DMD-MOS Project Smile Fit Progress Report

Matthew Leung
Dunlap Institute for Astronomy & Astrophysics, University of Toronto

December 29, 2020

1 Introduction

This report gives an overview of the current progress of the smile fit. For some wavelength λ, a point on the DMD given by (x, y) can be mapped to a point on the image plane, (x_d, y_d). Smile distortion can be fitted by the quadratic function:

$$y_d = f(x_d) = a_s x_d^2 + c_s \tag{1.1}$$

The main focus of this report will be on fitting c_s as a function of λ and y. Note that c_s has no dependence on x. I will also discuss a bit about fitting a_s as a function of λ and y.

2 Project Code Files

2.1 Project Code Files Structure

Since the previous progress report (from October 17, 2020), I restructured the Python scripts so that everything could be run from one script, and so that the fitting process is more flexible. The current structure is shown below.

```
project_directory/
├── get_ima_gia_files_info.py
├── find_centroids_functions.py
├── smile_fit_quadratic.py
├── Globals.py
├── main.py
├── a_fit_rough.py
└── dataset/
    ├── ima/
    ├── gia/
    ├── centroids/
    ├── visualizations/
    ├── smile_results/
    │   ├── baseline/
    │   ├── angle_add_corr_roots_quad_2/
    │   ...
    │   └── lambda_y_a_c.csv
    └── wavelengths.wav
```

For fitting c_s, only `main.py` needs to be run. There are options in `main.py` to find the centroids by k-means clustering (by calling functions in `find_centroids_functions.py`) and to fit equation (1.1) (by calling functions in `smile_fit_quadratic.py`) if needed. These only need to be done once. After finding the centroids and fitting equation (1.1), the results are saved in the `smile_results/` directory. The resulting values of λ, y, a_s, and c_s are saved in `lambda_y_a_c.csv`.

To fit c_s and a_s as functions of λ and y, only `lambda_y_a_c.csv` needs to be used, saving on time. The function `smile_c_fit` in `main.py` fits c_s using any function of choice. The resulting fit parameters and errors are stored in subdirectories of the `smile_results/` directory, with the same name as the fit function (e.g. `angle_add_corr_roots_quad_2/`). The script `a_fit_rough.py` has a similar function called `smile_a_fit` that fits a_s using any function of choice. I decided to separate this from `main.py` for now because I am still experimenting with the fit for a_s.

2.2 Some Notable Changes

Below are some notable changes to the code made since the last progress report:

- The quadratic function $y_d = a_s x_d^2 + c_s$ is now fitted in a separate function called `smile_fit_quadratic` in `smile_fit_quadratic.py`
- Initially, in `smile_fit.py`, this quadratic function, along with a_s and c_s as functions of λ, were all fitted in the *same* function, `smile_fit`. This is not ideal because a_s and c_s are actually functions of both λ and y. Separating the quadratic fitting process allows for better fits for a_s and c_s later on.
- `lambda_c_y.csv` from before is now `lambda_y_a_c.csv`; the column ordering was changed, and I also included the a_s values.
- The variable `slit_y_pos` in `smile_fit.py` was replaced with the variable `y_slit_dmd`, which is the slit position on the DMD. `slit_y_pos` corresponds to y_{slit} while `y_slit_dmd` corresponds to y (see equation (3.1) in the next section). The "y" in `lambda_y_a_c.csv` is now y rather than y_{slit}.
- Global variables are now stored in `Globals.py`

3 Derivation of Basic Fit Equation for c_s

Let N_{My} be the number of micromirrors on the DMD in the y-direction, and "row" be the row number of the DMD micromirror in the y-direction. Based on the design, $N_{My} = 4000$. We number the micromirrors such that $0 \leq \text{row} < N_{My}$, where "row" is an integer. From Geometric Image Analysis (GIA) in Zemax OpticStudio, we have output files in TXT format and some field_size. Given some row number for micromirrors that are ON, we can find the corresponding y on the DMD as:

$$y = -\left(\frac{\frac{1}{2}N_{My} - (\text{row} + 1) + 0.5}{\frac{1}{2}N_{My}}\right)\left(\frac{\text{field\_size}}{2}\right) \tag{3.1}$$

In the previous progress report, I mentioned some variable y_{slit}, which is the slit position on the detector (image plane). The relationship between y_{slit} and y is that $y_{slit} = yM_S$, where M_S is the magnification of the spectrograph.

Let G be the groove density of the grating, m be the diffraction order, θ be the entrance angle relative to the grating normal, and β be the diffraction angle relative to the grating normal. The grating equation states that:

$$Gm\lambda = \sin\alpha + \sin\beta \tag{3.2}$$

We use $m = 1$ in the design of the spectrograph. Solving for β, we obtain:

$$\beta = \arcsin\left(G\lambda - \sin\alpha\right) \tag{3.3}$$

Let us now find α. Suppose some row of micromirrors with y-coordinate y (found with equation (3.1)) on the DMD is ON. Light reflected by these micromirrors pass through a collimator with effective focal length f_{Coll}. If we trace the ray parallel to the optical axis (coming from the micromirror), this ray will make an angle $\theta' = \arctan\left(\frac{y}{f_{Coll}}\right)$ with the optical axis after passing through the collimator. However, the grating normal is tilted at an angle θ_0 relative to the optical axis containing the collimator. Thus, the entrance angle is:

$$\alpha = \theta_0 + \theta' = \theta_0 + \arctan\left(\frac{y}{f_{Coll}}\right) \tag{3.4}$$

In the design, we take $\theta_0 = 12.870°$. Substituting equation (3.4) into equation (3.3), we obtain:

$$\beta = \arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{y}{f_{Coll}}\right)\right]\right) \tag{3.5}$$

Now, let us find y_d. We assume by design that at centre wavelength $\lambda_c \equiv 0.55$ µm, $\alpha = \theta_0$ implies that $\beta = \alpha = \theta_0$. This is the Littrow configuration. Furthermore, once the ray leaves the grating, the optical axis is tilted at angle θ_0 relative to the grating normal. After leaving the grating, the ray passes through some camera optics with effective focal length f_{Cam}, and then hits the detector. We can then write:

$$\tan\left(\beta - \theta_0\right) = \frac{y_d}{f_{Cam}} \Rightarrow y_d = f_{Cam}\tan\left(\beta - \theta_0\right) \tag{3.6}$$

Substituting equation (3.5) into equation (3.6), we obtain:

$$y_d = f_{Cam}\tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \tag{3.7}$$

Now, if we choose $x = 0$, then $x_d = 0$ and so by equation (1.1) we have $y_d = c_s$. Thus,

$$c_s = f_{Cam}\tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \tag{3.8}$$

In the design, we have $G = 0.810$ lines/µm, $f_{Coll} = 60.115213$ mm, and $f_{Cam} = 79.131160$ mm. We will use equation (3.8), along with these values of G, f_{Coll}, f_{Cam}, and θ_0 as in the design, as a baseline fit for c_s.

