PACS Spectrometer Simulation and the Extended to Point Correction

Jeroen de Jong

February 11, 2016

Abstract

This technical note describes simulating a PACS observation with a model source and its application to computing the extended to point-source correction. The simulation is a forward modeling which computes how the light is distributed over the spaxels of the PACS detector given a source model, the measured beams and the orientation and position of the PACS footprint on this model source. The extended to source correction is then computed by dividing the simulation of an extended source by that of a point source.

In the next sections we describe the algorithm, its implementation in HCSS and show some results. In the results we discuss how the measured flux compares to the expected flux from the simulation.

1 Prerequisites

The algorithm basically projects the beams of the PACS spectrometer spaxels on a model source for a given pointing offset, position angle and wavelength. The beams are provided as calibration files in the calibration tree of HIPE. The source is defined by the user by means of an equation (see Sect. 3.3). For this the algorithm needs the following inputs:

- *The beams*, which are obtained from the PACS calibration WIKI page and contain the flux $B_{klij\lambda}$ (normalized to the maximum flux), where k, l are the spaxel indices, λ the wavelength of the beam and i, j is a pixel with 0.5 arcsec size.
- *The pointing offset*: $\Delta\alpha$ and $\Delta\delta$ are the RA and Declination offset of the Field-Of-View (FOV) with respect to a perfect pointing on the center of the beam of the central spaxel.
- *The position angle* θ of the FOV (degrees), counter-clockwise with respect to the declination axis.
- *A model of the source*, which is defined as a function $S(\alpha, \delta)$, where α and δ are the relative RA and Declination in arcsec with respect to the center of the source.
- In case you use the simulations for computing the extended- to point-source correction then the total flux of the extended- and point-source must be normalized, i.e.:

$$\iint S(\alpha, \delta) d\alpha d\delta = 1 \tag{1}$$

2 Algorithm Description

In order to compute how much light falls on a given spaxel the algorithm has to obtain a beam image for the given spaxel and wavelength, compute for each pixel in this beam image the relative Right Ascension and Declination (given position angle and pointing offset), compute the source model for these coordinates and multiply the source and beam. This is all done in the following steps:

1. Get the dimensions of the beam images (M, N) and size of each of their pixels in arcsec ($d_{\text{pix}} = 0.5$ for the current beams).
2. Compute the coordinates $\alpha(i, j)$ and $\delta(i, j)$ for each pixel in the beam images, given the offset ($\Delta\alpha$, $\Delta\delta$) and rotation (θ) of the beam with respect to the source:

$$x_{ij} = -d_{\text{pix}} \left(i - M/2 + 0.5\right) \tag{2}$$

$$y_{ij} = +d_{\text{pix}} \left(j - N/2 + 0.5\right) \tag{3}$$

$$\begin{pmatrix} \alpha_{ij} \\ \delta_{ij} \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_{ij} \\ y_{ij} \end{pmatrix} + \begin{pmatrix} \Delta\alpha \\ \Delta\delta \end{pmatrix} \tag{4}$$

3. Compute the source model for each pixel given the coordinates computed with Eq. 4: $S_{ij} = S(\alpha_{ij}, \delta_{ij})$.

4. Compute for all spaxels and wavelengths the effective signal $F_{kl\lambda}$:

$$F_{kl\lambda} = d_{\rm pix}^2 \sum_{ij} B_{klij\lambda} S_{ij} \tag{5}$$

To avoid numerical errors due to too much discretization each pixel is actually divided into $N_{\rm sub}$ sub-pixels and S_{ij} is always the average over $N_{\rm sub} \times N_{\rm sub}$ values computed for these sub-pixels.

