Technical note: telescope background model derivation in HIPE

1 Introduction *(Based on Albrecht Poglitsch's slides)*

Originally, the telescope background model was a pure physical model based on Planck functions. However, the "telescope background" derived from observations of celestial standards turned out to exhibit spectral features, for which there was no known explanation. It was then decided to go for an empirical representation of the telescope background.

The finally adopted approach is to derive a "reference" SED of the telescope on some reference date during the mission by using the mean of a number of SEDs derived from observations of well-characterized asteroids (Ceres and Pallas).

The SEDs were reduced with the normalization method. The signal can thus be expressed in terms of the ratio x between the source S and the telescope background T:

$$x = S/T \quad \text{therefore} \quad T = S/x$$

We know the absolute fluxes of the sources thanks to models provided by Thomas Müller and we then derive the telescope SEDs in Jy. The SEDs are then averaged by target.

Given the slight inconsistencies between the telescope SEDs derived form different asteroids, and taking into account potential non-linearity of the detectors for very bright sources, we decided to use Pallas as the absolute flux reference, but Ceres for a higher signal-to-noise ratio detailed SED. Ceres absolute flux was made to match Pallas with a linear fit

The time evolution of the model as a function of wavelength is derived from many key wavelength observations of the reproducibility source HD161796, also reduced with the normalization method, thus expressing the source flux in fraction of the telescope background.

2 Processing

The processing to derive the telescope background model can be described in 5 steps:

1. Reading of the mean telescope SEDs from Ceres and Pallas observations as ASCII files and performing a cubic spline interpolation
2. Scaling Ceres-based telescope absolute flux to match Pallas-based telescope flux
3. Reading the reproducibility source fluxes and express them in terms of $1/x$
4. Time evolution model
5. Telescope background SEDs as a function of operational day (OD)

2.1 Reading and interpolating asteroids-based mean telescope SEDs

The first step is to read the telescope SEDs built by reducing several Ceres and Pallas SED observations with the normalization method, converting the telescope signals into fluxes in Jansky by means of theoretical models of the asteroids and finally averaging them per target.

These SEDs were provided as ASCII files by Albrecht Poglitsch. They are read in HIPE and the data are stored in *TableDataset* objects.

Then, a cubic spline interpolation is performed on these SED data sets.

2.2 Scaling Ceres-based telescope SED to Pallas-based telescope SED

A linear model is fitted to the ratio of the Ceres-based telescope SED over the Pallas-based telescope SED. Then, the Ceres-based telescope SED is divided by this linear model to match Pallas-based telescope absolute flux, as shown in Fig. 1.

