

`rss_ringoccs.tools.CSV_tools.`**`get_tau`**(*tau*, *verbose=True*, *use_deprecate=False*)

Purpose: To extract a pandas DataFrame from a given TAU.TAB or TAU.CSV file.

Arguments:

tau (*str*) A string containing the location of the requested tau file. Ex: tau = "/path/to/tau.CSV" File must contain the following columns, in the following order: | rho_km_vals | rho_corr_pole_km_vals | rho_corr_timing_km_vals | phi_rl_deg_vals | phi_ora_deg_vals | p_norm_vals | raw_tau_vals | phase_deg_vals | raw_tau_threshold_vals | t_oet_spm_vals | t_ret_spm_vals | t_set_spm_vals | B_deg_vals

Keywords:

verbose (*bool*) A Boolean for printing out auxiliary information to the command line.

rss_ringoccs.tools.compare module

`rss_ringoccs.tools.compare.`**`compare`**(*NormDiff*, *geo*, *cal*, *dlp*, *tau*, *outfile*, *res=0.75*, *rng='all'*, *wtype='kbmd20'*, *norm=True*, *bfac=True*, *sigma=2e-13*, *verbose=True*, *psitype='Fresnel8'*)

`rss_ringoccs.tools.compare.`**`cringplots`**(*rev*, *geo*, *cal*, *dlp*, *res*, *outfile='outfile.pdf'*, *wtype='kbmd20'*, *psitype='cfresnel4'*)

`rss_ringoccs.tools.compare.`**`galleryplots`**(*rev, geo, cal, dlp, tau=None, res=[1.0], rng='all', wtype='kbmd20', psitype='Fresnel4', norm=True, bfac=True, sigma=2e-13, verbose=True, res_factor=0.75, outfile='galleryplot.pdf', ymin=-0.2, ymax=1.4*)

Purpose: Create a set of plots of the same ring feature at various resolutions as specified by the user.

Arguments:

rev (***str***) The rev number. Ex: rev = "Rev007"

geo (***str***) The location of the geo file. Ex: geo = "/path/to/geo"

:cal (***str***) The location of the cal file. Ex: dlp = "/path/to/cal"

dlp (***str***) The location of the dlp file. Ex: dlp = "/path/to/dlp"

Keywords:

tau (***str***) The location of the tau file. If set, this plots the PDS power, as well as the user reconstructed power. Ex: tau = "/path/to/tau"

res (***list***) The set of requested resolution to process and plot. The values should be floating point values and take the sampling theorem into consideration. Ex: res = [0.5, 0.7, 1.0, 1.2]

rng (***list*** **or** ***str***) The requested range for diffraction correction. Preferred input is rng = [a,b]. Arrays are allowed and the range will be set as:

rng = [MIN(array), MAX(array)]

Finally, certain strings containing a few of the regions of interests within the rings of Saturn are allowed. Permissable strings are:

'all' [1.0, 400000.0]
'cringripples' [77690.0, 77760.0]
'encke' [132900.0, 134200.0]
'enckegap' [132900.0, 134200.0]
'janusepimetheus' [96200.0, 96800.0]
'maxwell' [87410.0, 87610.0]
'maxwellringlet' [87410.0, 87610.0]
'titan' [77870.0, 77930.0]
'titanringlet' [77870.0, 77930.0]
'huygens' [117650.0, 117950.0]
'huygensringlet' [117650.0, 117950.0]

Strings are neither case nor space sensitive. For other planets use rng = [a,b]. Default value is set to 'all' which processes [1, 400000] Values MUST be set in kilometers.

wtype (*str) The requested tapering function for diffraction correction. A string with several allowed inputs:

'rect' Rectangular Window.

'coss' Squared Cosine Window.

'kb20' Kaiser-Bessel 2.0 Window.

'kb25' Kaiser-Bessel 2.5 Window.

'kb35' Kaiser-Bessel 3.5 Window.

'kbmd20' Modified kb20 Window.

'kbmd25' Modified kb25 Window.