2.1 Computing the Extended to Point Correction

Using the simulation from the previous section one can compute the extended to point correction as follows:

1. Compute the extended source ($S_{{\rm ext}_{ij}}$) image using the coordinates computed with Eq. 4. Do the same for the reference point-source image ($S_{{\rm point}_{ij}}$), except that no offset and rotation is used.
2. Normalize both images: $S_{ij} \rightarrow \frac{S_{ij}}{\sum_{ij} S_{ij}}$.
3. Compute for both the extended and point-source the fluxes with Eq. 5.
4. Then the extended- to pointsource correction for the central spaxel $C_{\rm cen}$ is:

$$C_{{\rm cen}_\lambda} = \frac{F_{22\lambda_{\rm ext}}}{F_{22\lambda_{\rm point}}} \tag{6}$$

5. and the same correction for the central 3x3 spaxels is:

$$C_{{\rm 3x3}_\lambda} = \frac{\sum_{k=1..3,l=1..3} F_{{\rm ext}_{kl\lambda}}}{\sum_{k=1..3,l=1..3} F_{{\rm point}_{kl\lambda}}} \tag{7}$$

3 Implementation

Steps 1 to 4 in Sect. 2 are done in the class `BeamProjector` in the package `herschel.pacs.spg.spec.beams`. This class accepts the extended source as an object which implements the `ExtendedSource` interface (see Sect. 3.3). For constructing a `BeamProjector` object one also needs to provide a `BeamsPerSpaxel` calibration product. The PACS calibration tree contains one such a product per band. Each of these products contain the beam images for the available wavelengths.

3.1 Task for Extended-to-Point Correction Computation

The `BeamProjector` class is used inside the task `specExtendedToPointCorrection` which implements the algorithm steps described in Sect. 2.1. This task runs the following steps:

1. Get input data:
   (a) Get an input reference spectrum product (`refSpec`).
   (b) Get the wavelength array from `refSpec`.
   (c) Get the position angle (`posAngle`) and band from `refSpec`.
   (d) Get a pointing offset from the task parameters (`raOffset, decOffset`).
   (e) Get all the beam calibration files for the channel belonging to the band.
   (f) Get or create the source object (using the standard disk with x and y radius).
2. Create `BeamProjector` objects for the source (`projExt`) and a reference point source (radius 0.1", `projPoint`). The 0.1" radius has been chosen to make sure that a single pixel of the beam image is used. Also, the pointsource is a delta function on the beam.
3. Compute the coordinates of the extended and point source on the beams using the `posAngle`, `raOffset` and `decOffset` values. The reference point source has no pointing offset using the method `BeamProjector.computeCoordinates()`.
4. Compute the normalized flux of the source using the method `BeamProjector.computeSource()` (called automatically within `computeCoordinates()`).
5. Loop over all available wavelengths in the calibration files and compute the effective signal $F_{kl\lambda}$ with Eq. 5 for the extended and point source using the method `BeamProjector.project(wave)`. This method returns a 5x5 array containing the signal for all spaxels.

6. Then compute for all wavelengths the correction factors using Eqs. 6 and 7.
7. Finally fit a 3rd order polynomial through all the $C_{\rm cen}(\lambda)$ and $C_{\rm 3x3}(\lambda)$ values. This model is chosen to get a smooth function throught the data points.
8. Compute the correction values for all the wavelengths in the reference spectrum using this 3rd order polynomial.

3.2 Task for computing simulated PACS spectrometer products

The task `pacsSpecFromModel` computes the flux with Eq. 5) for all spaxels and all wavelengths in the (rebinned) cubes of a reference observation. Thereby it simulates a complete observation. Such simulations can be used to test algorithms for a particular model source and observation. This task runs the following steps:

1. Get input data:
   (a) A reference cube which contains the wavelengths, positions and position angle for which the simulation will be computed (`refCube`).
   (b) The beam calibration files obtained from the PACS calibration tree.
   (c) A model of the source, which depends on wavelength. This model has to implement the `ExtendedSpectralSource` interface (see Sect. 3.3).
   (d) Right Ascension and Declination of the source with Epoch. Will be taken from the `refCube` product meta data if not provided.
   (e) The Signal-to-Noise of the simulated spectra.
2. Compute the flux for each spaxel and wavelength in the reference cube using the `BeamProjector` class, which implements the algorithm described in Sect. 2.

