COMPUTATIONAL RADIOMETRY WORK PACKAGE

DOCUMENT NUMBER	EQUIPMENT OR SUB-SYSTEM

SUBJECT

11 Infrared Measurement and Analysis

DISTRIBUTION

CONCLUSIONS/DECISIONS/AMENDMENTS

AUTHOR	SIGNATURE
CJ Willers	

DATE	DATE	DATE January 25, 2023
PREVIOUS PACKAGE NO.	SUPERSEDING PACKAGE NO.	CURRENT PACKAGE NO.

1 11 INFRARED MEASUREMENT AND ANALYSIS

This notebook forms part of a series on computational optical radiometry[1]. The notebooks can be downloaded from Github[2]. These notebooks are constantly revised and updated, please revisit from time to time.

DOI [3]

The source for the notebook from which this PDF is built is at `https://github.com/NelisW/ComputationalRadiometry/blob/master/11-InfraredMeasurementAndAnalysis.ipynb`

The date of this document and module versions used in this document are given at the end of the file.

Feedback is appreciated: neliswillers at gmail dot com.

1.1 Overview

This notebook demonstrates the use of the code used to read FLIR Inc ptw files, calculating the instrument function and applying the instrument function to an example measurement.

See Listing 2.1 for the code.

1.2 Download and open FLIR PTW files

First download the sample file. You have to be connected to the internet to download the file (875 KB). Alternatively copy the file (`pyradiSamplePtw.tgz`) from the pyradi data directory to the notebook's working directory. After downloading and opening the file there should be three ptw files, an XML file with calibration data, and a few spectral data files in the current working directory.

See Listing 2.2 for the code.

```
filesAvailable are ['LW100mmLens.txt', 'LWIR-BBref-150C-150us.ptw', '↵
    lwir100mm150us10ND20090910.xml', 'LWIRsensor.txt', 'LWND10.txt', '↵
    PyradiSampleLWIR.ptw', 'PyradiSampleMWIR.ptw', 'Unity.txt']
```

Open one of the ptw files and print the header contents.

See Listing 2.3 for the code first read the ptw file.

```
CED version 5.60
Main Header Size 3476
Frame Header Size 1016
Frame + Frame Header Size 5608
Frame Size 5100
Number of Frames 100
Year 2011 Month 2 Day 8
( 2011 / 02 / 08 )
Hour 15 Minute 46 Second 51
( 15 : 46 : 51 )
Camera Name Jade
Detector Code1 0
Detector Code2 0
```

```
Detector Gain 5699
Lens 50 mm
Filter NE_001%
Aperture Name
Emissivity 0.980000
Ambient Temperature 293.149994 (K)
Ambient Temperature 19.999994 (degC)
Distance to target 2.0
Atm Transmission 1.0
Ext Coef 0.0
Target 0
Optic 0
Atmo 0
Atm Temp 293.149994
Cut on Wavelength 3.700000
Cut off Wavelength 4.800000
PixelSize 25.0
PixelPitch 30.0
Detector Apperture 2.0
Optic Focal Length 1.0
Housing Temp1 311.429993 (K)
Housing Temp2 0.000000 (K)
Sensor Temp4 0.000000 (K)
Detector/FPA Temp 0.000000 (K)
Camera Serial Number
Min Threshold 5699
Max Threshold 5794
Gain 0.0
Offset 0.0
HZoom 2.0
VZoom 2.0
Field 75
AD converter 14 bit
Ext Sync OFF
Header.h_Signature = CED
CEDIP Period 100.000002 Hz
CEDIP Integration 0.150000 msec
NUC 1
Comment
Calibration File Name
Tools File Name
Palette Index 1
Palette Current 1
Palette Toggle 0
Palette AGC 1
Unit Index 1
Current Unit Index 0
Zoom Pos
Key Framenum 0
Num Keyframes
Player lock 0
Frame Select 0
ROI Start 0
ROI Stop 99
Player inf loop 0
Player Init Frame 0
Isoterm0 0
Isoterm0 DL Min -1
Isoterm0 DL Max -1
Isoterm0 Color RGB #1e32fa
```

```
Isoterm1 0
Isoterm1 DL Min -1
Isoterm1 DL Max -1
Isoterm1 Color RGB #fcdc14
Isoterm2 0
Isoterm2 DL Min -1
Isoterm2 DL Max -1
Isoterm2 Color RGB #fa1e1e
Zero 0
Zero DL -1
Palette Width 20
PaletteF Full 0
PTR Frame Buffer type 0
Thermoelasticity 0.0
Demodulation 0.0
Coordinate Type 0
X Origin 0
Y Origin 0
Coord Show Orig 0
Axe Colour RGB #6464ff
Axe Size 1
Axe Valid 1
Distance offset 0.0
Histogram 0
Histogram % 60
Calibration File Name
PTRFrame Valid 0
Subsampling 0
Camera flip H 0
Camera flip V 0
BB Temp 0.0
Capture Wheel Index 0
Capture Wheel Focal Index 0
```

Next load a number of frames from the file: select which frames by listing the required frames (zero-based) in the list of frames:

See Listing 2.4 for the code loading sequence of frames.

```
(3, 68, 75)
./PyradiSampleLWIR.ptw
(1) Data shape=(3, 68, 75)
Frame 0 summary data:
Image data:
[[5790. 5796. 5793. ... 5792. 5793. 5796.]
 [5791. 5794. 5794. ... 5793. 5790. 5790.]
 [5794. 5792. 5794. ... 5793. 5792. 5792.]
 ...
 [5793. 5789. 5794. ... 5794. 5791. 5791.]
 [5795. 5794. 5795. ... 5789. 5792. 5791.]
 [5794. 5792. 5791. ... 5792. 5793. 5792.]]
Sensor Temperature4 311.4100036621094 K
FPA Temperature 70.0 K
Time 15:46:52.118
```

The following code attempts to load a frame that does not exist in the input file:

See Listing 2.5 for the code loading sequence of frames, with an error in request.

```
getPTWFrames Error: at least one requested frame not in file
legal frames for this file are: 1 to 100
```

```
(1,)
```

Load single frames and concatenate into a numpy array. The first index into the array is the frame dimension and the second and third indices are row and columns in the image. First load the 0'th element to create the data array and then append the additional frames to the original data array.

See Listing 2.6 for the code loading single frames.

```
(100, 68, 75)
```

1.3 Instrument function and calibration

This information is summarised from (Sec 9.8) of my book *Electro-Optical System Analysis and Design: A Radiometry Perspective*, CJ Willers, SPIE, 2013 [4].

The radiance values used in the calibration calculations are calculated using Equation 9.59 in my book:

$$L_{\text{image}} = \int_0^\infty L_{bb\lambda}\left(T_m\right)\epsilon_{\lambda m}\tau_{a\lambda}\mathcal{S}_\lambda d\lambda \tag{1.1}$$

where L_{image} is the wideband radiance value measured by the imaging camera, $L_{bb\lambda}\left(T_m\right)$ is the black body source spectral radiance at temperature T_m, $\epsilon_{\lambda m}$ is the source spectral emissivity, $\tau_{a\lambda}$ is the spectral atmospheric transmittance, and $\mathcal{S}_\lambda$ is the imaging sensor spectral response (including the detector, optics and filter).

The imaging radiometers and spectrometer were calibrated (or characterized) against laboratory blackbody sources. In most cases the calibration is an elaborate process covering several instrument settings, filters, and environmental conditions, but it can be summarized as follows. Calibration entails measuring the instrument output (voltage or digital levels) versus source temperature for a series of source temperatures (ideally over the full dynamic range of the instrument). The set of measurements form a temperature calibration curve (see figure below). When using the instrument, the curve is read 'backward' such that a given instrument voltage returns the appropriate source temperature. If the test object's emissivity is the same as the laboratory source emissivity, the object's temperature will be equal to the source temperature.

Accurate measurement work requires that the spectral sensor response must be known. In this case, the calibration source temperature, together with the spectral sensor response, can be used to calculate the inband source radiance. A new curve is now constructed to relate signal voltage with inband radiance for an extended source completely filling the pixels. This curve can now be used to determine the radiance field incident on the instrument during a measurement. This radiance value lends itself to further analysis.

