DLA Report 8

Alexey Yermakov
18 January 2022

1 Description of what was done

1.1 Summary

Analyzing the source of the water drop's strange structure from report 7 led to identifying the problem being the primary FS-SS function in Pivmat: `surfheight`. A deeper look into the function alleviated the concern of the inverse gradient function `intgrad2` causing problems, but many questions are still unanswered and the water drop's structure was not resolved.

1.2 Analyzing the Water Drops

The last report contains water drops processed in FS-SS that have a strange structure. We expect the water drop surface reconstruction to contain concentric circles of similar height. Instead, however, we see concentric circles with peaks along the x-axis and troughs along the y-axis.

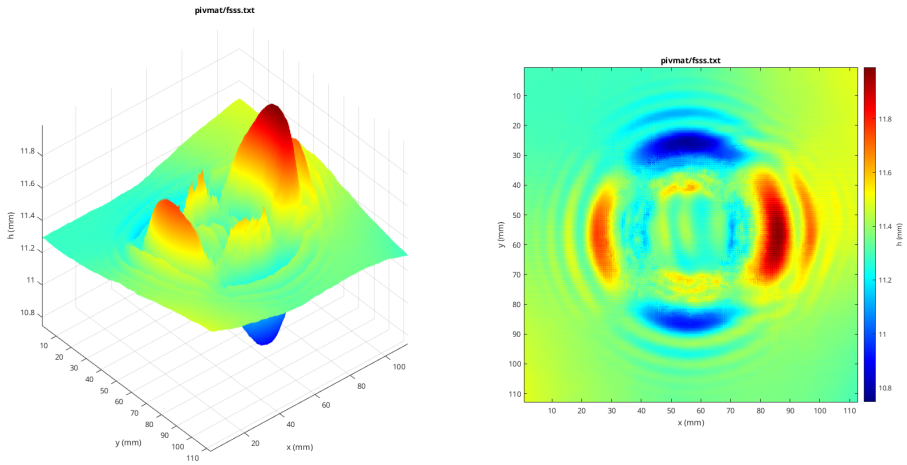

Figure 1: Water Drop from Report 7

To figure out what's causing the problem, I analyzed the three major parts going into making the surface reconstruction: photography, OpenPIV, and Pivmat.

1.2.1 Photography

Recall that we previously found a saddle shape when taking a picture of the glass tray without any water 20 seconds apart.

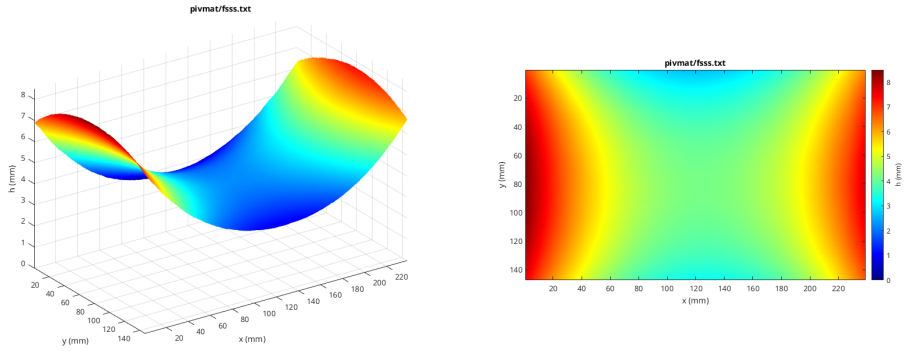

Figure 2: Two Images 20s Apart (Glass Tray)

It was brought up by Sam during one of the group meetings that the saddle shape could be causing the structure in figure 1. As a result, I followed up on Mark's suggestion to image the acrylic blocks and compare with the glass tray.

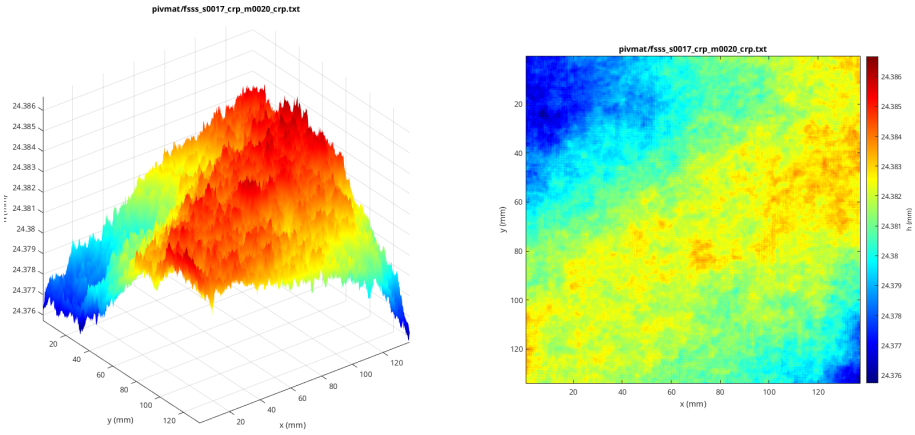

Figure 3: Two Images 20s Apart (Acrylic Slab)

Figure 3 shows us that the glass tray does have some unevenness resulting in a saddle-like shape since the acrylic does not. However, when I first saw this result I wasn't convinced that this accounts for the water drop's structure, since the outer edges of figure 1 seemed relatively flat, which is not what we'd expect with an underlying saddle. Furthermore, until the water table is set up I can't create water drops on anything but the glass tray. So, I analyzed the images of

the water drops in software.

Please note that the different in maximum and minimum height for the acrylic block was smaller in figure 3 than in figure 4 below, where the only difference is the latter uses image registration. Both images used the slope-removal technique.