The task can work also on a context of (rebinned) cubes and then computes the simulation for each cube in the context. You can then e.g. project these simulated cubes with the tasks `specProject, specInterpolate` or `drizzle`.

3.3 Defining a Source

A custom source can be defined by implementing the interface `ExtendedSource` or `ExtendedSpectralSource` in case a wavelength dependent source is needed. These interfaces contain only one method: `getFlux(double relRa, double relDec, [double wavelength])` to define the flux as function of relative sky coordinates and optionally wavelength (`ExtendedSpectralSource` interface). `relRa` and `relDec` are the relative RA and Dec in arcseconds, relative to the catalogue coordinates of the source. The wavelength and flux values are provided in the same unit as the observations which one wants to simulate. For the extended-to-point source method the flux units do not matter, since they are anyway normalized.

The `specExtendedToSourceCorrection` task uses by default the implementation `SimpleDiscSource` which defines a simple elliptical flat disk source with x and y radius. Furthermore, a class `ImageExtendedSource` is provided, which derives the flux from a provided image with WCS coordinates. The `pacsSpecFromModel` task always requires you to provide a specific model, but we provide the `SmallExtendedSpectralSource` implementation, which is a gaussian shaped source with a single spectral line.

The URM of the task `specExtendedToPointCorrection` contains an example of how to define your own extended source in Python by extending the `ExtendedSource` interface.

4 Results

4.1 Comparing Point- and Flat Extended Source Results with the Calibrations

As a first test we compared a point source and a flat extended source simulation with the point source loss and extended source loss calibration files. In these extreme cases the simulated fluxes of the central spaxel divided by the expected model fluxes should correspond to the values in the corresponding calibration files. The point-source loss fraction corresponds indeed within 1% to the calibration file values. However, the extended source loss correction from the simulation can be up to 15% higher compared to the calibration file values, especially in the red around 125μ the simulated values are much higher. This must be a problem of the normalization of the BeamsPerSpaxel calibration files, since one can easily compute that the extended source loss correction as:

$$C_{\mathrm{ext}_{kl\lambda}} = \frac{0.25}{9.4^2} \sum_{ij} B_{klij\lambda} \tag{8}$$

(by using Eq. 5, $S_{ij} = 1.0$, $d_{\rm pix} = 0.5$ and a spaxel size of 9.4")