4 Fitting c_s with SciPy Curve Fitting Module

4.1 Overview

In this section, I will attempt to fit c_s as a function of λ and y. For each fit function, I will show a plot of the error of the fit function (difference between the actual c_s and the c_s from the fit function), which we want to minimize. The calculated fit parameters are shown below `params`. Each fit is followed by a quick discussion. Every function in this section was fitted using the `scipy.optimize.curve_fit` module.

4.2 Direct Fitting

4.2.1 Baseline Fit

I first used equation (3.8) to fit c_s. The fit error is shown below.

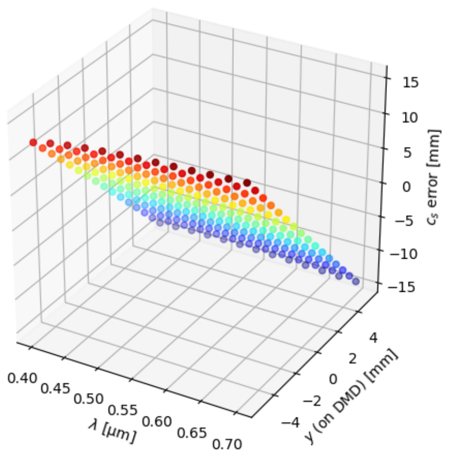


Figure 1: Baseline fit error using equation (3.8)

```
Maximum error is: 14.55811082466324 mm
Minimum error is: -14.31362582528057 mm
Maximum absolute error is: 14.55811082466324 mm
Minimum absolute error is: 0.030848188863705772 mm
```

It appears that the y coordinate is reversed. This is likely because of the negative sign in equation (3.1), and because the collimator and camera optics inverts the orientation of images, as shown below.

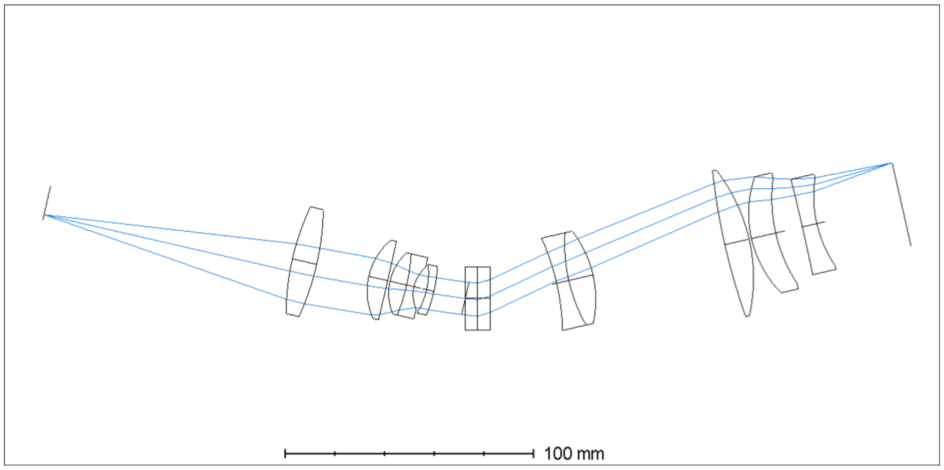


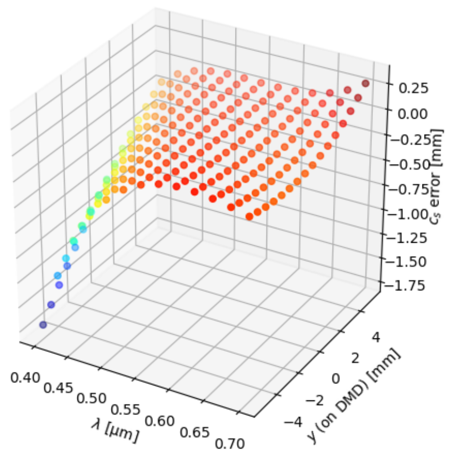
Figure 2: Images are inverted; rays from the bottom of the DMD end up at the top of the detector

4.2.2 Baseline Fit with negative y

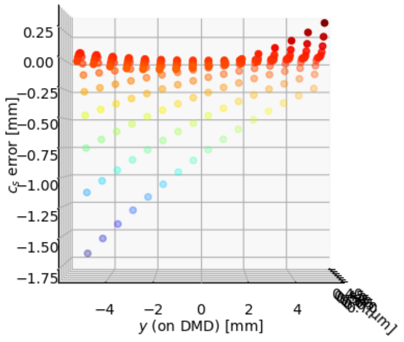
So instead, I added a negative sign in front of the y, and tried the following fit formula:

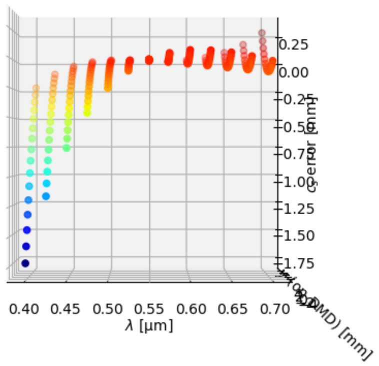
$$c_s = f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \tag{4.1}$$

The fit error is shown below.



(a)





(b) (c)

Figure 3: Baseline fit error using equation (4.1)

```
Maximum error is: 0.280110542304989 mm
Minimum error is: -1.7080647759522378 mm
Maximum absolute error is: 1.7080647759522378 mm
Minimum absolute error is: 0.0007947945838429O6 mm
Mean squared error is: 0.12539364015897364 mm^2
```

This baseline fit function looks more correct. Note that near $y = 0$ and $\lambda = \lambda_c$, the error is noticeably "flat" and near 0, which is desired. The error increases greatly at the "corners" of the plot, when the values of λ and y are both small or both large relative to when $y = 0$ and $\lambda = \lambda_c$. The error is more significant when λ and y are both small. To model this nonlinearity, we could try fitting variants of equation (4.1).

4.2.3 Fit with G, f_{Coll}, f_{Cam}, and θ_0 as fit parameters

Now, I let G, f_{Coll}, f_{Cam}, and θ_0 be fit parameters rather than constants. The fit error and fit parameters are shown below.

