DMD-MOS Project Smile Fit Progress Report

Matthew Leung
Dunlap Institute for Astronomy & Astrophysics, University of Toronto
October 17, 2020

1. Introduction

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

$$y_d = f(x_d) = a_s x_d^2 + c_s \tag{1}$$

Note that c_s is a function of λ and slit position y_{slit}. This report will focus on c_s. For each fixed y_{slit}, we will try fitting c_s as a linear function of λ:

$$c_s = m\lambda + b \tag{2}$$

2. Project Code Files Structure

Since we are now working with many IMA files, resulting in many Zemax Geometric Image Analysis (GIA) outputs in TXT format, I had to restructure all of the Python scripts and the way the project files were set up, so that the large amount of data files could be dealt with efficiently. The directory structure of the project is shown below:

```
project_directory/
├── get_ima_gia_files_info.py
├── find_centroids_V2.py
├── smile_fit.py
├── plot_lambda_c_y.py
└── dataset/
    ├── ima/
    ├── gia/
    ├── centroids/
    ├── visualizations/
    ├── smile_results/
    └── wavelengths.wav
```

The script `get_ima_gia_files_info.py` contains the necessary functions to extract data from the IMA and GIA data files (which are located in the `ima/` and `gia/` directories respectively). The script `find_centroids_V2.py` uses k-means clustering to cluster points together, and saves the coordinates of the centroids of the clusters into CSV format in the `centroids/` directory, along with plots of the GIA data in the `visualizations/`

directory. The script `smile_fit.py` does all of the fitting and places most of the results in the `smile_results/` directory. The script `plot_lambda_c_y.py` creates a 3D plot of λ, c_s, and the slit position y_{slit} on the image detector.

First, we run `find_centroids_V2.py` in order to obtain all of the centroids. k-means algorithm can take a long time to run with many points, and so I decided to separate the k-means clustering from the rest of the fitting to save on time. This script only needs to be run once, and centroids are saved in CSV format and can be imported to the other scripts later, without having to go through the entire k-means clustering process again. Next, we run `smile_fit.py` in order to fit the centroids using equation (1). A CSV file `lambda_c_y.csv` is saved, and this stores the set of all points (λ, c_s, y_{slit}). This can be visualized using `plot_lambda_c_y.py`.

3. Results

I used equation (2) to fit c_s as a function of λ and the results are shown in the figures below.

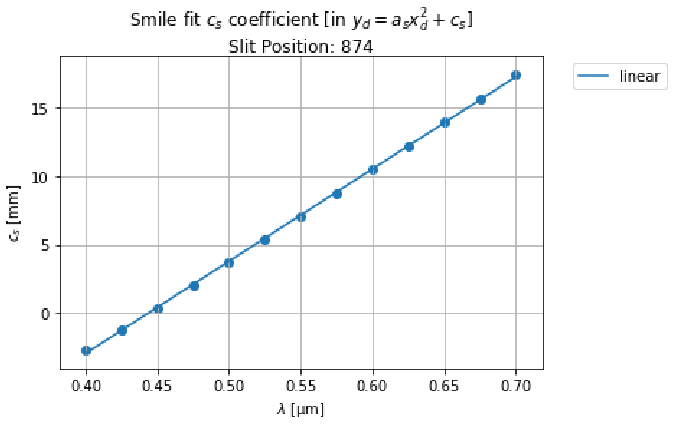

Figure 1: Example: fit using equation (2) for slit position at micromirror number 874

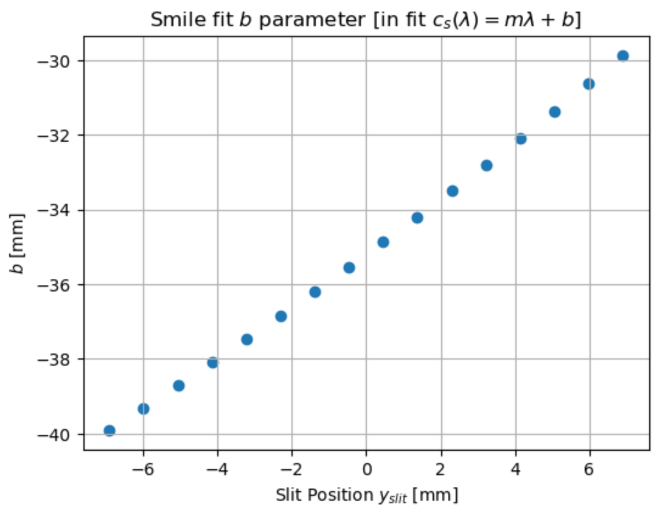

Figure 2: b compared with slit position y_{slit}

From Figure 2, it appears that b is proportional to y_{slit} which is expected. Upon closer inspection of the graph, the relationship appears nonlinear, as a line cannot pass through all of the points. A concave up function would appear to fit all of the points. However, the relationship between b and y_{slit} appears sufficiently linear in the plot above.