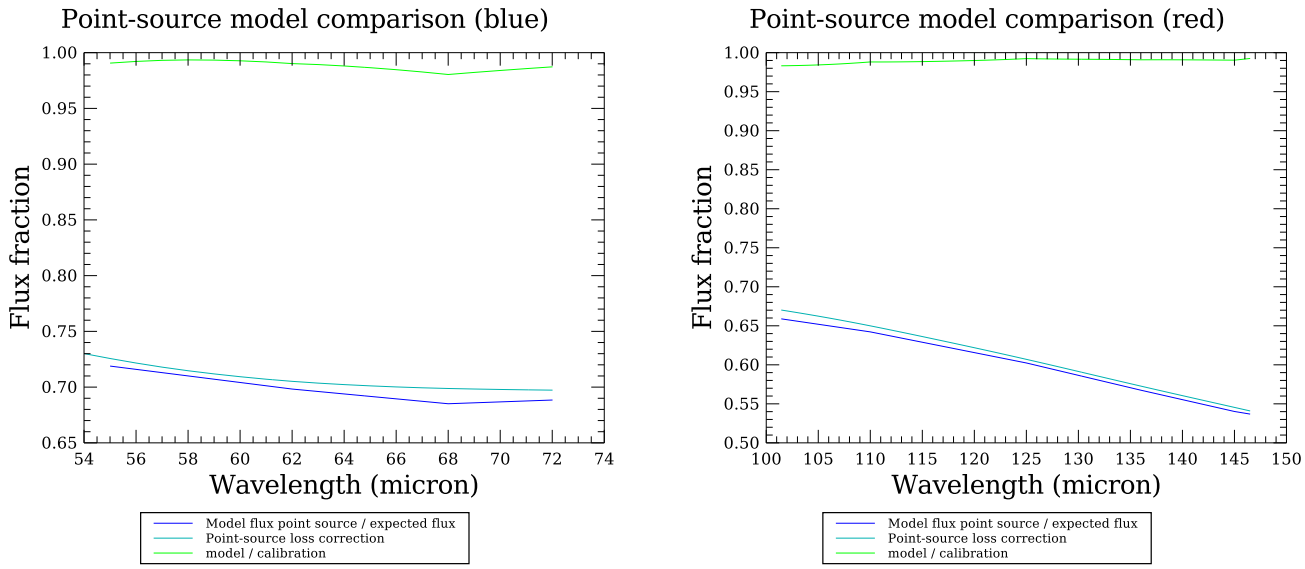

Figure 1: Point-source loss correction from simulation compared the calibration file point-source loss correction.

Measure	3”	15”	Neptune Obs.
$F_{\rm cen}/F_{\rm exp}$	0.850	0.794	—
$F_{\rm drz}/F_{\rm exp}$	0.863	0.859	—
$F_{\rm prj}/F_{\rm exp}$	0.777	0.796	—
$F_{\rm int}/F_{\rm exp}$	0.778	0.764	—
$\rm FWHM_{drz}$	9.965	17.914	9.559
$\rm FWHM_{int}$	12.284	18.932	12.284
$\rm FWHM_{prj}$	9.605	17.769	9.660

Table 1: The measured fluxes compared to the expected fluxes from the forward model for a 3” source, a 15” source (2D gaussian functions) and the actual Neptune observation (1342246387). Furthermore, the FWHM in arcsec is shown as well for the drizzled, interpolated and projected cubes.

For the point-source we took a Gaussian shaped source with a FWHM of 0.1”, a flat continuum of 1.0 Jy and no line (using Eq. 10). The expected flux for this source is the integral under this 2D Gaussian, which is computed as follows ($d_{\rm source}$ is the FWHM of the model source):

$$F_{\rm total} = 2\pi F_{\rm cont} \left(\frac{d_{\rm source}}{2\sqrt{2\ln 2}} \right)^2 \tag{9}$$

Using this we compute the simulated point-source correction with Eq. 6 with $F_{22\lambda_{\rm point}} = F_{\rm total}$. For simulating the extended source correction we used $S_{ij} = 1.0$ and computed the expected flux as $F_{\rm exp} = 9.4^2$, which is the integration of the source model over a 9.4” square spaxel.

We used the SED observation 1342197794 as a reference and remove the pointing offset and jitter to simulate a perfectly pointed point source observation. The resulting simulated and calibration point-source loss corrections are shown for the red and blue in Fig. 1. Fig. 2 shows the results with the flat extended source model ($S_{ij} = 1.0$), which is the same as the values of Eq. 8, compared to the extended source loss correction calibration file. One can see in this figure that the extended source loss correction computed from Eq. 8 is identical to the model results for a flat extended source. Therefore, the deviations from the calibration file extended source loss correction must be due to the normalization of the BeamsPerSpaxel calibration files.