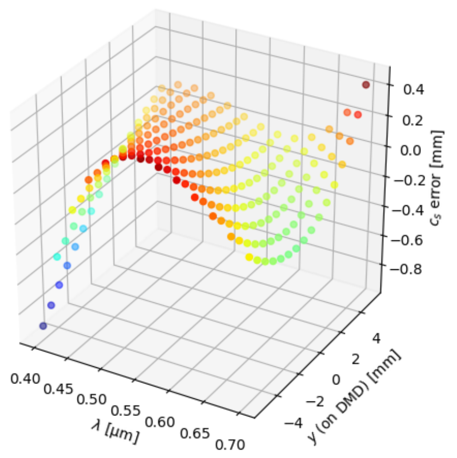


Figure 4: Baseline fit error using equation (4.1) with G, f_{Coll}, f_{Cam}, and θ_0 as fit parameters

Order of fit parameters: $G, \theta_0, f_{Coll}, f_{Cam}$

```
params
1.03695551
12.85481974
45.92014286
58.31682884
Maximum error is: 0.4171076681773158 mm
Minimum error is: -0.9005198900877449 mm
Maximum absolute error is: 0.9005198900877449 mm
Minimum absolute error is: 0.002041577879855261 mm
Mean squared error is: 0.03950138551838381 mm^2
```

The maximum absolute error is reduced by around a factor of 2. However, even with G, f_{Coll}, f_{Cam}, and θ_0 as fit parameters, the nonlinearity is still present, and the error is less "flat" now. Using `scipy.optimize.curve_fit` and allowing G, f_{Coll}, f_{Cam}, and θ_0 to vary does not seem to work too well. In this section, from now on, I will keep G, f_{Coll}, f_{Cam}, and θ_0 constant and use the values in the design. From now on, all variables except for λ, y, λ_c, G, f_{Coll}, f_{Cam}, and θ_0 are fit parameters. We will explore other variants of fit functions.

4.2.4 arcsin and arctan with multiplicative factors

I scaled the arcsin and arctan terms with some factors a_1 and a_2:

$$c_s = f_{Cam} \tan\left(a_1 \arcsin\left(G\lambda - \sin\left[\theta_0 + a_2 \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \tag{4.2}$$

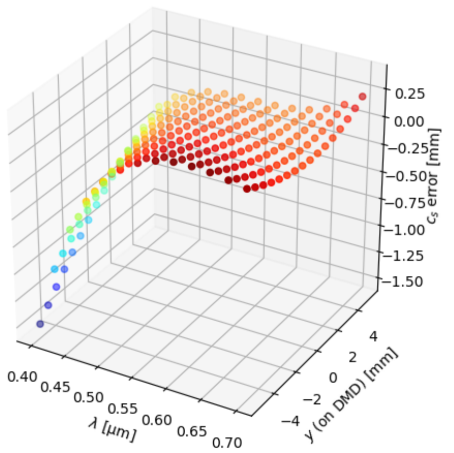


Figure 5: Fit error using equation (4.2)

Order of fit parameters: a_1, a_2

```
params
1.00458235
0.96954961
Maximum error is: 0.3285395980997161 mm
Minimum error is: -1.4921680402278064 mm
Maximum absolute error is: 1.4921680402278064 mm
Minimum absolute error is: 0.000639682820818166 mm
Mean squared error is: 0.10588488181785341 mm^2
```

The error here is reduced relative to equation (4.1), but by a less significant amount as when G, f_{Coll}, f_{Cam}, and θ_0 were fit parameters. Furthermore, the calculated values for the fit parameters a_1 and a_2 were both around 1, suggesting that this method is not very effective. Further experimentation showed that the a_2 factor was not useful, and the a_1 factor was more beneficial.

4.2.5 Cubic function in tan

To correct the nonlinearity, I tried a fit function which is cubic in tan. That is:

$$c_s = f_{Cam} \sum_{i=0}^{3} c_i \left[\tan \left(\arcsin \left(G\lambda - \sin \left[\theta_0 + \arctan \left(\frac{-y}{f_{Coll}} \right) \right] \right) - \theta_0 \right) \right]^i \tag{4.3}$$

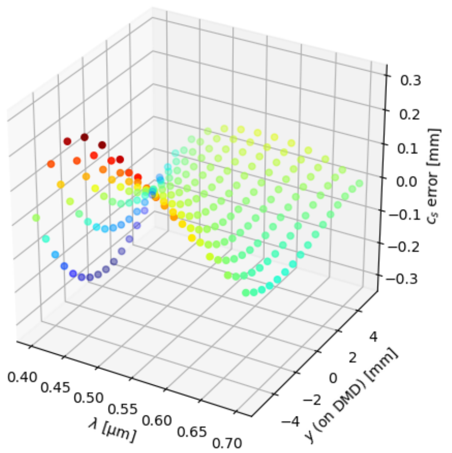


Figure 6: Fit error using equation (4.3)

Order of fit parameters: c_0, c_1, c_2, c_3

```
params
0.00015406
0.99298540
0.21276011
-1.15422118
Maximum error is: 0.2866263923598833 mm
Minimum error is: -0.3073479408301534 mm
Maximum absolute error is: 0.3073479408301534 mm
Minimum absolute error is: 1.8598423970672684e-05 mm
Mean squared error is: 0.010018644223583673 mm^2
```

The maximum absolute error reduced by a factor of around 6 relative to equation (4.1). However, the error is still too much since the pitch of a pixel on the detector is $p_D = 9.00$ µm $= 0.009$ mm. Ideally, the error should be on the same order of magnitude as p_D.

4.2.6 Some other Fit Functions

Functions using the small-angle approximation $\tan\Theta \approx \Theta$, such as the two below, yielded errors on the same order of magnitude as those without small-angle approximation. This suggests that the small-angle approximation does not matter significantly (for now).

$$c_s = f_{Cam}\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \frac{-y}{f_{Coll}}\right]\right) - \theta_0\right) \tag{4.4}$$

$$c_s = f_{Cam}\sum_{i=0}^{3} c_i\left[\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \frac{-y}{f_{Coll}}\right]\right) - \theta_0\right)\right]^i \tag{4.5}$$

The fit functions below were not too effective, or had a greater number of fit parameters but a negligible effect on decreasing the error. The following function is somewhat of a combination of equations (4.2) and (4.3). The error was similar to equation (4.3).

$$c_s = f_{Cam} \sum_{i=0}^{3} c_i \left[\tan\left(a_1 \arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}} \right) \right] \right) - \theta_0 \right) \right]^i \tag{4.6}$$

The following function is very ineffective, and in fact, resulted in much greater error:

$$c_s = f_{Cam} c_3 \left[\tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}} \right) \right] \right) - \theta_0 \right) \right]^3 \tag{4.7}$$

4.3 Fitting by Constraining the Error

4.3.1 Rationale

The above methods did not work too well since the smallest maximum absolute error (with equation (4.3)) is still an order of magnitude greater than what is desired. This is because all of the fit functions above were variants of equation (4.1), which is what c_s should be ideally. While (4.1) models the nonlinearity of the grating, this equation assumes that the collimator and camera optics do not cause distortion. The nonlinearity in the error likely arises not because of the grating, but rather because the collimator and camera optics cause distortion. As in equation (4.1), most of the above fit functions have errors that significantly increase when the values of λ and/or y are different from λ_c and 0 respectively. That is, the errors are generally large for large $|\lambda - \lambda_c|$ and/or large $|y|$. These values of λ and y would have light rays passing through the collimator and camera optics in regions further away from the optical axis and closer to the edge of the lenses. This would result in greater distortion and thus the nonlinearity in error.