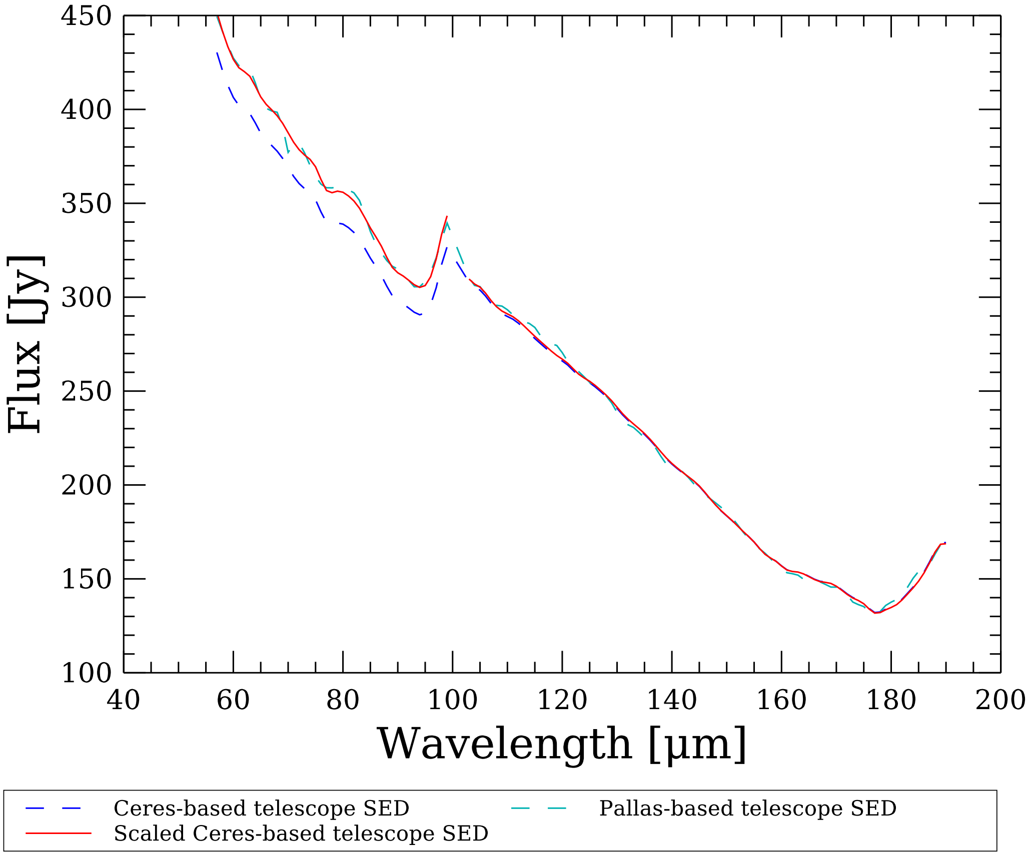

Figure 1: Ceres and Pallas-based mean telescope SEDs and scaled Ceres-based telescope SED scaled to match Pallas flux

2.3 Reading the reproducibility source fluxes

Key wavelengths observations all along the mission of the reproducibility source HD161796 were also reduced with the normalization method. The fluxes are therefore expressed in terms of x, the ratio of the source S over the telescope T.

These data, provided as ASCII files by Albrecht Poglitsch, are read for each key wavelength: 60 μm (B2A and B3A), 75 μm (B2B), 120 μm, 150 μm and 180 μm (R1). They are also stored in *TableDatset* products.

Then, to study the time variations of the telescope flux at the key wavelengths, we take the inverse of x:

$$x = S/T \quad \text{therefore} \quad 1/x = T/S$$

As we are only interested in the relative variations of the telescope flux T at a given wavelength, we do not need to know the source flux S. These $1/x$ values are stored in the same *TableDataset* products.

2.4 Time evolution model

The time evolution of the telescope background is modelled by a linear growth, combined with a periodic modulation at 60 μm.

The model used at 60 μm is a combination of a linear model and a sinusoidal model. It is written as follows in Albrecht Poglitsch's Mathematica notebook:

```
smodel = a * x + b + b * c * Sin[2 * Pi / 365 * x + d];
```

This needs a little bit of mathematics to implement it in HIPE:

$$\begin{aligned}
\text{smodel}(x) &= ax + b + bc\sin\left(\frac{2\pi x}{365} + d\right) \\
&= ax + b + bc\left(\sin d\cos\frac{2\pi x}{365} + \cos d\sin\frac{2\pi x}{365}\right) \\
&= \underbrace{\overbrace{a}^{\text{p[1]}}x + \overbrace{b}^{\text{p[0]}}}_{\text{PolynomialModel(1)}} + \underbrace{\overbrace{bc\sin d}^{\text{p[2]}}\cos\frac{2\pi x}{365} + \overbrace{bc\cos d}^{\text{p[3]}}\sin\frac{2\pi x}{365}}_{\text{SineAmpModel(1.0/365)}}
\end{aligned}$$

We derive:

$$a = p[1], b = p[0], d = \arctan\frac{p[2]}{p[3]}, c = \frac{p[2]}{b\sin d}$$

Figure 2 shows the fluxes of the telescope background at the key wavelengths, relative to the ones of the reproducibility source HD161796, as a function of time, together with the fitted models.

