SOFIA-EXES Exposure Time Calculator Manual

Benjamin Cain

October 1, 2013

The SOFIA-EXES Exposure Time Calculator (ETC) is a web application that uses a combination of HTML/PHP pages and Python scripts to collect user input and ultimately calculate the necessary amount of exposure time for a given set of observational conditions. Since the particular instrument setup will change the options available downstream, a multi-step procedure is necessary. The basic overview of the process is:

1. Collect the central observing wavelength/wavenumber and the instrument mode.
2. Inform the user of the possible observing orders and slit widths. Collect these parameters, as well as the source geometry (point-like or extended), source flux (or surface brightness, for extended objects) and the desired signal to noise ratio (S/N) at the central wavelength. User also selects an observing altitude for the aircraft.
3. Calculate the integration time, the estimated total clock time due to nodding/mapping overheads, and the S/N over the observable wavelength range in the mode selected. A summary of the setup and plots of S/N vs. wavelength and atmospheric transmission vs. wavelength are also provided.

In the subsequent sections we will outline the specific steps in each of these calculations and the inputs and assumptions that go into the exposure time calculation.

1 `etc.html`

The first two steps in the ETC, which take place on the `etc.html` page (symbolically linked from `index.html` at `http://irastro.physics.ucdavis.edu/exes/etc`), are basic data collection. Step 1 is to collect from the user the central or target wavelength, in microns, for the observation via an HTML text entry form. The target wavelength can alternatively be specified by entering a target wavenumber ($1/\lambda$, in cm^{-1}) into the same form as the wavelength would be entered, and the acceptable ranges for both quantities are listed for the user's reference (4.5-28.5 μm or 350-2220 cm^{-1}.

The user is asked to enter the target wavelength in the rest frame of the source to allow for Doppler-shifted observations of moving sources. There is an optional checkbox that the user can select if the source is moving, and there is a second HTML text-entry form for the source velocity, in km/s. Negative values will be interpreted as a source moving towards the observer.

In Step 2, the user is also asked to select one of the four instrument modes to be used: Cross-dispersed High-Medium, Cross-dispersed High-Low, Single-order Long Slit Medium, or Single-order Long Slit Low. By default, High-Medium is selected. A radio button is used for entry to ensure that one and only one mode is selected.

Once the target wavelength (or wavenumber), mode, and any of the optional Doppler information have been entered, the user must click the submit button at the bottom of the page to proceed to the subsequent parts of the ETC. Doing so passes the following data on to `etc1.php`:

- Target rest-frame wavelength/wavenumber [text]
- Doppler shift flag [checkbox]
- Doppler velocity [text, optional]
- Instrument Mode [radio button]

2 `etc1.php`

From this point on, the web interface relies heavily on PHP scripting within the HTML pages. The page `etc1.php` (and `etc2.php` thereafter) receives the user input from the previous page via the PHP `POST` method, meaning that the data from one page to the next is passed within the browser, rather as a query string in the URL.

Within `etc1.php`, the first action is to error check the input. If the Doppler shift checkbox is unchecked or if the velocity is blank then the source is assumed to have no velocity relative to the observer. Otherwise, the input value is taken. Non-numerical inputs to either velocity or wavelength/wavenumber will evaluate to zero for that variable.

The maximum observable wavelength is 28.5 microns, and the minimum is 4.5 microns. These limits also set the wavenumber minimum and maximium, respectively, by their inverses. If the user entry for rest-frame wavelength/wavenumber is between 350.9 and 2222.2, we assume that the user intended wavenumber input in units of cm^{-1} and convert that input into wavelength in microns. Otherwise, we assume the user intended wavelength input in microns. We then use the Doppler shift information to calculate the observed wavelength corresponding to the user's target rest-frame wavelength. Note that the Doppler shifting assumes non-relativistic velocities entered in km/s.

