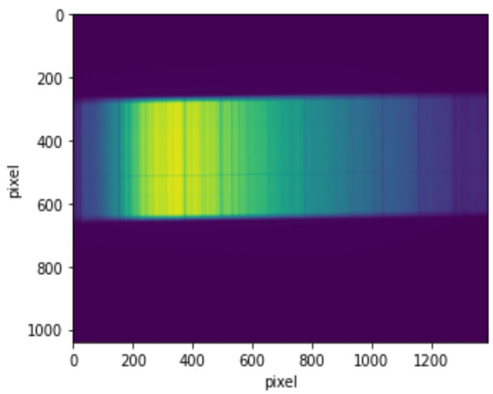
Import all data

```
Light Frame Light Frame Dark Frame Bias Frame

Light Frame Flat Field Dark Frame Bias Frame
```

Out[ ]: <matplotlib.image.AxesImage at 0x1ca8736d580>



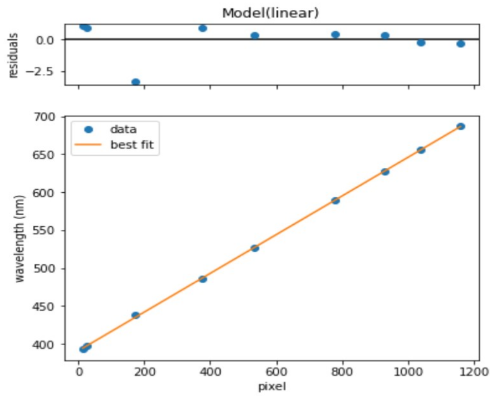
Out[ ]: <matplotlib.image.AxesImage at 0x1caa2269d30>

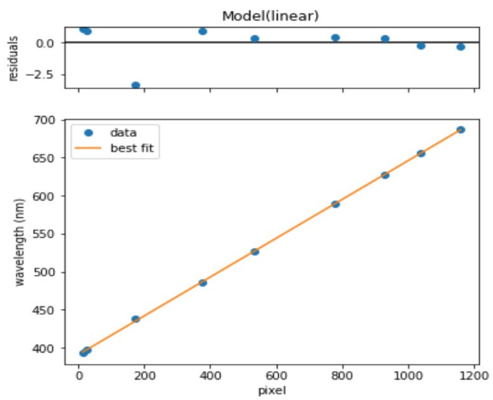
Calibration

With Fraunhofer

```
<Parameter 'slope', value=0.25525659690182534 +/- 0.00116, bounds=[-inf:inf]>
<Parameter 'intercept', value=390.82847973147994 +/- 0.809, bounds=[-inf:inf]>
```

Out[ ]:





Out[ ]:

Model

Model(linear)

Fit Statistics

fitting method	leastsq
# function evals	6
# data points	9
# variables	2
chi-square	14.5750765
reduced chi-square	2.08215378
Akaike info crit.	8.33879563
Bayesian info crit.	8.73324478

Variables

name	value	standard error	relative error	initial value	min	max	vary
slope	0.25525660	0.00116398	(0.46%)	1.0	-inf	inf	True

intercept	390.828480	0.80904219	(0.21%)	0.0	-inf	inf	True

Correlations (unreported correlations are < 0.100)

Single fit file case

`np.median(2darray)` gives the median of all numbers in the 2darray.

`np.median(2darray, axis=0)` gives the medians of the numbers alongside the columns.

`np.median(2darray, axis=1)` gives the medians of the numbers alongside the rows.

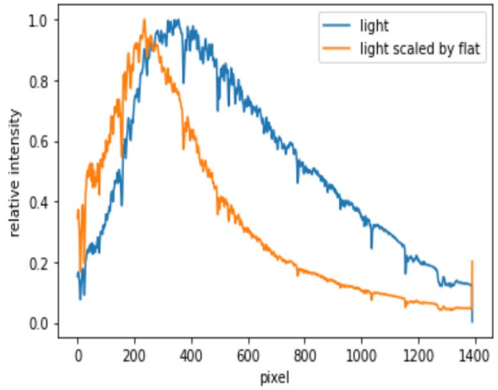
`np.sum(2darray)` gives the sum of all numbers in the 2darray.