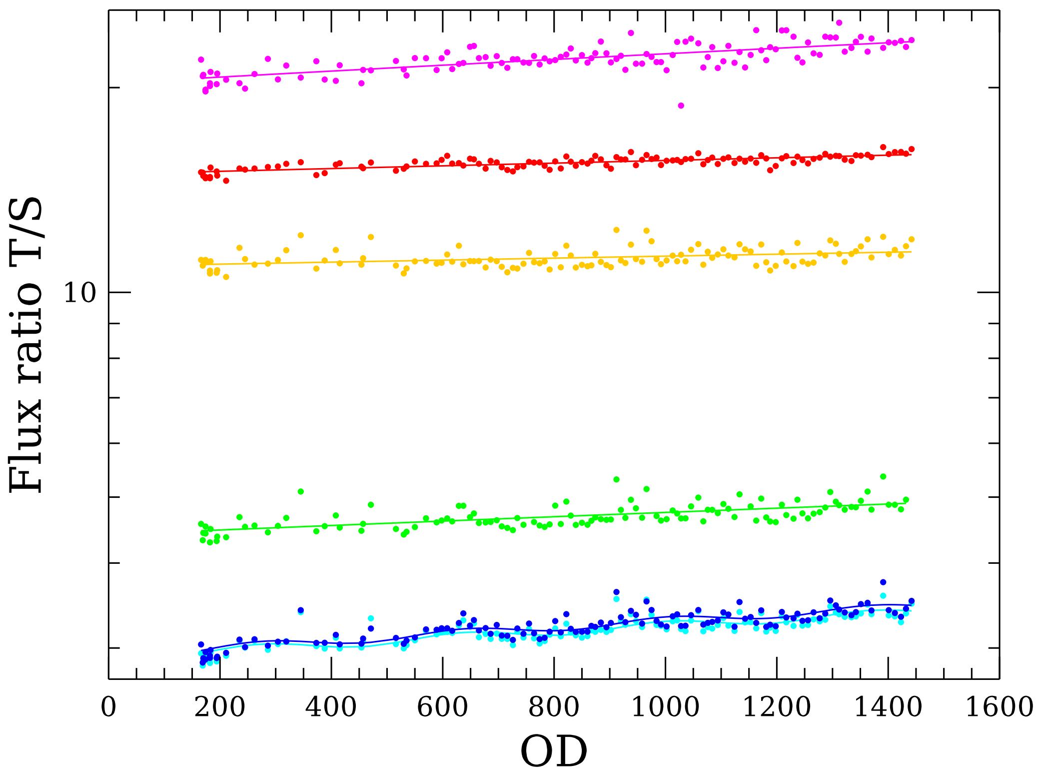

Figure 2: Time evolution of the telescope background at the key wavelengths. A combination of linear growth and periodic modulation is fitted at the 60 μm data (B2A and B3A). A simple linear model is fitted at the other key wavelengths.

2.5 Wavelength dependency

The wavelength dependency of the linear growth is fitted by a parabola in order to have a growth rate per 1000 ODs (see Fig. 3).