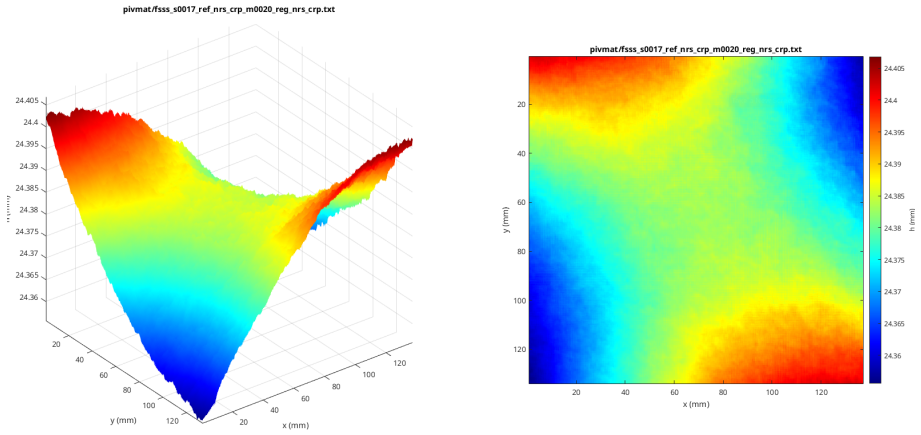

Figure 4: Two Images 20s Apart (Acrylic Slab) with Image Registration

1.2.2 Pivmat or OpenPIV?

So, I had to narrow down the source of the problem to Pivmat and OpenPIV. Luckily, Pivmat contains functions to analyze vector fields. I used these functions to analyze the output of OpenPIV's particle displacement field. We expect concentric circles from the water drop, where all the vectors in the concentric circle are either pointing away or towards the center of the water drop.

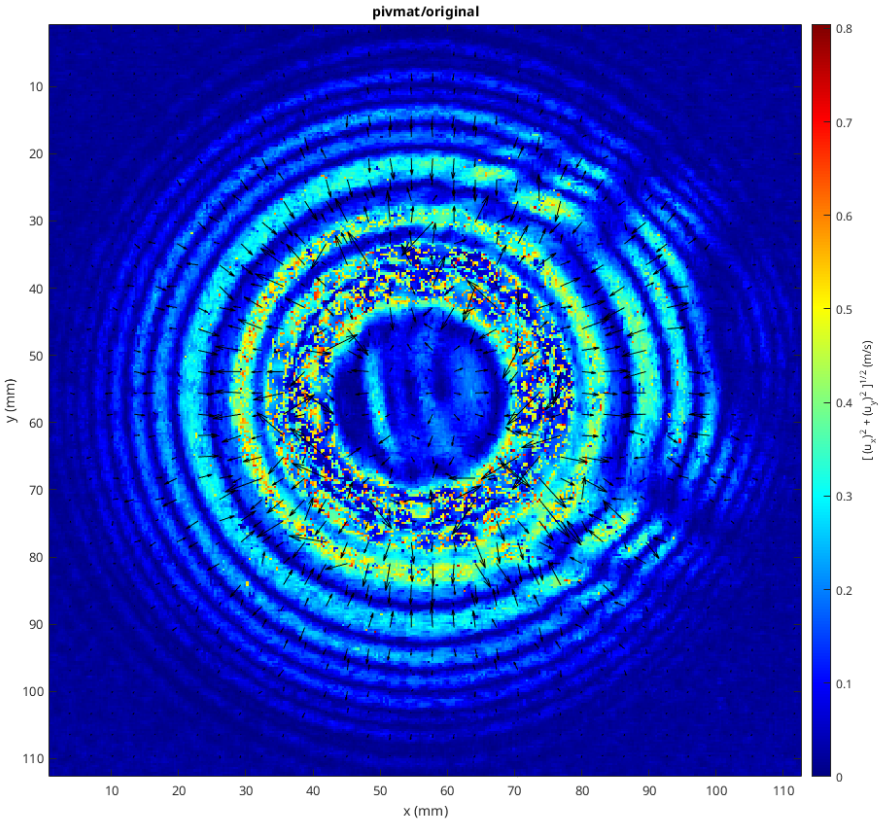

Figure 5: Displacement Field Magnitude

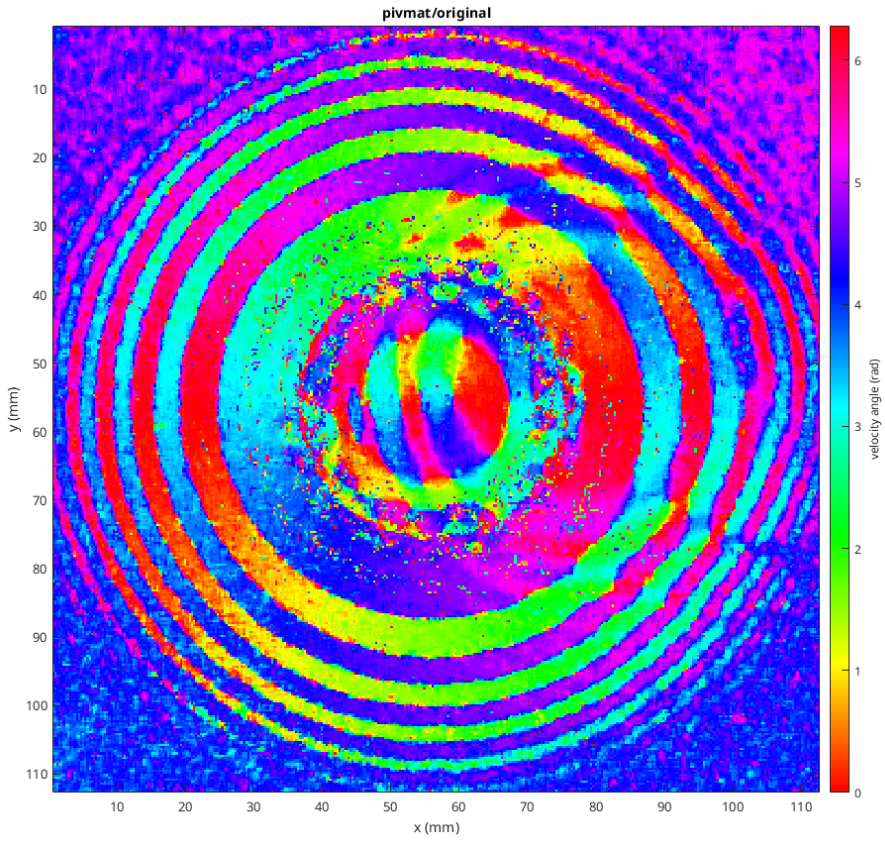

Figure 6: Displacement Field Argument (Angle)

Figure 5 is a 2D plot of the displacement field of the water drop, where the scale represents the magnitude of the vectors. Superimposed on figure 5 are sample vectors showing their directions. Figure 6 is a 2D plot of the displacement field of the water drop, where the scale represents the argument (angle) of the vectors. These two images together suggest that OpenPIV was able to successfully identify that the water droplet created concentric waves of water, since each vector in each concentric circle shares a common magnitude and direction (either away or towards the center). Thus, the problem of the surface reconstruction should belong to Pivmat.