The figure below shows typical calibration data. The top curve relates source temperature and instrument voltage (digital level). The bottom curve relates source irradiance on the sensor entrance aperture to instrument voltage. For an extended source, the source apparent radiance is related to irradiance by $E = L\omega$, where ω is the pixel FOV. In this case the instrument was calibrated for three different gain settings (integration times of 30, 120, and 500 μs) and two different sensor internal temperature conditions (16 and 42.6 $^\circ$C). Observe the importance of calibration at different internal temperatures: the curves (for the same instrument and settings) show significantly different responses. A good strategy to allow for temperature changes in the sensor is to measure the calibra-

tion curve at several different internal temperatures and then interpolate between these, according to the actual instrument temperature during the measurement.

The effect of hot optics is also shown in this figure. It is evident that the hot optics flux sets an asymptotically lower measurable flux limit (called the 'floor'). When measuring low-temperature test samples, a small variation in internal temperature shifts the floor up or down, playing havoc with calibration. In this particular instrument setting, an ND2 neutral density filter (0.01 transmittance) was used. Much of the hot optics flux emanates from this filter. So in all fairness, there is little point in using an ND2 filter when measuring a low-temperature target. The situation does arise, however, if there is a requirement to measure both hot and cold test samples without changing filters.

See Listing 2.7 for the code.

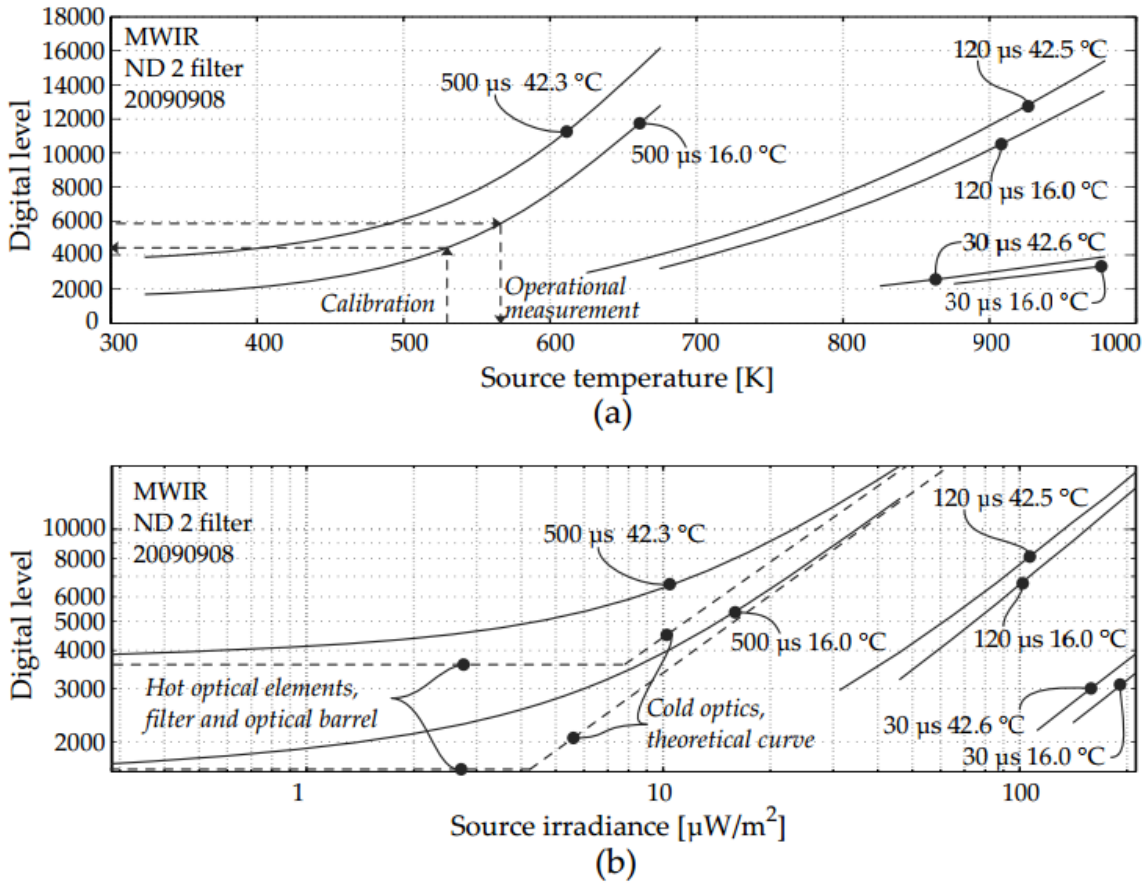


Figure 9.16 MWIR imaging radiometer calibration curves for (a) digital level vs. blackbody temperature, and (b) digital level vs. irradiance.

1.4 Calculating a practical instrument function

Load the calibration data, presumably done with a blackbody and at short range.

See Listing 2.8 for the code.

The calibration file contains all the information required to calculate the instrument function of the camera. It lists the filenames for the various spectral parameters (sensor response, optics transmittance, source emissivity, and atmospheric transmittance). The file also contains key sensor parameters such as detector pitch and fill factor and focal length. The most important information is the list of camera

digital levels versus source temperature, for one or two instrument temperatures. By convention, the temperature in this file must be in degrees celcius. An example file contents is shown below.

```
<JadeCalibration Name="lwir100mm0150us10ND20090910" ID="lwir100mm10ND" Version="" Summar
    <SensorResponse Filename="LWIRsensor.txt"/>
    <OpticsTransmittance Filename="LW100mmLens.txt"/>
    <Filter Filename="LWND10.txt"/>
    <SourceEmis Filename="Unity.txt"/>
    <AtmoTau Filename="Unity.txt"/>
    <DetectorPitch Value="30e-6"/>
    <FillFactor Value="0.7"/>
    <Focallength Value="0.1"/>
    <IntegrationTime Value="150e-6"/>
    <Fnumber Value="2"/>
    <Nu Min="700" Max="1665" Inc="5" />
    <Caldatas>
        <Caldata InstrTemperature="17.1" DlFloor="3625" Power="10" >
            <CalPoint Temp="50" DL="4571" Comment="temp in deg C" />
            <CalPoint Temp="100" DL="5132" />
            <CalPoint Temp="150" DL="5906" />
            <CalPoint Temp="200" DL="6887" />
            <CalPoint Temp="250" DL="8034" />
            <CalPoint Temp="300" DL="9338" />
            <CalPoint Temp="350" DL="10834" />
            <CalPoint Temp="400" DL="12386" />
            <CalPoint Temp="450" DL="14042" />
            <!-- <CalPoint Temp="500" DL="15324" /> -->
        </Caldata>
        <Caldata InstrTemperature="34.4"  DlFloor="4210" Power="10"  >
            <CalPoint Temp="50" DL="5477"  Comment="temp in deg C" />
            <CalPoint Temp="100" DL="6050" />
            <CalPoint Temp="150" DL="6817" />
            <CalPoint Temp="200" DL="7789" />
            <CalPoint Temp="250" DL="8922" />
            <CalPoint Temp="300" DL="10262" />
            <CalPoint Temp="350" DL="11694" />
            <CalPoint Temp="400" DL="13299" />
            <CalPoint Temp="450" DL="14921" />
        </Caldata>
    </Caldatas>
</JadeCalibration>
```

Two elements in this file are not measured but must be determined as part of the data analysis process:

```
JadeCalibration/Caldatas/Caldata@DlFloor
JadeCalibration/Caldatas/Caldata@Power
```

The filename to five spectral vectors must be supplied (see the `RadLookup` class in `rylookup.py`,

and the functions called in `ryfiles.py`). If any filename is None, then unity spectral values will be used.

1. `SensorResponse` the detector/sensor spectral response.
2. `OpticsTransmittance` optics/lens transmittance.
3. `Filter` the spectral filter transmittance.
4. `SourceEmis` calibration source emissity.
5. `AtmoTau` transmittance of the (atmospheric) path between the source and the camera.

The spectral data file must have two columns: first column is wavelength in µm, and second column is the spectral value at this wavelength. The spectral data in the second column must be normalised. Any other colums are ignored. Data read in from the file will be interpolated to (nuMin, nuMax,nuInc) scale.

The __init___() function loads the data and calculates the mapping functions between digital level, radiance and temperature. Using the spectral curves and DL vs. temperature calibration inputs it calculate the various mapping functions between digital level, radiance and temperature.

