In [2]:

```
# aa_spectraltool
# 2017 francesco.anselmo@aaschool.ac.uk
```

In [3]:

```
%matplotlib inline
```

In [51]:

```
# import libraries
import colour
import colour.plotting
import colour.io
from colour.plotting import *
import PIL
import numpy as np
from numpy import array, zeros, linspace, float64, savetxt
from scipy import interpolate, vectorize
from scipy.signal import argrelextrema, argrelmax, savgol_filter, resample
from sklearn.preprocessing import normalize
import matplotlib.pyplot as plt
import pylab
from itertools import izip
import csv
import warnings
warnings.filterwarnings("ignore")
```

In [30]:

```
img = PIL.Image.open("DSC_0064.JPG") # EDIT THIS: change name of your file
```

In [31]:

```
# show image format, size and colour mode
print(img.format, img.size, img.mode)
```

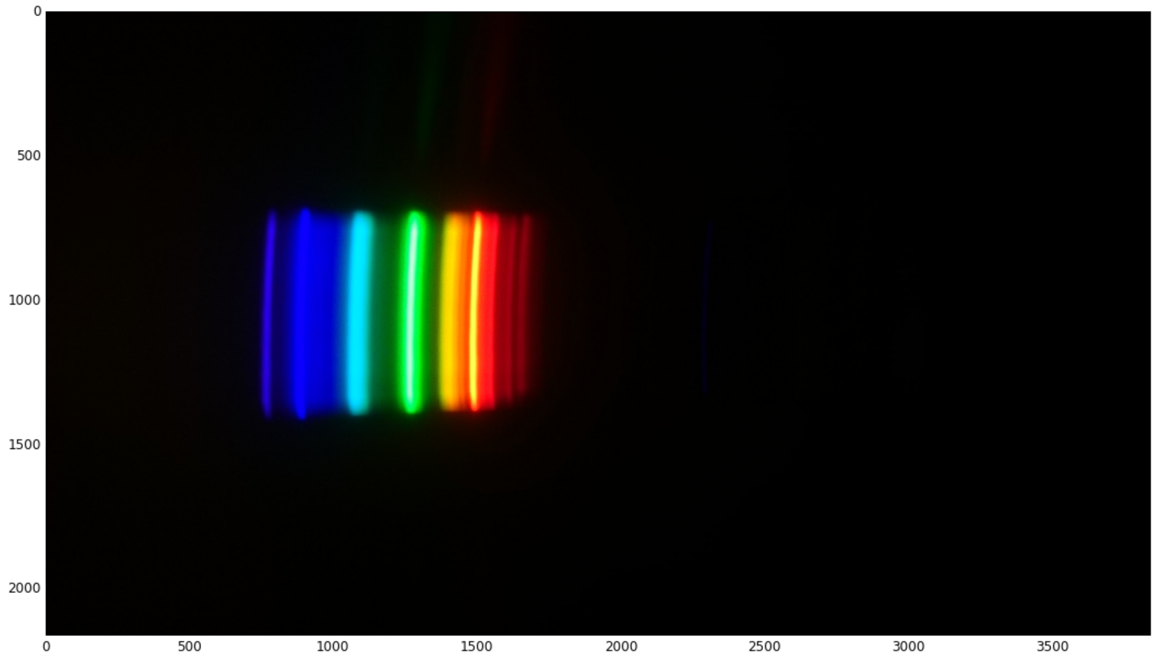
```
JPEG (3840, 2160) RGB
```

In [32]:

```
# show image - ensure that the image has the blue part of the spectrum on the le
ft
# if not, rotate it with an image editing software, save it and import it again
plt.imshow(np.asarray(img))
```

Out[32]:

```
<matplotlib.image.AxesImage at 0x13034b898>
```



In [33]:

```
# convert image to grey scale
greyscale_img = img.convert("LA")
print(greyscale_img.format, greyscale_img.size, greyscale_img.mode)
```