1.2.3 Troubleshooting Pivmat

To prove that the problem exists in Pivmat, I took the original pair of water drop images (the still and moving images) and rotated them 90 degrees. I then rotated back the rotated image in Pivmat by using a provided Pivmat function called `rotatef`.

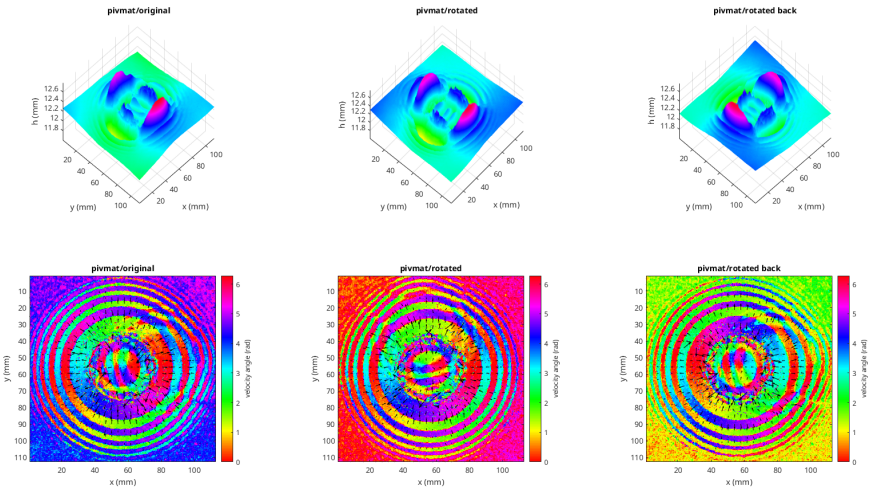

Figure 7: Rotation using `rotatef`. Top row is surface height, bottom row is displacement field's argument (angle). The pairs of images, from left to right, are the original image, rotated by 90 degrees image, and rotated back by 90 degrees image.

The astute observer will notice strange results in figure 7:

- We expect the first two images in the top row to be rotated 90 degrees about each other, but they are not. You can see the peaks and troughs are in the same location!
- The first two pairs of images are using input images that are rotated 90 degrees about each other (see the bottom row's first two images) despite having similar maxima and minima in the surface height.
- The first and last image in the bottom row are not identical, which is what we would expect after two rotations of equal magnitude in different directions (90 degrees counter-clockwise then 90 degrees clockwise).

The only *normal* result is that the final two images in the first row are rotated 90 degrees from one another, as expected.

I found that the above problems are a result of `rotatef` improperly preserving the direction of the vectors relative to the center of the water drop. In fact, the angles of the vectors between the bottom left and bottom right image in the previous figure appear 180 degrees off! (Compare the colors of these images, they are off by about π in the scale.) This, I believe, is what results in the peaks and troughs being inverted between the top right and top left image in the above figure, since FS-SS accounts for the particles moving in opposite directions. I wrote my own `rotatef` function as a result of the above strange results.

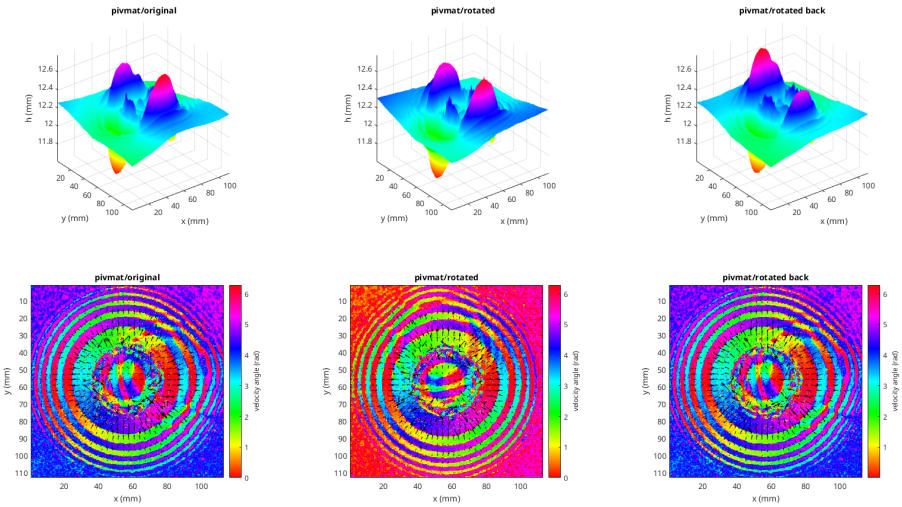

Figure 8: Rotation using my own `rotatef`. Top row is surface height, bottom row is displacement field's argument (angle). The pairs of images, from left to right, are the original image, rotated by 90 degrees image, and rotated back by 90 degrees image.

I believe the above figure to contain a proper rotation function because the bottom left and bottom right image are nearly identical. The interesting bit, however, is that the entire top row of images contains peaks along the y-axis and troughs along the x-axis, despite rotations of 90 degrees between the images. This indicates that the problem is in the `surfheight` function of Pivmat, which runs the Surface height reconstruction for FS-SS.

Once I identified the problem down to the function, I did a couple of troubleshooting steps. First, I recalled a tutorial for FS-SS that mentions using MATLAB 7. So, I installed MATLAB 7 (R2012a) and got the exact same results as shown above. I also decided to see if maybe it was a Linux problem, so I installed MATLAB 7 (R2012a) and the current version of MATLAB (R2012b) on Windows and saw no difference in the figures.

1.2.4 Troubleshooting `surfheight`

We met as a group to discuss the next steps for troubleshooting `surfheight`:

- Define an analytic "radial wave" function $h(x, y)$ centered at the origin by rotating a gaussian curve

  The function is defined as $h(x, y) = 5 \times e^{-(\frac{\sqrt{x^2+y^2}-5.4}{1.7})^2} + 2$

  I also tested with $g(x, y) = 5 \times e^{-(\frac{\sqrt{2\times x^2+y^2}-5.4}{1.7})^2} + 2$ as a sort of "oval gaussian" wave (the only difference between this and $h(x, y)$ is that there's a constant of 2 before x^2)

Lastly, I tested a built-in MATLAB function called `peaks` which is multiple Gaussian hills

- Obtain the gradient field of $h(x, y)$ and use it in `intgrad2`, the inverse gradient function used in `surfheight`
- Check the MATLAB file exchange for other inverse gradient functions
- Try different inverse gradient functions on the gradient field I have from OpenPIV

Below are the relevant figures collected from the above steps followed by a discussion.