The variable is neither case nor space sensitive. Default window is set to 'kb25'. See window_functions submodule for further documentation.

norm (*bool*) A Boolean for determining whether or not the reconstructed complex transmittance is normalize by the window width. This normalization is the complex transmittance that is computed by using free space divided by the complex transmittance that is computed using free space weighted by the selected tapering function. Default is True.

bfac (*bool*) A Boolean for determining whether or not the 'b' factor in the window width computation is used. This is equivalent to setting the Allen Deviation for the spacecraft to a positive value or to zero. If set to False, the Allen Deviation is assumed to be zero. If set to True the Allen Deviation is set to 2e-13, or whichever number you wish to specify in the sigma keyword (See below). Default is True.

sigma (*float*) The Allen deviation for the spacecraft. If the bfac keyword (See above) is set to False, this is ignored. If bfac is set to True, and sigma is NOT specified, then sigma=2e-13 will be used, which is the Allen deviation for Cassini with 1 second integration time. For spacecraft other than Cassini, you should provide the Allen deviation yourself. Default is sigma=2e-13

psitype (*str*) A string for determining what approximation to the geometrical 'psi' function is used. Several strings are allowed:

'full' No Approximation is applied.

'MTR2' Second Order Series from MTR86.

'MTR3' Third Order Series from MTR86.

'MTR4' Fourth Order Series from MTR86.

'Fresnel' Standard Fresnel approximation.

The variable is neither case nor space sensitive. Default is set to 'full'.

verbose (*bool*) A Boolean for determining if various pieces of information are printed to the screen or not. Default is False.

outfile (*str*) Path to the output folder and the name of the pdf that is to be created. Ex: outfile = "/path/to/outfile.pdf"

res_factor (*float*) Floating point number used as a scale factor for the resolution for the sake of consistency with the PDS results. The definition of resolution adopted in the PDS and the definition specified in MTR86 differs by a scale of about 0.75. To skip this, set res_factor = 1.0.

ymin (*float*) The minimum y value to be plotted. Ex: ymin = -0.2

ymax (*float*) The maximum y value to be plotted. Ex: ymax = 1.5

rss_ringoccs.tools.dtau_miescatt_partsize_grid module

rss_ringoccs.tools.et_to_spm module

Purpose Convert ephemeris time to seconds past midnight (SPM).

Dependencies

1. numpy
2. spiceypy

`rss_ringoccs.tools.et_to_spm.`**`et_to_spm`**(*et_vals*, *kernels=None*, *ref_doy=None*)

Convert ephemeris time to seconds past midnight.

Arguments

et_vals (*float* or *np.ndarray*) ET seconds past J2000

Keyword Arguments

kernels (*str* or *list*) Path to NAIF kernels

ref_doy (*int*) Reference day of year, typically used for occultations that occur over multiple days

Returns

spm_vals (*float* or *np.ndarray*) Seconds past midnight

rss_ringoccs.tools.history module

Purpose Functions related to recording processing history.

Dependencies

1. sys
2. time
3. os
4. platform
5. pandas
6. numpy

`rss_ringoccs.tools.history.`**`date_to_rev`**(*year*, *doy*, *rss_file='../tables/RSSActivities_all_rings_only.txt'*)

Pull rev number from a table given the year and doy from a RSS activities file with columns for CIMS request, sequence number, year, doy, start earth-received time in HH:MM, end earth-received time in HH:MM

Arguments

year (*int*) Year of occultation

doy (*int*) Day of year of occultation

Returns

rev_number (*str*) 3-digit rev number (e.g. '007')

Note:

1. **Given default 'rss_file' location, this script must be run** one directory from the top-level rss_ringoccs directory

`rss_ringoccs.tools.history.`**`get_rev_info`**(*rsr_inst*)

This returns a dictionary with information related to the ring occultation.

Arguments

rsr_inst (*class*) Instance of RSRReader class

Returns:

rev_info (*dict*) Dictionary with keys: rsr_file, band, year, doy dsn, rev, occ_dir, planetary_occ_flag

`rss_ringoccs.tools.history.`**`rev_to_occ_info`**(*rev*, *sroc_info_file='../tables/list_of_sroc_dir_all_events.txt'*)

Pull occultation direction from a text file given rev.

Arguments

rev (*str*) Revolution/orbit number in 'XXX' format

Keyword Arguments

sroc_info_file (*str*) Path to csv file with columns: rev number, occultation direction, planetary occultation flag

Returns

occ_dir (*str*) Occultation direction (over entire, I&E, occultation) This is not to be confused with profile direction.

Note:

1. **Given default 'sroc_info_file' location, this script must be run** one directory from the top-level rss_ringoccs directory

`rss_ringoccs.tools.history.`**`write_history_dict`**(*input_vars*, *input_kwds*, *source_file*, *add_info=None*)

This creates a dictionary of processing history for an instance.

Arguments:

input_vars (*dict*) Dictionary of all input variables to the instance.

input_kwds (*dict*) Dictionary of all input keywords to the instance.

source_file (*str*) Full path to the script used to run the instance.