For the remainder of this section, I will use fit functions that have the form of equation (4.1) but with some "correction terms" added to reduce the error. That is, I will use fit functions of the form:

$$c_s = f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}} \right) \right] \right) - \theta_0 \right) - \Delta c_s \tag{4.8}$$

where Δc_s is the error in c_s caused by distortion. The $-\Delta c_s$ term above would be the "correction term", which I will attempt to model.

4.3.2 Basic cubic correction term

From figure 3a, the error looks somewhat like the surface $\Delta c_s = \tilde{a}(\lambda - \lambda_c)^3 + \tilde{b}y^3$, where $\tilde{a}, \tilde{b} > 0$. Thus, I tried the fit function:

$$c_s = f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}} \right) \right] \right) - \theta_0 \right) + a(\lambda - \lambda_c)^3 + by^3 \tag{4.9}$$

where $a = -\tilde{a}$ and $b = -\tilde{b}$. The fit error is shown below.

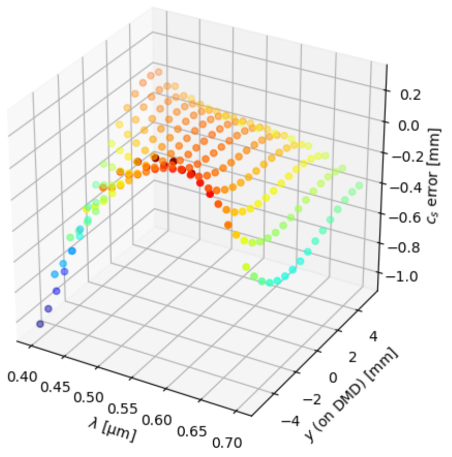


Figure 7: Fit error using equation (4.9)

Order of fit parameters: a, b

```
params
-132.77445999
-0.00158276
Maximum error is: 0.26540026612995193 mm
Minimum error is: -1.0318352242566284 mm
Maximum absolute error is: 1.0318352242566284 mm
Minimum absolute error is: 4.7865423178139466e-05 mm
Mean squared error is: 0.07027255738403945 mm^2
```

The values of the fit parameters satisfy $a < 0$ and $b < 0$ which is consistent with what I expected. The maximum absolute error is less than when using equation (4.1) but is greater than the error when using equation (4.3). This suggests that equation (4.9) does not quite capture the error caused by distortion.

4.3.3 Cubic function in λ with coefficients that are functions of y

Upon closer inspection of figure 3, we see in figure 3c that Δc_s appears to be a cubic function of λ if y is fixed. Similarly, Δc_s appears to be a cubic function of y if λ is fixed. So, I used the following fit function:

$$\begin{aligned} c_s =& f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \\ &+ \left(\sum_{i=0}^{3} a_i y^i\right)(\lambda - \lambda_c)^3 + \left(\sum_{i=0}^{3} b_i y^i\right)(\lambda - \lambda_c) \end{aligned} \tag{4.10}$$

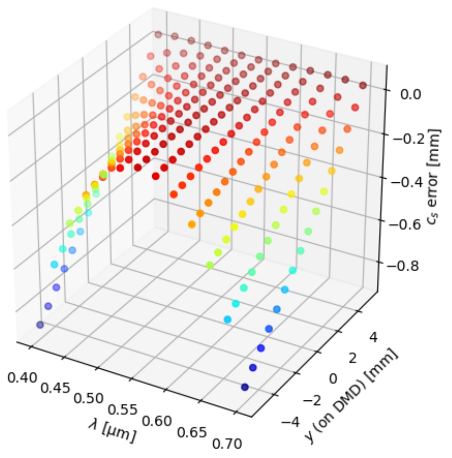


Figure 8: Fit error using equation (4.10)

Order of fit parameters: $a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0$

```
params
-0.00312199
-0.00779984
4.15174268
-104.12148031
-0.00057715
-0.04231994
0.29537506
-0.08826998
Maximum error is: 0.04092198883052345 mm
Minimum error is: -0.8764230262587542 mm
Maximum absolute error is: 0.8764230262587542 mm
Minimum absolute error is: 0.00015115735809256847 mm
Mean squared error is: 0.06809868926635546 mm^2
```

From figure 8 above, we can clearly see that the resulting error of the fit has the form:

$$(\tilde{c_1}y + \tilde{c_0})(\lambda - \lambda_c)^2$$

where $\tilde{c_1} > 0$ and $\tilde{c_0} < 0$. Therefore, equation (4.10) can be improved by adding this term.

4.3.4 Cubic function in λ with coefficients that are functions of y, with $(\lambda - \lambda_c)^2$ term

I then used the following fit function:

$$\begin{aligned} c_s =& f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \\ &+ \left(\sum_{i=0}^{3} a_i y^i\right)(\lambda - \lambda_c)^3 + \left(\sum_{i=0}^{3} b_i y^i\right)(\lambda - \lambda_c) + (c_1 y + c_0)(\lambda - \lambda_c)^2 \end{aligned} \tag{4.11}$$

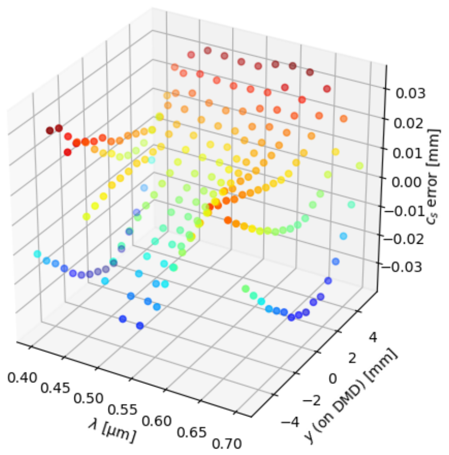


Figure 9: Fit error using equation (4.11)

Order of fit parameters: $a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0, c_1, c_0$

```
params
-0.00317003
-0.00789347
4.15304053
-104.11772851
-0.00057638
-0.04231818
0.29535577
-0.08832732
-3.78370436
18.64086944
Maximum error is: 0.03267779360071188 mm
Minimum error is: -0.03528356788464482 mm
Maximum absolute error is: 0.03528356788464482 mm
Minimum absolute error is: 4.341172193900178e-05 mm
Mean squared error is: 0.00028732642785843127 mm^2
```

We can see that the fit error has reduced significantly when compared to the baseline fit using equation (4.1). In addition, $c_1 = -\tilde{c_1} < 0$ and $c_0 = -\tilde{c_0} > 0$ as expected. However, maximum absolute error is still around $4p_D$, and from figure 9, it appears that the error is still noticeable for large $|y|$. The error is also noticeable for λ near the maximum and minimum wavelengths ($\lambda_{max} = 0.7$ µm and $\lambda_{min} = 0.4$ µm respectively).