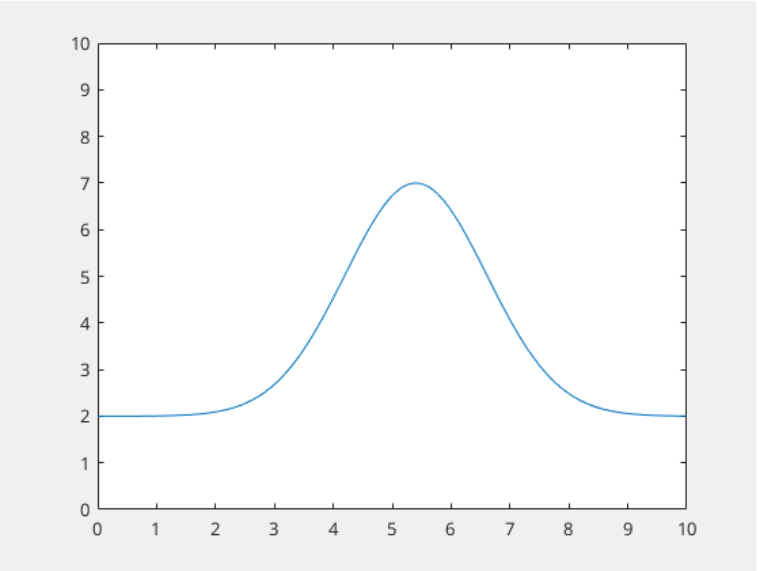

Figure 9: 2D Gaussian curve defined by $f(x) = 5 \times e^{-(\frac{x-5.4}{1.7})^2} + 2$

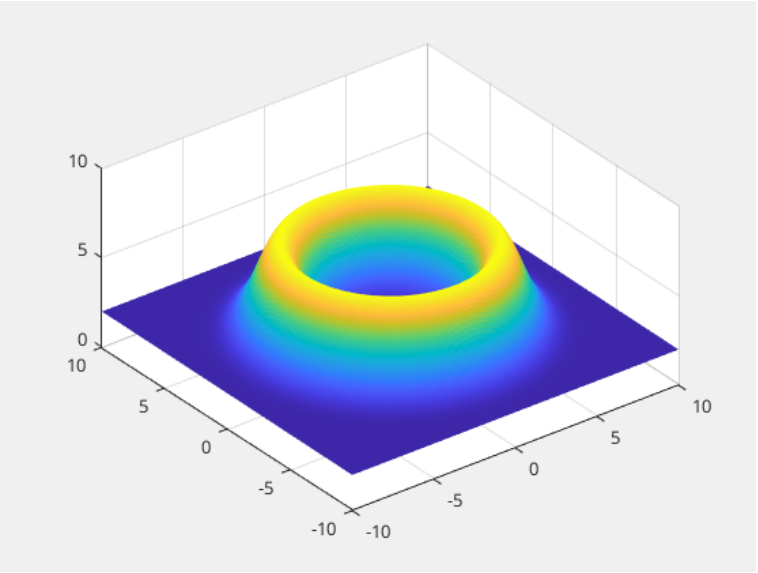

Figure 10: Radial wave defined by $h(x, y)$ above

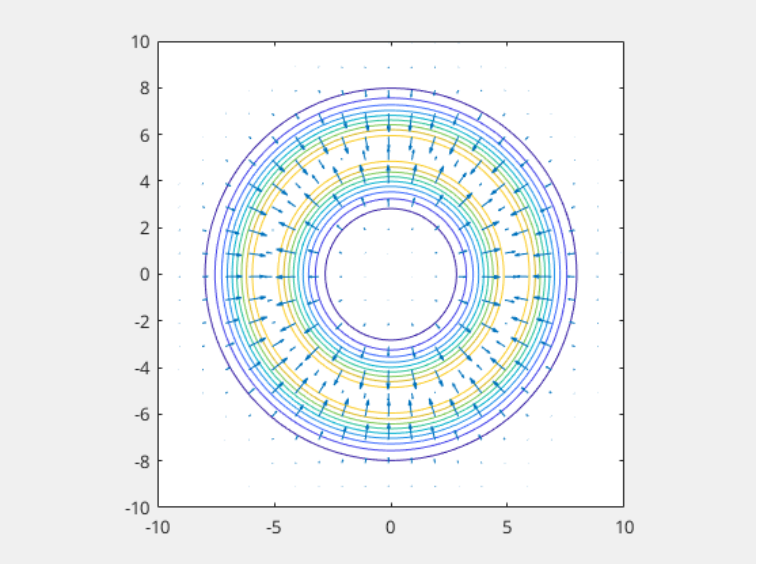

Figure 11: Contour plot of $h(x, y)$ with its superimposed gradient field

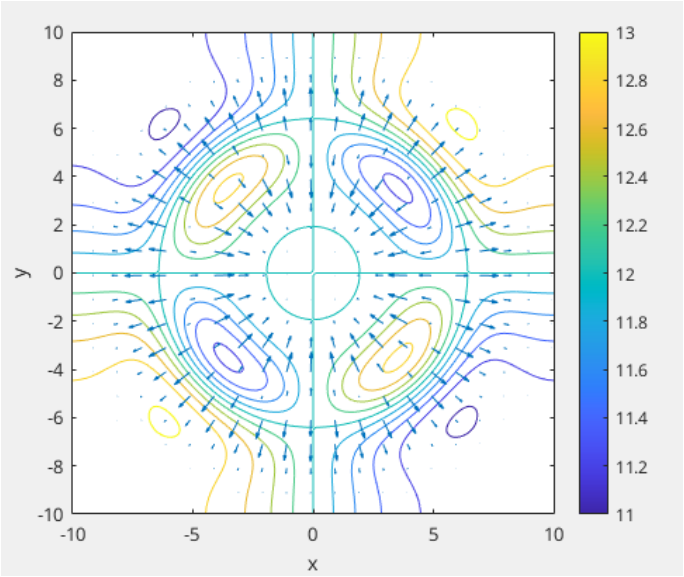

Figure 12: Contour plot of `surfheight` on $h(x, y)$'s gradient field with the gradient field of $h(x, y)$ superimposed

