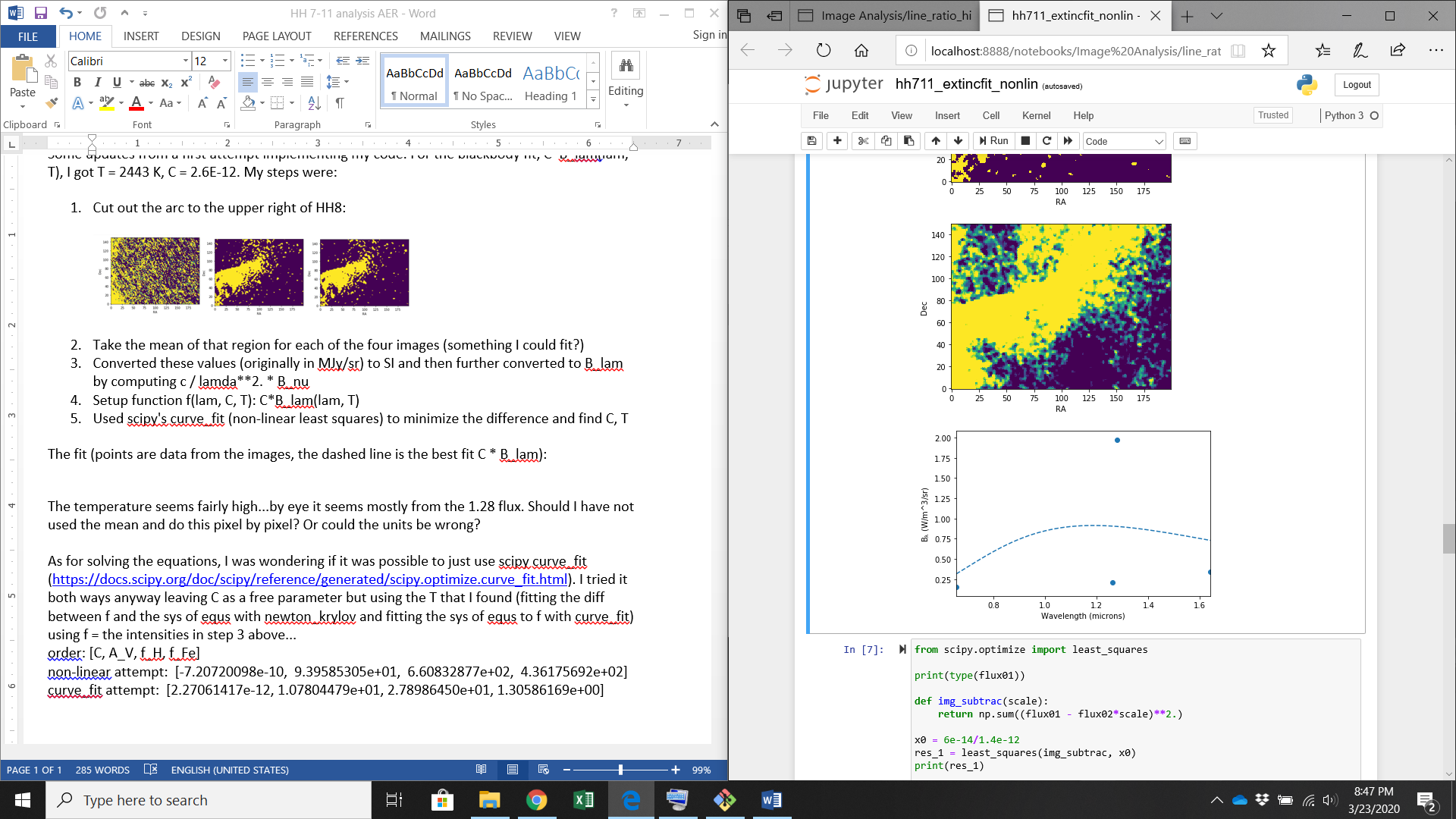