4.3.5 Cubic function in λ with coefficients that are functions of y, with $(\lambda - \lambda_c)^2$ term and small-angle approximation

Consider equation (4.11) with the small-angle approximation:

$$\begin{aligned} c_s = & f_{Cam}\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \frac{-y}{f_{Coll}}\right]\right) - \theta_0\right) \\ & + \left(\sum_{i=0}^{3} a_i y^i\right)(\lambda - \lambda_c)^3 + \left(\sum_{i=0}^{3} b_i y^i\right)(\lambda - \lambda_c) + (c_1 y + c_0)(\lambda - \lambda_c)^2 \end{aligned} \tag{4.12}$$

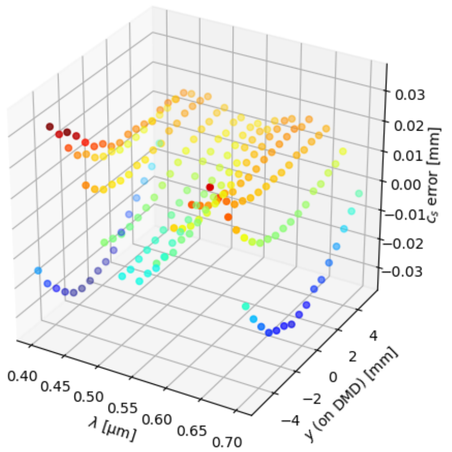


Figure 10: Fit error using equation (4.12)

Order of fit parameters: $a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0, c_1, c_0$

```
params
-0.00208969
0.03851854
4.51848259
-88.68649333
-0.00036567
-0.02401991
0.29526356
-0.09032228
-2.69538390
18.87678109
Maximum error is: 0.03381003378278713 mm
Minimum error is: -0.03310682424703337 mm
Maximum absolute error is: 0.03381003378278713 mm
Minimum absolute error is: 0.00011115364114677 82 mm
Mean squared error is: 0.00022816661928701704 mm^2
```

The mean squared error reduced slightly, and it appears that the error at large y is smaller and also more “flat”.

4.3.6 Factoring out $(\lambda - \lambda_c)$

Equations (4.11) and (4.12) have significantly lower errors than equation (4.1). However, the maximum absolute error is still around $4p_D$ but 10 fit parameters were used. In addition to having a low error, is also desired that a smaller number of fit parameters is used. Looking more closely at figure 3c, for each fixed y, Δc_s is a cubic function of λ with a zero at $\lambda = \lambda_c$. Thus, it seems reasonable to express the Δc_s from equation (4.1) as:

$$\Delta c_s = (\lambda - \lambda_c)h(\lambda, y)$$

where $h(\lambda, y)$ is a function of λ and y, represented by:

$$h(\lambda, y) = \tilde{a}(y)\lambda^2 + \tilde{b}(y)\lambda + \tilde{c}(y)$$

$\tilde{a}(y)$, $\tilde{b}(y)$, and $\tilde{c}(y)$ are some functions of y. $h(\lambda, y)$ is plotted below. Note that the points $\lambda = \lambda_c$ are not included in the plot because of error due to division by 0. I plotted $h(x, y)$ by plotting $\Delta c_s/(\lambda - \lambda_c)$, where Δc_s is from the fit with equation (4.1).

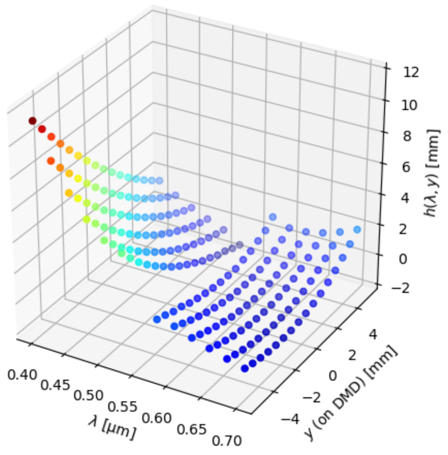


Figure 11: Plot of $h(\lambda, y)$

Through further experimentation, I found that it was sufficient to model $\tilde{a}(y)$, $\tilde{b}(y)$, and $\tilde{c}(y)$ as quadratic functions of y. I then used the following fit equation:

$$\begin{aligned} c_s =& f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \\ &+ (\lambda - \lambda_c)\left(\left[\sum_{i=0}^{2} a_i y^i\right]\lambda^2 + \left[\sum_{i=0}^{2} b_i y^i\right]\lambda + \left[\sum_{i=0}^{2} c_i y^i\right]\right) \end{aligned} \tag{4.13}$$

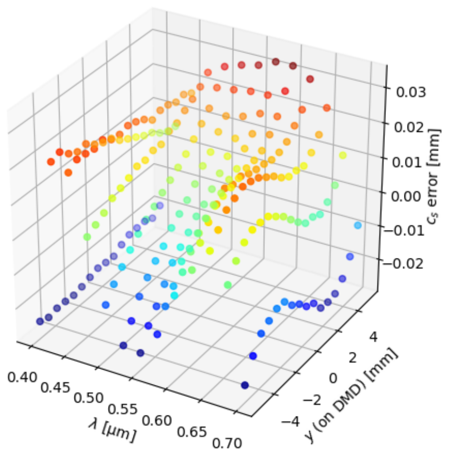


Figure 12: Fit error using equation (4.13)

Order of fit parameters: $a_2, a_1, a_0, b_2, b_1, b_0, c_2, c_1, c_0$

```
params
-0.00810652
4.09428871
-104.11580816
-0.03910319
-8.28740967
133.66642856
-0.01835615
3.60418737
-42.10985694
Maximum error is: 0.03198724034124467 mm
Minimum error is: -0.025522982468480393 mm
Maximum absolute error is: 0.03198724034124467 mm
Minimum absolute error is: 4.811537086446549e-05 mm
Mean squared error is: 0.0002626999557272172 mm^2
```

9 fit parameters were used, and the maximum absolute error was less than that of equation (4.11) by about 0.003 mm (where 10 fit parameters were used). The mean squared error was also slightly less.