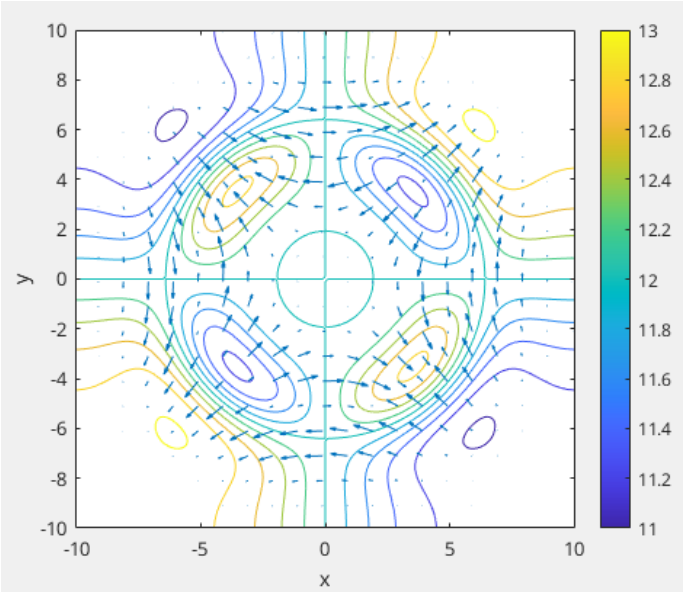

Figure 13: Contour plot of `surfheight` on $h(x, y)$'s gradient field with the gradient field of $h(x, y)$ superimposed and each gradient field's vector components swapped

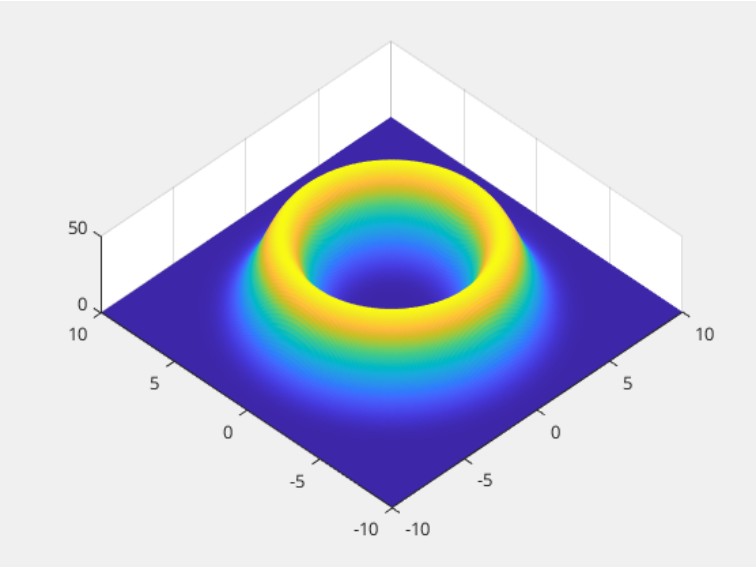

Figure 14: The output of the inverse gradient function `intgrad2` when passed the gradient field of $h(x, y)$