`np.sum(2darray, axis=0)` gives the sums of the numbers alongside the columns.

`np.sum(2darray, axis=1)` gives the sums of the numbers alongside the rows.

```
<class 'numpy.ndarray'> 1039 <class 'numpy.ndarray'>
<class 'numpy.ndarray'> 1391 <class 'numpy.float64'>
<class 'numpy.ndarray'> 1391 <class 'numpy.float64'>
<class 'numpy.ndarray'> 1391 <class 'numpy.float64'>
<class 'numpy.ndarray'> 1391 <class 'numpy.float64'>
```

Out[ ]: <matplotlib.legend.Legend at 0x1caa22a7820>

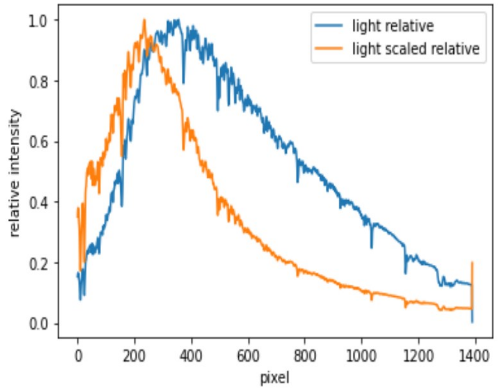


Stacking file case

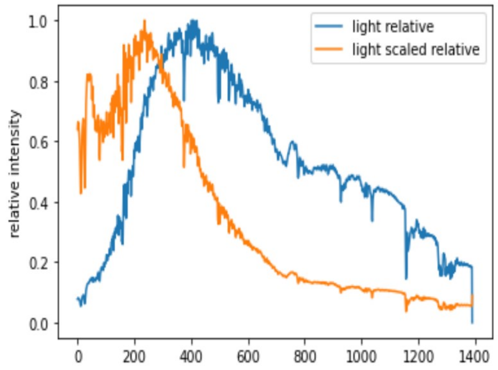
This is not much different; the `flat_stack` and `light_stack` act just like `flat` and `light` above.

```
<class 'numpy.ndarray'> 1039 <class 'numpy.ndarray'>
<class 'numpy.ndarray'> 1039 <class 'numpy.ndarray'>
<class 'numpy.ndarray'> 1391 <class 'numpy.float64'>
<class 'numpy.ndarray'> 1391 <class 'numpy.float64'>
```

Out[ ]: <matplotlib.legend.Legend at 0x1caa2712af0>



Out[ ]: <matplotlib.legend.Legend at 0x1caa279bf40>

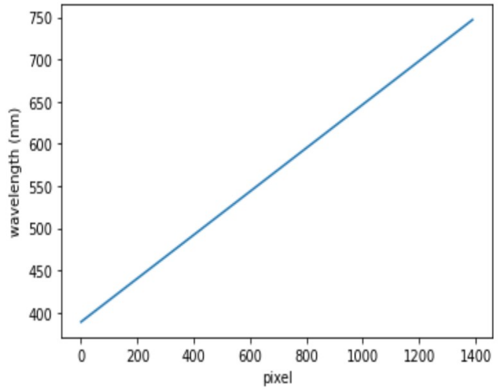


Neon calibration

Convert to wavelengths

Wavelength -- pixel relation

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1caa2848f40>]
```

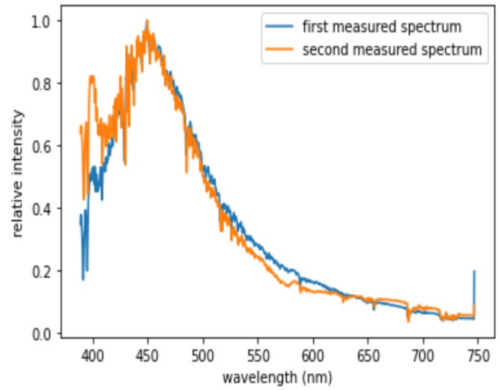


Crude method to determine effective temperature.