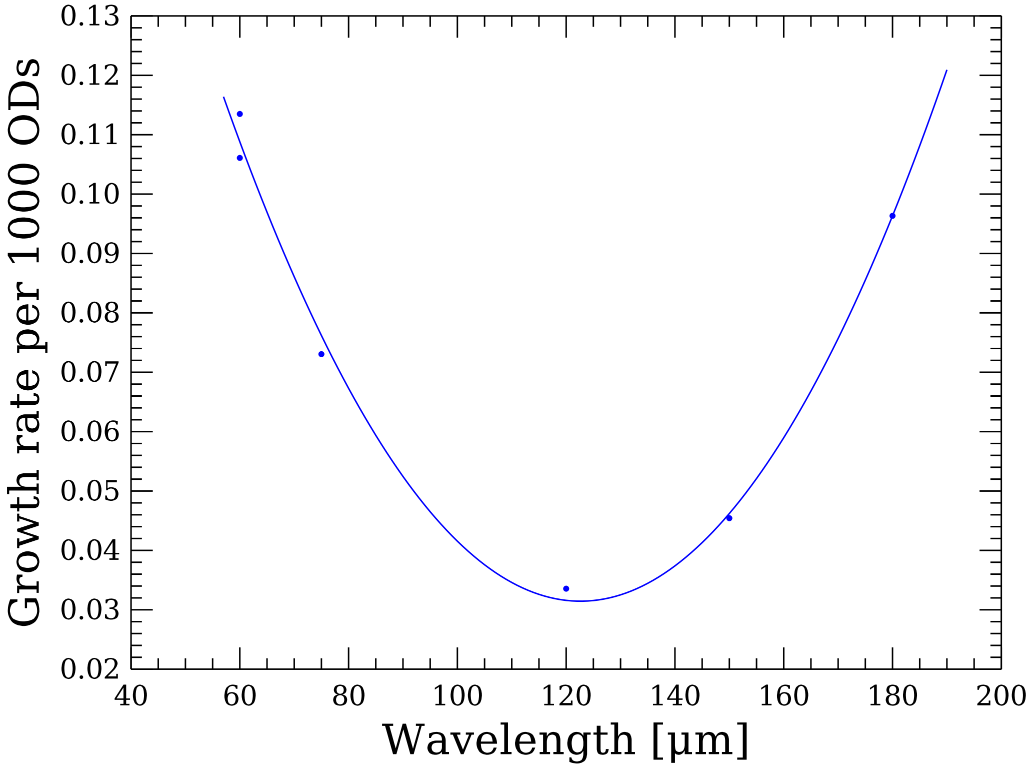

Figure 3: Growth rate per 1000 ODs as a function of the wavelength. A parabola has been fitted to the data.

2.6 Telescope background model

The seasonal variation, linearly decreasing from 55 μm to 101 μm (there is no seasonal correction in the red), combined with the linear growth, which follows a parabola with wavelength and linearly increases with OD, formed a correction factor which is applied to the reference SED. No correction at all is applied for ODs below 180.

A reference OD (791), roughly corresponding to the middle of the period covered by the model, was chosen.

Tables of telescope background SEDs are then built for a complete range of ODs from 1 to 1446 in the blue ($55 < \lambda < 101\,\mu$m) and in the red ($103 < \lambda < 190\,\mu$m). These tables are then included in the PACS spectrometer calibration tree.

Figure 4 shows different telescope background SEDs for different ODs from 200 to 1400.