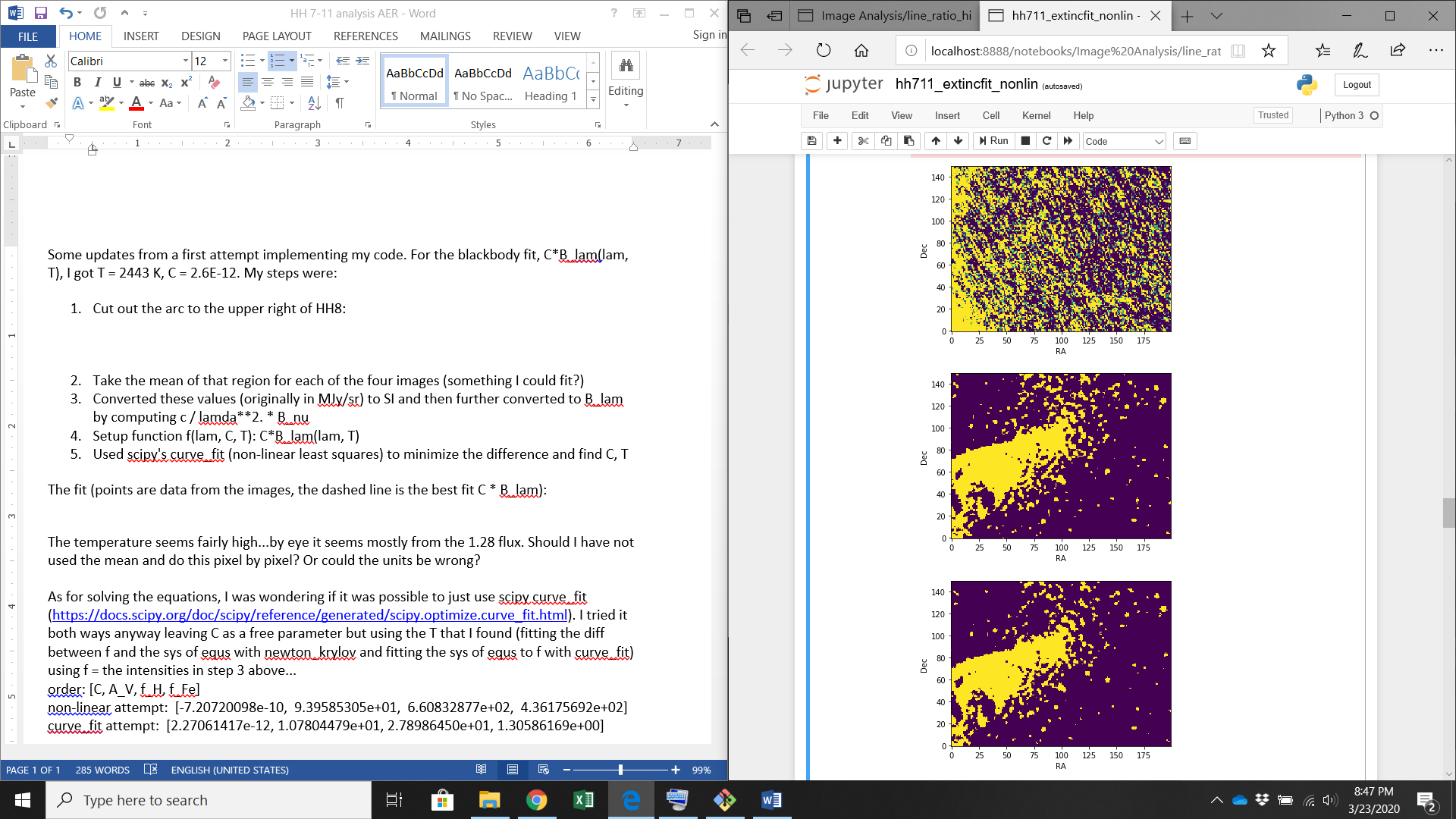