```
4.4940945446e-07
6447.955035752187
11.710933% deviation from the accepted value.
```

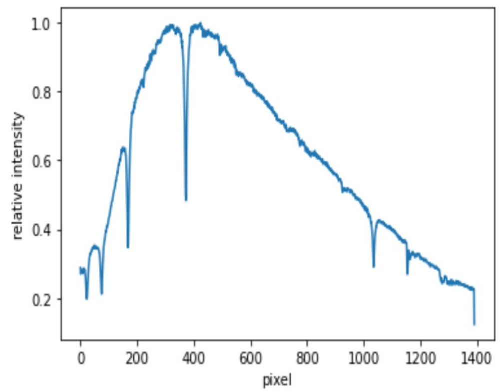
```
4.49667005296e-07
6444.261911305897
11.646949% deviation from the accepted value.
```

Out[ ]: <matplotlib.legend.Legend at 0x1caa287ea60>

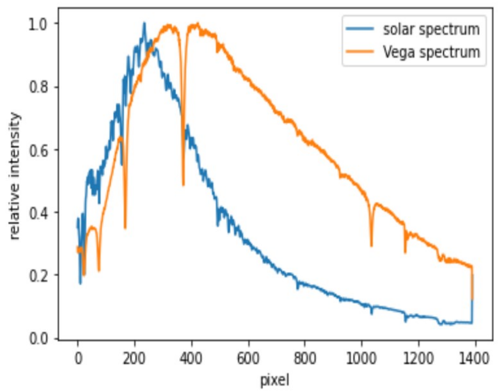


Vega

Out[ ]: [<matplotlib.lines.Line2D at 0x1caa2920d60>]

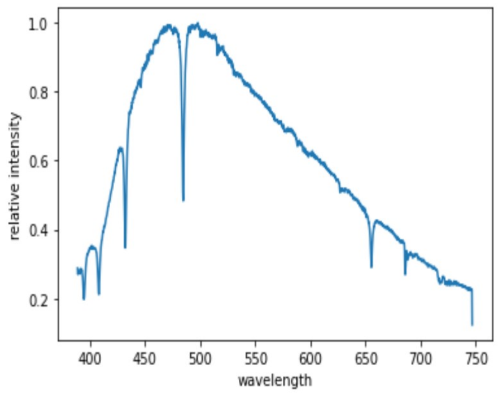


Out[ ]:



Vega with wavelength

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1caa2a239d0>]
```

