1 Frequently Desired Operations

- Find fluxes in a list of transmission bands of a large number of SEDs. The wavelengths of SEDS may be at different grids. The use case is to find the fluxes/ mags of every object of a certain class (galaxy/AGN/SN) on a focal plane. Since these might be at different redshifts, the wavelength grid could be different), unless we store that at the top of the atmosphere on the same wavelength grid. So we are looking at

  ```
          # Fix all SEDs to the wavelength grid of the bandpass system
  bandpassdict.flux([sed0, sed1, sed2, sed3], bandpassindices=['u', 'g', 'r', 'i'])
  ```

  where the length of bandpassindices matches the length of the bandpass string. We should also allow for the case, where the flux for each sed is evaluated in multiple bands (for example in a SDSS like case):

  ```
          # Fix all SEDs to the wavelength grid of the bandpass system
  bandpassdict.flux([sed0, sed1, sed2, sed3], bandpassindices=[['u', 'g', 'r', 'i'],
                                                               ['u', 'g', 'r', 'i', 'z'],

                                                               ['u', 'g', 'r', 'i', 'z'],

                                                               ['u', 'g', 'r', 'i', 'z']]
  ```

- Transients: The focal plane will have a number of transients of each class. When doing a set of instance catalogs at different times at the same position, rather than working instance catalog by instance catlog, it might be useful to query all the objects in the field of view and obtain the flux values for the SEDS at different times and fill the instance catalogs simultaneously. While this could be limited by memory issues, this has the potential of speeding up the calculations if the transient SEDs are calculated through a causal calculation.

  ```
              # Fix all SEDS to the wavelength grid of the SED
  for Transientsed in sedList:
      TransientSed.setModelParams.fromCatalog()
      Transientsed.flux([t0, t1, t2, t3], bands=['u', 'g', 'r', 'i', 'z'])


  Class TransientSed(object):
      def __init__(self, modelParams):
          ...
      def specificFlux(self, times):
          # sed = f_\lambda (dE/d\lambda) at the top of the atmosphere
  ```

```
        # observer frame time
        return list of sed(times)
    def flux(self, bandpassDict, time, bandpassInd):
        return bandpassdict.flux(self.specificFlux(times) , bandpassdict,
                                 bandpassind)
```

- Noisy spectrum
- Zero points and other magsystems

2 Glossary: Definitions, Conventions and Equations

2.1 Transmission Functions

$T(\lambda)$ The transmission functions describing bandpasses. There are two conventions in this. We should try to support both: `https://github.com/sncosmo/sncosmo/issues/111`

2.2 Specific Flux:

$f_\lambda = \frac{dE}{d\lambda}$ or $f_\nu = \frac{dE}{d\nu}$, we can write f_ν as a function of wavelength: $f_\nu(\lambda) = f_\nu(\nu = c/\lambda)c\frac{d\lambda}{\lambda^2}$

2.3 Flux

As it is measured by our CCDs, the important quantity is number of photons per unit area per unit time. The units are thus $cm^{-2}s^{-1}$. A second unit of interest is the ratio of this quantity for our source to that of a standard source as defined by the Magnitude System.

$$\mathrm{Flux} = \int \frac{\mathrm{dE}(\lambda)}{\mathrm{d}\lambda}\mathrm{T}(\lambda)\frac{\lambda\mathrm{d}\lambda}{\mathrm{hc}} \tag{1}$$

We can also write this as

$$\mathrm{Flux} = \int \frac{\mathrm{dE}(\lambda)}{\mathrm{d}\nu}\mathrm{T}(\lambda)\frac{\mathrm{d}\lambda}{\mathrm{h}\lambda} \tag{2}$$

2.4 Magnitude System

2.5 Magnitudes

3 Data Structures and PseudoCode

3.1 Transmission Functions: BandPass

- The main class of objects should be a Bandpass class, and a Bandpassdict class. It is expected that the bandpassdict class will encapsulate all the bands of a survey and will be the most commonly used object. The user should try to obtain the appropriate bandpassdict and use it.
- use a registry to be able to refer to the bandpasses and bandpassdict-sthrough strings (connected to next point)
- Use builtins for well known transmission functions so that instantiation is unnecessary, along with the ability to add transmission functions through an instantiation of bandpass/bandpassdict objects
- Not really throught about: How to extend bandpassdict to heterogeneous transmission function models?

Possible Attributes of BandPass

- Names: unique string identifying bandpass in registry ? This will allow these objects to be loaded via builtins
- bestwavelen : large wavelen range with fine grid
- wavelen: Wavelength grid, used for flux/mag calculation, usually smaller than bestwavelen
- minWavelen, maxWavelen : To be used to check if flux can be calculated at all for a Sed
- Sb: Transmission probabilities
- bestSB: Transmission probabilities over bestwavelen
- units as metadata or maybe our current documentation is good enough
- phi : Normalized transmission for multiplying fnu
- flux of standard SED in band: This frees us up to calculate in flambda or fnu. So if we read in the SED in flambda, we don't need to convert the entire SED to fnu just for flux evaluation. The calculation is just Flux as calculated by 1, 2.

Possible methods of BandPass

- griddedFlux(SedgriddedtoBP, zeroPoint=None, MagSys='ab', wavelen='wavelen') single sed, no checks about gridding, but potentially out of band if bestwavelen is used in place of wavelen using wavelen_min, wavelen_max criteria if zeroPoint is None, return is on units of $num/cm^2/sec$, if zeroPoint is a number, the return is $Flux/Fs * 10^{0.4zp}$ if magsystem is not 'ab', we would have to evaluate the flux in the bands. (This is new anyway, we do not have non-ab mags in our system yet, we dont have to do this))

- Flux(Sed, zeroPoint=None, MagSys='ab', wavelen='wavelen', regrid='sed') This is what we already have (moduly zeropt, Magsys) Check wavelen limits, if outside return None if OK Check gridding, if not same, interpolate SED unless regrid='bp', store in copy and use to perform calculations. We should not be using this very often, unless we have a very noisy sed, and feel that it is more appropriate to resample the bandpass

- ManyFluxCalc(SedList, zeroPoint=None, MagSys='ab', wavelen='wavelen', regrid='sed') UseCase: SedList is quite homogeneous, and has been set to the same wavelength grid, but can have different wavelen_min, wavelen_max valuess.

3.2 BandPassDict

Orderded dict of BandPass Objects, so that we can refer to a particular band in terms of both indices and strings.

3.2.1 Possible Methods:

- Construction: using __init__ passing in the arrays of wavelen, sB in order and providing string names. with .fromThroughputsDir() method. For the most frequently used (eg. LSST) we should use a registry and builtins so that the code 'knows' "LSST" bandpasses. then it should be possible to construct the correponding BandPassDict as :

  ```
  LSSTBP = BandPassDict('LSST')
  ```

- Flux for SedList : FluxesForSeds(SedList, BandList, zp, magsys) BandList can be a list of indices[0, 1, 2, 3] or strings ['u', 'g', 'r', 'i'] SedList is an instance of Sed. zp should be None or an array the length of SedList, BandList and zp, should match Would be interesting if BandList could be a list of lists

3.3 SED

3.4 SEDList