Keyword Arguments:

add_info (*dict*) Dictionary of additional info

Returns:

history (*dict*) Dictionary with keys: "User Name", "Host Name", "Run Date", "Python Version", "Operating System", "Source File", "Positional Args", "Keyword Args"

rss_ringoccs.tools.spm_to_et module

Purpose Calculate ephemeris time given a set of SPM values and appropriate kernels.

Dependencies

1. numpy
2. spiceypy
3. sys

`rss_ringoccs.tools.spm_to_et.`**`spm_to_et`**(*spm*, *doy*, *year*, *kernels=None*)

Convert seconds past midnight to ephemeris seconds past J2000.

Arguments

spm (*np.ndarray*) SPM values

doy (*int*) Day of year of observation

year (*int*) Year of observation

Keyword Arguments

kernels (*str*) String specifying the appropriate ephemeris kernel file. If `None`, sets the kernel file to ../../kernels/naif/CASSINI/kernels/lsk/naif0012.tls Default is `None`.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Bibliography

[MTR1986] Essam A. Marouf, G. Leonard Tyler, Paul A. Rosen, “Profiling Saturn’s rings by radio occultation”. Icarus, Volume 68, Issue 1, 1986, Pages 120-166, https://doi.org/10.1016/0019-1035(86)90078-3

[CRSUG2018] Cassini Radio Science User’s Guide.

[GRESH86] Gresh et al. (1986) “An analysis of bending waves in Saturn’s rings using Voyager radio occultation data”. Icarus 68, 481-502.

[NICH14] Philip D. Nicholson, Richard G. French, Colleen A. McGhee-French, Matthew M. Hedman, Essam A. Marouf, Joshua E. Colwell, Katherine Lonergan, Talia Sepersky. “Noncircular features in Saturn’s rings II: The C ring”. Icarus, Volume 241, 2014, Pages 373-396, ISSN 0019-1035. https://doi.org/10.1016/j.icarus.2014.06.024 or http://www.sciencedirect.com/science/article/pii/S0019103514003443

[NAIF] NASA JPL/NAIF spice toolkit https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials/pdf/individual_docs/27_derived_quant.pdf

Python Module Index

r

rss_ringoccs, 1
rss_ringoccs.calibration, 2
rss_ringoccs.calibration.calc_f_sky_recon, 2
rss_ringoccs.calibration.calc_freq_offset, 3
rss_ringoccs.calibration.calc_tau_thresh, 4
rss_ringoccs.calibration.calibration_class, 5
rss_ringoccs.calibration.dlp_class, 7
rss_ringoccs.calibration.freq_offset_fit, 9
rss_ringoccs.calibration.power_normalization, 11
rss_ringoccs.calibration.resample_IQ, 13
rss_ringoccs.diffrec, 14
rss_ringoccs.diffrec.advanced_tools, 15
rss_ringoccs.diffrec.diffraction_correction, 16
rss_ringoccs.diffrec.special_functions, 20
rss_ringoccs.diffrec.window_functions, 25
rss_ringoccs.occgeo, 28
rss_ringoccs.occgeo.calc_occ_geometry, 28
rss_ringoccs.occgeo.occgeo, 37
rss_ringoccs.rsr_reader, 39
rss_ringoccs.rsr_reader.rsr_reader, 40
rss_ringoccs.scatter, 41
rss_ringoccs.scatter.spectro_reader, 41
rss_ringoccs.scatter.spectrogram, 41
rss_ringoccs.tools, 43
rss_ringoccs.tools.compare, 46
rss_ringoccs.tools.CSV_tools, 43
rss_ringoccs.tools.et_to_spm, 48
rss_ringoccs.tools.history, 48
rss_ringoccs.tools.spm_to_et, 50

