DLA Report 9

Alexey Yermakov
1 February 2022

1 Description of what was done

1.1 Summary

Re-writing my own FSSS code revealed the problem to be how Pivmat and OpenPIV were communicating. OpenPIV's output file isn't structured the way Pivmat is expecting it to be, causing problems when imported. This is easily resolved by swapping the columns of the displacement field's matrices in MATLAB. We are now able to continue using Pivmat and also have a backup repository in case more problems arise with Pivmat.

1.2 Writing FSSS Code

After our last meeting I re-wrote my own FSSS code in a much cleaner and well-documented manner. As a result, we have a repository with working and modular FSSS code (note that scaling is currently a problem, see the section "Scaling" below). This code follows the equations from Moisy's paper and is my attempt at replicating the functionality of Pivmat that we've been using. Please note that the repository is currently private.

The result that we saw last time we met was that analytic functions worked with FSSS without any problems yet for some reason the water droplet still contained the strange structure.

1.2.1 Pivmat's `Loadvec` function

All of the code in the aforementioned repository is written by me except for `intgrad2` and `loadvec`. In the previous report I showed that `intgrad2` was behaving properly. The source of the strange water droplet structure turned out to be in Pivmat's `loadvec` function, which takes in the text file output from OpenPIV and turns it into a struct usable in Pivmat.

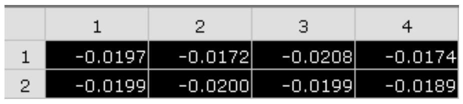
Below is a part of the file generated by OpenPIV from the water droplet images. Each four line section of the file has the same y-value but an incremental increase in x-values. The first two four line pairs are separated by a decrease of one y-value and the second two four line pairs are also separated by a decrease of one y-value. Most of the data from the file is between the two four line pairs and is omitted.

```
# x y u v mask
0.7137 112.6883  -0.0197   0.0172   0.0000
1.0309 112.6883  -0.0199   0.0179   0.0000
```

```
1.3481 112.6883  -0.0323   0.0273   0.0000
1.6653 112.6883  -0.0220   0.0251   0.0000
...
0.7137 112.3711  -0.0172   0.0124   0.0000
1.0309 112.3711  -0.0200   0.0131   0.0000
1.3481 112.3711  -0.0345   0.0211   0.0000
1.6653 112.3711  -0.0126   0.0224   0.0000
...
0.7137   1.0309  -0.0026   0.0229   0.0000
1.0309   1.0309  -0.0057   0.0272   0.0000
1.3481   1.0309  -0.0079   0.0194   0.0000
1.6653   1.0309  -0.0052   0.0149   1.0000
...
0.7137   0.7137  -0.0004   0.0167   0.0000
1.0309   0.7137  -0.0014   0.0214   0.0000
1.3481   0.7137  -0.0015   0.0158   0.0000
1.6653   0.7137  -0.0042   0.0163   1.0000
```

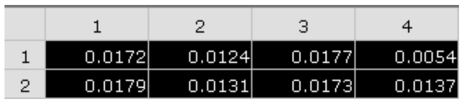
Note that each row contains the (x,y) position of the displacement field vector with components ⟨u,v⟩. The final column labeled `mask` is used to inform us if the vector didn't pass the signal to noise ratio threshold during OpenPIV's processing (1 means it didn't pass, 0 means it did). The final column can be ignored here.

Below is an image of the x and y components of the gradient fields in matrices from the MATLAB struct created by `loadvec`:



	1	2	3	4
1	-0.0197	-0.0172	-0.0208	-0.0174
2	-0.0199	-0.0200	-0.0199	-0.0189

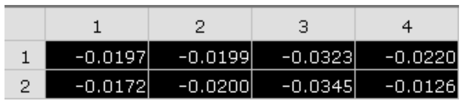
Figure 1: Part of the displacement field's x-components in a 2D matrix.



	1	2	3	4
1	0.0172	0.0124	0.0177	0.0054
2	0.0179	0.0131	0.0173	0.0137

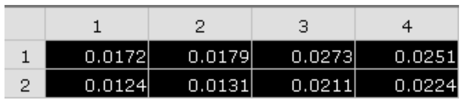
Figure 2: Part of the displacement field's y-components in a 2D matrix.

We first notice that a change in x-coordinate corresponds to a change in rows. This is opposite of what we'd expect since we're using MATLAB's meshgrid function, which relates an increase in x-coordinates to an increase in columns. Transposing the displacement field's x and y component matrices yields the following:



	1	2	3	4
1	-0.0197	-0.0199	-0.0323	-0.0220
2	-0.0172	-0.0200	-0.0345	-0.0126

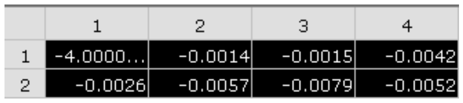
Figure 3: Part of the displacement field's x-components in a transposed 2D matrix.