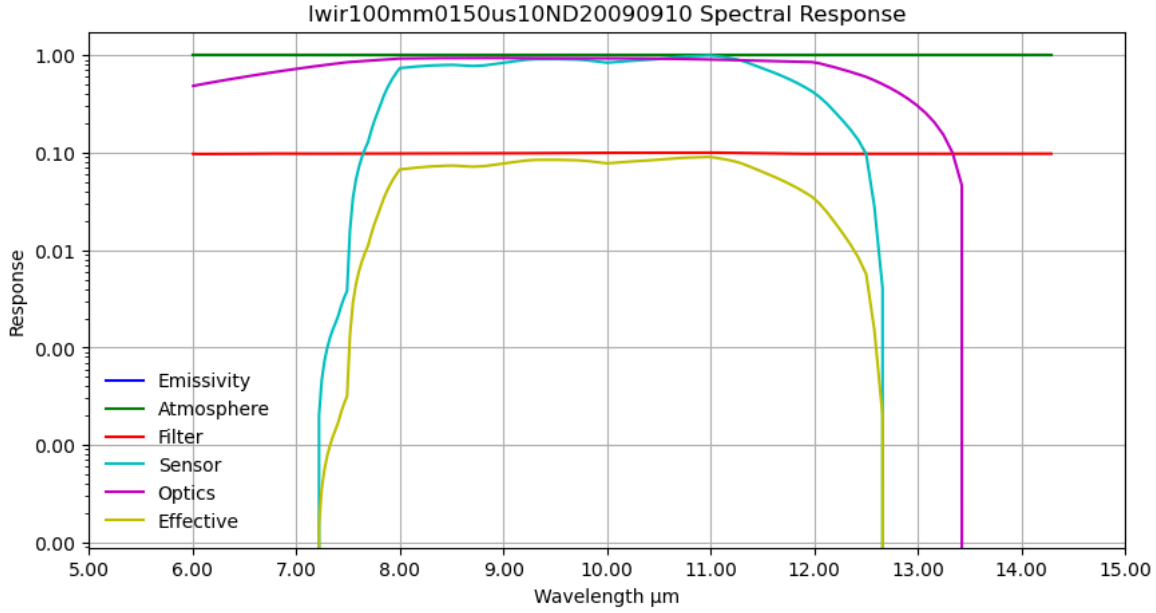
When loaded, plot the spectral data for inspection. The various plot functions plot the data to a series of files with hard-coded filenames, built up from the name of the calibration file, with an appropriate added string.

See Listing 2.9 for the code.

```
Path to XML                  = .
Path to datafiles            = .
Calibration Name             = lwir100mm0150us10ND20090910
ID                           = lwir100mm10ND
Version                      =
Summary                      =
DetectorPitch                = 3e-05
FillFactor                   = 0.7
Focallength                  = 0.1
integrationTime              = 0.00015
Fnumber                      = 2.0
Lookup Name          = lwir100mm0150us10ND20090910
Sensor Response      = ./LWIRsensor.txt
Optics Transmittance = ./LW100mmLens.txt
Filter               = ./LWND10.txt
Source Emissivity    = ./Unity.txt
Atmospheric Transm   = ./Unity.txt
Nu (Min, Max, Inc)   = (700.0, 1665.0, 5.0)
Calibration data set

Instrument temperature = 17.1 C
DL floor               = 3625.0
DL power               = 10.0
     Temp K           DL       L W/(sr.m2) [with filter]
[[3.2315e+02 4.5710e+03 4.4507e+00]
 [3.7315e+02 5.1320e+03 8.3094e+00]
 [4.2315e+02 5.9060e+03 1.3496e+01]
 [4.7315e+02 6.8870e+03 1.9920e+01]
 [5.2315e+02 8.0340e+03 2.7452e+01]
 [5.7315e+02 9.3380e+03 3.5957e+01]
```

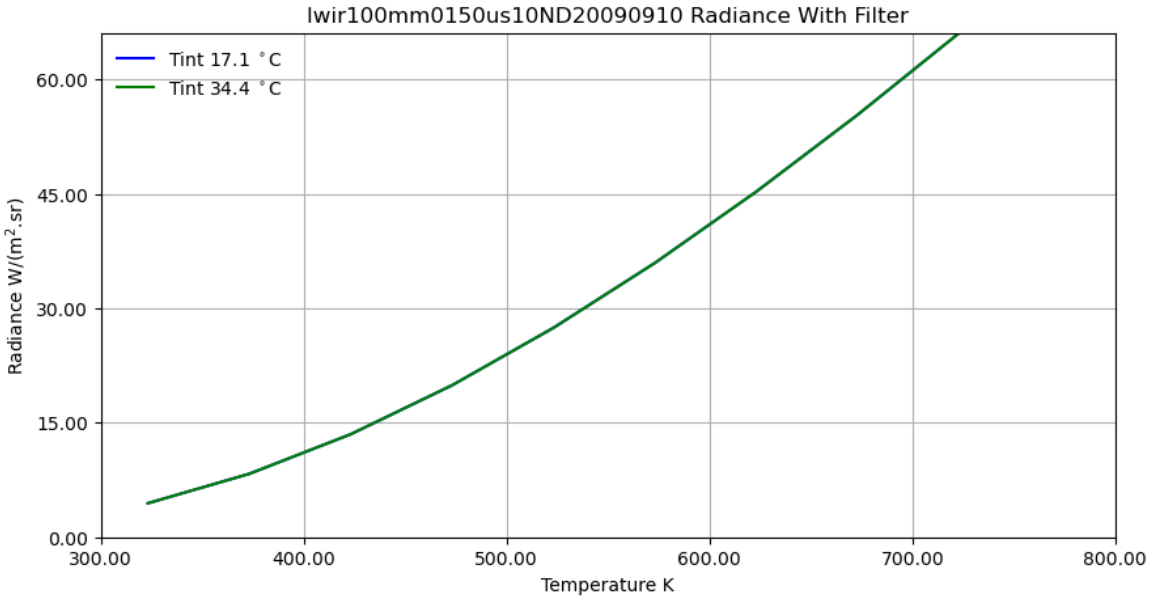
```
 [6.2315e+02 1.0834e+04 4.5307e+01]
 [6.7315e+02 1.2386e+04 5.5385e+01]
 [7.2315e+02 1.4042e+04 6.6092e+01]]
straight line fit L = 6.4889192972e-03 DL + -2.4902412790e+01

Instrument temperature = 34.4 C
DL floor               = 4210.0
DL power               = 10.0
     Temp K            DL        L W/(sr.m2) [with filter]
[[3.2315e+02 5.4770e+03 4.4507e+00]
 [3.7315e+02 6.0500e+03 8.3094e+00]
 [4.2315e+02 6.8170e+03 1.3496e+01]
 [4.7315e+02 7.7890e+03 1.9920e+01]
 [5.2315e+02 8.9220e+03 2.7452e+01]
 [5.7315e+02 1.0262e+04 3.5957e+01]
 [6.2315e+02 1.1694e+04 4.5307e+01]
 [6.7315e+02 1.3299e+04 5.5385e+01]
 [7.2315e+02 1.4921e+04 6.6092e+01]]
straight line fit L = 6.5071938467e-03 DL + -3.0916223162e+01
```

See Listing 2.10 for the code.