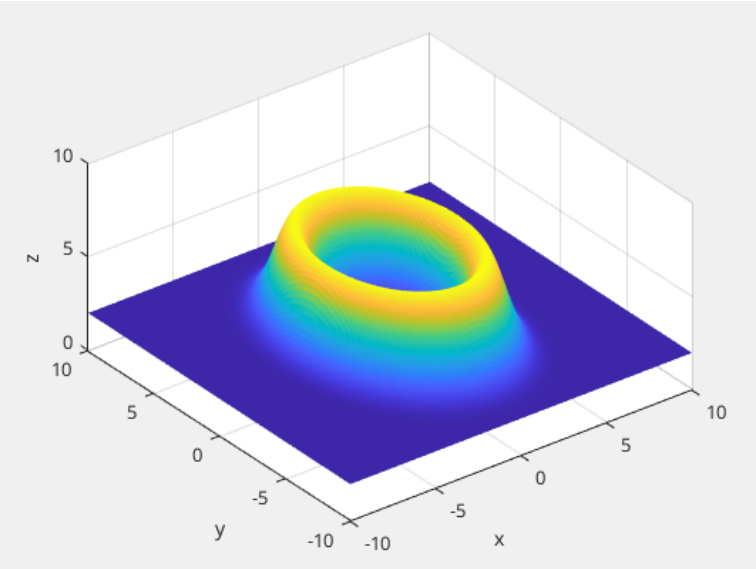

Figure 15: Oval wave defined by $g(x, y)$ above

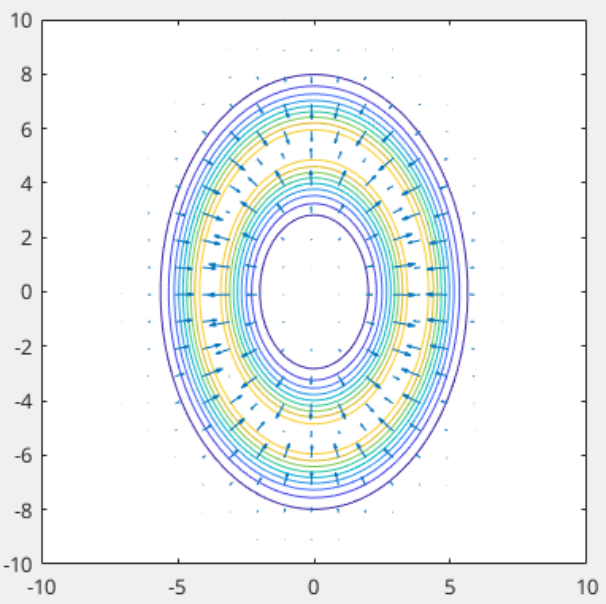

Figure 16: Contour plot of $g(x, y)$ with its superimposed gradient field

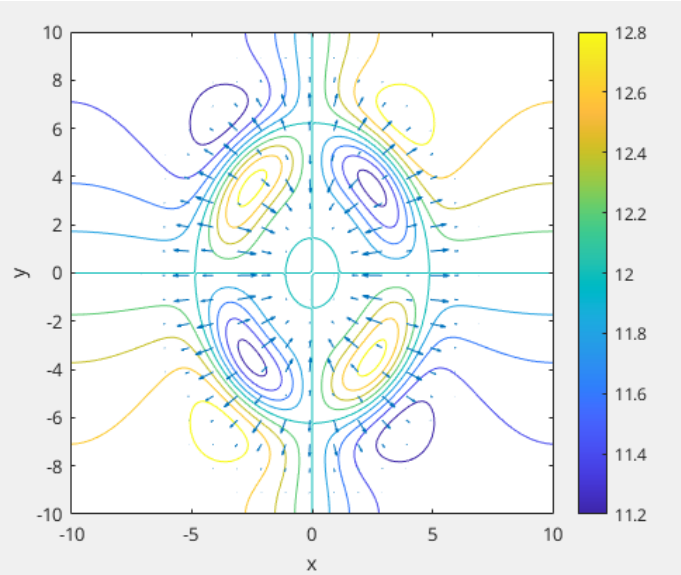

Figure 17: Contour plot of `surfheight` on $g(x, y)$'s gradient field with the gradient field of $g(x, y)$ superimposed

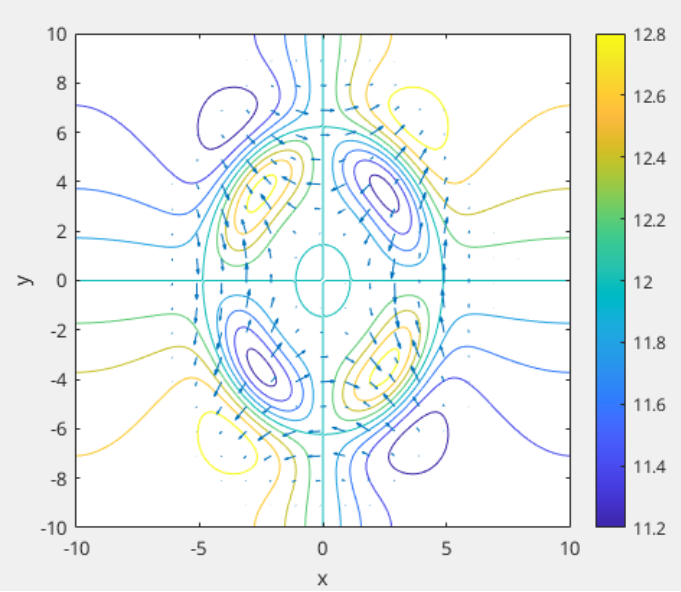

Figure 18: Contour plot of `surfheight` on $g(x, y)$'s gradient field with the gradient field of $g(x, y)$ superimposed and each gradient field's vector components swapped

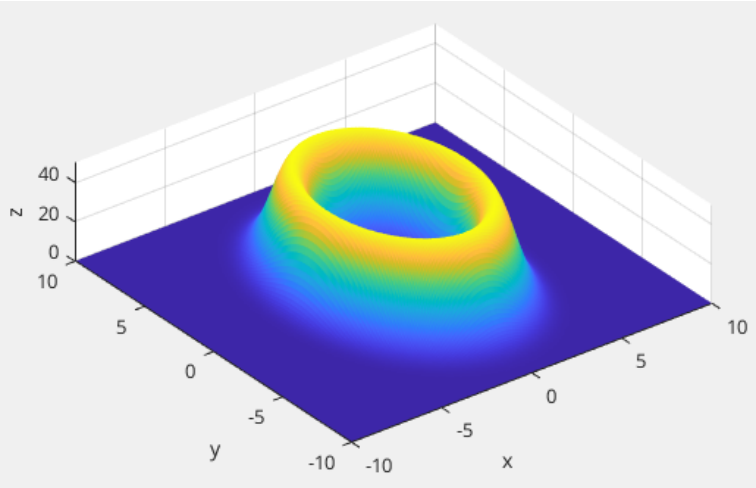

Figure 19: The output of the inverse gradient function `intgrad2` when passed the gradient field of $g(x, y)$