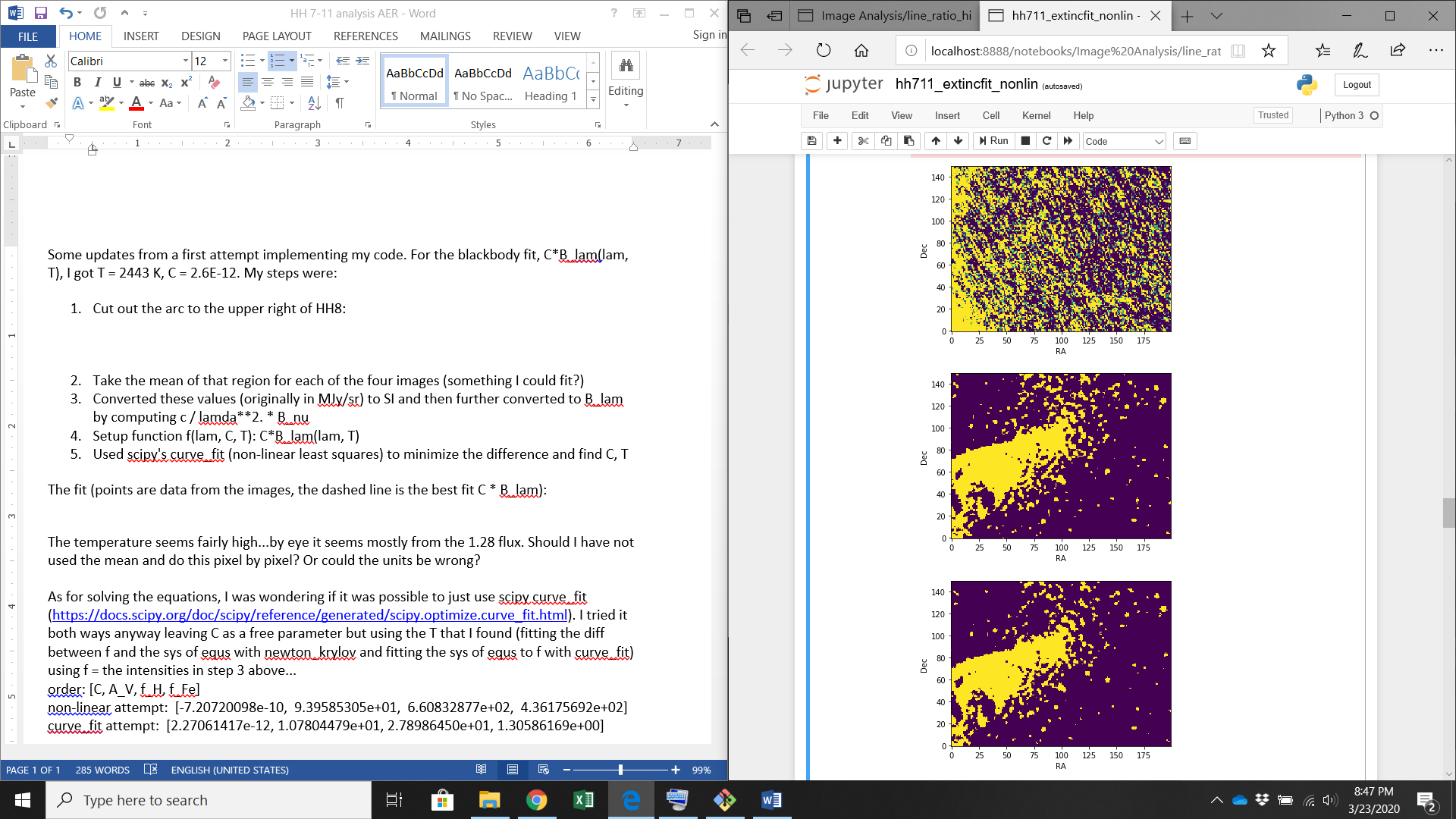