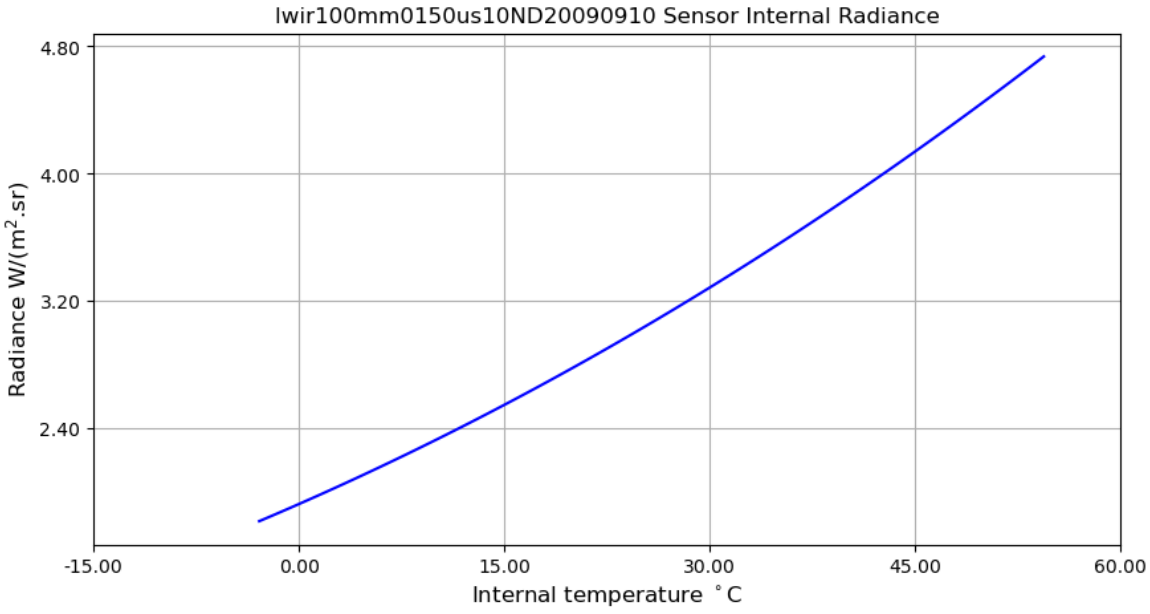
The following graph plots the source radiance versus source temperature, for the instrument spectral response, as defined in the input files.

See Listing 2.11 for the code.



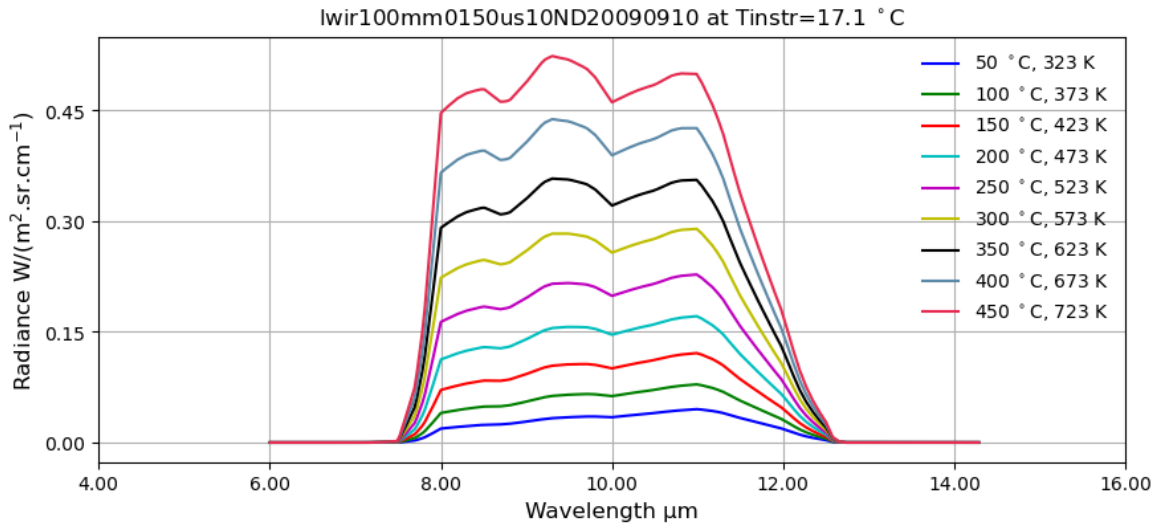
The radiance caused by the hot optics in the spectral band is as follows:

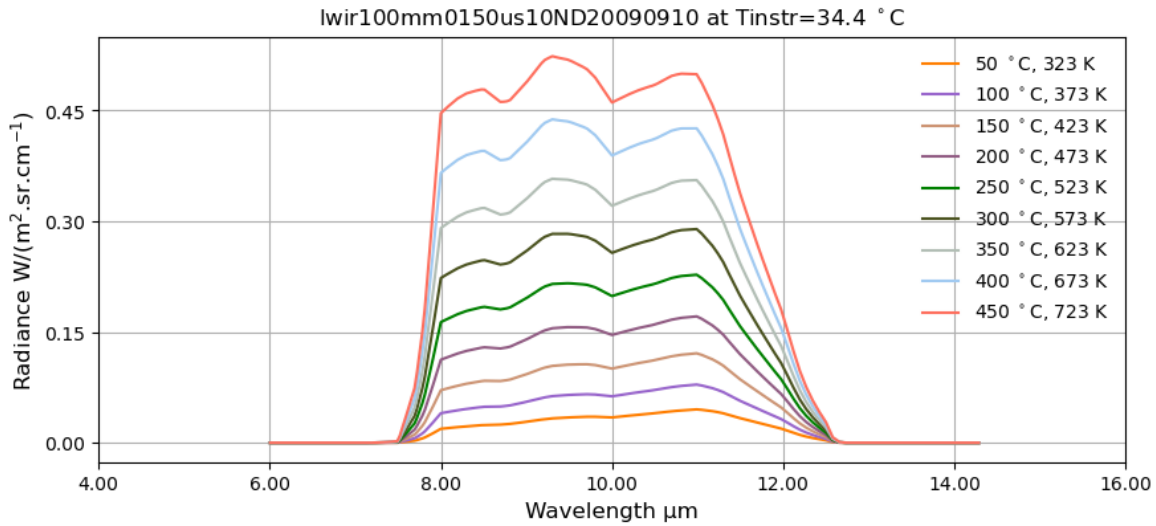
See Listing 2.12 for the code.



Plot the spectral radiance data for the temperatures used in the calibration process:

See Listing 2.13 for the code.





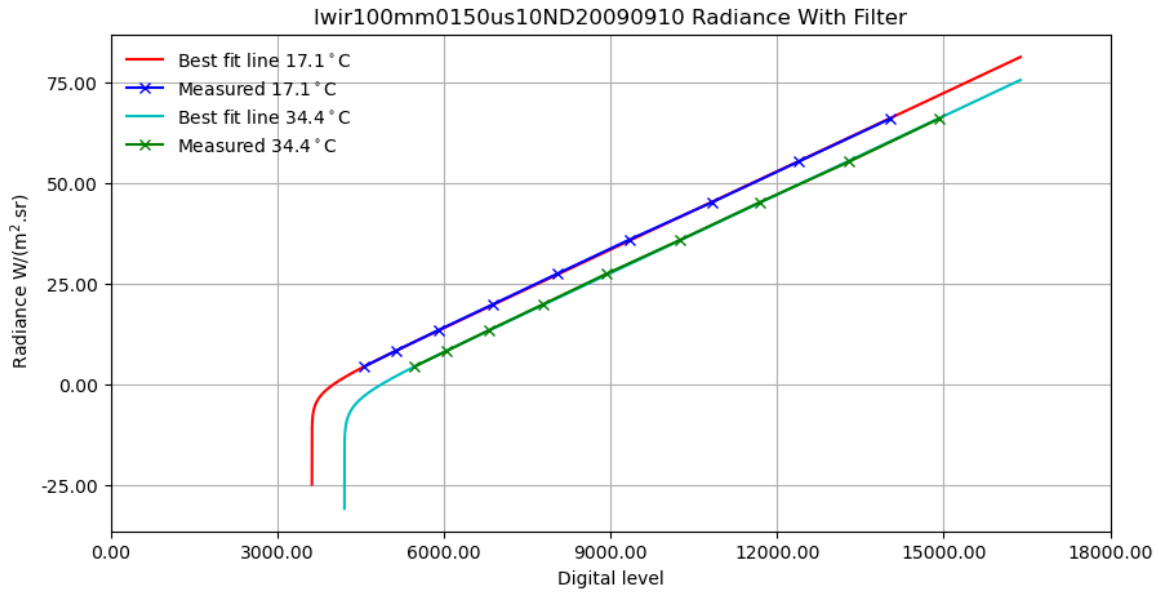
The instrument function is nominally a straight line, but flattens out at the low and high ends. At the low end, the (constant) radiance in the optics and instrument dominates the incoming flux on the detector, resulting in an asymptotic lower limit in digital level. At the top end, the detector or digitiser saturates, resulting in an asymptotic line at the maximum recordable level. In practice, the top-end asymptotic line is avoided simply by not using the instrument at these signal levels. The low-end asymptotic line is always present and must always be accounted for.