4.2 Simulations of a Raster Observation

We tested the simulator with the blue rebinned cubes of observation 1342246387, which is a 5x5 raster observation with 2.5 arcsec steps. The model source is a circular 2D gauss function with a FWHM of 3 and 15 arcsec (compare point and slightly extended source), a continuum of 1.0 Jy and with a gauss function line with an amplitude of 3 Jy, FWHM of 0.05 μ and line center at 66.45 μ. The Source function of the model is:

$$S(\Delta\alpha, \Delta\delta, \lambda) = e^{-\frac{(\Delta\alpha - \alpha_{\rm offset})^2}{2\sigma^2_{\rm source}}} \left((F_{\rm cont} + \Delta F_{\rm cont}(\lambda - \lambda_{\rm line}))(1 + Ae^{-\frac{(\lambda - \lambda_{\rm line})^2}{2\sigma^2_{\rm line}}}) \right) \tag{10}$$

The following analysis steps were done on the simulation results:

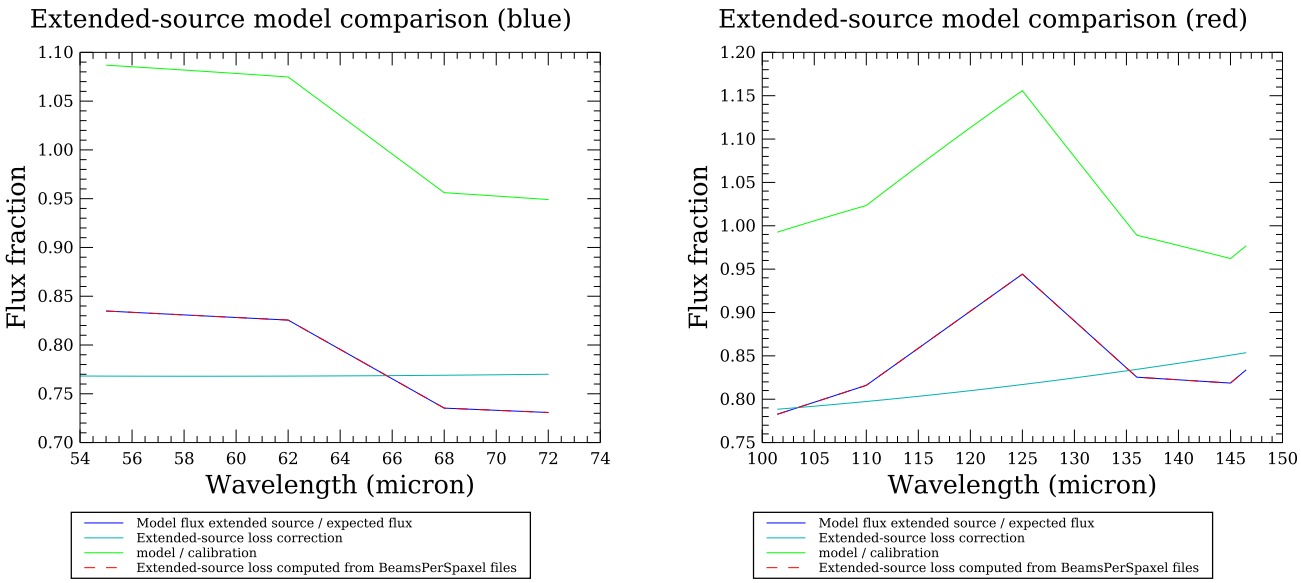

Figure 2: Extended source loss correction from simulation compared the calibration file extended source loss correction. In addition the result of Eq. 8 is plotted as a red dashed line to show that this is identical to the forward model results as expected.