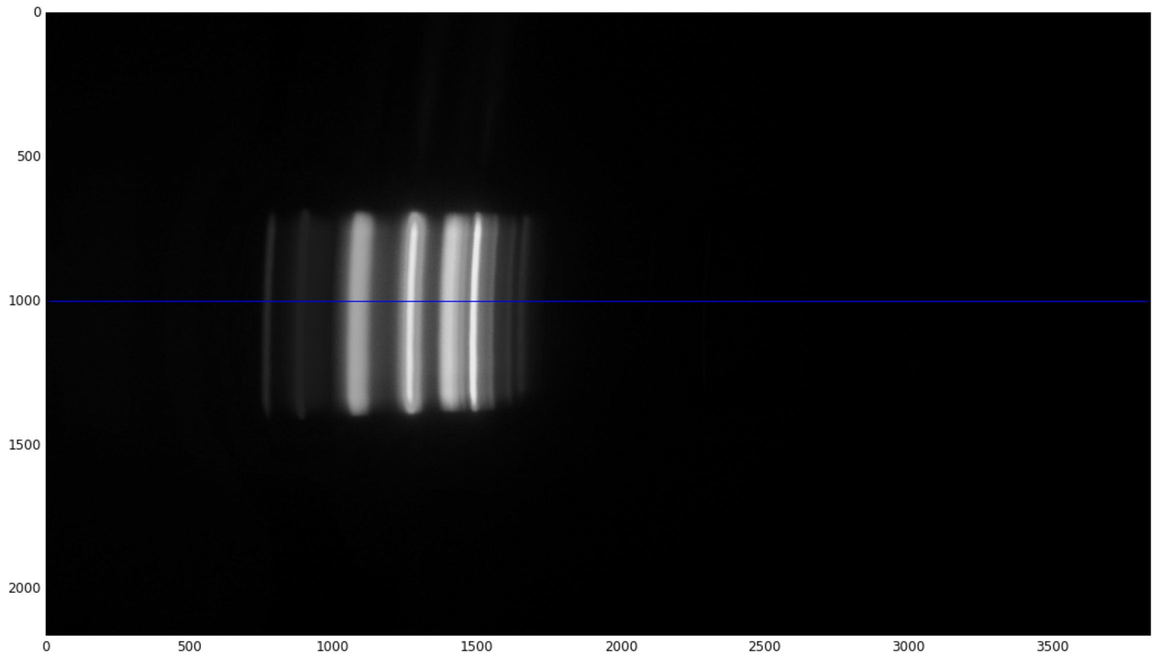
```
None (3840, 2160) LA
```

In [34]:

```
# show grey scale image
plt.imshow(greyscale_img)
# show sampling line
sampling_line = 1000 # EDIT THIS: change to the line you need to sample
plt.axhline(sampling_line)
```

Out[34]:

<matplotlib.lines.Line2D at 0x12d81a6d8>



In [35]:

```
# load pixels into numpy array
pix = array(greyscale_img)
```

In [36]:

```
# sample at the specified pixel row / horizontal line
data_row = pix[sampling_line]
```