The low-end asymptotic line is defined by a set of two parameters in the input file (one set each for each instrument temperature):

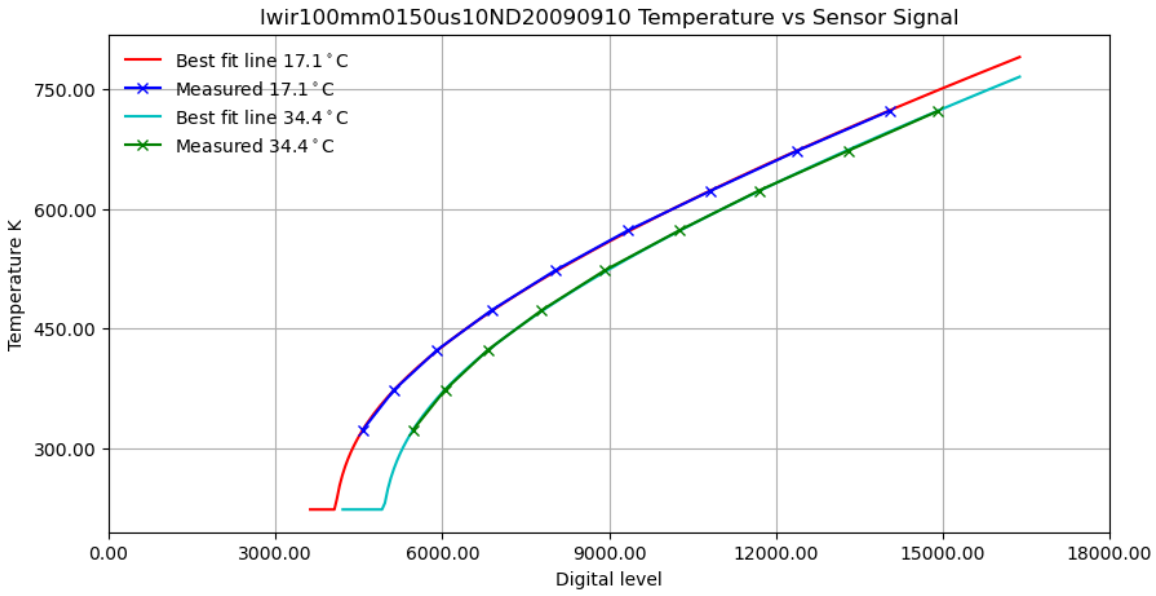
`JadeCalibration/Caldatas/Caldata@DlFloor`
`JadeCalibration/Caldatas/Caldata@Power`

These two parameters are not directly measurable and must be determined by an iterative process. This is done by entering estimates into the data file and then plotting the two graphs below, changing the values until you are satisfied with the graph. The `DlFloor` parameter determines the asymptotic limit, and the `Power` value determines the shape of the transition between the asymptotic line and the linear portion of the instrument function.

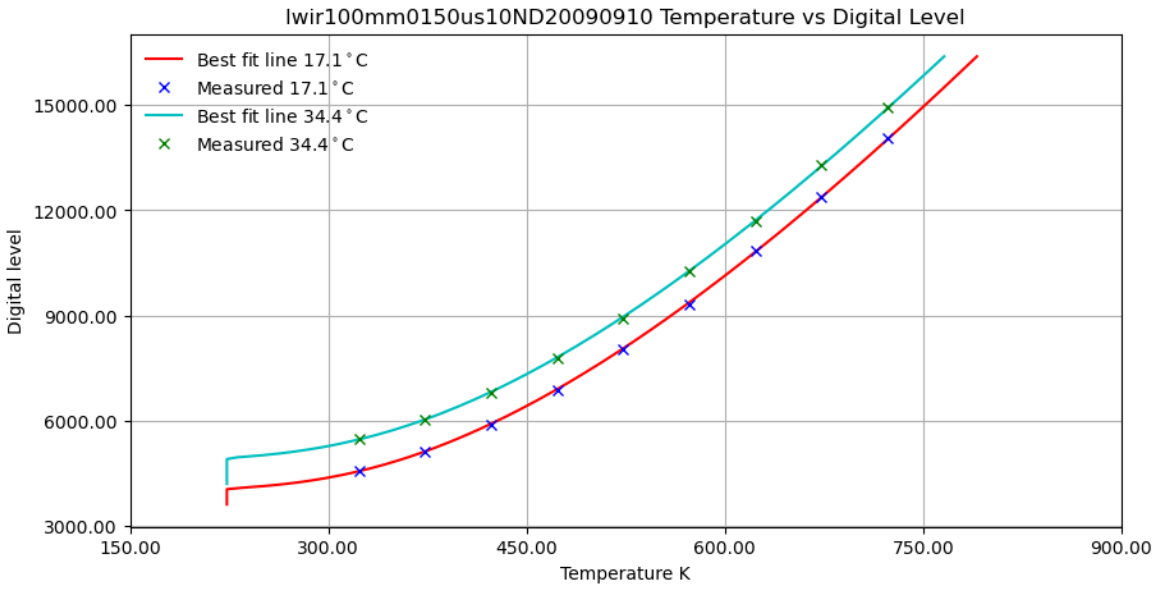
See Listing 2.14 for the code.



See Listing 2.15 for the code.



See Listing 2.16 for the code.



Finally, print out the calibration data and instrument function:

See Listing 2.17 for the code.

```
Lookup Name           = lwir100mm0150us10ND20090910
Sensor Response       = ./LWIRsensor.txt
Optics Transmittance = ./LW100mmLens.txt
Filter                = ./LWND10.txt
Source Emissivity     = ./Unity.txt
Atmospheric Transm    = ./Unity.txt
Nu (Min, Max, Inc)    = (700.0, 1665.0, 5.0)
Calibration data set

Instrument temperature = 17.1 C
DL floor               = 3625.0
DL power               = 10.0
     Temp K            DL        L W/(sr.m2) [with filter]
[[3.2315e+02 4.5710e+03 4.4507e+00]
 [3.7315e+02 5.1320e+03 8.3094e+00]
 [4.2315e+02 5.9060e+03 1.3496e+01]
 [4.7315e+02 6.8870e+03 1.9920e+01]
 [5.2315e+02 8.0340e+03 2.7452e+01]
 [5.7315e+02 9.3380e+03 3.5957e+01]
 [6.2315e+02 1.0834e+04 4.5307e+01]
 [6.7315e+02 1.2386e+04 5.5385e+01]
 [7.2315e+02 1.4042e+04 6.6092e+01]]
straight line fit L = 6.4889192972e-03 DL + -2.4902412790e+01

Instrument temperature = 34.4 C
DL floor               = 4210.0
DL power               = 10.0
     Temp K            DL        L W/(sr.m2) [with filter]
[[3.2315e+02 5.4770e+03 4.4507e+00]
 [3.7315e+02 6.0500e+03 8.3094e+00]
 [4.2315e+02 6.8170e+03 1.3496e+01]
 [4.7315e+02 7.7890e+03 1.9920e+01]
 [5.2315e+02 8.9220e+03 2.7452e+01]
 [5.7315e+02 1.0262e+04 3.5957e+01]
 [6.2315e+02 1.1694e+04 4.5307e+01]
 [6.7315e+02 1.3299e+04 5.5385e+01]
 [7.2315e+02 1.4921e+04 6.6092e+01]]
straight line fit L = 6.5071938467e-03 DL + -3.0916223162e+01
```

The two instrument functions (at low and high instrument temperatures) are known, the instrument function for any arbitrary instrument temperature can now be determined by interpolation (linear in this case).

The following code is used to verify digital-level-values for temperatures at the measured instrument temperatures and then demonstrates the interpolation of the instrument function at an intermediate instrument temperature.