We then compare the observed wavelength to the observable 4.5 – 28.5 micron range. Should the observed wavelength fall outside that range, the user is warned and the observed wavelength is reset to the nearest limit (i.e., $\lambda < 4.5\ \mu\text{m} \rightarrow 4.5\ \mu\text{m}$; $\lambda > 28.5\ \mu\text{m} \rightarrow 28.5\ \mu\text{m}$) and the calculation can proceed from there.

The observed wavelength, the instrument mode, and an output flag are then passed to the Python script `run_etc1.py` via the PHP `exec` method, and the output is captured. Because the full amount of output from the Python script occasionally exceeds the maximum size for a single PHP array object, `run_etc1.py` is called twice with two different output flags to capture all the required output. The specific workings of the Python script, `run_etc1.py`, are outlined in another section below. For this section, we indicate the inputs and the outputs of the script as an overview.

For its first call, `run_etc1.py` returns the number of available observing orders (N_{ord}) at the given wavelength, the image quality (IQ, FWHM) in both arcseconds and pixels, and arrays of the following quantities:

- The available orders at the target wavelength.
- The grating angle (in degrees) for each order.
- The resolving power for each order, assuming the smallest usable slit width.
- The minimum wavelength (in microns) observable for each order.
- The maximum wavelength (in microns) observable for each order.
- The slit length in arcseconds at each order.
- The slit length in pixels at each order.
- A flag noting whether the slit is too short for on-slit point source nodding.

Each of these arrays has N_{ord} elements.

For the second call, `run_etc1.py` returns the number of available slit widths (N_{slit}), and two arrays including:

- The available slit widths in arcseconds.
- The available slit widths in pixels.

These two arrays have N_{slit} elements.

The output strings from both calls of `run_etc1.py` are split into PHP variables and arrays. The user is reminded of their input from Steps 1 & 2 of the ETC and are informed of the expected SOFIA IQ. In the case that a Doppler shift has been entered, the user is informed of both the rest-frame and observed wavelengths. The user is also presented with a link that allows them to restart the ETC if they so choose.

In Step 3 of the ETC the user is asked to select one of the available orders. A table is presented which includes the orders available and (for each order): the grating angle (in degrees), the resolving power with the default (smallest) slit width, the minimum and maximum wavelengths (in microns) as well as the minimum and maximum wavenumbers (in cm^{-1}), the slit length (in arcsec) and a note as to whether on-slit nodding is available for point source observations for that order. The user selects one of the order options using a radio button, which defaults to the lowest order. The page notes that the resolving power for high-resolution modes is entirely dependent on selected slit width, and not dependent on the grating angle.

In Step 4 of the ETC the user is asked to select a slit width. Again the user is presented with a table including the available slit widths from `run_etc1.py`, in arcseconds, and for each slit width: the extended source aperture (the product of the slit width and the IQ, in square arcseconds), and the resolving power for each combination of order and slit width. Again, the user selects an option using a radio button, which by default is set to the smallest slit width. The page notes that since most EXES observations will be background limited, selecting a larger slit width may increase the the noise disproportionately to any increase in signal counts. Users should be sure to compare the IQ with the slit width.

In Step 5 of the ETC, the user is asked to select a representative set of atmospheric conditions for the observation. This is done by selecting an altitude for the SOFIA aircraft. There are four options for the atmosphere: 39,000 feet altitude, 41,000 feet altitude, 43,000 feet altitude, and no atmosphere. For each altitude, a 45° elevation angle for the telescope is assumed. The user selects one of these atmospheres with a radio button, which defaults to the 39,000 feet altitude atmosphere.

In Step 6 of the ETC the user is asked to select a source geometry, quantify the brightness of the source, and to enter the desired signal to noise ratio for the observation. As noted on the page, the signal to noise ratio requested is per resolution element. The options for geometry are 1) point source, which requires that the user enter the source flux in Janskys, and 2) extended source, which requires that the user enter the source surface brightness, in Jansky per square arcsecond. The geometry is selected by a radio button and the flux (or surface brightness) is entered via text forms. Also included in Step 6 is an optional input for a detector shift which only affects the high-resolution cross-dispersed modes. The detector shift (in pixels) alters the relative position of the wavelength centers with respect to the blaze maxima.