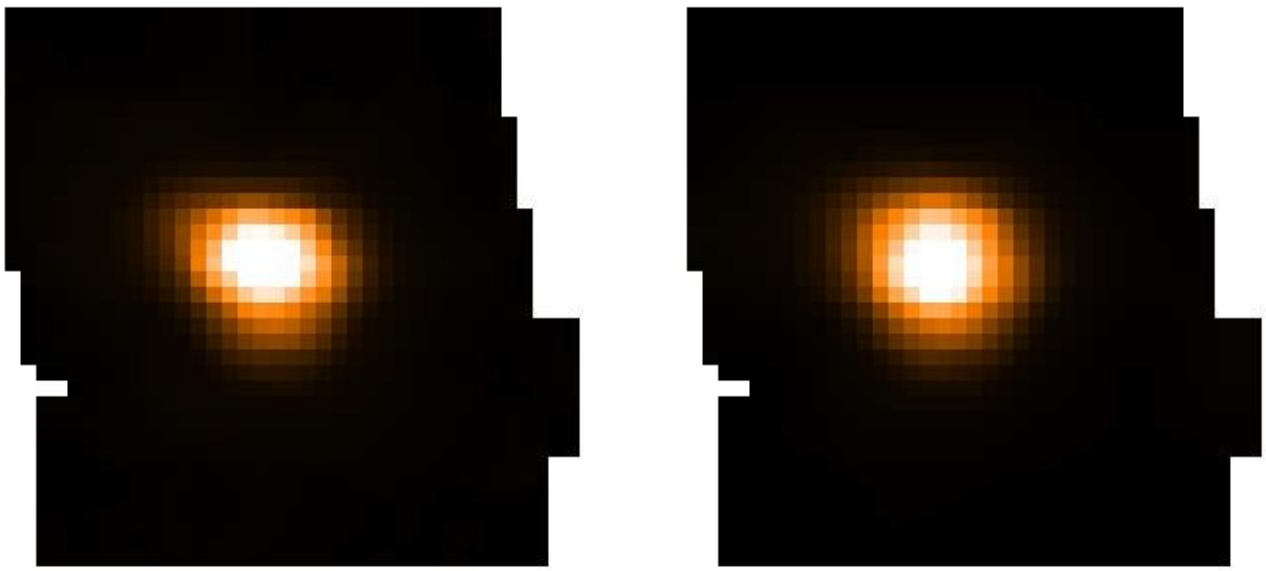

Figure 3: Drizzled cubes of Neptune 5x5 2.5 arcsec observation (left) and of the corresponding simulation with a 3" source (right).

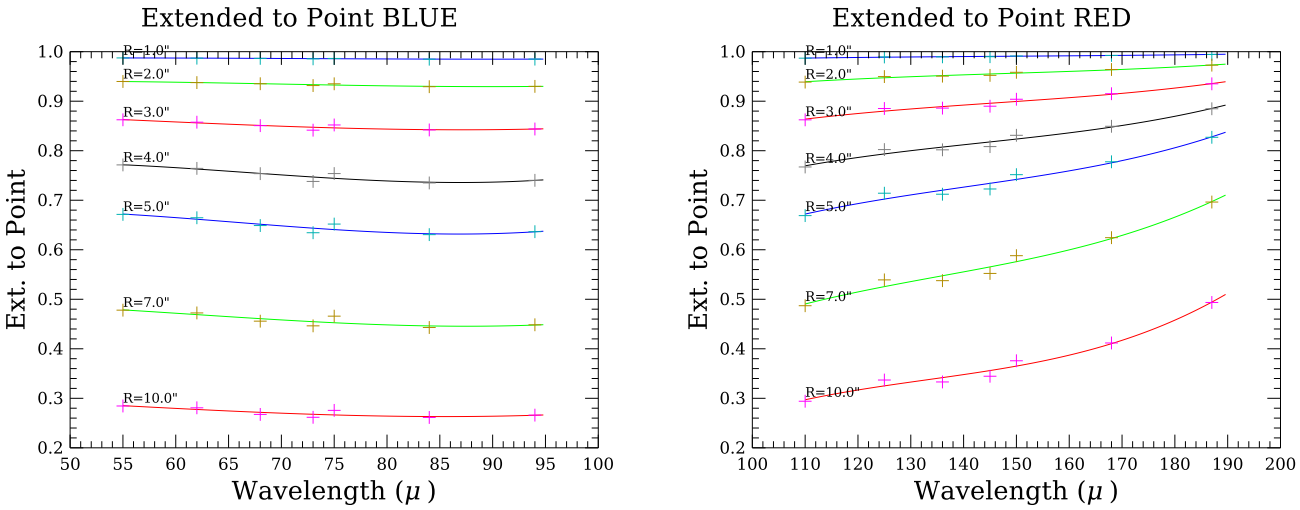

Figure 4: Extended to point source correction curves in the blue (left) and red (right) for the central spaxel only and uniform disks with the indicated radius