See Listing 2.18 for the code.

```
Tint=17.1 DL=4571 T=50.0 L=4
Tint=17.1 DL=5132 T=99.8 L=8
Tint=17.1 DL=5906 T=149.1 L=13
Tint=17.1 DL=6887 T=199.0 L=20
Tint=17.1 DL=8034 T=248.6 L=27
Tint=17.1 DL=9338 T=298.5 L=36
Tint=17.1 DL=10834 T=350.5 L=45
Tint=17.1 DL=12386 T=400.4 L=55
Tint=17.1 DL=14042 T=450.6 L=66
---------------------------------------
Tint=34.4 DL=5477 T=50.1 L=4
Tint=34.4 DL=6050 T=100.4 L=8
Tint=34.4 DL=6817 T=149.2 L=13
Tint=34.4 DL=7789 T=198.8 L=20
Tint=34.4 DL=8922 T=248.0 L=27
Tint=34.4 DL=10262 T=299.5 L=36
Tint=34.4 DL=11694 T=349.3 L=45
Tint=34.4 DL=13299 T=401.1 L=56
Tint=34.4 DL=14921 T=450.4 L=66
---------------------------------------
Tint=25 DL=5477 T=95.6 L=8
Tint=25 DL=6050 T=134.3 L=12
Tint=25 DL=6817 T=176.5 L=17
Tint=25 DL=7789 T=221.8 L=23
Tint=25 DL=8922 T=268.2 L=30
Tint=25 DL=10262 T=317.6 L=39
Tint=25 DL=11694 T=366.0 L=48
Tint=25 DL=13299 T=416.6 L=59
Tint=25 DL=14921 T=465.0 L=69
```

1.5 Instrument function applied for temperature measurement (unity emissivity)

The following measurement observed a large-area blackbody in the lab. The blackbody set point was 150 °C and a recording made. The Cedip Altair s/w calculated 157 °C in the centre area of the source. The ryptw module calculates a temperature of 149 °C.

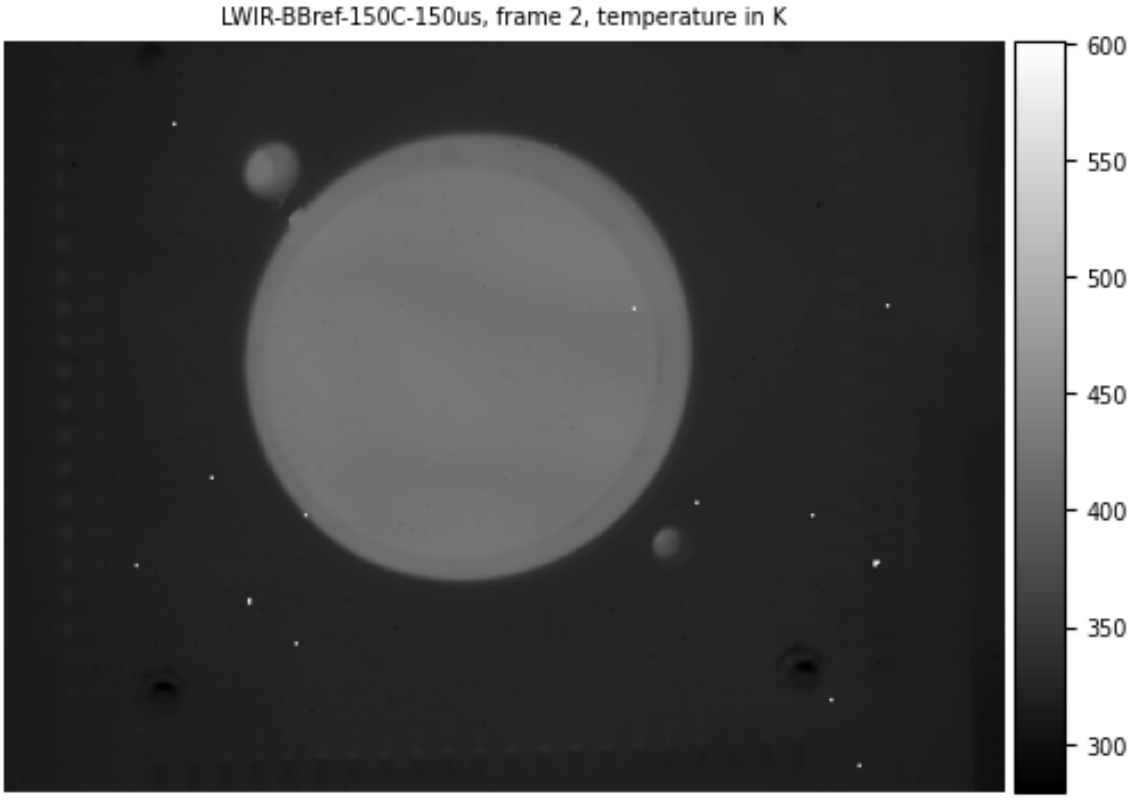
In this analysis we use the calibration file that was analysed above. The instrument (housing) temperature for the relevant frame is extracted from the header frame. Once the instrument internal temperature is known, it is used to interpolate a curve between the two extreme instrument functions. With the instrument function known at the operational temperature, the source temperature is calculated from the digital levels in the ptw file.

The ptw file may have nill data for the FPA and sensor4 temperatures, from the file description document it appears that a nill value means that it is 'disabled' (what ever that may mean). The example

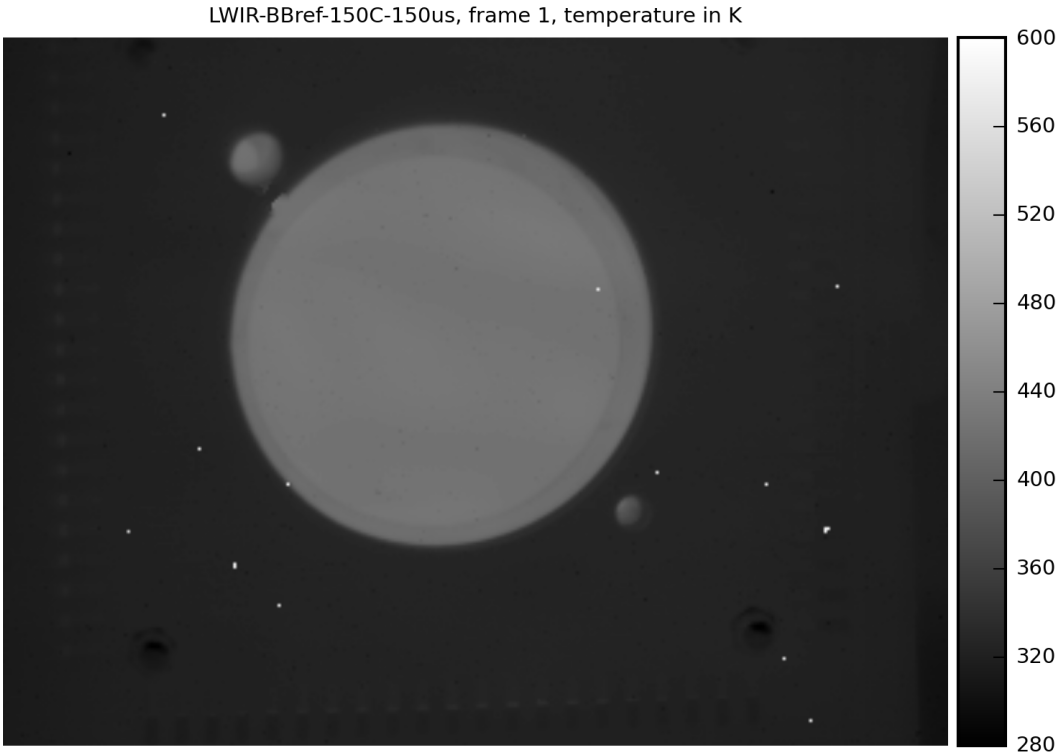
below has nill values.