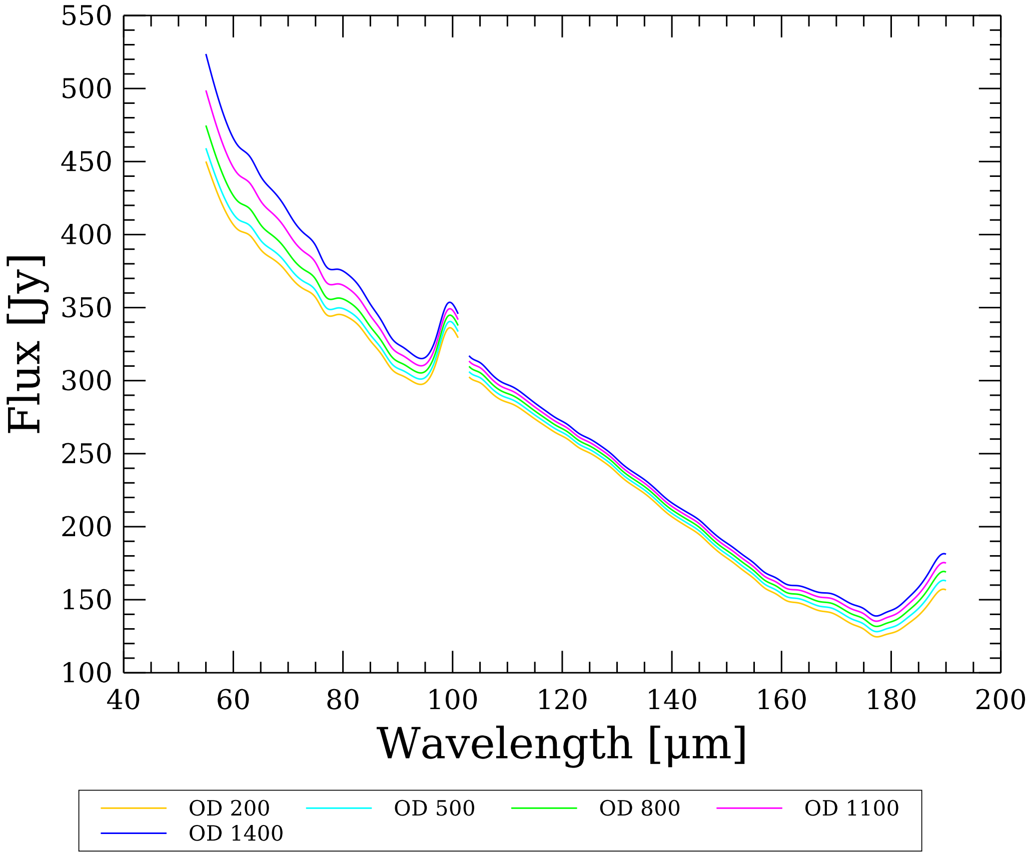

Figure 4: Telescope background SEDs for different ODs

3 Comparisons

3.1 With the calibration tree

The model currently used in the PACS spectrometer calibration tree (version 8 of the calibration file, see JIRA ticket PACS-5612) is not the latest version of the model provided by Albrecht Poglitsch but an earlier version, introduced in calibration tree version 62 (version 7 of the calibration file, see JIRA ticket PACS-5598), which has been scaled to match the absolute flux level of the calibration blocks based method.

Figure 5 shows the ratio, for different ODs between 200 and 1400, between the model described in this document and the latest version currently in use in the PACS spectrometer calibration tree. Although they are different, they do not differ by more than 5% in the red and even less than 2% in the main part of the blue range.