In Step 7, the final step, the user is asked to select a pointing mode. The two options are nodding mode and mapping mode. The nodding will either be on-slit or off-slit as determined by the relative size of the IQ and the slit width. For mapping mode, the user is asked to enter the number of desired mapping points. As is noted on the page, the mapping will be done in stripes oriented perpendicular to the slit, and that the slit orientation will be set by the SOFIA flight plan.

Once the forms for each of the steps on `etc1.php` are completed, the user clicks a submit button at the bottom of the page to continue on to `etc2.php` and complete the exposure time calculation. Upon submission, the data listed below are passed to `etc2.php`. Note that, as before, any text entries which are left blank or have non-numerical entries will evaluate to zero:

- The index of the order selected, 0,...,($N_{ord} - 1$) [radio button]
- The index of the slit width selected, 0,...,($N_{slit} - 1$) [radio button]
- The source geometry, point or extended source [radio button]
- The source flux [text, optional for extended sources]
- The source surface brightness [text, optional for point sources]
- The signal to noise ratio [text]
- The atmosphere option selected, [radio button]
- The pointing mode [radio button]
- The number of mapping points [text, optional for nodding mode]
- The detector shift, in pixels [text]

In addition to these direct user inputs from `etc1.php`, the form also passes along these data from `etc.html`:

- Rest-frame and observed wavelength (in microns)
- The Doppler velocity (in km/s)
- The IQ (FWHM, in both pixels and arcseconds)

as well as several strings with pipe-separated values (i.e., 'entry1|entry2|...') to include output by `run_etc1.py`. First, those which would be indexed by the order selected:

- The available orders
- The grating angles (in degrees)
- The product of the resolving power and the default slit width
- The minimum and maximum wavelengths (one string each)
- The slit lengths in pixels and arcseconds (one string each)
- The on-slit nodding availability flag

And second, those which would be indexed by the slit width selected:

- The slit width in pixels and arcseconds (one string each)

3 `etc2.php`

The final output of the exposure time calculator is presented on the `etc2.php` page. Each of the pipe delimited array strings is split into a PHP array variable using the PHP `explode` method. Other data processing is done to unpack the rest of the data passed from `etc1.php` via the `POST` method (e.g., the resolving power is calculated from the value of the R/default slit width product determined by the order selected and the actual slit width selected). A link to reset the ETC is again placed at the top of the page.

The user is then presented with three blocks of output information. The first is a comprehensive instrument setup summary. This includes:

- The instrument mode
- The observed wavelength (in microns)
- The rest-frame wavelength (in microns)
- The observed wavenumber (in cm^{-1})
- The rest-frame wavenumber (in cm^{-1})
- The Doppler velocity (in km/s)
- The observing order
- the resolving power for the selected order and slit width
- The minima and maxima for both the wavelength (in microns) and wavenumber (in cm^{-1})
- The slit length (in arcseconds)
- The slit width (in arcseconds)
- The extended object aperture (in square arcseconds, but only if the source type is an extended object)
- The pointing mode (nodding or mapping, and whether nodding is on- or off-slit)
- The SOFIA image quality (FWHM, in arcseconds)
- The detector shift (in pixels)

The second block is an observation summary. This includes:

- The source type (point source or extended object)

- The source flux (in Jy, if a point source) or the surface brightness (in Jy/arcsec^2, if an extended object)
- The selected atmosphere
- The desired signal to noise ratio at the target wavelength.