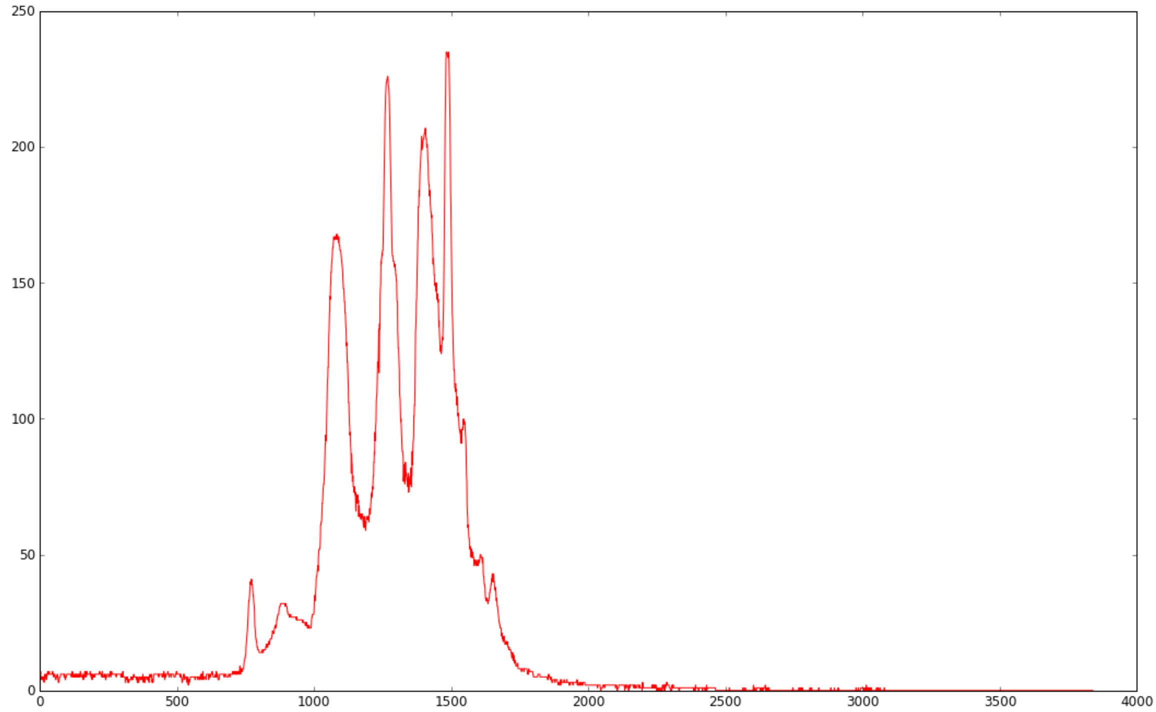
In [37]:

```
# take data in sampling line as an array of values - this is the raw spectral da
ta
spectrum = np.take(data_row,0,1)
# print the number of values, which is equal to the image horizontal size
print(len(spectrum))
# plot the raw spectrum
plt.plot(spectrum)
```

```
3840
```

Out[37]:

```
[<matplotlib.lines.Line2D at 0x133e34fd0>]
```



In [41]:

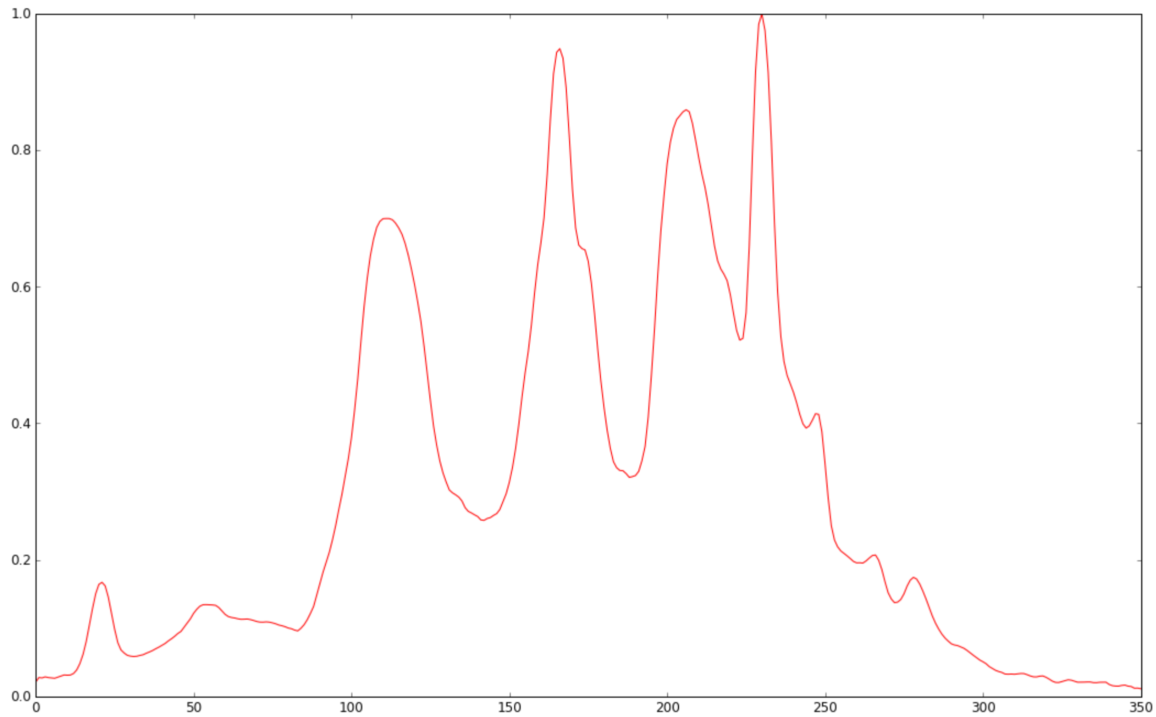
```
# this is the calibration process:
# using a compact fluorescent spectrum, choose how to trim the left and right ed
ge of the data
# so that the second blue wavelength is 436 nm and the green wavelength is 546 n
m
# this might need a little bit of trial and error and running iteratively this c
ell and
# the next two cells too

trimLeft = 700 # EDIT THIS: change this for calibration
trimRight = 1900 # EDIT THIS: change this for calibration

# create a linear integer space arrangement with x trimmed between the two selec
ted values
x = np.linspace(trimLeft,trimRight,trimRight-trimLeft+1)

# smooth the spectrum using the Savitzky–Golay filter
newSpectrum = savgol_filter(spectrum, 21, 2)

# normalise the spectrum so that the values are between 0 and 1
normalizedSpectrum =
normalize(newSpectrum[trimLeft:trimRight+1],'max').reshape(-1,1)

# resample the spectrum so that the sampling is every nanometer
resampledSpectrum = resample(normalizedSpectrum,730-380+1)
```

In [42]:

```
# plot the resampled and smoothed spectrum
plt.plot(resampledSpectrum)
```