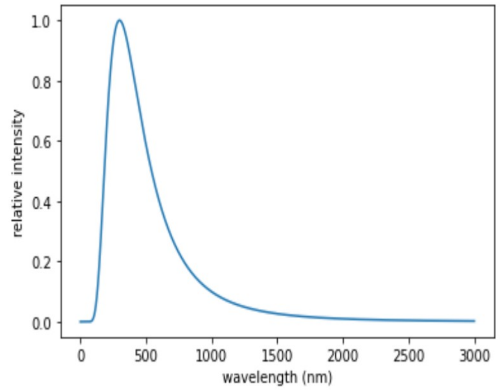


crude method

```
4.980865624640001e-07
5817.807934156908
```

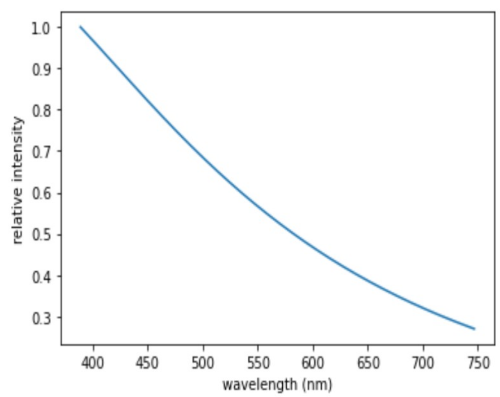
```
C:\Users\bvptr\AppData\Local\Temp/ipykernel_1376/2665269975.py:30: RuntimeWarning: overflow encountered in e
xp
  ( (wavelength**5) * ( np.exp( ( sc.Planck * sc.speed_of_light) / ( wavelength * sc.k * temperature ) ) - 1
) ) )
```

Out[ ]: `[<matplotlib.lines.Line2D at 0x1caa2ac23d0>]`




```
4.980865624640001e-07
5817.807934156908
-40.022599% deviation from the accepted value.
```

Out[ ]: `[<matplotlib.lines.Line2D at 0x1caa2b125e0>]`

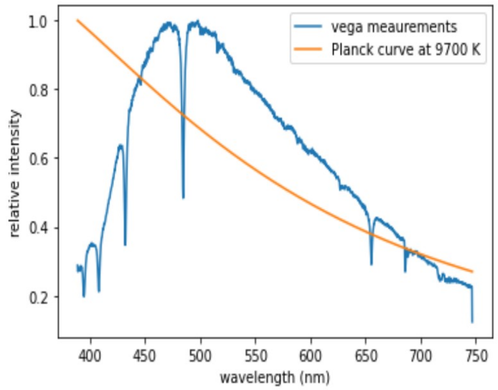


switching the coefficient around changes the spectrum significantly.

Perhaps we need to calculate the relative coefficient.

```
[3.47880251 3.69892729 3.64840053 ... 1.20492438 1.19465939 2.18259458]
```

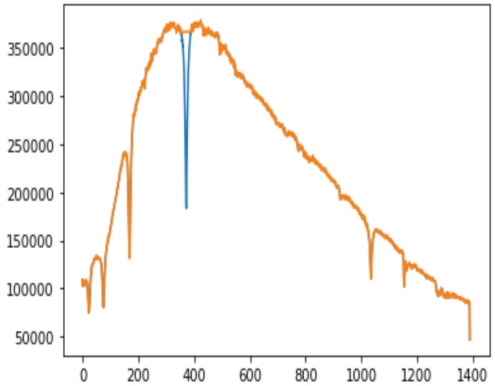
Out[ ]:
```
<matplotlib.legend.Legend at 0x1caa2b37850>
```

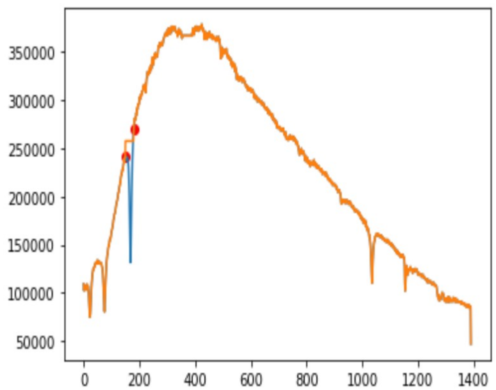


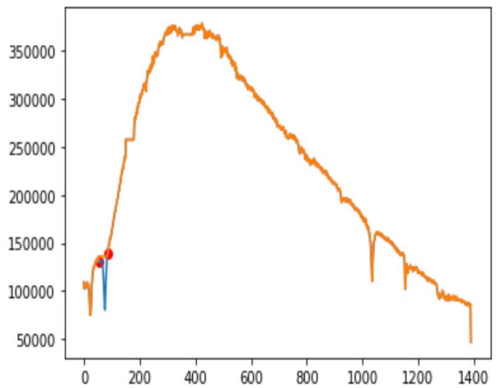
We now eliminate the absorption lines of Vega

so that it does not mess up the shifting of the solar spectrum.