These first two blocks contain the full input to the exposure time calculation, as well as some corollary properties of the instrument and the observation based on the instrument setup. `etc2.php` at this point calls the checks to ensure that the flux/surface brightness are positive values and then calls the Python script `run_etc2.py` using the PHP `exec` method to get the output of the ETC. As before, the details of this Python script are described in a subsequent section. The script returns

- The total integration time (in seconds)
- The estimated clock time, including overhead and nodding (in seconds)
- The source count rate (in e^-/s)
- The background count rate (in e^-/s)
- The total read noise counts (in e^-/readout)
- The number of readouts required to avoid nonlinearity
- The path to an image file (in PNG format) plotting the S/N versus wavelength/wavenumber over the observed range (upper subplot) and the atmospheric transmission for each of the four aircraft altitudes as a function of wavelength/wavenumber over the observed range (lower subplot)
- The path to an ASCII text file which lists the input data for the S/N plot and the atmospheric transmission plot (though the text file includes only the selected atmosphere.

The third output block lists the first six of these outputs for the user. To the right of the three data blocks, the S/N and atmospheric transmission plots are displayed, as is a link to the text file with the plot data. If the observed wavelength is longer than 19.0 microns, the user is alerted to the possibility of gaps in the wavelength coverage as the individual orders become too large to fit on the detector, and the user is instructed to use the pixel shift parameter to adjust the wavelength coverage to ensure that wavelength regions of interest other than the target wavelength are observable and do not fall into any of these gaps.

4 Python Overview

The PHP pages described above each call a Python script, `run_etc1.py` and `run_etc2.py` respectively. These scripts interface with a package written for this ETC - `etc_utils.py`. In addition, the `sys`, `os`, `numpy`, `matplotlib`, and `random` packages are required. The `etc_utils` package also requires two additional packages to run: `constants.py`, which outlines a set of basic physical constants for use in unit conversions and calculations, and `exespars.py`, which defines a set of EXES-specific parameters which can be referenced and modified as needed without altering the rest of the ETC code. The next two sections here describe the two `run_etc*.py` scripts.

5 `run_etc1.py`

The major calculations for the ETC are done in Python. `etc1.php` calls `run_etc1.py` with three input arguments (calls it twice, in fact). The arguments are:

- Wavelength (in microns)
- The mode - a string, containing “`himed`”, “`hilo`”, “`med`”, or “`lo`”.
- A flag to determine what kind of output is returned, either 1 or 2.

In actuality, if the mode string does not match any of the first three, the script defaults to low resolution single-slit mode. Similarly, though the flag is intended to be an integer of either 1 or 2, the script only considers input as “1 or not 1”.

The script passes the wavelength and the mode into the `setup_options` method in the `etc_utils` package, which returns a list with five entries. These are:

- An $N \times 7$ array of setup options, where N is the number of orders available at this wavelength. This array includes the orders, the grating angle for each order, the minimum and maximum wavelengths for each order, the slit length for each order in arcseconds and in pixels, and a flag as to whether on-slit nodding is available for this order.
- An $M \times 2$ array of slit widths in arcseconds and in pixels, where M is the number of available slits at this wavelength.
- An $N \times M$ array of spectral resolutions, with N and M defined as above (though the resolution only depends on angle for the low-resolution modes.
- The image quality (FWHM) in arcseconds.
- The image quality (FWHM) in pixels.

There are two output flag options for `run_etc1.py`. If the flag into `run_etc1.py` is set to 1, then the script prints out to standard output

- The number of orders
- The image FWHM in arcseconds
- The image FWHM in pixels

and for each order the script loops through and prints

- The order number
- The grating angle
- The resolving power assuming the default slit width, which is defined to be the smallest of the available slit widths.
- The minimum and maximum wavelengths observable
- The slit length in arcseconds and in pixels
- The flag for nodding (1 if on-slit is ok, 0 if nodding must be off slit)

These printed outputs are captured by the PHP `exec` command and parsed by `etc1.php`.