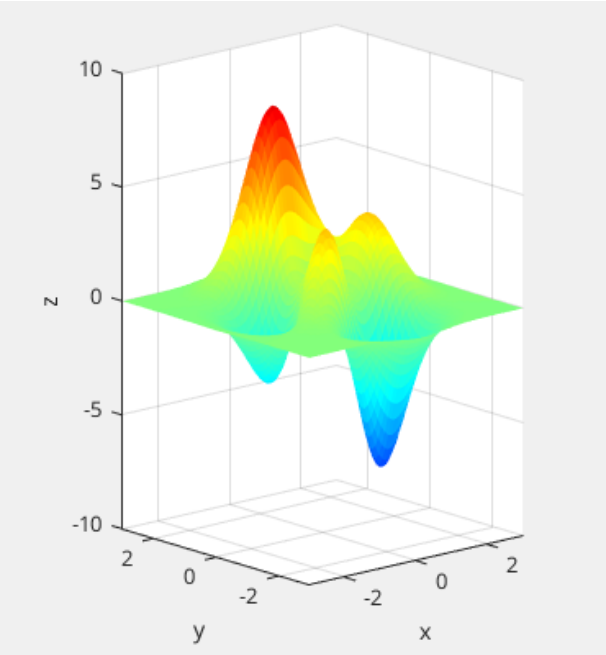

Figure 20: 3D function defined by `peaks` above

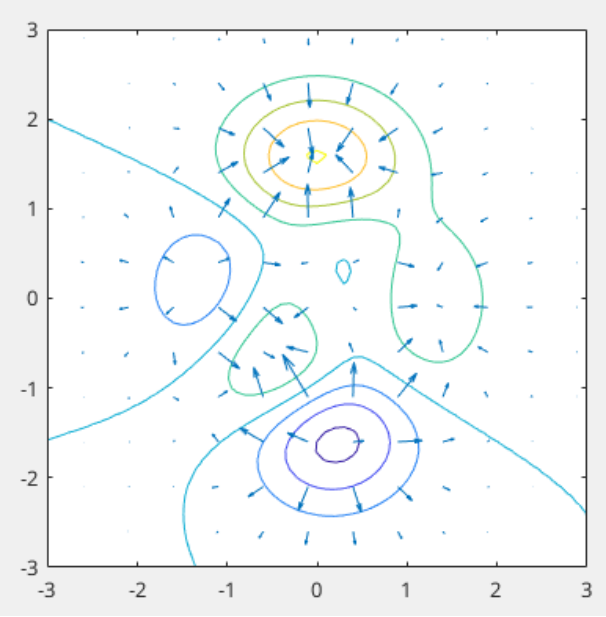

Figure 21: Contour plot of `peaks` with its superimposed gradient field

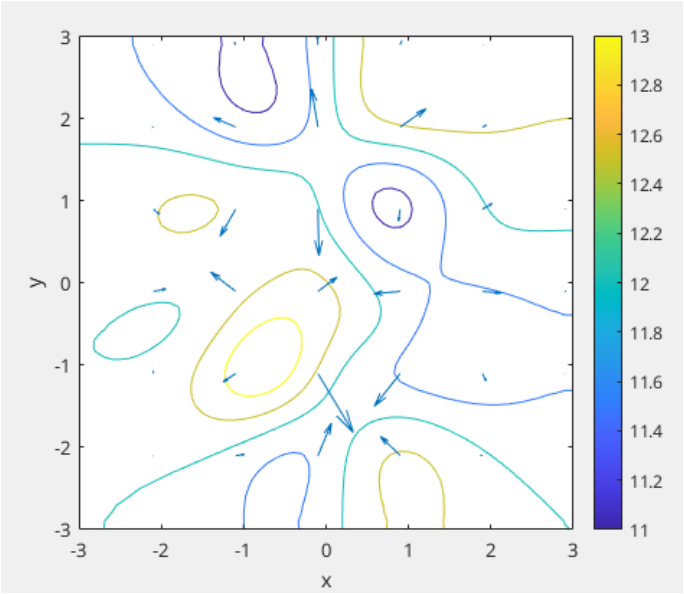

Figure 22: Contour plot of `surfheight` on `peaks`'s gradient field with the gradient field of `peaks` superimposed

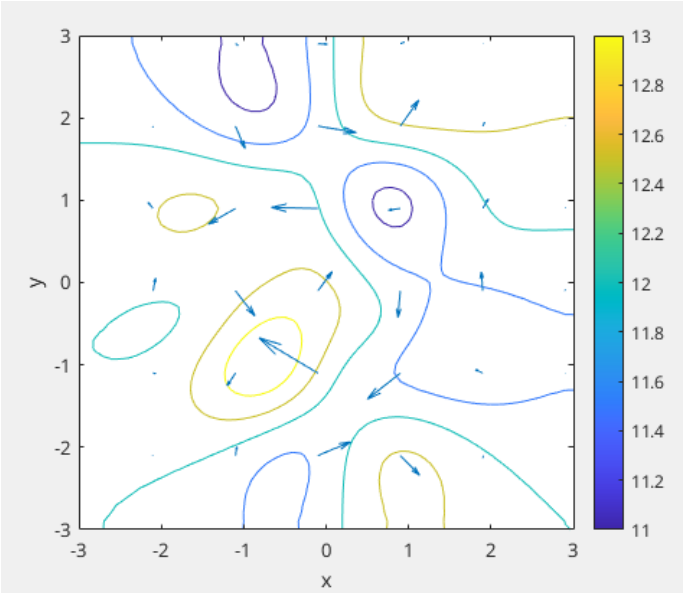

Figure 23: Contour plot of `surfheight` on `peaks`'s gradient field with the gradient field of `peaks` superimposed and each gradient field's vector components swapped

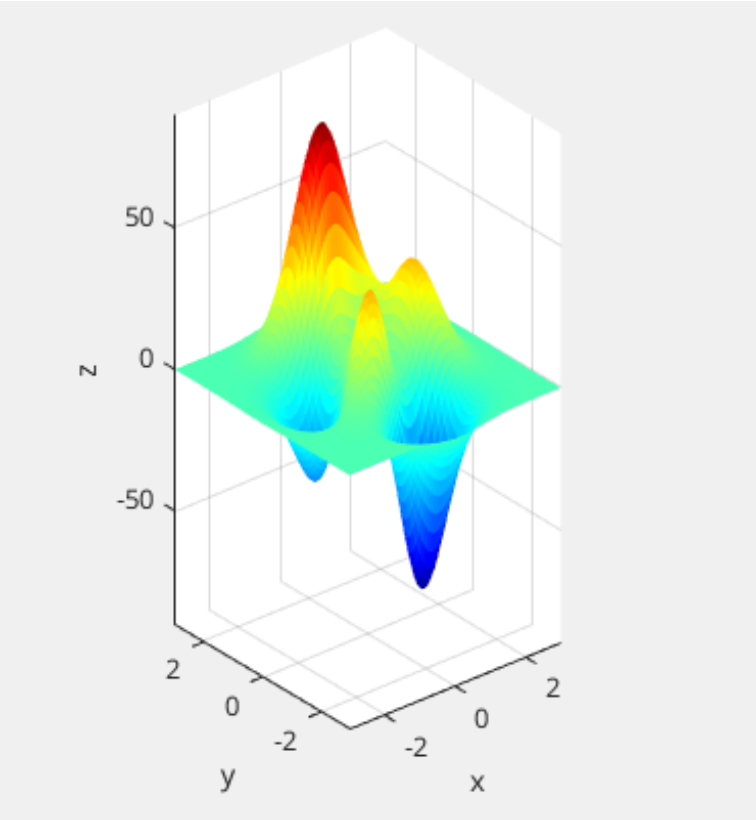

Figure 24: The output of the inverse gradient function `intgrad2` when passed the gradient field of `peaks`