Index

C

calc_B_deg() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 28
calc_B_eff_deg() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 29
calc_beta() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 29
calc_D_km() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 29
calc_elevation_deg() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 30
calc_F_km() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 29
calc_f_sky_recon() (*in module rss_ringoccs.calibration.calc_f_sky_recon*), 2
calc_freq_offset (*class in rss_ringoccs.calibration.calc_freq_offset*), 3
calc_impact_radius_km() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 30
calc_phi_deg() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 30
calc_poly_order() (*rss_ringoccs.calibration.freq_offset_fit.FreqOffsetFit method*), 9
calc_rho_km() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 31
calc_rho_vec_km() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 31
calc_rip_velocity() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 32
calc_sc_state() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 32
calc_set_et() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 32
calc_tau_thresh (*class in rss_ringoccs.calibration.calc_tau_thresh*), 4
Calibration (*class in rss_ringoccs.calibration.calibration_class*), 5
compare() (*in module rss_ringoccs.tools.compare*), 46
CompareTau (*class in rss_ringoccs.diffrec.advanced_tools*), 15
compute_norm_eq() (*in module rss_ringoccs.diffrec.special_functions*), 20
cont_stft() (*in module rss_ringoccs.scatter.spectrogram*), 42
correct_IQ() (*rss_ringoccs.calibration.calibration_class.Calibration method*), 6
coss() (*in module rss_ringoccs.diffrec.window_functions*), 25
create_dlps() (*rss_ringoccs.calibration.dlp_class.DiffractionLimitedP class method*), 8
create_mask() (*rss_ringoccs.calibration.freq_offset_fit.FreqOffsetFit method*), 9
create_mask() (*rss_ringoccs.calibration.power_normalization.Normal method*), 11
cringplots() (*in module rss_ringoccs.tools.compare*), 46

D

d2psi() (*in module rss_ringoccs.diffrec.special_functions*), 20
date_to_rev() (*in module rss_ringoccs.tools.history*), 48

derlt() (*in module rss_ringoccs.calibration.calc_f_sky_recon*), 3
derpt() (*in module rss_ringoccs.calibration.calc_f_sky_recon*), 3
DiffractionCorrection (*class in rss_ringoccs.diffrec.diffraction_correction*), 16
DiffractionLimitedProfile (*class in rss_ringoccs.calibration.dlp_class*), 7
double_slit_diffraction() (*in module rss_ringoccs.diffrec.special_functions*), 21
downsample_IQ() (*rss_ringoccs.calibration.power_normalization.Normalization method*), 12
dpsi() (*in module rss_ringoccs.diffrec.special_functions*), 21
dpsi_ellipse() (*in module rss_ringoccs.diffrec.special_functions*), 21

E

et_to_spm() (*in module rss_ringoccs.tools.et_to_spm*), 48
extract_list_from_str() (*rss_ringoccs.calibration.power_normalization.Normalization method*), 12
ExtractCSVData (*class in rss_ringoccs.tools.CSV_tools*), 43

F

find_gaps() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 33
find_noise() (*rss_ringoccs.calibration.calc_tau_thresh.calc_tau_thresh method*), 5
FindOptimalResolution (*class in rss_ringoccs.diffrec.advanced_tools*), 15
fit_check() (*rss_ringoccs.calibration.power_normalization.Normalization method*), 12
fit_freespace_power() (*rss_ringoccs.calibration.power_normalization.Normalization method*), 12
fit_freq_offset() (*rss_ringoccs.calibration.freq_offset_fit.FreqOffsetFit method*), 10
FreqOffsetFit (*class in rss_ringoccs.calibration.freq_offset_fit*), 9
fresnel_cos() (*in module rss_ringoccs.diffrec.special_functions*), 22
fresnel_scale() (*in module rss_ringoccs.diffrec.special_functions*), 22
fresnel_sin() (*in module rss_ringoccs.diffrec.special_functions*), 22

G

galleryplots() (*in module rss_ringoccs.tools.compare*), 46
Geometry (*class in rss_ringoccs.occgeo.occgeo*), 37
get_cal() (*in module rss_ringoccs.tools.CSV_tools*), 44
get_chord_ind() (*rss_ringoccs.occgeo.occgeo.Geometry method*), 39
get_dlp() (*in module rss_ringoccs.tools.CSV_tools*), 45
get_f_sky_pred() (*rss_ringoccs.rsr_reader.rsr_reader.RSRReader method*), 41
get_freespace() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 33
get_freespace_km() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 34
get_geo() (*in module rss_ringoccs.tools.CSV_tools*), 45
get_planet_occ_times() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 34
get_pole() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 35
get_profile_dir() (*rss_ringoccs.occgeo.occgeo.Geometry method*), 39
get_rev_info() (*in module rss_ringoccs.tools.history*), 49
get_start_jd() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 35
get_tau() (*in module rss_ringoccs.tools.CSV_tools*), 45

H

hfit_med() (*rss_ringoccs.calibration.power_normalization.Normalizati method*), 13

I

inverse_square_well_diffraction() (*in module rss_ringoccs.diffrec.special_functions*), 23