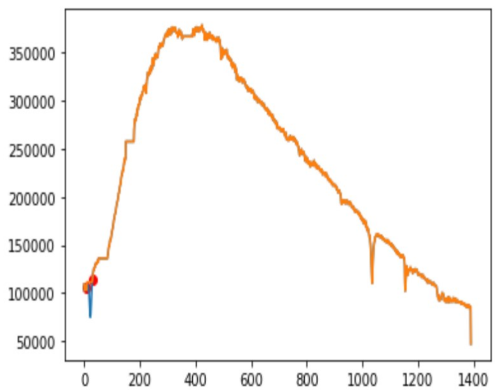
Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1caa2bdddf0>]
```



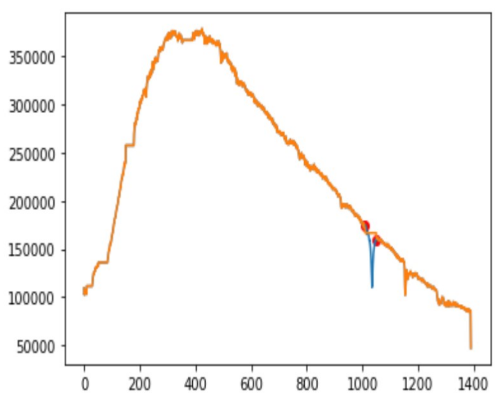

350000
300000
250000
200000
150000
100000
50000
0
200
400
600
800
1000
1200
1400



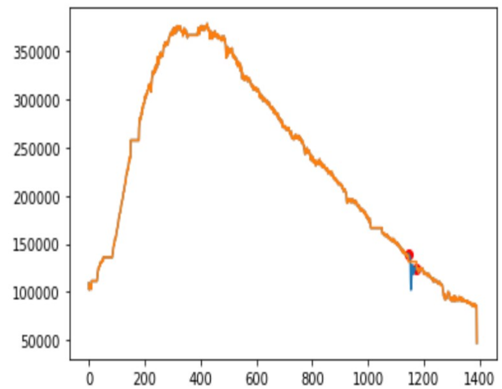
Out[ ]: [<matplotlib.lines.Line2D at 0x1caa5c739d0>]



Out[ ]: [<matplotlib.lines.Line2D at 0x1caa5cea340>]



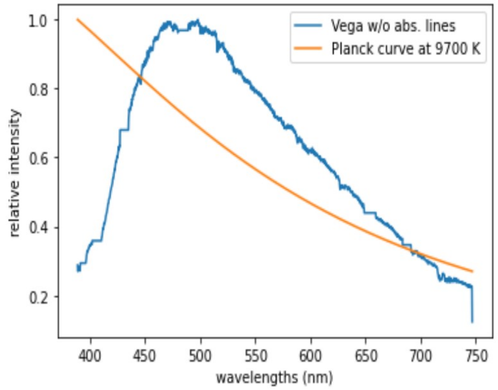
Out[ ]:


350000
300000
250000
200000
150000
100000
50000
0
200
400
600
800
1000
1200
1400

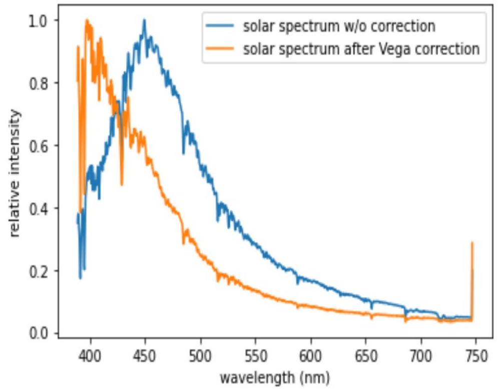
Do the same thing we did before but with `vega_sum_no_abs`