Out[42]:

```
[<matplotlib.lines.Line2D at 0x133deba90>]
```

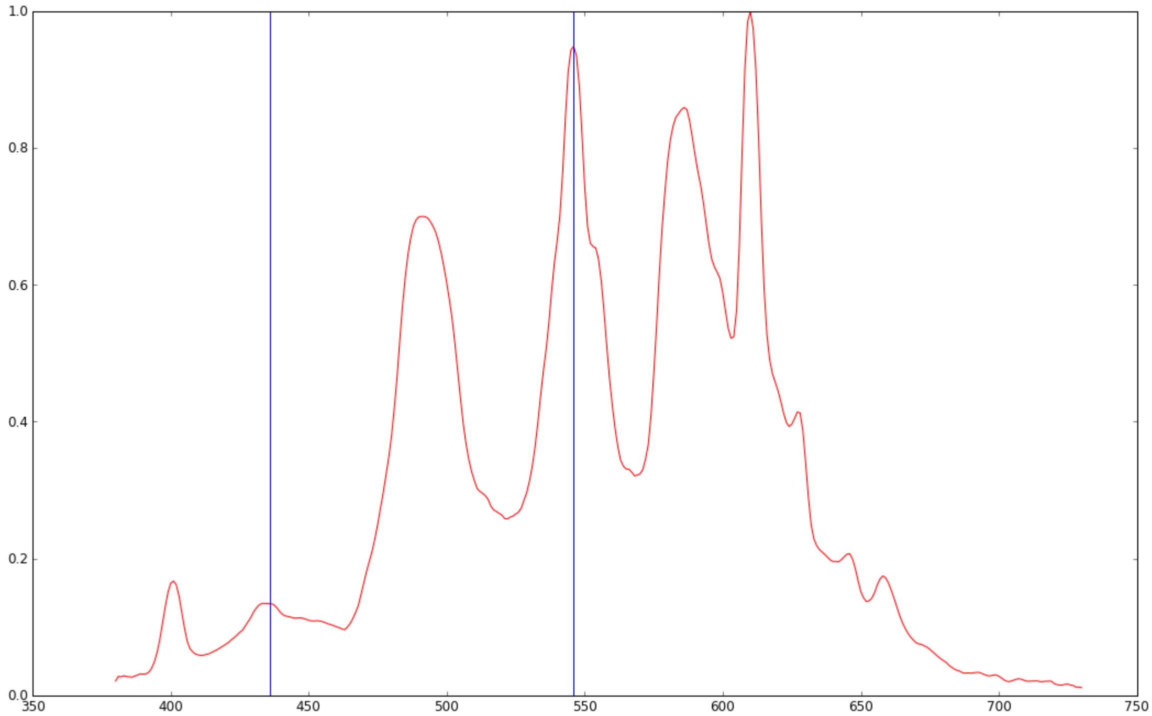


In [44]:

```
# create a linear integer space arrangement with x between 380 and 730 nm
xx = np.linspace(380,730,730-380+1)

# plot the new resampled, smoothed spectrum, transposed between 380 and 730 nm
plt.plot(xx,resampledSpectrum)

# plot the second blue wavelength (436 nm) and the green wavelength (546 nm)
plt.axvline(x=436)
plt.axvline(x=546)

# if the blue lines are marking the second blue wavelength and the green wavelen
gth,
# proceed to the next cell, otherwise revise the trimLeft and trimRight values a
nd run
# another iteration
```

Out[44]:

<matplotlib.lines.Line2D at 0x13b740a20>

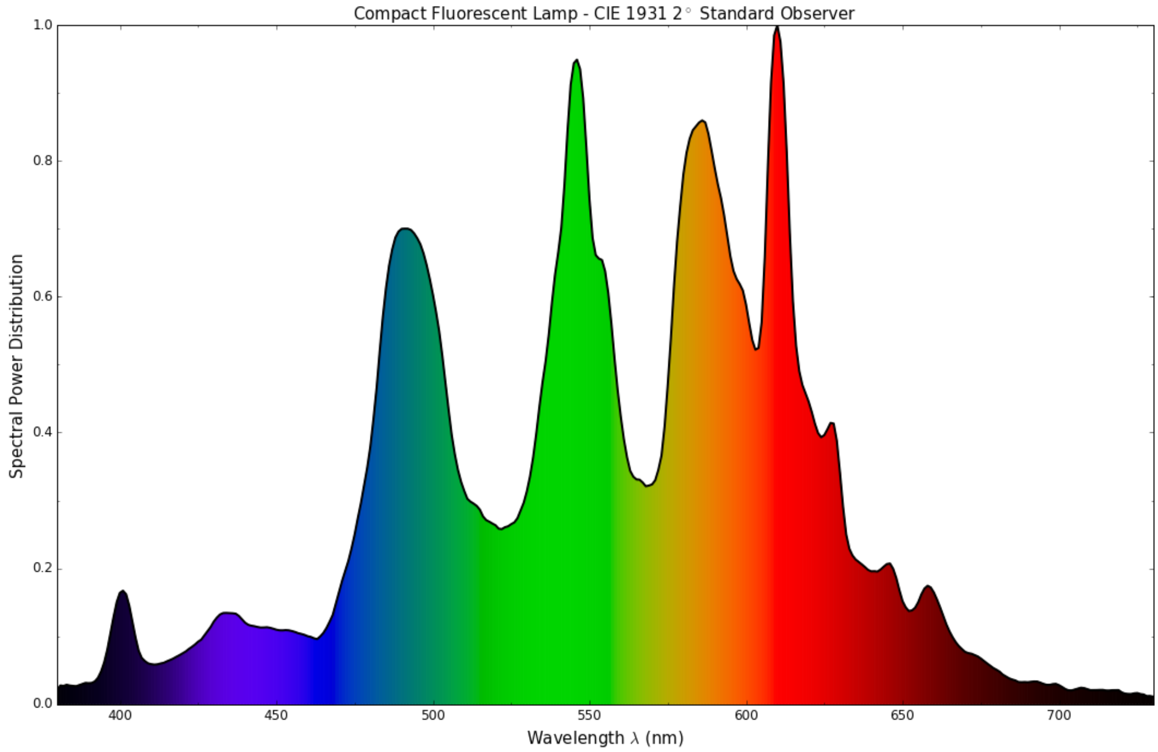


In [45]:

```
# transform the spectrum into a dictionary that the Colour library can process
spd_data = dict(izip(xx, resampledSpectrum))
```