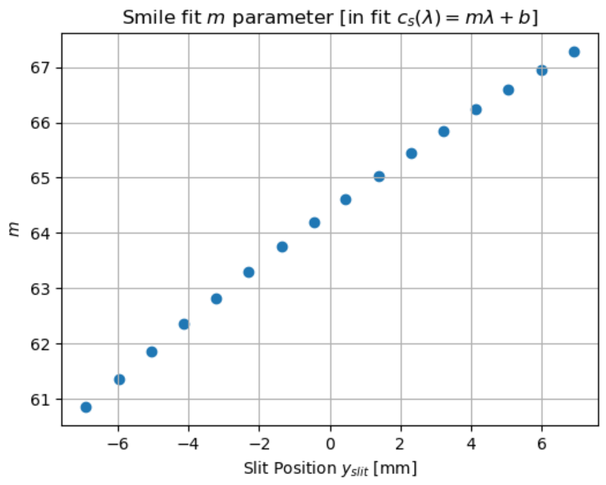

Figure 3: m compared with y_{slit}

I did not expect the relationship shown in Figure 3 between m and y_{slit}. Figure 3 shows that m changes with y_{slit}, and is not a constant. Thus for different y_{slit}, we cannot assume that the set of lines each with equation $c_s = m\lambda + b$ are merely just vertically shifted versions of each other. It appears that m varies nonlinearly with y_{slit} and can be fitted with a concave down function.

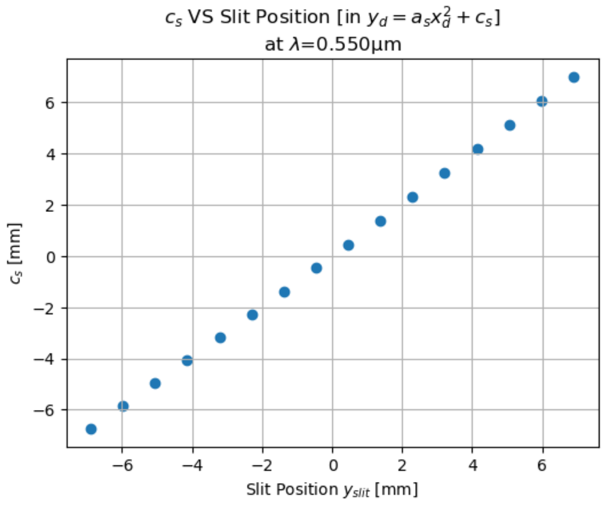

Figure 4: c_s compared with y_{slit} at centre wavelength λ = 0.550µm

In Figure 4, we should expect that the straight line passing through all the points should obey the relationship $c_s = y_{slit}$ since we are using the centre wavelength. However, upon closer inspection, when I zoomed into the plot, I realized this is not the case. In fact, at λ = 0.550µm, I noticed that the difference between c_s and y_{slit} could be greater than 0.1mm. Consider when the slit position is at micromirror number 874, meaning y_{slit} = 6.90149mm, shown in Figure 5 below. It can be seen that the c_s value here is around 7.01mm. The difference between c_s and y_{slit} is more than 0.1mm. Thus, c_s could vary nonlinearly with y_{slit}.