If the output flag for `run_etc1.py` is not set to 1, the script will print the following to standard output:

- The number of available slit widths
- The width of each slit in arcseconds and in pixels

Again, these output are captured by the PHP `exec` command and parsed by `etc1.php`. By calling the script twice, once with each output option, the full slate of setup options can be presented to the user and a well-specified mode can be chosen.

6 `run_etc2.py`

After the user selects a configuration and characterizes the source properties and atmospheric conditions in `etc1.php`, the `etc2.php` page will call `run_etc2.py` to complete the exposure time calculation. This script requires 16 input parameters:

- Target S/N at the target wavelength
- Flux (in Jy) **or** surface brightness (in Jy/arcsec^2)
- Target wavelength

- Minimum wavelength
- Maximum wavelength
- Slit length (in pixels)
- Slit width (in pixels)
- Image quality (FWHM, in pixels)
- Observing mode (himed, hilo, med, or lo)
- Spectral resolution
- Flag for an extended source (1 if yes, 0 if no)
- Flag for nodding (1 if off-slit, 0 if on-slit)
- Atmosphere selection (1-3,5, corresponding to aircraft altitudes of 39,000 feet, 41,000 feet, 43,000 feet, and no atmosphere, in that order. There is a deprecated option 4 for 45,000 feet as well.)
- Number of mapping points (-1 if nodding)
- Number of samples per FWHM for mapping (-1 if nodding)

The one new parameter not yet described is the number of samples. In mapping mode, the default assumption is that the map will be made in stripes, stepping the slit by half of the slit width perpendicular to the long edge of the slit. This means that each resolution element will be spatially sampled multiple times. This sampling rate, which is the ratio of the image FWHM to the selected slit width, rounded to the nearest integer, will affect the S/N and thus the exposure time in mapping mode.

Once these inputs are parsed, `run_etc2.py` calculates a set of wavelengths to plot based on the given spectral resolution and calculates the high resolution blaze efficiency for each of these wavelengths and for the target wavelength. This blaze efficiency is set to unity for the low resolution modes, and is calculated for the high resolution modes by the `blaze_efficiency` method of the `etc_utils` package.

The target wavelength, the desired S/N, the source flux/surface brightness, the slit dimensions (in pixels), the image quality (also in pixels), the spectral resolution, the instrument mode, the selected atmosphere, the extended source and nodding flags, the number of mapping points, the number of samples per FWHM, and the blaze efficiency at the target wavelength are all input into the `calc_exp_time` method of the `etc_utils` package to calculate the required exposure time. This method calculates the atmospheric transmission internally for the target wavelength. The method will return a list of five elements, those being:

- The required exposure time
- The required clock time, including nodding overheads and background exposures
- The total source count rate per resolution element (in e^-/s)
- The total background count rate per resolution element (in e^-/s)
- The read noise electrons per resolution element per readout (in e^-)
- The number of readouts required to avoid saturation

All five of these are printed out to standard output to be caught by `etc2.php`. Once the exposure time is calculated at the target wavelength, the script calculates the atmospheric transmission at each of the wavelengths in the observed range and adds that input (plus the exposure time replacing desired S/N) to calculate the S/N which will be achieved across the entire observed wavelength range. The atmospheric data exists in both ASCII format (which is not directly used) and the `numpy` binary data format (`.npy` files). The latter is much quicker to load and should there be updates to the atmospheric data these should be converted to the `.npy` format.

The observed wavelengths, the corresponding wavenumbers, the S/N for each wavelength, and the atmospheric transmission for each wavelength are all loaded into an array to be saved to an ASCII text file which will be available for the user via a link on the `etc2.php` page. This text file will be named `exes-etc-snr-atmXXXXXX.txt`, where `XXXXXX` is a randomly generated 6-digit number. This is to prevent file conflicts with multiple users accessing the ETC at approximately the same time. A new file number will be generated each time that `run_etc2.py` is called. The same number will be used for the subsequent plot output, where the plotting will be saved into `exes-etc-snr-atmXXXXXX.png` and displayed to the user. In either case (plot or text), should the number already exist, the existing file will be deleted and overwritten. Both files will be saved into the `plt_txt_out` subdirectory. The generated file names are then printed to standard output so that `etc2.php` will be able to find the correct plot and text file.