```
Out[ ]: array([3.47880251, 3.69892729, 3.64840053, ..., 1.20492438, 1.19465939,
               2.18259458])
```

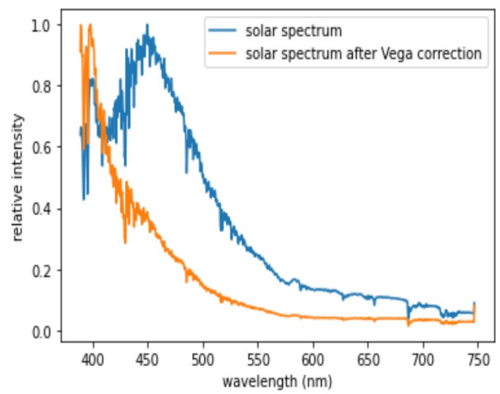
```
Out[ ]: <matplotlib.legend.Legend at 0x1caa5dbefd0>
```




```
Out[ ]: <matplotlib.legend.Legend at 0x1caa5df0190>
```



```
Out[ ]: <matplotlib.legend.Legend at 0x1caa613f880>
```



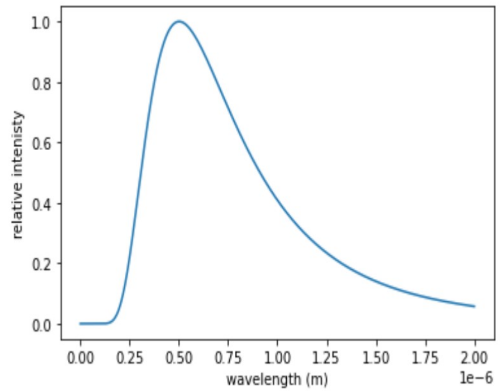

```
3.9738418558800007e-07
7292.116949023109
26.336052% deviation from the accepted value.

3.978992872600001e-07
7282.676917957139
26.172504% deviation from the accepted value.
```

Plotting the planck curve of the sun

```
C:\Users\bvptr\AppData\Local\Temp/ipykernel_1376/2665269975.py:30: RuntimeWarning: overflow encountered in e
xp
  ( (wavelength**5) * ( np.exp( ( sc.Planck * sc.speed_of_light) / ( wavelength * sc.k * temperature ) ) - 1
) ) )
```

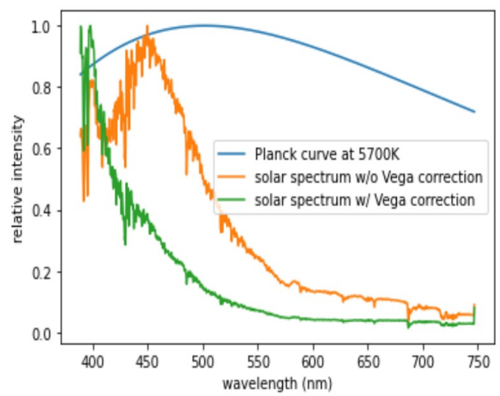
Out[ ]:

```
[<matplotlib.lines.Line2D at 0x1caa6212df0>]
```



Out[ ]:

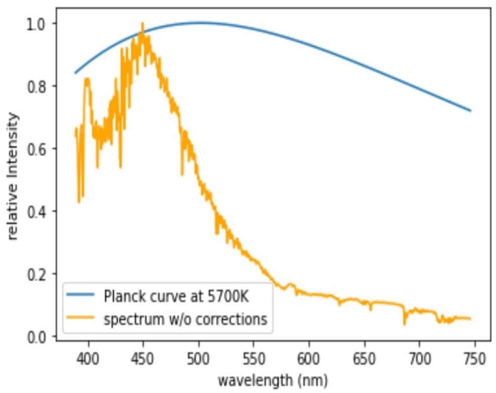
```
<matplotlib.legend.Legend at 0x1caa623aa60>
```



Correcting for Rayleigh scattering and final results

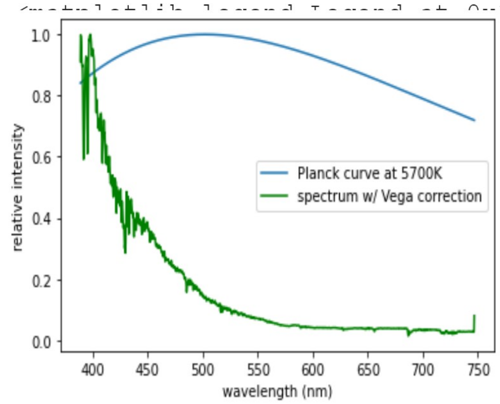
Out[ ]:

```
<matplotlib.legend.Legend at 0x1caa62a59d0>
```

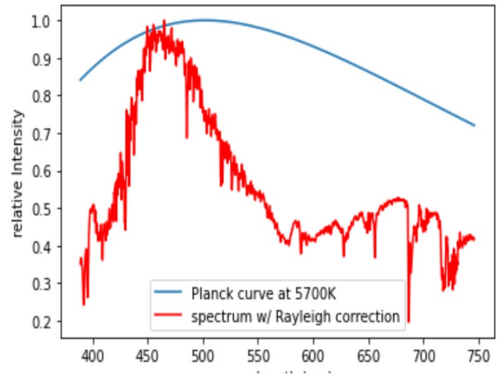



```
Out[ ]:
```

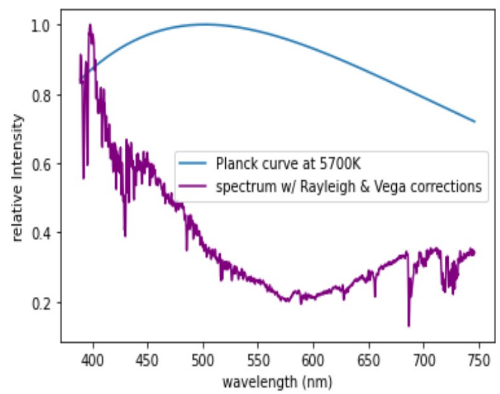
<matplotlib.legend.Legend at 0x1caa634cb20>



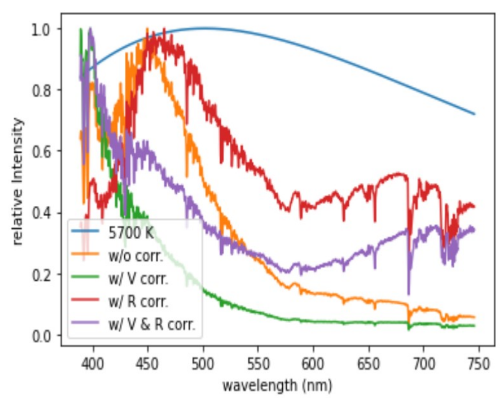
Out[ ]: <matplotlib.legend.Legend at 0x1caa63b8640>



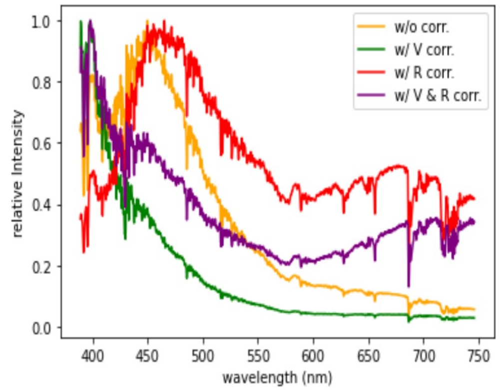
Out[ ]: <matplotlib.legend.Legend at 0x1caa6f41d60>



Out[ ]: <matplotlib.legend.Legend at 0x1caa6fbd2b0>



Out[ ]: <matplotlib.legend.Legend at 0x1caa70409a0>




```
4.651200554560001e-07
6230.159119152681
7.937615% deviation from the accepted value.