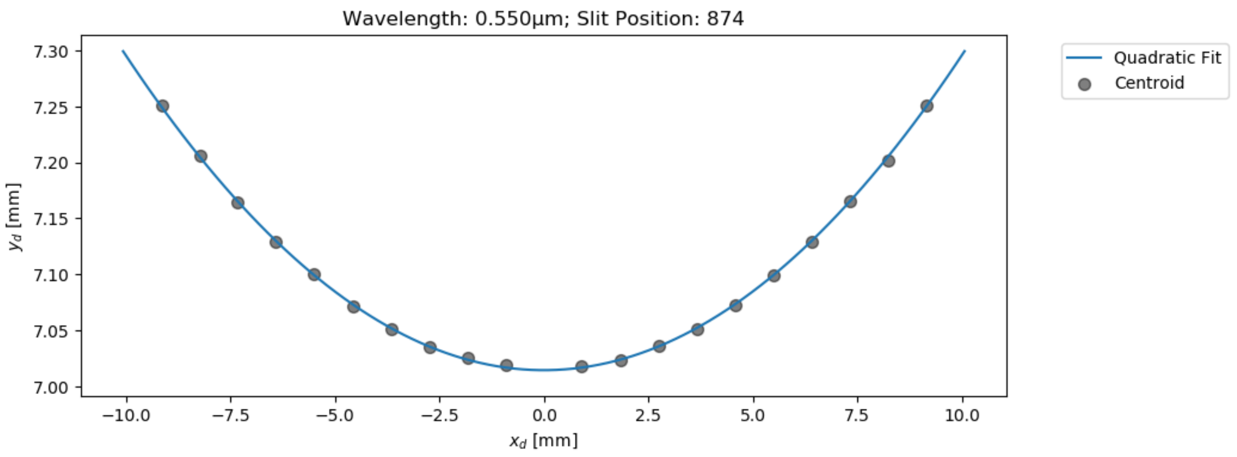

Figure 5: Quadratic fit at λ = 0.550µm for slit position at micromirror 874

The nonlinear relationships are shown when we see 3D plots of each (λ, c_s, y_{slit}). Consider the 3D plots shown below, generated using `plot_lambda_c_y.py`. If the relationships were linear, then a plane could pass through all of the points. However, this is not the case. A curved surface would be required to pass through all of the points. The relationships between the variables may need to be reconsidered.

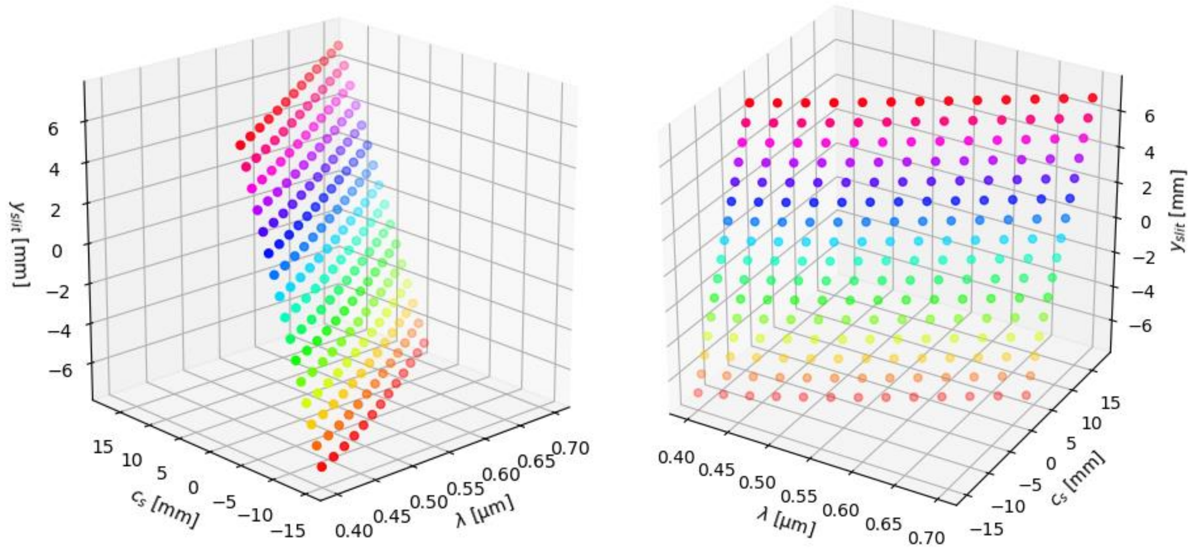

Figure 6: 3D plots comparing λ, c_s, and y_{slit}

4. Next Steps

Next, I will explore the concept of expressing c_s as a function of y_{slit} using the grating equation. However, modelling c_s as a linear function of λ may not be accurate as shown in particular by the relationship between m and y_{slit} in Figure 3.