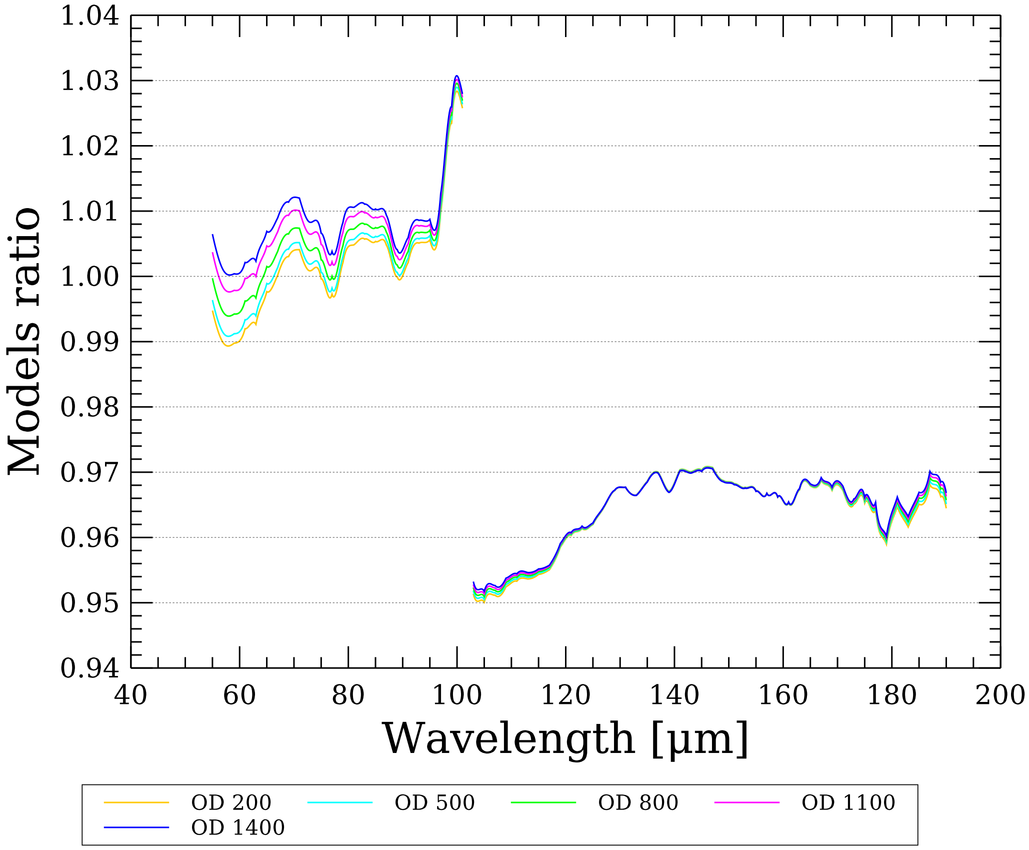

Figure 5: Ratio between the latest telescope background model with the version currently in use in the calibration tree (calibration file version 8)

3.2 With SPIRE

The comparison between the PACS and SPIRE telescope background models is much more difficult because of the different wavelengths domains and because they are conceptually different.

The telescope background in SPIRE is modelled as an extended source radiating as the sum of two black bodies, corresponding to the primary and secondary mirrors M1 and M2, with an additional correction to take into account the OD-dependent emissivity of the primary mirror M1.

On the other hand, the telescope background model in PACS is seen as an absolute flux calibration model for data reduced with the normalization method, *i.e.* with the source signal expressed as a fraction of the telescope signal. This model is empirically built from observations of well calibrated asteroids. For a given telescope background signal value, the corresponding source of light is a spaxel.

To compare the two models, we will assume that the telescope background is seen as a point source. Ivan Valtchanov wrote a script which compute the SPIRE telescope background model for OD 600 assuming it is a point source, *i.e.* multiplying the telescope background spectral radiance in W/m^2/Hz/sr by the beam solid angle for a point source. For PACS, we apply the point souce correction to the telescope background SED for OD 600. The results are shown in Fig. 6. The wavelength ranges do not overlap but we already see that the two models match quite well, if we exclude the band edges (red leak region beyond 180 μm for PACS).