	1	2	3	4
1	0.0172	0.0179	0.0273	0.0251
2	0.0124	0.0131	0.0211	0.0224

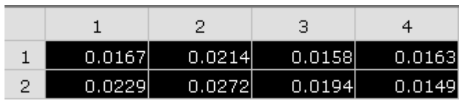
Figure 4: Part of the displacement field's y-components in a transposed 2D matrix.

Now an increase in x-coordinates is be equivalent to an increase in columns. This, however, doesn't solve the problem entirely since we want an increase in y-coordinates to be equivalent to an increase in rows. The latter two figures, however, show that an increase in columns is equivalent to a decrease in y-coordinates. Swapping the top and bottom rows of both matrices results in the displacement field that we'd expect:



	1	2	3	4
1	-4.0000...	-0.0014	-0.0015	-0.0042
2	-0.0026	-0.0057	-0.0079	-0.0052

Figure 5: Part of the displacement field's x-components in a transposed and row-swapped 2D matrix.



	1	2	3	4
1	0.0167	0.0214	0.0158	0.0163
2	0.0229	0.0272	0.0194	0.0149

Figure 6: Part of the displacement field's y-components in a transposed and row-swapped 2D matrix.

Here, an increase in x-coordinates is an increase in columns and an increase in y-coordinates is an increase in rows. Running my FSSS code with the "correct" displacement field matrices produces the desired result of a water droplet with concentric circles:

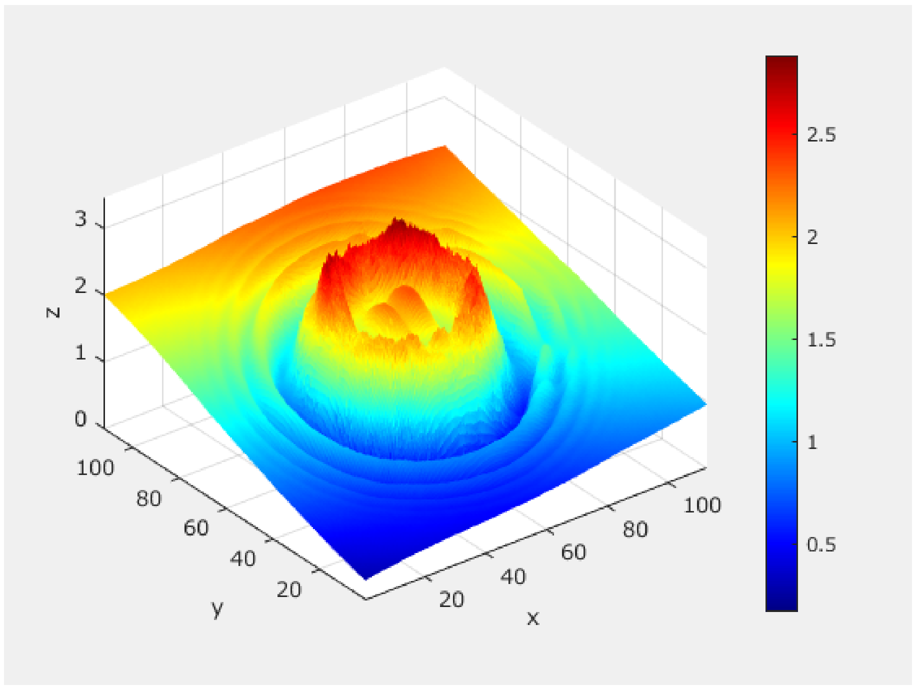


Figure 7: FSSS of a water droplet. No post-OpenPIV processing was done. Experimental measurements weren't used.

Below is the final result using my code and experimental measurements.