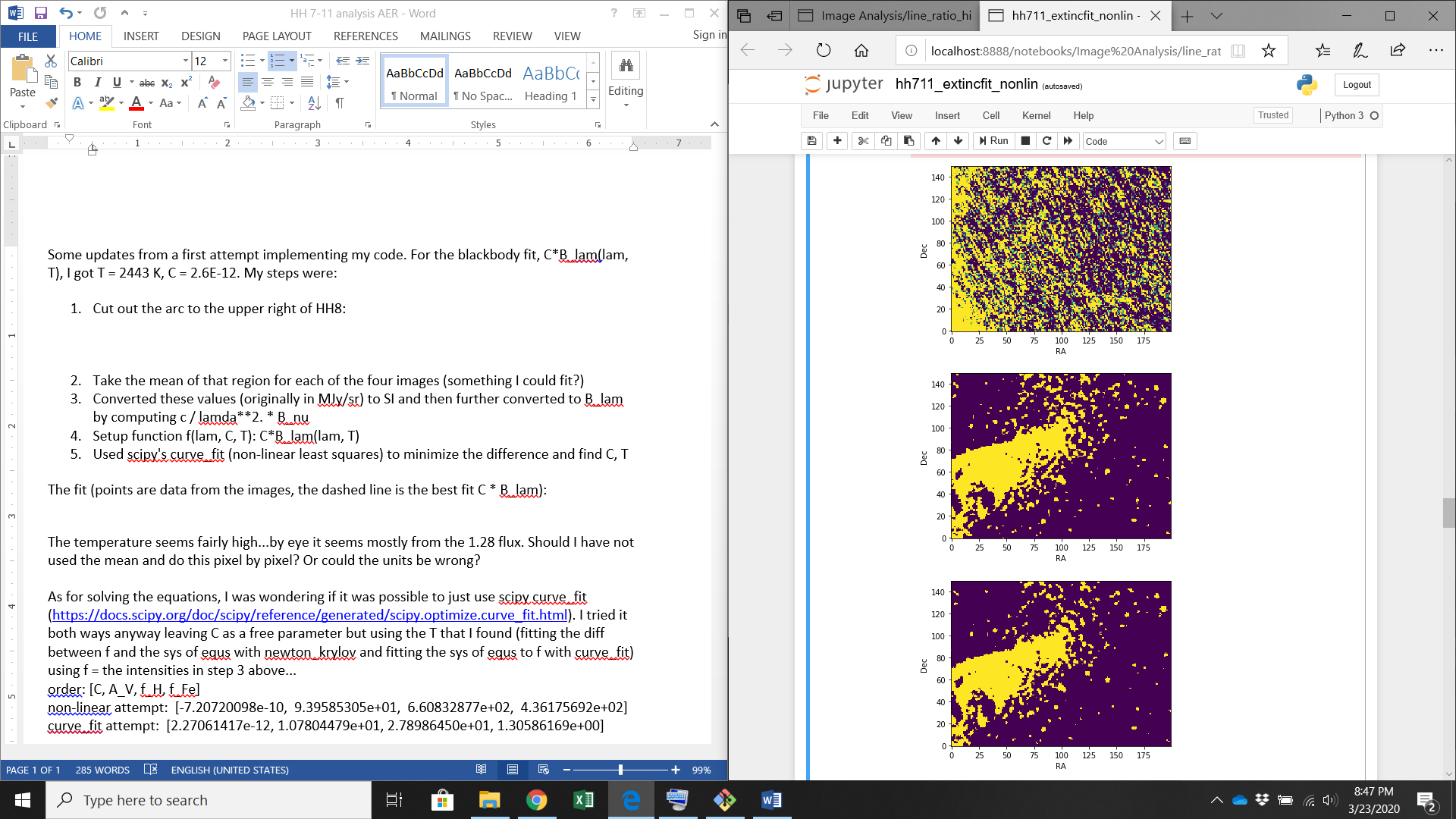
**Blackbody Fit**