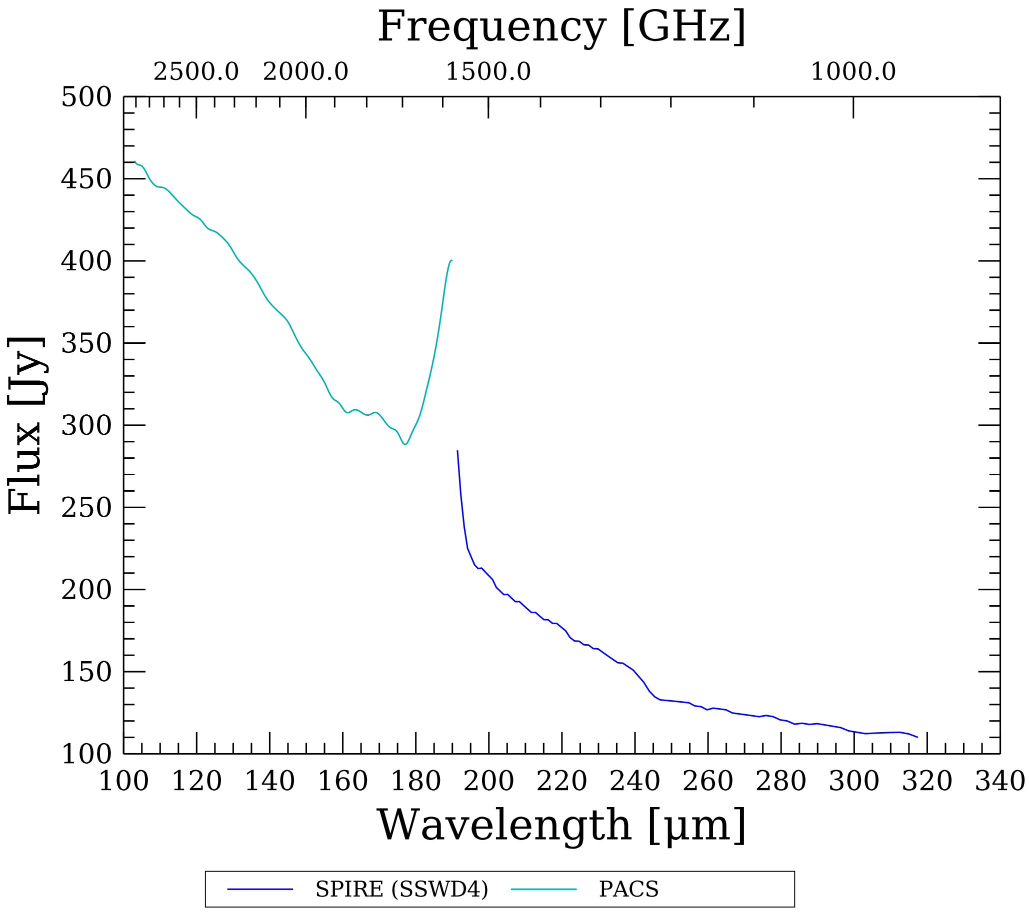

Figure 6: Comparison between SPIRE and PACS telescope background models assuming the telescope background as a point source