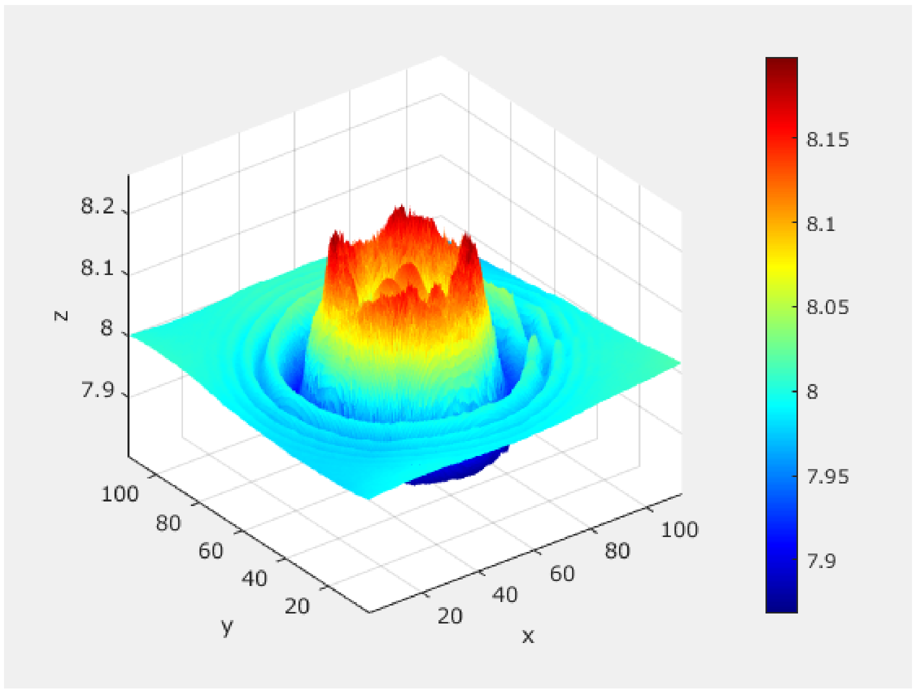


Figure 8: FSSS of a water droplet. Effective water height calculated with experimental measurements. Mean of the displacement field was subtracted before integration. Average height adjusted to the one measured experimentally.

1.2.2 Is Pivmat to Blame?

Looking futher led me to PIVMat's import policy. Here we see that the output from OpenPIV should follow the following format:

```
Headerlines (optional)
x1 y1 u v
x2 y1 u v
x3 y1 u v
...
x1 y2 u v
x2 y2 u v
...
```

Looking at the sample output above shows that the y-values are decreasing with an increase in rows. This means that OpenPIV isn't outputting to the text file in the way Pivmat is expecting it to. Therefore, to use Pivmat with our data instead of using the repository I wrote we can either: swap the columns of the displacement field matrices (since Pivmat associates columns with y-values instead of x-values) or modify the output file from OpenPIV. The former solution is much easier to implement. In fact, I've already implemented it and it looks like it works as we'd expect.

1.2.3 Scaling

Please note that the images I've provided here which are from my code don't have the same z-axis scaling as with Pivmat. I don't know what the problem is immediately, but I think we can rely on Pivmat for now, since we are importing the displacement field properly.

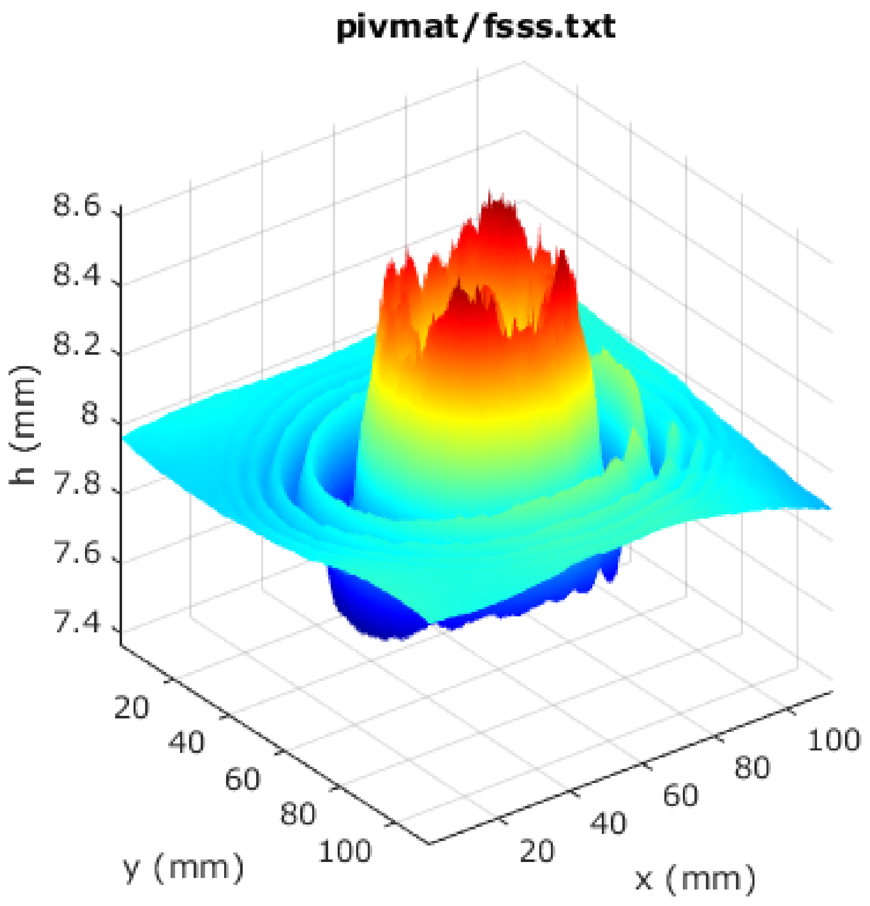


Figure 9: FSSS of a water droplet using Pivmat. The mean of the displacement field was subtracted and the average height was moved to the experimental value.

2 Description of Next Steps

Now that we have a working FSSS solution, the first step should be setting up the water table. The next step would be performing experiments.

3 Questions

- Should I read through another paper about another Synthetic Schlieren method?
- How are we going to implement image registration with the water table?
- What experiment are we planning on setting up first?