In [53]:

```
# EDIT THIS: change name according to name of spectrum
spd = colour.SpectralPowerDistribution('Compact Fluorescent Lamp', spd_data)
# plot the spectrum using the Colour library
single_spd_plot(spd)
```



In [54]:

```
# calculate the sample spectral power distribution *CIE XYZ* tristimulus values
# using the CIE 1931 2 degree standard observer and D65 (daylight) standard illu
minant
cmfs = colour.STANDARD_OBSERVERS_CMFS['CIE 1931 2 Degree Standard Observer']
illuminant = colour.ILLUMINANTS_RELATIVE_SPDS['D65']
XYZ = colour.spectral_to_XYZ(spd, cmfs, illuminant)
print(XYZ)
```

```
[ 45.02891476  52.32765966  20.07891099]
```

In [55]:

```
# calculate *xy* chromaticity coordinates for the spectrum
xy =  colour.XYZ_to_xy(XYZ)
print(xy)
```

```
[ 0.38343534  0.44558644]
```

In [57]:

```
# display the colour coordinate of the spectral sample in the CIE chromaticity d
iagram

# plot the *CIE 1931 Chromaticity Diagram* including the Planckian Locus
# the argument *standalone=False* is passed so that the plot doesn't get display
ed
# and can be used as a basis for other plots
#CIE_1931_chromaticity_diagram_plot(standalone=False)
planckian_locus_CIE_1931_chromaticity_diagram_plot(standalone=False)

# plot the *xy* chromaticity coordinates of the spectrum
x, y = xy
pylab.plot(x, y, 'o-', color='white')

# Annotating the plot.
pylab.annotate(spd.name.title(),
               xy=xy,
               xytext=(-50, 30),
               textcoords='offset points',
               arrowprops=dict(arrowstyle='->', connectionstyle='arc3,
rad=-0.2'))

# Displaying the plot.
display(standalone=True)
```


A, B, C Illuminants - Planckian Locus
CIE 1931 Chromaticity Diagram - CIE 1931 2 Degree Standard Observer
CIE y
CIE x
520
540
510
560
500
580
Compact Fluorescent Lamp
A
600
B
C
620
700
490
4000K
3000K
2500K
2000K
1667K
6000K
10000K
480
470
460
390
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