4.3.7 Factoring out $(\lambda - \lambda_c)$, with small-angle approximation

Small-angle approximation for equation (4.11) was somewhat useful (in equation (4.12)), and so I tried this with equation (4.13) as well:

$$\begin{aligned} c_s =& f_{Cam}\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \frac{-y}{f_{Coll}}\right]\right) - \theta_0\right) \\ &+ (\lambda - \lambda_c)\left(\left[\sum_{i=0}^{2} a_i y^i\right]\lambda^2 + \left[\sum_{i=0}^{2} b_i y^i\right]\lambda + \left[\sum_{i=0}^{2} c_i y^i\right]\right) \end{aligned} \tag{4.14}$$

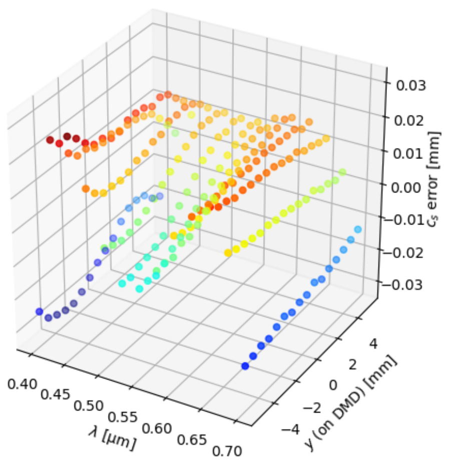


Figure 13: Fit error using equation (4.14)

Order of fit parameters: $a_2, a_1, a_0, b_2, b_1, b_0, c_2, c_1, c_0$

```
params
0.03842264
4.47961235
-88.68547392
-0.07471063
-7.62295749
116.76737318
0.00544959
3.12600700
-37.48503776
Maximum error is: 0.029804677826783532 mm
Minimum error is: -0.030871752550767795 mm
Maximum absolute error is: 0.030871752550767795 mm
Minimum absolute error is: 6.0235198588109995e-05 mm
Mean squared error is: 0.00021680640529372536 mm^2
```

Like with the small-angle approximation in equation (4.12), the mean squared error reduced slightly, and it appears that the error at large y is smaller and also more "flat". The maximum absolute error here was less than that of equation (4.12) by about 0.003 mm.

4.3.8 Factoring out $(\lambda - \lambda_c)$, with small-angle approximation and quartic term in λ

Looking at figure 13, for each fixed y, the resulting error seems to be a quartic function of λ with a zero at $\lambda = \lambda_c$ of order 2. For each fixed y, the error in equation (4.14) can be roughly modelled by:

$$\tilde{d_1}(\lambda - \lambda_c)^2(\lambda - \lambda_c + \tilde{d_0})(\lambda - \lambda_c - \tilde{d_0})$$

For simplicity (and so that there are not too many fit parameters), I will take $\tilde{d_1}$ and $\tilde{d_0}$ as constants and not as functions of y. Note that this equation above assumes that the error is symmetric about

$\lambda = \lambda_c$, which is not actually the case in figure 13, but is a rough approximation (again, to avoid too many fit parameters). I then used the fit function:

$$\begin{aligned} c_s =& f_{Cam}\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \frac{-y}{f_{Coll}}\right]\right) - \theta_0\right) \\ &+ (\lambda - \lambda_c)\left(\left[\sum_{i=0}^{2} a_i y^i\right]\lambda^2 + \left[\sum_{i=0}^{2} b_i y^i\right]\lambda + \left[\sum_{i=0}^{2} c_i y^i\right]\right) + d_1(\lambda-\lambda_c)^2(\lambda-\lambda_c+d_0)(\lambda-\lambda_c-d_0) \end{aligned} \tag{4.15}$$

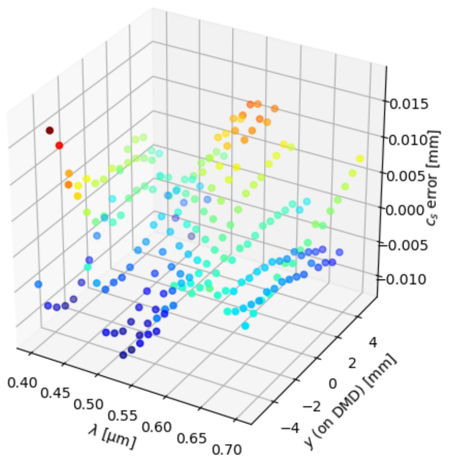


Figure 14: Fit error using equation (4.15)

Order of fit parameters: $a_2, a_1, a_0, b_2, b_1, b_0, c_2, c_1, c_0, d_1, d_0$

```
params
0.03850292
4.47961615
-88.68630459
-0.07479918
-7.62296167
219.10272296
0.00547277
3.12600809
-93.76921598
222.59958104
0.69150408
Maximum error is: 0.01761576165692169 mm
Minimum error is: -0.011036183970152713 mm
Maximum absolute error is: 0.01761576165692169 mm
Minimum absolute error is: 1.0133890437025173e-05 mm
Mean squared error is: 3.1481205813288916e-05 mm^2
```

There are 11 fit parameters in equation (4.15). However, as seen above, the mean squared error has reduced by an order of magnitude relative to equation (4.14). In addition, the maximum absolute error is now around $2p_D$. Equation (4.15) has the best performance among all of the fit functions discussed so far. A plot of (4.15) is shown below, represented by a surface (with a red-blue colormap to represent the c_s values). The actual points (λ, y, c_s) are also plotted. As seen from the plot, equation (4.15) fits the actual data well.