K

kb20() (*in module rss_ringoccs.diffrec.window_functions*), 25
kb25() (*in module rss_ringoccs.diffrec.window_functions*), 25
kb35() (*in module rss_ringoccs.diffrec.window_functions*), 25
kbal() (*in module rss_ringoccs.diffrec.window_functions*), 26

kbmd20() (*in module rss_ringoccs.diffrec.window_functions*), 26
kbmd25() (*in module rss_ringoccs.diffrec.window_functions*), 26
kbmd35() (*in module rss_ringoccs.diffrec.window_functions*), 26
kbmdal() (*in module rss_ringoccs.diffrec.window_functions*), 27

M

ModelFromGEO (*class in rss_ringoccs.diffrec.advanced_tools*), 15

N

Normalization (*class in rss_ringoccs.calibration.power_normalization*), 11
normalize() (*in module rss_ringoccs.diffrec.window_functions*), 27

P

plot_power_profile() (*rss_ringoccs.calibration.power_normalization.Normalization method*), 13
plotFORFit() (*rss_ringoccs.calibration.freq_offset_fit.FreqOffsetFit method*), 10
pre_resample() (*in module rss_ringoccs.calibration.resample_IQ*), 13
psi() (*in module rss_ringoccs.diffrec.special_functions*), 23

R

rad_converge() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 35
read_spectro() (*in module rss_ringoccs.scatter.spectro_reader*), 41
rect() (*in module rss_ringoccs.diffrec.window_functions*), 27
remove_blocked() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 36
resample_IQ() (*in module rss_ringoccs.calibration.resample_IQ*), 14
resolution_inverse() (*in module rss_ringoccs.diffrec.special_functions*), 23
rev_to_occ_info() (*in module rss_ringoccs.tools.history*), 49
RSRReader (*class in rss_ringoccs.rsr_reader.rsr_reader*), 40
rss_ringoccs (*module*), 1
rss_ringoccs.calibration (*module*), 2
rss_ringoccs.calibration.calc_f_sky_recon (*module*), 2
rss_ringoccs.calibration.calc_freq_offset (*module*), 3
rss_ringoccs.calibration.calc_tau_thresh (*module*), 4
rss_ringoccs.calibration.calibration_class (*module*), 5
rss_ringoccs.calibration.dlp_class (*module*), 7
rss_ringoccs.calibration.freq_offset_fit (*module*), 9
rss_ringoccs.calibration.power_normalization (*module*), 11
rss_ringoccs.calibration.resample_IQ (*module*), 13
rss_ringoccs.diffrec (*module*), 14
rss_ringoccs.diffrec.advanced_tools (*module*), 15
rss_ringoccs.diffrec.diffraction_correction (*module*), 16
rss_ringoccs.diffrec.special_functions (*module*), 20
rss_ringoccs.diffrec.window_functions (*module*), 25
rss_ringoccs.occgeo (*module*), 28
rss_ringoccs.occgeo.calc_occ_geometry (*module*), 28
rss_ringoccs.occgeo.occgeo (*module*), 37
rss_ringoccs.rsr_reader (*module*), 39
rss_ringoccs.rsr_reader.rsr_reader (*module*), 40
rss_ringoccs.scatter (*module*), 41
rss_ringoccs.scatter.spectro_reader (*module*), 41
rss_ringoccs.scatter.spectrogram (*module*), 41
rss_ringoccs.tools (*module*), 43
rss_ringoccs.tools.compare (*module*), 46
rss_ringoccs.tools.CSV_tools (*module*), 43
rss_ringoccs.tools.et_to_spm (*module*), 48
rss_ringoccs.tools.history (*module*), 48
rss_ringoccs.tools.spm_to_et (*module*), 50

S

savitzky_golay() (*in module rss_ringoccs.diffrec.special_functions*), 24
Scatter() (*in module rss_ringoccs.scatter.spectrogram*), 41
single_slit_diffraction() (*in module rss_ringoccs.diffrec.special_functions*), 24
spectro() (*in module rss_ringoccs.scatter.spectrogram*), 42
split_chord_arr() (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 36

`spm_to_et()` (*in module rss_ringoccs.tools.spm_to_et*), 50
`square_well_diffraction()` (*in module rss_ringoccs.diffrec.special_functions*), 25
`stack_spec()` (*in module rss_ringoccs.scatter.spectrogram*), 43

V

`verify_chord()` (*rss_ringoccs.occgeo.occgeo.Geometry method*), 39

W

`window_width()` (*in module rss_ringoccs.diffrec.window_functions*), 28
`write_history_dict()` (*in module rss_ringoccs.tools.history*), 49

X

`xform_j2k_to_pcf()` (*in module rss_ringoccs.occgeo.calc_occ_geometry*), 37