3.978992872600001e-07
7282.676917957139
26.172504% deviation from the accepted value.
```

Trying to fit

What we can do is remove all the constants from the fit; for we are only interested in the relative intensities.

$$\frac{a}{\lambda^5\left(e^{\frac{b}{\lambda T}}-1\right)}$$

Out[ ]:

Model

Model(fit)

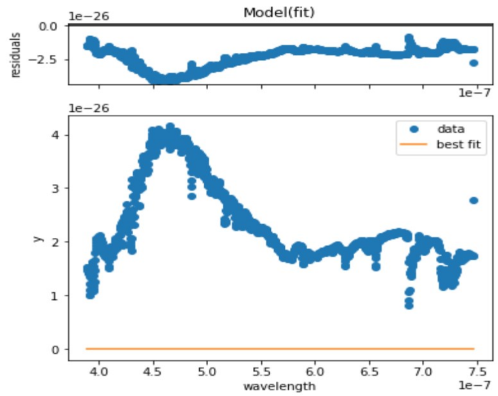
Fit Statistics

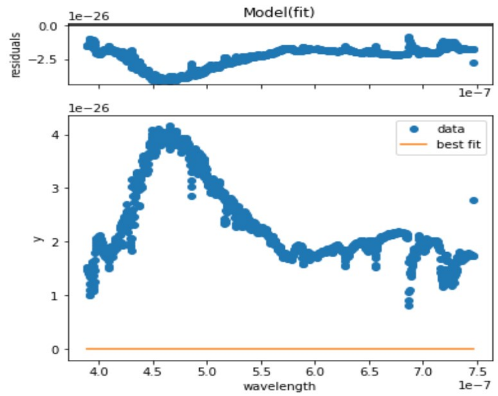
fitting method	leastsq
# function evals	2
# data points	1391
# variables	1
chi-square	8.2537e-49
reduced chi-square	5.9379e-52
Akaike info crit.	-164071.712
Bayesian info crit.	-164066.474

Variables

name	value	initial value	min	max	vary
temperature	5772.00000	5772	-inf	inf	True

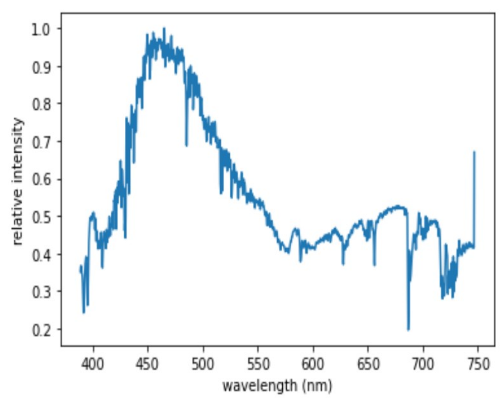
Out[ ]:



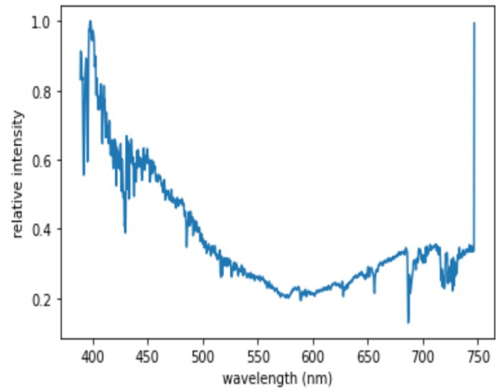


To mathematica

Out[ ]: [<matplotlib.lines.Line2D at 0x1caa9f819d0>]



Out[ ]: [<matplotlib.lines.Line2D at 0x1caa9fbe490>]



Multiple planck curves

To look which one is *the best*

Out[ ]: <matplotlib.legend.Legend at 0x1caa9ff8c70>