Some updates from a first attempt implementing my code. For the blackbody fit, C\*B\_lam(lam, T), I got **T = 3566 +/- 523 K, C = 1.36E-23 +/- 9.44E-24** (the error bar here is 1σ determined from the covariance that the curve\_fit function outputs…in other words it could be more precise). My steps were:

1. Cut out the arc to the upper right of HH8:

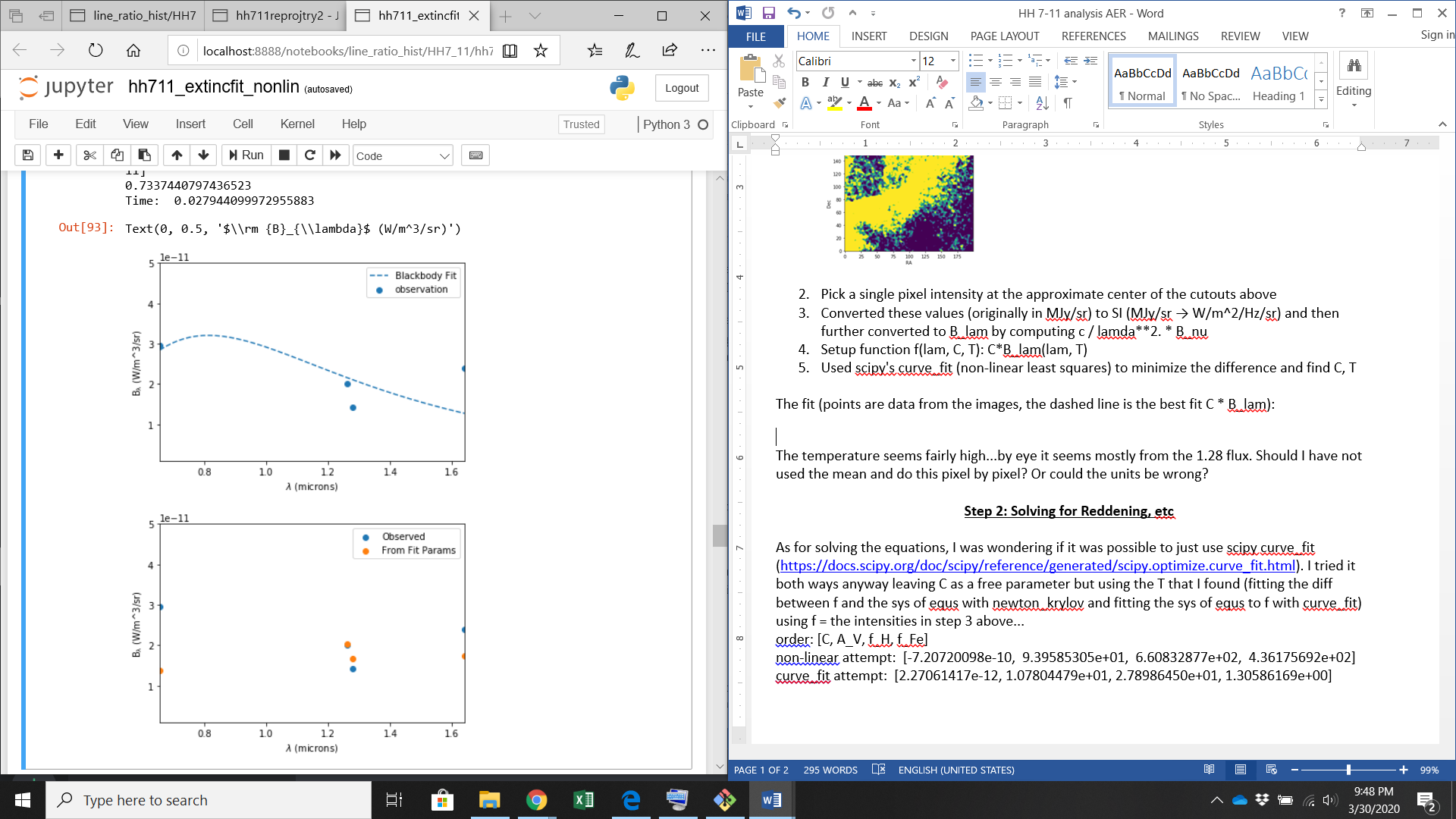


1. Pick a single pixel intensity at the approximate center of the cutouts above
2. Converted these values (originally in MJy/sr) to SI (MJy/sr → W/m^2/Hz/sr) and then further converted to B\_lam by computing c / lamda\*\*2. \* B\_nu
3. Setup function f(lam, C, T): C\*B\_lam(lam, T)
4. Used scipy's curve\_fit (non-linear least squares) to minimize the difference and find C, T

The intensity values to be fit are then:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lambda (microns) | 0.656 | 1.26 | 1.28 | 1.64 |
| Intensity (W/m^3/sr) | 2.95e-11 | 2.01e-11 | 1.44e-11 | 2.40e-11 |

The fit (points are data from the images, the dashed line is the best fit C \* B\_lam):