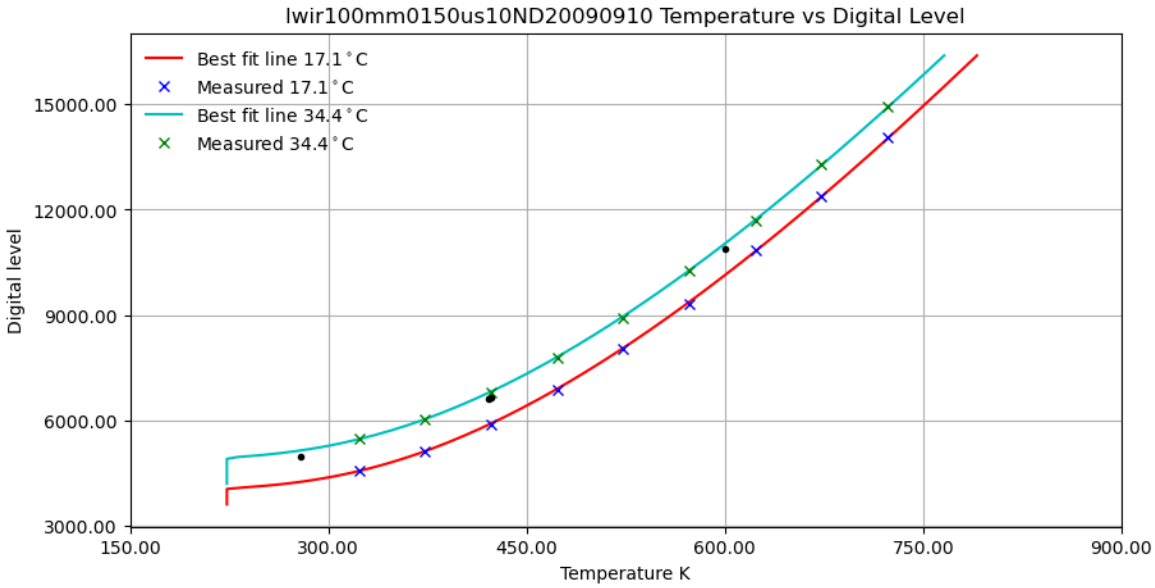
See Listing 2.19 for the code.

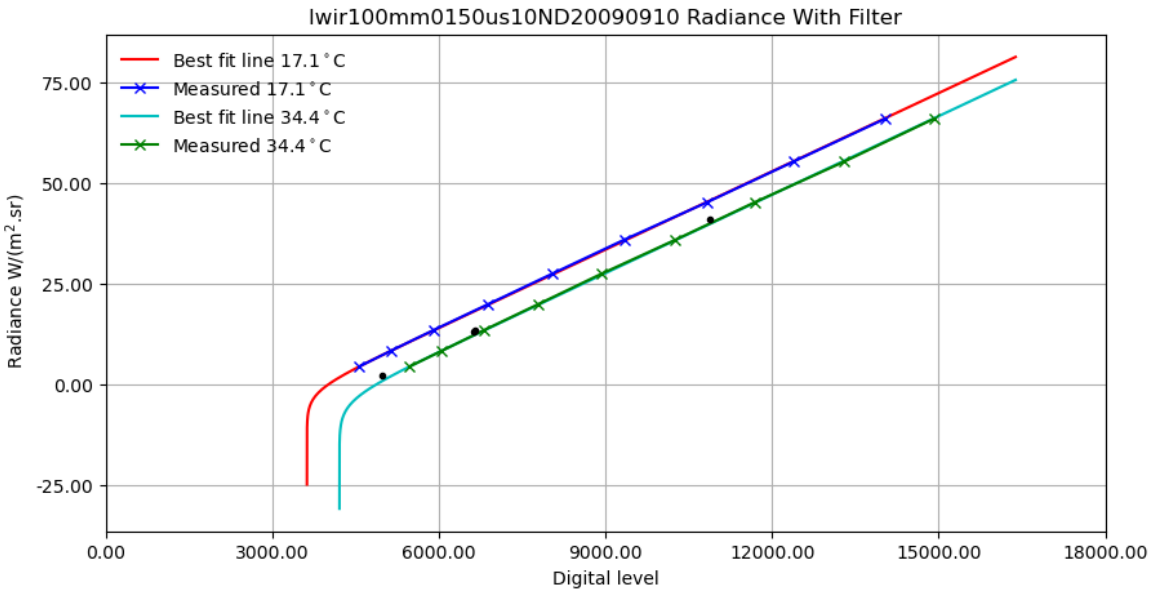
```
    Frame          Time  CamT4 C  CamT1 C  FPA K    Row    Col       DL  T ↵
       deg C    T K  L W/(m2.sr)
0     2.0  11:51:36.905   -273.1     31.2    0.0  199.0  256.0   4986.0      ↵
    5.9  279.1          2.16
0     2.0  11:51:36.905   -273.1     31.2    0.0  160.0  120.0   6618.0    ↵
    148.2  421.3         13.28
0     2.0  11:51:36.905   -273.1     31.2    0.0  140.0  110.0   6642.0    ↵
    149.5  422.7         13.44
0     2.0  11:51:36.905   -273.1     31.2    0.0  139.0   66.0  10873.0    ↵
    327.5  600.6         41.00
0     NaN           NaN       NaN       NaN    NaN    NaN    NaN       NaN      ↵
    NaN    NaN           NaN
```



See Listing 2.20 for the code display(Image(filename='LWIR-BBref-150C-150us-1.png')).


LWIR-BBref-150C-150us, frame 1, temperature in K
600
560
520
480
440
400
360
320
280


lwir100mm0150us10ND20090910 Temperature vs Digital Level
Best fit line 17.1 °C
Measured 17.1 °C
Best fit line 34.4 °C
Measured 34.4 °C
15000.00
12000.00
9000.00
6000.00
3000.00
Digital level
150.00
300.00
450.00
600.00
750.00
900.00
Temperature K



1.6 Instrument function applied for temperature measurement (spectral emissivity)

To follow.

1.7 Python and module versions, and dates

See Listing 2.23 for the code to get software versions.

```
Python implementation: CPython
Python version       : 3.9.15
IPython version      : 8.7.0

numpy : 1.21.5
scipy : 1.9.3
pyradi: 1.1.2

Compiler    : MSC v.1916 64 bit (AMD64)
OS          : Windows
Release     : 10
Machine     : AMD64
Processor   : Intel64 Family 6 Model 165 Stepping 2, GenuineIntel
CPU cores   : 16
Architecture: 64bit

Git hash: fb38ce73da5c38ed01c181a59c475645c5ecd512
```

2 LISTINGS

Listing 2.1: Code Listing in cell 6

```
from IPython.display import display
from IPython.display import Image
from IPython.display import HTML
import numpy as np
import pandas as pd

import pyradi.ryptw as ryptw
import pyradi.ryplot as ryplot
import pyradi.ryfiles as ryfiles
import pyradi.ryplanck as ryplanck
import matplotlib as mpl
mpl.rc("savefig", dpi=150)
mpl.rc('figure', figsize=(10,8))
%matplotlib inline
pd.set_option('display.width', 150)
```

Listing 2.2: Code Listing in cell 10

```
tgzFilename = 'pyradiSamplePtw.tgz'
destinationDir = '.'
tarFilename = 'pyradiSamplePtw.tar'
url = 'https://raw.githubusercontent.com/NelisW/pyradi/master/pyradi/data/'
dlNames = ryfiles.downloadUntar(tgzFilename, url, destinationDir, ↵
    tarFilename)
print('filesAvailable are {}'.format(dlNames))
```

Listing 2.3: Code Listing in cell 12

```
# first read the ptw file
ptwfile  = './PyradiSampleLWIR.ptw'
outfilename = 'PyradiSampleLWIR.txt'

header = ryptw.readPTWHeader(ptwfile)
ryptw.showHeader(header)
```

Listing 2.4: Code Listing in cell 14

```
#loading sequence of frames
framesToLoad = [3,4,10]
data, fheaders = ryptw.getPTWFrames (header, framesToLoad)
print(data.shape)
print(ptwfile)
print('(1) Data shape={}'.format(data.shape))
ifrm = 0
print('Frame {} summary data:'.format(ifrm))
print('Image data:\n{}'.format(data[ifrm]))
print('Sensor Temperature4 {} K '.format(fheaders[ifrm].h_sensorTemp4))
print('FPA Temperature {} K '.format(fheaders[ifrm].h_detectorTemp))
print('Time {}:{}:{} '.format(fheaders[ifrm].h_frameHour,
        fheaders[ifrm].h_frameMinute,fheaders[ifrm].h_frameSecond))
```

Listing 2.5: Code Listing in cell 16

```
#loading sequence of frames, with an error in request
```

```
framesToLoad = [0,4,10]
data, fheaders = ryptw.getPTWFrames (header, framesToLoad)
print(data.shape)
```

Listing 2.6: Code Listing in cell 18

```
# loading single frames
framesToLoad = list(range(1, 101, 1))
frames = len(framesToLoad)
data, fheaders = ryptw.getPTWFrame (header, framesToLoad[0])
for frame in framesToLoad[1:]:
    f, fheaders  = ryptw.getPTWFrame (header, frame)
    data = np.concatenate((data, f))

rows = header.h_Rows
cols = header.h_Cols
img = data.reshape(frames, rows ,cols)
print(img.shape)
```