The plotted output is a figure with two subplots. The first is a plot of S/N versus both wavelength (in microns, bottom ordinate) and wavenumber (in cm^{-1}, top ordinate) for the exposure time required to reach the targes S/N at the target wavelength. The second subplot is the atmospheric transmission as a function of both wavelength and wavenumber (same units as above) for each of the three aircraft altitudes. This is intended to give the user a sense of how sensitive their observational goals are to the selected atmosphere. Note that only the selected atmosphere is used in the S/N calculation, as well as written to the text file. The full plot is then saved to the generated file. The plot file and the text data file will each only be present on the server for a short period of time (approximately 48 hours) prior to being deleted, so users should download these files immediately if they would like to keep them.

7 Notes on Parameter Files

There are several files which contain essential data for the exposure time calculation. These have been mentioned before, but we add some important notes here for reference. The files are:

- `atm39K.npy`/`atm39K.txt` - 39,000 feet altitude, 45° elevation angle atmospheric tranmission, 4-50μm, in `numpy` binary and ASCII text formats.
- `atm41K.npy`/`atm41K.txt` - 41,000 feet altitude, 45° elevation angle atmospheric tranmission, 4-50μm, in `numpy` binary and ASCII text formats.
- `atm43K.npy`/`atm43K.txt` - 43,000 feet altitude, 45° elevation angle atmospheric tranmission, 4-50μm, in `numpy` binary and ASCII text formats.
- `atm45K.npy`/`atm45K.txt` - 45,000 feet altitude, 45° elevation angle atmospheric tranmission, 4-50μm, in `numpy` binary and ASCII text formats.
- `constants.py` - physical constants and unit conversions.
- `exespars.py` - empirical parameters specific to the EXES instrument.

The last file, `exespars.py`, contains mostly simple numerical parameters and some basic algebraic combinations thereof. The two exceptions to this are `img_poly` and `qe_poly`. These are `numpy` arrays of polynomial coefficients used to calculate the image FWHM and the quantum efficiency, respectively, at arbitrary wavelengths. Directly above each polynomial coefficient array definition is the Python code used to generate the array, for reference.

8 etc_utils.py

In this section we detail the specifics of the code which is used to perform each of the calculations for the ETC. Since this file contains a series of Python methods, we will detail each in turn, in the order they appear in the file. Additional detailed comments can be found within the code itself. Using any of the methods in this package will require the `constants`, `numpy`, and `exespars` packages to be importable. The calling syntax and the default keyword values for each method are included with the descriptions below.

```
geom_slit_efficiency( slitwidth, slitlength )
```

This method takes the slit width and length in units of the point source image FWHM and calculates the fraction of the photons incident on that slit which will pass through. In the limit that both parameters are much larger than unity, this method's output will approach unity.

`b_nu_cgs( wavelen, temp )`

This method takes a wavelength (in centimeters) and a temperature (in Kelvin) and returns the Planck thermal spectral radiance in CGS units (erg/s/cm^2/Hz/Sr).

`best_order( lambdas, nlines, blaze, [order1=1, norders=30, minang=2.0, maxang=75.0])`

This method takes in a wavelength (or an array of wavelengths) in microns, the number of lines per millimeter for the grating, the blaze angle (in degrees) for the grating. The method returns an $N \times 3$ `numpy` array, where N is the number of wavelengths input. This array contains the best order (0th index), the grating angle for that order (1st index), and an estimated grating efficiency for that angle (2nd index). These outputs are calculated by finding the angles which place the target wavelengths as close as possible to a blaze maximum and assuming that each order's efficiency curve is a Gaussian with FWHM equal to the order spacing.