The temperature (**3566 K**) seems reasonable for scattered starlight from a relatively cool star. So far seems like main thing to check is whether the observed Bλ values are reasonable…

**Solving Reddening for a Single Pixel (mostly relying on defaults)**

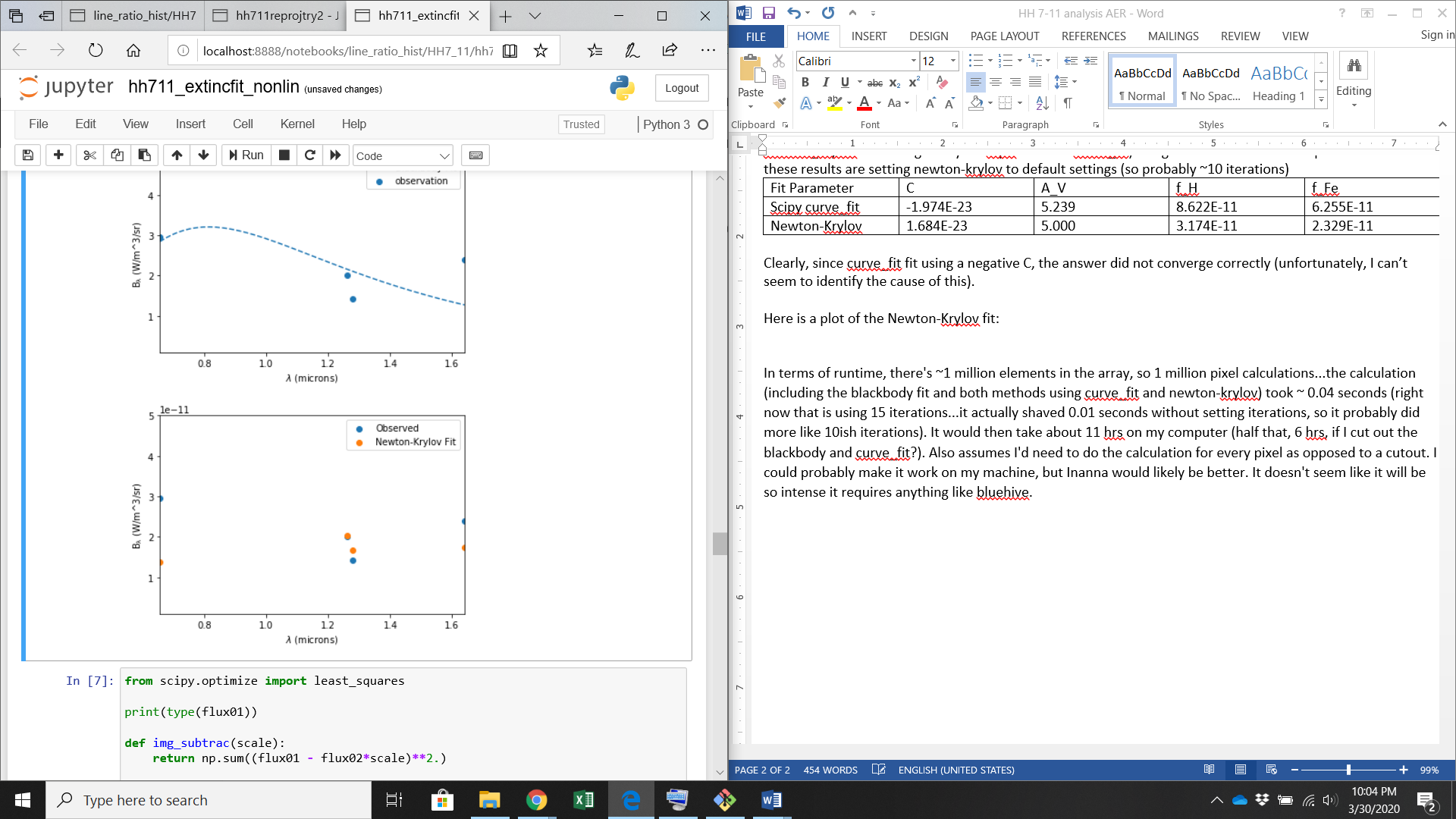
For solving the equations, I was wondering if it was possible to just use scipy curve\_fit (<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html>). I tried it both ways leaving C as a free parameter but using the T I found (fitting the diff between f and the sys of equs with newton\_krylov and fitting the sys of equs to f with curve\_fit) using f = the intensities in step 3 above...I found the following when setting to default settings (so probably ~10 iterations)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fit Parameter | C | A\_V | f\_H | f\_Fe |
| Scipy curve\_fit | -1.974E-23 | 5.239 | 8.622E-11 | 6.255E-11 |
| Newton-Krylov | 1.684E-23 | 5.000 | 3.174E-11 | 2.329E-11 |