1. Compute the simulated level-2 cube and rebinned cube, based on the wavelengths and coordinates in the corresponding cubes from the observation.
2. Run the tasks drizzle, specProject and specInterpolate on these simulations, as well as on the original observation cubes.
3. Compute in both the simulation and the observations the integrated flux in the continuum of the single central pointing (rebinned cube), drizzled, interpolated and projected cubes.
4. In case of the simulation: compare the integrated flux with what is expected from the model (area under the 2D gauss function).
5. In all cases: compare the fluxes with each other.
6. Do a 2D gaussian fit to the simulated and real source in the drizzled, interpolated and projected cubes. Compare this source size with the resolution at the corresponding wavelength.

The results of this simulation analysis is shown in Table 1. The most important conclusion of this simulation is that for a single pointing observation of a point source the flux lost is already around 15%. If the source is 15" then the flux loss can be up to 20%.

4.3 Extended to Point Correction Curves

Figures 4 and 5 show some correction curves for uniform disk extended sources with a radius between 1.0 and 10.0 arcseconds. One can see that the curves don't follow the computed correction values exactly, but instead are limited to the more smooth 3rd polynomial fits. This avoids that single values introduce too many wiggles in the curves, which may not be real.

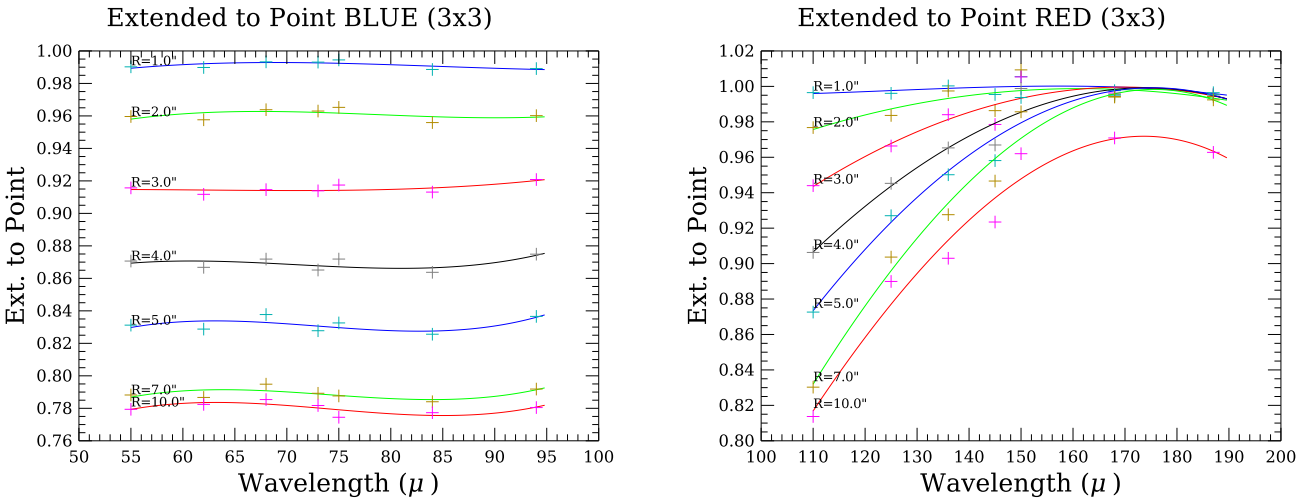

Figure 5: Extended to point source correction curves in the blue (left) and red (right) for the central 3x3 spaxels and uniform disks with the indicated radius