In [58]:

```
# show the spectrum wavelengths
print(spd.wavelengths)
```

```
[ 380.  381.  382.  383.  384.  385.  386.  387.  388.  389.  390.
 391.
  392.  393.  394.  395.  396.  397.  398.  399.  400.  401.  402.
 403.
  404.  405.  406.  407.  408.  409.  410.  411.  412.  413.  414.
 415.
  416.  417.  418.  419.  420.  421.  422.  423.  424.  425.  426.
 427.
  428.  429.  430.  431.  432.  433.  434.  435.  436.  437.  438.
 439.
  440.  441.  442.  443.  444.  445.  446.  447.  448.  449.  450.
 451.
  452.  453.  454.  455.  456.  457.  458.  459.  460.  461.  462.
 463.
  464.  465.  466.  467.  468.  469.  470.  471.  472.  473.  474.
 475.
  476.  477.  478.  479.  480.  481.  482.  483.  484.  485.  486.
 487.
  488.  489.  490.  491.  492.  493.  494.  495.  496.  497.  498.
 499.
  500.  501.  502.  503.  504.  505.  506.  507.  508.  509.  510.
 511.
  512.  513.  514.  515.  516.  517.  518.  519.  520.  521.  522.
 523.
  524.  525.  526.  527.  528.  529.  530.  531.  532.  533.  534.
 535.
  536.  537.  538.  539.  540.  541.  542.  543.  544.  545.  546.
 547.
  548.  549.  550.  551.  552.  553.  554.  555.  556.  557.  558.
 559.
  560.  561.  562.  563.  564.  565.  566.  567.  568.  569.  570.
 571.
  572.  573.  574.  575.  576.  577.  578.  579.  580.  581.  582.
 583.
  584.  585.  586.  587.  588.  589.  590.  591.  592.  593.  594.
 595.
  596.  597.  598.  599.  600.  601.  602.  603.  604.  605.  606.
 607.
  608.  609.  610.  611.  612.  613.  614.  615.  616.  617.  618.
 619.
  620.  621.  622.  623.  624.  625.  626.  627.  628.  629.  630.
 631.
  632.  633.  634.  635.  636.  637.  638.  639.  640.  641.  642.
 643.
  644.  645.  646.  647.  648.  649.  650.  651.  652.  653.  654.
 655.
  656.  657.  658.  659.  660.  661.  662.  663.  664.  665.  666.
 667.
  668.  669.  670.  671.  672.  673.  674.  675.  676.  677.  678.
 679.
  680.  681.  682.  683.  684.  685.  686.  687.  688.  689.  690.
 691.
  692.  693.  694.  695.  696.  697.  698.  699.  700.  701.  702.
 703.
  704.  705.  706.  707.  708.  709.  710.  711.  712.  713.  714.
 715.
  716.  717.  718.  719.  720.  721.  722.  723.  724.  725.  726.
 727.
  728.  729.  730.]
```