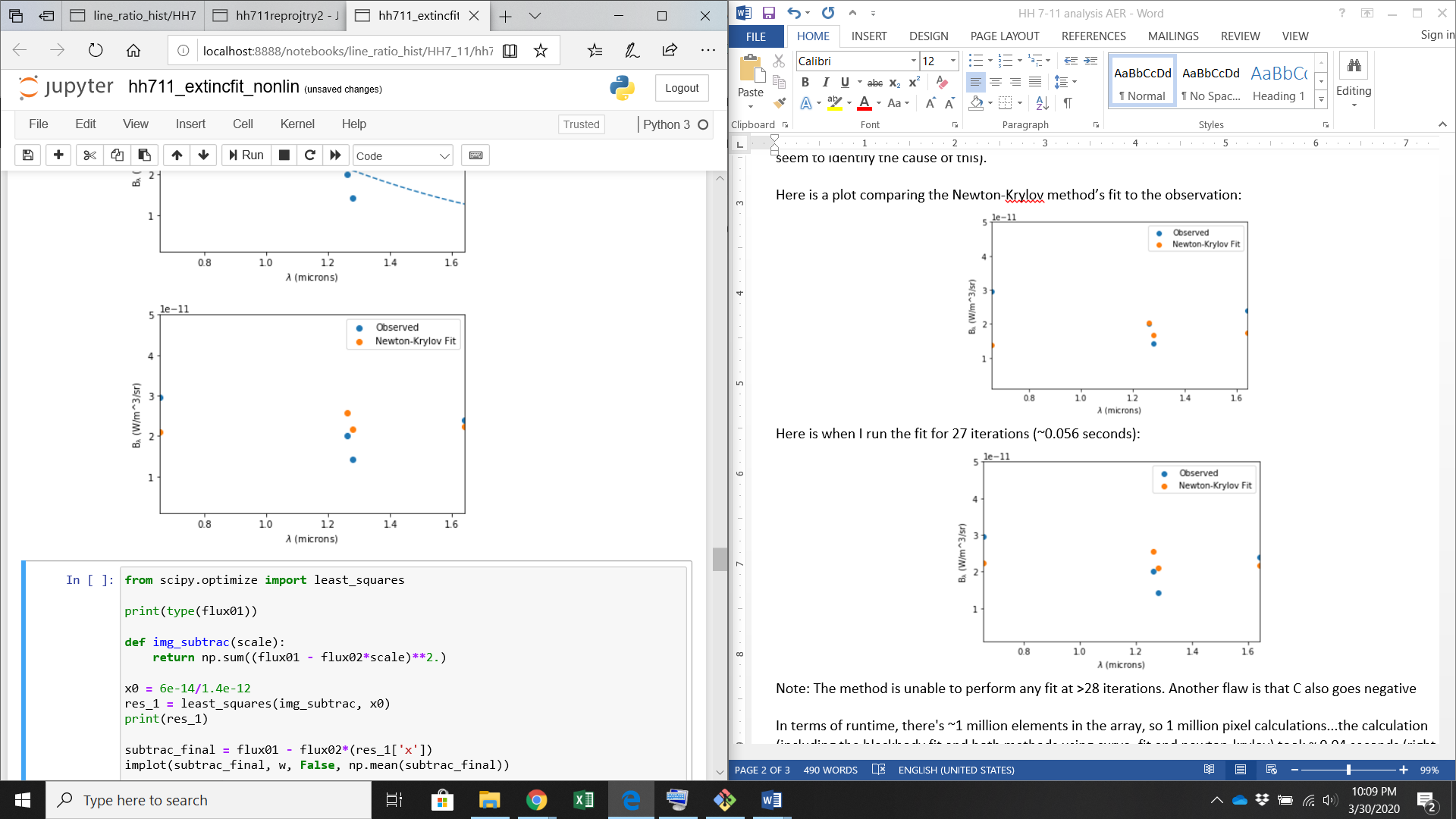
Noticeably, curve\_fit produces a negative C, which seems peculiar to me…I’ll proceed only using Newton-Krylov, though the other fit parameters are quite similar, which seems promising.