Listing 2.7: Code Listing in cell 23

```
display(Image(filename='images/radiometry08.PNG'))
```

Listing 2.8: Code Listing in cell 26

```
calData = ryptw.JadeCalibrationData('./lwir100mm150us10ND20090910.xml', '.')
```

Listing 2.9: Code Listing in cell 33

```
print(calData.Info())
```

Listing 2.10: Code Listing in cell 34

```
calData.LU.PlotSpectrals()
# HTML('<img src="images/lwir100mm0150us10ND20090910-spectrals.png" width↵
    =800 />')
# display(Image(filename='lwir100mm0150us10ND20090910-spectrals.png'))
```

Listing 2.11: Code Listing in cell 36

```
calData.LU.PlotCalTempRadiance()
# HTML('<img src="images/lwir100mm0150us10ND20090910-CalTempRadiance.png" ↵
    width=800 />')
# display(Image(filename='lwir100mm0150us10ND20090910-CalTempRadiance.png'))
```

Listing 2.12: Code Listing in cell 38

```
calData.LU.PlotCalTintRad()
# HTML('<img src="images/lwir100mm0150us10ND20090910-CalInternal.png" width↵
    =800/>')
# display(Image(filename='lwir100mm0150us10ND20090910-CalInternal.png'))
```

Listing 2.13: Code Listing in cell 40

```
calData.LU.PlotCalSpecRadiance()
# HTML('<img src="images/lwir100mm0150us10ND20090910-CalRadiance.png" width↵
    =800/>')
# display(Image(filename='lwir100mm0150us10ND20090910-CalRadiance.png'))
```

Listing 2.14: Code Listing in cell 42

```
calData.LU.PlotCalDLRadiance()
# HTML('<img src="images/lwir100mm0150us10ND20090910-CaldlRadiance.png" ↵
   width=800/>')
# display(Image(filename='lwir100mm0150us10ND20090910-CaldlRadiance.png'))
```

Listing 2.15: Code Listing in cell 43

```
calData.LU.PlotCalDLTemp()
# HTML('<img src="images/lwir100mm0150us10ND20090910-CalDLTemp.png" width↵
   =800 />')
# display(Image(filename='lwir100mm0150us10ND20090910-CalDLTemp.png'))
```

Listing 2.16: Code Listing in cell 44

```
calData.LU.PlotCalTempDL()
# HTML('<img src="images/lwir100mm0150us10ND20090910-CalTempDL.png" width↵
   =800 />')
# display(Image(filename='lwir100mm0150us10ND20090910-CalDLTemp.png'))
```

Listing 2.17: Code Listing in cell 46

```
print(calData.LU.Info())
```

Listing 2.18: Code Listing in cell 48

```
for Tint in [17.1]:
  for DL in [4571, 5132, 5906, 6887, 8034, 9338, 10834, 12386, 14042 ]:
    print('Tint={} DL={} T={:.1f} L={:.0f}'.format(Tint, DL, calData.LU.↵
       LookupDLTemp(DL, Tint)-273.15, calData.LU.LookupDLRad(DL, Tint)))
print('---------------------------------------- ')
for Tint in [34.4]:
  for DL in [5477, 6050, 6817, 7789, 8922, 10262, 11694, 13299, 14921 ]:
    print('Tint={} DL={} T={:.1f} L={:.0f}'.format(Tint, DL, calData.LU.↵
       LookupDLTemp(DL, Tint)-273.15, calData.LU.LookupDLRad(DL, Tint)))
print('---------------------------------------- ')
for Tint in [25]:
  for DL in [5477, 6050, 6817, 7789, 8922, 10262, 11694, 13299, 14921 ]:
    print('Tint={} DL={} T={:.1f} L={:.0f}'.format(Tint, DL, calData.LU.↵
       LookupDLTemp(DL, Tint)-273.15, calData.LU.LookupDLRad(DL, Tint)))
```

Listing 2.19: Code Listing in cell 51

```
import os.path

#load the calibration file and calculate the instrument functon
calData = ryptw.JadeCalibrationData('./lwir100mm150us10ND20090910.xml', '.')

#load the ptw file header
ptwfile  = 'LWIR-BBref-150C-150us.ptw'
header = ryptw.readPTWHeader(ptwfile)
# showHeader(header)

dfpts = pd.DataFrame()

#loading sequence of frames
framesToLoad = [1,2]
for frame in framesToLoad:
    data,fheader = ryptw.getPTWFrame (header, frame)
    strTime ='{}:{}:{}'.format(fheader.h_frameHour,
```

```
            fheader.h_frameMinute,fheader.h_frameSecond)
    columns=['Frame','Time','CamT4 C','CamT1 C','FPA K','Row','Col','DL','T ↵
       deg C','T K','L W/(m2.sr)']


#     print('Sensor Temperature4 {} K '.format(fheader.h_sensorTemp4))
#     print('FPA Temperature {} K '.format(fheader.h_detectorTemp))
#     print('Time {}:{}:{} '.format(fheader.h_frameHour,
#             fheader.h_frameMinute,fheader.h_frameSecond))


    #get the internal temperature from the header and use here
    tempIm = calData.LU.LookupDLTemp(data, header.h_HousingTemperature1↵
       -273.15)
    # get a radiance image
    radIm = calData.LU.LookupDLRad(data, header.h_HousingTemperature1↵
       -273.15)

#     print('Instrument temperature during measurement was {:.2f} C'.format(↵
   header.h_HousingTemperature1-273.15))

#     print('Temperature at ({},{})={} C'.format(160,120,tempIm↵
   [160,120]-273.15))
#     print('Temperature at ({},{})={} C'.format(140,110,tempIm↵
   [140,110]-273.15))
    ldfpts = []
    for coords in [np.argwhere(np.min(radIm)==radIm)[0],[160,120],[140,110],↵
       np.argwhere(np.max(radIm)==radIm)[0]]:
        df2 = pd.DataFrame([[frame,strTime,
                             round(fheader.h_sensorTemp4-273.15,1),
                             round(header.h_HousingTemperature1-273.15,1),
                             round(fheader.h_detectorTemp,1),
                             coords[0],coords[1],int(data[coords[0],coords↵
                                [1]]),
                             round(tempIm[coords[0],coords[1]]-273.15,1),
                             round(tempIm[coords[0],coords[1]],1),
                             round(radIm[coords[0],coords[1]],2),
                            ]],
                           columns=columns)
        ldfpts.append(df2)

    ldfpts.append(pd.DataFrame([len(columns)*[np.NaN]],columns=columns))
    dfpts = pd.concat(ldfpts)

    I = ryplot.Plotter(4, 1, 1,'', figsize=(6,6))
    I.showImage(1, tempIm, ptitle='{}, frame {}, temperature in K'.format(↵
       ptwfile[:-4],frame), titlefsize=7, cbarshow=True, cbarfontsize=7)
#     I.saveFig('{}-{}.png'.format(os.path.basename(ptwfile)[:-4],frame))

print(dfpts)
```

Listing 2.20: Code Listing in cell 52

```
# display(Image(filename='LWIR-BBref-150C-150us-1.png'))
HTML('<img src="images/LWIR-BBref-150C-150us-1.png" width=800 />')
```

Listing 2.21: Code Listing in cell 53

```
##
pts = dfpts[['T K','DL']].values.T
```

```
calData.LU.PlotCalTempDL(plotPoints=pts)
```

Listing 2.22: Code Listing in cell 54

```
##
pts = dfpts[['DL','L W/(m2.sr)']].values.T
calData.LU.PlotCalDLRadiance(plotPoints=pts)
```

Listing 2.23: Code Listing in cell 58

```
# to get software versions
# https://github.com/rasbt/watermark
# you only need to do this once
# pip install watermark

%load_ext watermark
%watermark -v -m -p numpy,scipy,pyradi -g
```

BIBLIOGRAPHY

[1] [Online]. Available: https://github.com/NelisW/ComputationalRadiometry#computational-optical-radiometry-with-pyradi

[2] [Online]. Available: https://github.com/NelisW/ComputationalRadiometry#computational-optical-radiometry-with-pyradi

[3] [Online]. Available: http://dx.doi.org/10.5281/zenodo.9910

[4] [Online]. Available: http://spie.org/Publications/Book/2021423?origin_id=x646