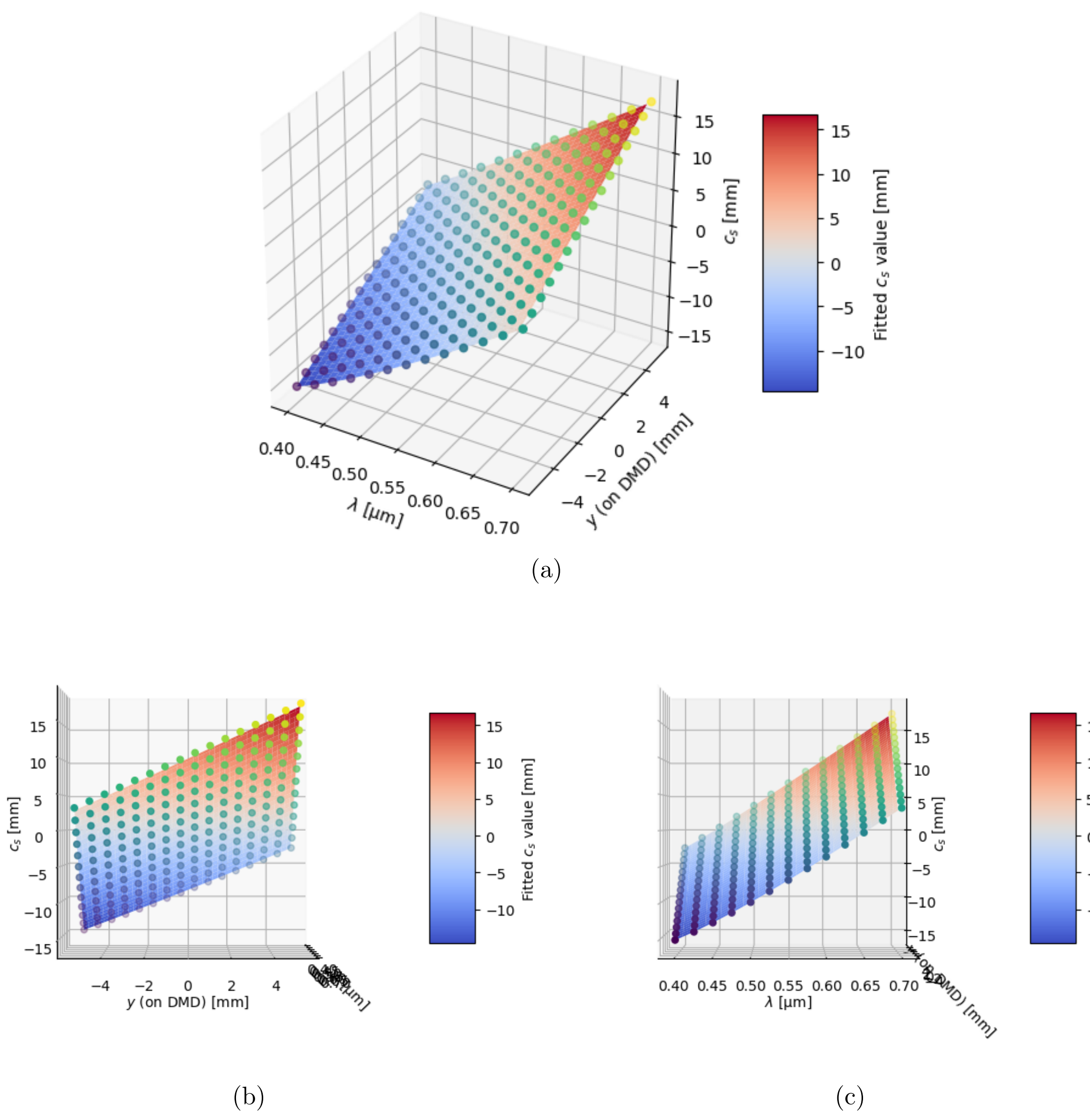


Figure 15: Plot of the fit using (4.15), and actual points (λ, y, c_s)

4.3.9 Some other Fit Functions

I also tried some other fit functions, which had more fit parameters, but also a negligible effect on decreasing the error. They are provided below just for information, and should not be used.

Cf. equation (4.11):

$$\begin{aligned} c_s =& f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \\ &+ \left(\sum_{i=0}^{3} a_i y^i\right)(\lambda-\lambda_c)^3 + \left(\sum_{i=0}^{3} b_i y^i\right)(\lambda-\lambda_c) + \left(\sum_{i=0}^{3} c_i y^i\right)(\lambda-\lambda_c)^2 \end{aligned} \tag{4.16}$$

$$\begin{aligned} c_s =& f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \\ &+ (d_1 y + d_0)(\lambda-\lambda_c)^4 + \left(\sum_{i=0}^{3} a_i y^i\right)(\lambda-\lambda_c)^3 + \left(\sum_{i=0}^{3} b_i y^i\right)(\lambda-\lambda_c) + (c_1 y + c_0)(\lambda-\lambda_c)^2 \end{aligned} \tag{4.17}$$

$$\begin{aligned} c_s =& f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \\ &+ \left(\sum_{i=0}^{3} d_i y^i\right)(\lambda-\lambda_c)^4 + \left(\sum_{i=0}^{3} a_i y^i\right)(\lambda-\lambda_c)^3 + \left(\sum_{i=0}^{3} b_i y^i\right)(\lambda-\lambda_c) + (c_1 y + c_0)(\lambda-\lambda_c)^2 \end{aligned} \tag{4.18}$$

Cf. equation (4.13):

$$\begin{aligned} c_s =& f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \\ &+ \left(\sum_{i=0}^{3} a_i y^i\right)(\lambda-\lambda_c)(\lambda-(b_1 y + b_0))(\lambda-(c_1 y + c_0)) \end{aligned} \tag{4.19}$$

$$\begin{aligned} c_s =& f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \\ &+ \left(\sum_{i=0}^{3} a_i y^i\right)(\lambda-\lambda_c)\left(\lambda - \left[\sum_{i=0}^{3} b_i y^i\right]\right)\left(\lambda - \left[\sum_{i=0}^{3} c_i y^i\right]\right) \end{aligned} \tag{4.20}$$

$$\begin{aligned} c_s =& f_{Cam} \tan\left(\arcsin\left(G\lambda - \sin\left[\theta_0 + \arctan\left(\frac{-y}{f_{Coll}}\right)\right]\right) - \theta_0\right) \\ &+ (\lambda-\lambda_c)\left(\left[\sum_{i=0}^{3} a_i y^i\right]\lambda^2 + \left[\sum_{i=0}^{3} b_i y^i\right]\lambda + \left[\sum_{i=0}^{3} c_i y^i\right]\right) \end{aligned} \tag{4.21}$$

5 Fitting a_s with SciPy Curve Fitting Module

5.1 Relationships between a_s, λ, y, and c_s

Now, I will attempt to find a fit function for a_s. Perhaps a_s is a function of c_s or a function of y. Consider the plots below comparing a_s with c_s, and comparing a_s with y.