An attempt at a “χ^2”: , where O is the observed value, p is the theoretical value. This gets a value of **χ^2 = 2.94E-22**. Cannot do further without some estimate for σ.

I think a visual might be of interest in addition (as it’s always possible to mess up your statistics). Here is a plot comparing the Newton-Krylov method’s fit to the observation:



Here is when I run the fit for 17 iterations (~0.046 seconds):



20 iterations (“converged”):

Note: The method is unable to perform any fit at >28 iterations. Another flaw is that C also goes negative when I perform > 18 iterations (dependent on initial choice of parameters, but that is difficult to find). I also had to adjust values at 17 and ~15 iterations. When I left it default, it did not seem to require much choosing of initial parameters.

In terms of runtime, there's ~1 million elements in the array, so 1 million pixel calculations...the calculation (including the blackbody fit and both methods using curve\_fit and newton-krylov) took ~ 0.04 seconds (right now that is using 15 iterations...it actually shaved 0.01 seconds without setting iterations, so it probably did more like 10ish iterations. It would then take about 11 hrs on my computer (half that, 6 hrs, if I cut out the blackbody and curve\_fit?). Also assumes I'd need to do the calculation for every pixel as opposed to a cutout. I could maybe make it work on my machine, but Inanna would likely be better. It doesn't seem like it will be intense enough to require the likes of bluehive at least!

Fitting a Pixel (no defaults)

Adjusting method:

Adjusting iterations:

Adjusting step size:

Extra Check:



Solve for f\_Fe/f\_H:

**= χ\_Fe \* λ4 / λ3 \* γ09 \* Σ[A9i]/A90 \* 1/ α\_Paβ**

χ:

λ4/λ3 = 1.64 / 1.28

γ =

A coeffs…

α =