A Complete script in HIPE

```
#
#  This file is part of Herschel Common Science System (HCSS).
#  Copyright 2001-2015 Herschel Science Ground Segment Consortium
#
#  HCSS is free software: you can redistribute it and/or modify
#  it under the terms of the GNU Lesser General Public License as
#  published by the Free Software Foundation, either version 3 of
#  the License, or (at your option) any later version.
#
#  HCSS is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
#  GNU Lesser General Public License for more details.
#
#  You should have received a copy of the GNU Lesser General
#  Public License along with HCSS.
#  If not, see <http://www.gnu.org/licenses/>.
#

from java.lang.Math import PI
from herschel.share.unit.Length import MICROMETERS
from herschel.share.unit.FluxDensity import JANSKYS

### Put here the directory path where the data files are saved.
dataDir = "/home/christophe/PACS/TelescopeBackground/"

# Read the SED data from an ASCII file and store them in a TableDataset
def readMeanTelescope(fileName):
  tab = asciiTableReader(file=dataDir+fileName,tableType=’SPACES’)
  tab.setColumnName(0,’Wavelength’)
  tab.setColumnName(1,’Flux’)
  tab[’Wavelength’].setUnit(MICROMETERS)
  tab[’Flux’].setUnit(JANSKYS)
  return tab

meanTelescopeCeres = readMeanTelescope(’MeanTelescope_Ceres.txt’)
meanTelescopePallas = readMeanTelescope(’MeanTelescope_Pallas.txt’)

# Perform a cubic spline interpolation of the data
def interpolation(table):
  interp = CubicSplineInterpolator(Double1d(table[’Wavelength’].data), table[’Flux’].data)
  return interp

interpCeres = interpolation(meanTelescopeCeres)
interpPallas = interpolation(meanTelescopePallas)

# Fit a polynomial model to data
# Return a dictionary with the fitted model parameters and the model
def linearModel(xdata,ydata,order):
  output = {}
  model = PolynomialModel(order)
  fit = Fitter(xdata,model)
  output[’parameters’] = fit.fit(ydata)
  output[’model’] = model
  return output

# Define the wavelength ranges
waveBlue = Double1d(Int1d(range(57,100)))
waveRed = Double1d(Int1d(range(103,191)))

# Build the data sets containing the ratio between the Ceres-based telescope
# background (TB) and the Pallas-based telescope background
dataBlue = TableDataset()
dataBlue[’Wavelength’] = Column(data=waveBlue)
dataBlue[’Flux’] = Column(data=interpCeres(waveBlue) / interpPallas(waveBlue))

dataRed = TableDataset()
dataRed[’Wavelength’] = Column(data=waveRed)
dataRed[’Flux’] = Column(data=interpCeres(waveRed) / interpPallas(waveRed))

# Fit the ratios Ceres-based TB / Pallas-based TB
modelBlue = linearModel(dataBlue[’Wavelength’].data, dataBlue[’Flux’].data,1)[’model’]
modelRed = linearModel(dataRed[’Wavelength’].data, dataRed[’Flux’].data,1)[’model’]

# Define the mean telescope background SEDs
# (Ceres-based TB scaled to Pallas absolute flux)
```

```
def meantelsedblue(x):
  return interpCeres(x) / modelBlue(x)

def meantelsedred(x):
  return interpCeres(x) / modelRed(x)

# Read the reproducibility data from an ASCII file and store them in a TableDataset
def readReproducibility(fileName):
  tab = asciiTableReader(file=dataDir+fileName,tableType=’SPACES’)
  tab.setColumnName(0,’OD’)
  tab.setColumnName(1,’Flux’)
  return tab

# Convert the fluxes in 1/x
fluxRepr = {}
for band in (’B2A_60’,’B3A_60’,’B2B_75’,’R1_120’,’R1_150’,’R1_180’):
  fluxRepr[band] = readReproducibility(’flux3x3_’+band+’.txt’)
  fluxRepr[band][’Flux_1/x’] = Column(data=1.0 / fluxRepr[band][’Flux’].data)

# Degradation model
degradm = TableDataset()
degradm[’Wavelength’] = Column(data=Short1d([60,60,75,120,150,180]))
degradm[’Data’] = Column(data=Double1d([\
  1000 * linearModel(fluxRepr[band][’OD’].data,fluxRepr[band][’Flux_1/x’].data,1)[’parameters’][1] /\
  MEAN(fluxRepr[band][’Flux_1/x’].data) for band in\
  (’B3A_60’,’B2A_60’,’B2B_75’,’R1_120’,’R1_150’,’R1_180’)]))

# Reference OD, roughly corresponding to the middle of the period covered by the model
day0 = 791

# Parabola fitted to the linear growth rate parameters
degradModel = linearModel(degradm[’Wavelength’].data,degradm[’Data’].data,2)[’model’]

# Evolution function
def evolution(x,od):
  return (od-day0) / 1000.0 * degradModel(x)

# Fit seasonal variations
# Return a dictionary with the fitted model parameters and the model
def seasonalModel(xdata,ydata):
  output = {}
  model = PolynomialModel(1)
  model += SineAmpModel(1.0 / 365)
  fit = Fitter(xdata,model)
  p = fit.fit(ydata)
  a = p[1]
  b = p[0]
  d = ARCTAN(p[2]/p[3])
  c = p[2] / (b * SIN(d))
  output[’parameters’] = (a,b,c,d)
  output[’model’] = model
  return output

a,b,c,d = seasonalModel(fluxRepr[’B2A_60’][’OD’].data,\
  fluxRepr[’B2A_60’][’Flux_1/x’].data)[’parameters’]

# Telescope background model, including degradation and seasonal variations
def telsedblue(x,od,sed=meantelsedblue):
  return sed(x) * (1 + (100-x) / (100-60) *\
    c * (SIN(2*PI/365 * od + d) - SIN(2*PI/365 * day0 + d)) +\
    evolution(x,od))

def telsedred(x,od,sed=meantelsedred):
  return sed(x) * (1 + evolution(x,od))

# Define wavelengths ranges [55,101] and [103,190] by step of 0.1
waveBlueFinal = Double1d(Int1d(range(550,1011))/10.)
waveRedFinal = Double1d(Int1d(range(1030,1901))/10.)
# Define ODs range [1,1446]
ods = range(1,1447)

# Build telescope background tables to be included in the calibration tree
bluetelod = TableDataset()
bluetelod[’Wavelength’] = Column(data=waveBlueFinal)
data = Double2d(waveBlueFinal.size, len(ods))
for i,od in enumerate(ods):
  data[:,i] = telsedblue(waveBlueFinal, MAX([180,od]))
bluetelod[’Flux’] = Column(data=data)
```

```
redtelod = TableDataset()
redtelod['Wavelength'] = Column(data=waveRedFinal)
data = Double2d(waveRedFinal.size, len(ods))
for i,od in enumerate(ods):
  data[:,i] = telsedred(waveRedFinal, MAX([180,od]))
redtelod['Flux'] = Column(data=data)
```