The optional keyword inputs are:

- `order1` - the minimum order to consider, defaulting to 1
- `norders` - the number of orders to consider, defaulting to 30
- `minang` - the minimum allowable grating angle (in degrees), defaulting to 2.0
- `maxang` - the maximum allowable grating angle (in degrees), defaulting to 75.0

Note: this method is not currently used in the ETC.

`atmos_trans( wavelen, atmos, [filename=‘’] )`

This method calculates the atmospheric transmission at the wavelength (or wavelengths, if `wavelen` is a `numpy` array) given an atmosphere selection (`atmos`). For `atmos`=1,...,4, the atmospheric data from `atm39K.npy`,... `atm45K.npy` are used to determine the transmission(s), otherwise unity is returned for each wavelength. The values of `wavelen` are assumed to be in microns.

There is an optional keyword input `filename`, which defaults to an empty string. If this keyword is not an empty string, it is assumed to be the appropriate path to a binary `.npy` file which stores an alternative set of wavelength data in $N \times 2$ array format. Setting this keyword supersedes the value of `atmos`.

The values returned by this method for each method are interpolated using the `numpy.interp` method.

setup_options(wavelen, [low=True, med=False, high=False])

This method takes in a target wavelength (assumed to be in microns) and returns a list with five entries. These are:

- An $N \times 7$ array of setup options, where N is the number of orders available at this wavelength. This array includes the orders, the grating angle for each order, the minimum and maximum wavelengths for each order, the slit length for each order in arcseconds and in pixels, and a flag as to whether on-slit nodding is available for this order.
- An $M \times 2$ array of slit widths in arcseconds and in pixels, where M is the number of available slits at this wavelength.
- An $N \times M$ array of spectral resolutions, with N and M defined as above (though the resolution only depends on angle for the low-resolution modes).
- The image quality (FWHM) in arcseconds.
- The image quality (FWHM) in pixels.

There are three optional boolean keyword inputs to characterize the instrument mode, which default to low resolution mode. Furthermore, one and only one of `low` and `med` must be `True` and the other `False`. If this is not the case, a warning message will be printed and the variables `low=True` and `med=False` will be used.

For a range of potential orders (2 to 14 for the medium resolution grating, `med=True`; -4 to -1 for the medium resolution grating, `low=True`) the grating angle necessary is calculated for each order. The orders which fall into the acceptable range of angles (35 to 65 degrees for the medium resolution grating and -30 to -7 degrees for the low resolution grating) are the ones returned, along with the associated properties in the first array. The available slit widths are set by the target wavelength. The nodding flag is set to unity (i.e., on-slit nodding is possible) if the image quality (FWHM) is less than a set factor of `exespars.nod_factor` smaller than the slit length. The image quality is determined from a polynomial fit to empirical data. For each mode setting, the order and the slit width determine the spectral resolution options.

`calc_exp_time( wavelen, snr, flux, slitlen, slitwid, imgsize, rnum, [low=True, med=False, high=False, ext_src=False, atmos=1, atm_trans=-1, nodoff=False, nmap=-1, nsamp=-1, blaze_eff=1.0] )`

This method takes as positional arguments the target wavelength (in microns), the target S/N at that wavelength, the flux (in Jy) or the surface brightness (in Jy/arcsec^2), the slit length (in pixels), the slit width (in pixels), the image FWHM (in pixels), and the spectral resolution. The optional keyword arguments are the three mode booleans (`low`, `med`, and `high`, which default to the low resolution single slit mode), a boolean `ext_src` for extended source observations (defaulting to `False`), the `atmos` variable to select the atmosphere (defaulting to unity, which selects the 39,000 foot atmosphere), `atm_trans` which defines the atmospheric transmission at the target wavelength and defaults to -1 indicating that the method should calculate the transmission using the atmosphere selected with `atmos`, a boolean indicating whether to nod off-slit (defaulting to False), the number of mapping points (defaulting to -1, indicating single object observation), the number of samples per FWHM in image mode (defaulting to -1), and the blaze efficiency (defaulting to unity).