In [59]:

```
# show the spectrum values
print(spd.values)
```

```
[ 0.0208532   0.0276647   0.02705774  0.02857712  0.02757241  0.0270
4047
  0.02655208  0.02816369  0.02960833  0.03137507  0.03104737  0.0313
0543
  0.03362139  0.03885933  0.04821104  0.06173291  0.08048775  0.1046
6583
  0.12883401  0.15049445  0.16405011  0.16713666  0.16188531  0.1452
5878
  0.12155411  0.09827836  0.07876268  0.06809213  0.06355497  0.0602
3504
  0.05897934  0.05840131  0.05876845  0.06003597  0.06100284  0.0631
6679
  0.0650546   0.06717051  0.06977669  0.07214287  0.07483113  0.0775
9553
  0.0812703   0.08445041  0.08798542  0.09211443  0.0951888   0.1013
09
  0.10721708  0.11378383  0.12170485  0.12757968  0.13217332  0.1342
912
  0.13441399  0.13424532  0.13394442  0.13332682  0.13011293  0.1250
9383
  0.12012495  0.11697253  0.11561395  0.11486569  0.11371808  0.1129
253
  0.11304927  0.11336467  0.11251656  0.11122885  0.10972158  0.1088
4345
  0.10875118  0.1091733   0.10876491  0.10771535  0.10634419  0.1045
5106
  0.10347094  0.10208042  0.10016011  0.09912679  0.09710617  0.0959
3964
  0.09982956  0.10501397  0.11227592  0.12153846  0.13160431  0.1482
2223
  0.16511767  0.18192386  0.19656192  0.21123455  0.22954628  0.2501
5141
  0.27383041  0.29623251  0.3220319   0.34798454  0.37899008  0.4196
2653
  0.46520778  0.51960679  0.57046239  0.61244045  0.64596871  0.6697
3736
  0.68707329  0.69587873  0.69986553  0.70011914  0.70006321  0.6981
8976
  0.69318454  0.68604671  0.67711478  0.66371328  0.64647006  0.6259
6193
  0.60306604  0.57746577  0.54903081  0.51312023  0.47342207  0.4335
0917
  0.39636979  0.36787321  0.34472369  0.32735881  0.31408608  0.3024
5582
  0.29824511  0.2953585   0.29193385  0.28639587  0.27651715  0.2711
2556
  0.26874644  0.26599569  0.26349957  0.25849622  0.25788944  0.2606
1897
  0.26199981  0.26514868  0.26775772  0.2738861   0.28511369  0.2967
5908
  0.31361297  0.33528754  0.36328957  0.39814369  0.43774585  0.4738
5855
  0.50519948  0.54422731  0.59084599  0.63234286  0.6643888   0.7014
2019
  0.76447728  0.84500756  0.91214098  0.94366321  0.94895798  0.9350
4173
  0.89205414  0.82036985  0.74303304  0.6864942   0.66124573  0.6561
4826
  0.65399752  0.63744469  0.60528323  0.56061692  0.50895642  0.4622
9187
  0.42332402  0.38986833  0.36337942  0.34365058  0.33506426  0.3310
```

```
1434
  0.33059806  0.32644338  0.32092394  0.32199736  0.32367027  0.3298
5068
  0.34506855  0.36600491  0.4091305   0.47005863  0.53863144  0.6156
584
  0.68202372  0.73385218  0.78035866  0.81183814  0.83237448  0.8450
9714
  0.85084503  0.85637306  0.85948471  0.85648523  0.8396101   0.8150
6428
  0.78928083  0.76549763  0.74543108  0.71999868  0.68934411  0.6596
9543
  0.63792409  0.62610721  0.61846006  0.60907596  0.58860875  0.5614
7873
  0.53624808  0.52198981  0.52454641  0.56233434  0.65876969  0.7938
206
  0.91558636  0.9848981   0.99986573  0.97536923  0.91407373  0.8133
0012
  0.69259416  0.5922527   0.52761976  0.49054903  0.4702032   0.4577
8523
  0.44557041  0.43037878  0.41325657  0.39938376  0.39313505  0.3962
489
  0.40471458  0.41418736  0.41318507  0.38803135  0.34138332  0.2906
9322
  0.24997929  0.22902382  0.21909617  0.21313405  0.20958483  0.2058
5604
  0.20192641  0.1977575   0.19564312  0.19582315  0.19533689  0.1984
5998
  0.20262221  0.20640468  0.20715235  0.19941753  0.18579084  0.1678
3557
  0.15180272  0.14271603  0.13728926  0.13810497  0.141982    0.1501
7986
  0.16165584  0.17032694  0.17446937  0.17195729  0.16459166  0.1541
7384
  0.14263512  0.13024197  0.11764841  0.10722557  0.09860438  0.0913
2427
  0.08554107  0.08107416  0.07705725  0.07528649  0.07441203  0.0724
5976
  0.07045841  0.06729825  0.06369236  0.06013696  0.0567705   0.0532
9291
  0.0507702   0.04793462  0.04371459  0.04090486  0.03819947  0.0364
3053
  0.03510026  0.03276661  0.03248427  0.03273332  0.03242645  0.0330
703
  0.03353955  0.03343585  0.03181774  0.02996533  0.02864418  0.0285
4561
  0.02964049  0.02993882  0.02801315  0.02532085  0.02235417  0.0205
0332
  0.02033057  0.02169528  0.02323981  0.02434216  0.02373819  0.0222
9067
  0.02076481  0.02080465  0.02086707  0.02108103  0.02108124  0.0201
847
  0.02013291  0.02066616  0.02064667  0.02079031  0.01748933  0.0157
0138
  0.01512702  0.0150428   0.01608278  0.01650037  0.01498474  0.0143
9265
  0.01194791  0.0121482   0.01127465]
```

In [61]:

```
# export the spectral dataset to a csv file
# EDIT THIS: change filename of the csv file (csv means comma separated values)
out = csv.writer(open("compact_fluorescent.csv","w"),
delimiter=',',quoting=csv.QUOTE_ALL)
out.writerow(spd.wavelengths)
out.writerow(spd.values)
```

Out[61]:

```
6036
```

In [62]:

```
# well done! now do the same with other spectral pictures!
```