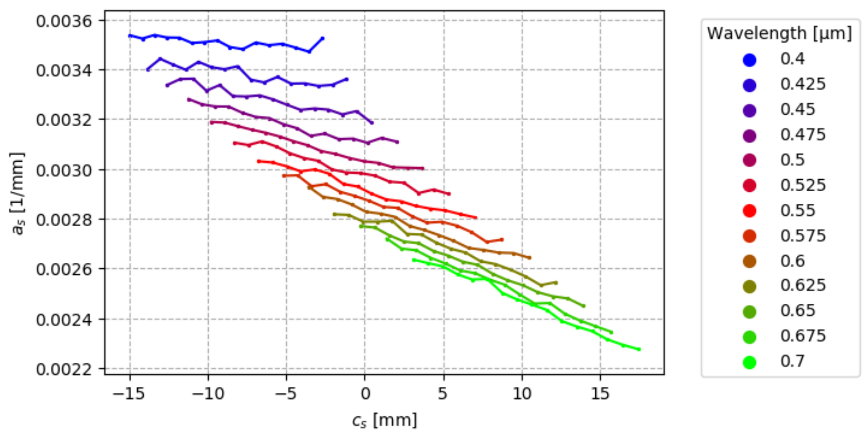


Figure 16: a_s VS c_s

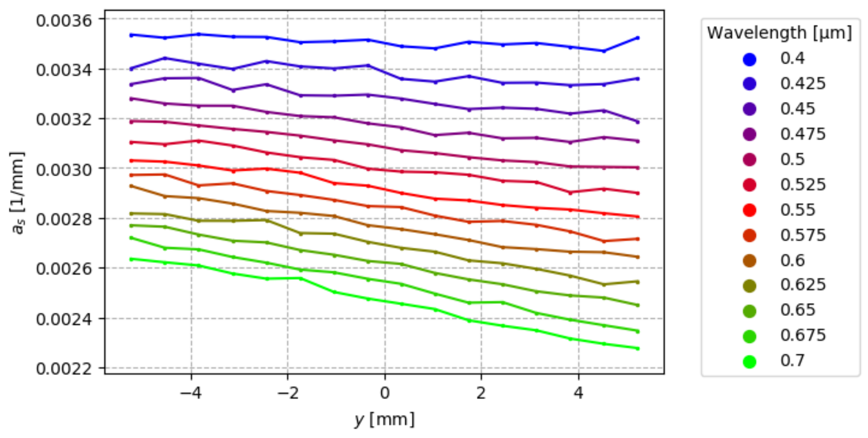
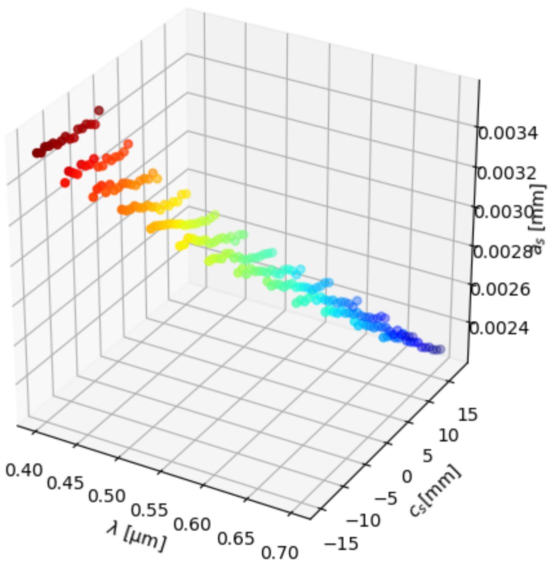
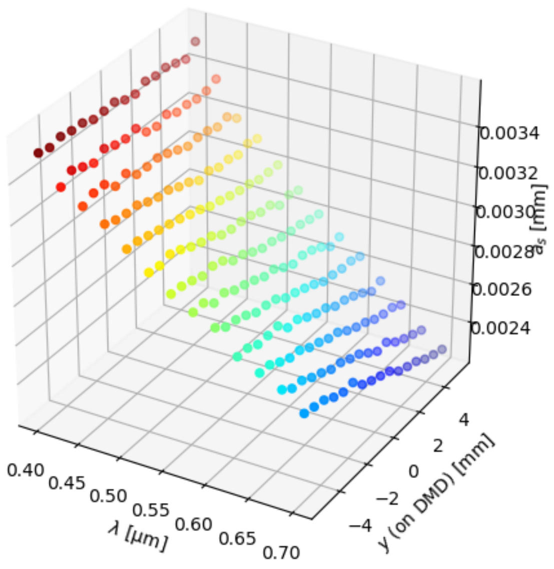


Figure 17: a_s VS y

As seen in the figures above, the relationship between a_s and c_s looks more complicated, possibly due to the nonlinearity in c_s, as discussed in section 4. Looking more closely at the relationship with λ in figure 18 below, it appears that the surface in figure 18a is more curved than the surface in figure 18b. However, there appears to be nonlinear relationships between a_s, λ, and y as well, as was with the case of c_s. The relationship does not appear to be too trivial, since for each fixed λ, the relationship between a_s and y seems to be roughly linear, but with some "fluctuations" as seen in figure 17. These "fluctuations" may be due to distortion and other nonidealities, or may just be uncertainties from the GIA simulation since only 1000 rays were used for each IMA file for this current fit. I will assume that the latter is the case for now.



(a) a_s VS c_s and λ



(b) a_s VS y and λ

Figure 18: Plots showing the relationship between a_s, λ, y, and c_s

5.2 Basic Fit

It may be easier to fit a_s as a function of λ and y, rather than a_s as a function of λ and c_s. For each fixed λ, we could possibly model a_s as:

$$a_s = a(\lambda)y + b(\lambda)$$

where $a(\lambda)$ and $b(\lambda)$ are nonlinear functions of λ. By looking at figure 18b, I decided to represent $a(\lambda)$ and $b(\lambda)$ as quadratic functions of λ. I used the fit function:

$$a_s = \left(\sum_{i=0}^{2} a_i\right) y + \left(\sum_{i=0}^{2} b_i\right) \tag{5.1}$$

where a_2, a_1, a_0, b_2, b_1, and b_0 are fit parameters. Using the `scipy.optimize.curve_fit` module, equation (5.1) was fitted and the resulting error is shown below.

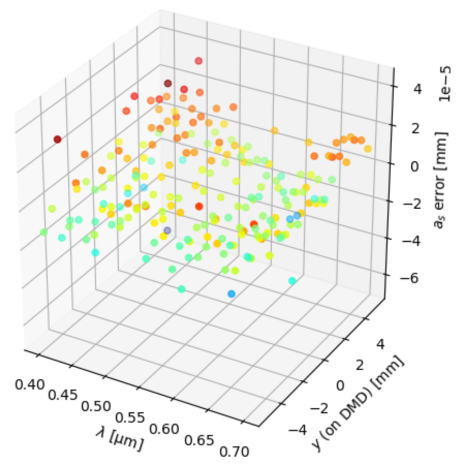


Figure 19: Fit error using equation (5.1)

Order of fit parameters: $a_2, a_1, a_0, b_2, b_1, b_0$

```
params
0.00014553
-0.00025758
0.00007347
0.00272086
-0.00638631
0.00560963
Maximum error is: 4.124831242875281e-05 mm
Minimum error is: -6.55823105701705e-05 mm
Maximum absolute error is: 6.55823105701705e-05 mm
Minimum absolute error is: 5.5548406800980804e-08 mm
Mean squared error is: 2.2926823147538938e-10 mm^2
```

The maximum absolute error was a bit less than 3% of a_s. The error of the fit (Δa_s) in figure 19 does not seem to exhibit clear patterns in that the points are mostly distributed evenly above and below the horizontal plane $\Delta a_s = 0$, which is desired. However, further analysis needs to be conducted in order to determine if this error is acceptable.

6 Conclusion and Next Steps

The best fit for c_s was equation (4.15), which had a maximum absolute error of less than $2p_D$. A fit was found for a_s (equation (5.1)) with a maximum absolute error less than 3% of a_s. Next, I will try using Markov chain Monte Carlo to fit c_s using equation (4.1) and using G, f_{Coll}, f_{Cam}, and θ_0 as fit parameters, seeing if a better fit could be obtained. For this part, I would have to first obtain the errors/uncertainties of the centroid coordinates from k-means. Later, I will use 500000 to 1 million rays with GIA for each IMA file, and then try out the fits again to see if the errors are acceptable. I will also explore other fit functions for a_s.