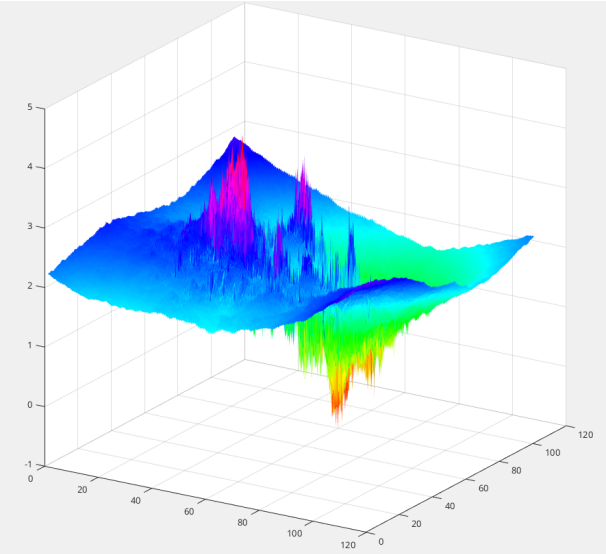

Figure 25: Inverse gradient function `intgrad2` output of the displacement field obtained through a water drop experiment (side view) (plane subtracted)

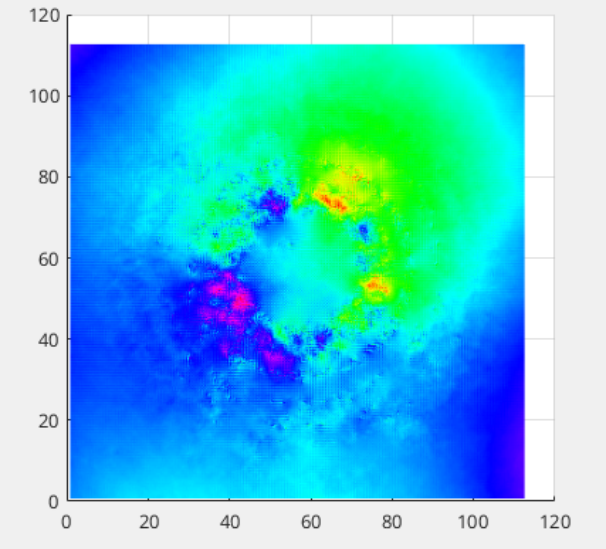

Figure 26: Inverse gradient function `intgrad2` output of the displacement field obtained through a water drop experiment (top view) (plane subtracted)

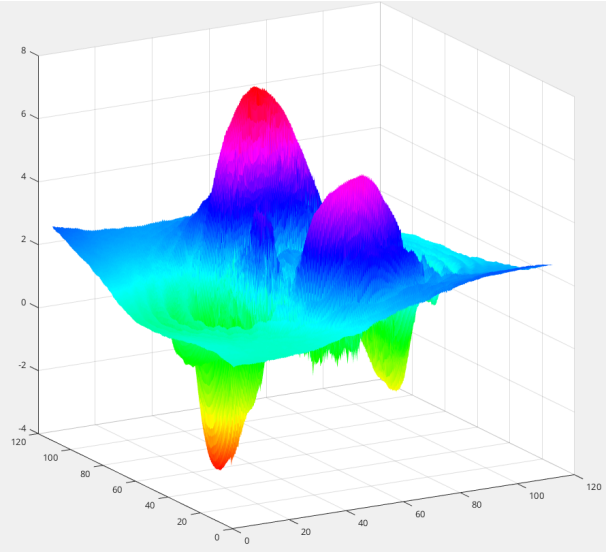

Figure 27: Inverse gradient function `intgrad2` output of the displacement field obtained through a water drop experiment with the x and y components of the displacement field swapped (side view) (plane subtracted)

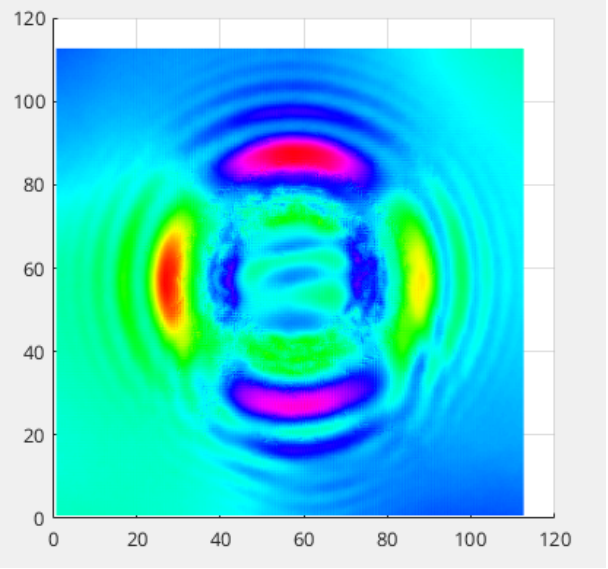

Figure 28: Inverse gradient function `intgrad2` output of the displacement field obtained through a water drop experiment with the x and y components of the displacement field swapped (top view) (plane subtracted)

What's clear from these figures is that the inverse gradient function used in Pivmat, `intgrad2`, is working correctly with the analytic functions. Each of the three different functions looks similar to the output `intgrad2` when their gradient fields are passed in. This tells us that the problem is in Moisy's `surfheight` pre-processing steps before calling `intgrad2`. This is supported by the gradient field from the input functions not corresponding to the contour plot of `surfheight`'s output. In fact, to make it correspond the gradient field's x and y components of each vector needed to be swapped. This is because Moisy swaps the components of the gradient field in the input to `intgrad2` within `surfheight`. It's reassuring to see the water droplet structure emerge in `surfheight` with these analytic functions.

Fortunately, the analytic functions don't have a problem with `intgrad2`. Unfortuntely, the data I collected did. Using `intgrad2` the same way it was used to get the analytic functions did not result in clear, concentric circles we'd expect from the water drop. Instead, it just appears like a bunch of noise. Even more concerning is that swapping the x and y components of the gradient field before passing it into `intgrad2` gave results similar to those that started this investigation: concentric circles with 2 maxima and minima along the x and y axes.

I was unable to figure out why `intgrad2` didn't like the data obtained from OpenPIV. I also searched the MATLAB file exchange for inverse gradient functions and found 2: `intgrad2` and another function which took more than 30 minutes to run without succeeding, so I stopped considering it as a viable option.

2 Description of Next Steps

The next step would be to discuss as a group what we should do next. I can continue looking into the source code of `surfheight` in depth along with following along Moisy's research paper in more detail. I could also go into the lab and help move and then work on getting the water table set-up. We should also discuss if we should reach out to Frédéric Moisy with our problem with his code.

3 Questions

- Should we contact Frédéric Moisy?
- When are we moving the lab?
- What should be my next steps?