Returned is a six-element Python list containing

- The exposure time to reach the target S/N at the target wavelength, in seconds
- The clock time
- The source count rate for this flux/surface brightness and wavelength, in e^-/s
- The background count rate at this wavelength, in e^-/s
- The read noise counts, in in e^-/readout
- The number of readout frames that will be necessary to prevent saturation

The method uses the various physical parameters from `exespars.py` to calculate the optical efficiency (folding in the input blaze efficiency as well if using a high resolution mode). This allows a background count rate per pixel to be calculated, as well as a total source count rate (including the geometric slit efficiency for point sources and the resolution element size for extended objects). The exposure time is calculated by finding the time t that solves:

$$S = \frac{C_{src}t}{\sqrt{C_{src}t + 2C_{bkg}t + 2N_{frame}\sigma^2_{read}}},$$

where S is the target S/N, C_{src} and C_{bkg} are the source and background count rates, N_{frame} is the number of frames, and σ_{read} is the standard deviation of counts per readout.

Both t and N_{read} are determined iteratively. The total number of background counts per pixel is used to estimate a time to full well (the maximum number of counts in a full well is set as a parameter in `exespars.py`). The number of frames is increased to ensure that the number of counts is below the full well depth for each readout. Since most observations are background limited, we neglect the source counts in the full well time calculation, though in principle these should be included as well. As an additional consideration, for on-slit nodding we further require that the number of frames be even to allow for proper background subtraction.

For mapping mode, the exposure time is calculated by considering a target S/N which is a factor of $\sqrt{N_{samp}}$ smaller than the true target to account for the additional samples per resolution element gained from the mapping scheme. The clock time is calculated by adding a number of overhead/background frames (defined in `exespars.py`) to the required number of target exposures and then multiplying the sum by the calculated exposure time per frame. For nodding, on-slit or off, the clock time is determined by an efficiency factor (also defined in `exespars.py`). Note that on- and off-slit nodding will have different efficiencies.

`calc_snr ( wavelen, exptime, flux, slitlen, slitwid, imgsize, rnum, [low=True, med=False, high=False, ext_src=False, atmos=1, atm_trans=-1, nframe=-11, blaze_eff=1.0] )`

Apart from the replacement of `snr` with `exptime` (the exposure time in seconds) the positional arguments for `calc_snr` are the same as for `calc_exp_time`. The keywords are also the same, except `nodoff`, `nmap`, and `nsamp` have been removed and replaced by `nframe`, the number of frames for this observation. This method uses the same S/N formula and method as `calc_exp_time` to calculate the S/N for a given exposure time. This method is primarily intended for using the required exposure time to reach a particular S/N at one wavelength to infer the S/N at nearby wavelengths. The output is a four element Python list containing

- The S/N at the given wavelength
- The source count rate for this flux/surface brightness and wavelength, in e^-/s
- The background count rate at this wavelength, in e^-/s
- The read noise counts, in in e^-/readout

Notable about this function is that `wavelen` can be passed as either a single wavelength or a `numpy` array. If passed as an array, `atm_trans` must either be a single value or must be the same length as `wavelen` to avoid an error.

`blaze_efficiency (wl_min, wl_max, rpower, [shift=0.0] )`

Given a minimum and maximum wavelength and a spectral resolution (along with an optional detector shift, in pixels) this method returns two arrays. The first is an array of wavelength from the given minimum to the maximum with spacing defined by the spectral resolution. The second is an array with the same number of elements containing the blaze efficiency of at each wavelength. The pixel shift (which defaults to 0.0) alters where the blaze maxima fall on the detector, which is important particularly at long wavelengths when the detector is smaller than the blaze order and there are gaps in the wavelength coverage. This blaze efficiency only factors into the calculations for high-resolution cross-dispersed modes.