Several matlab programs have been created for my project. The following is a description of these programs and how they should be used and altered. Below I have attached the matlab programs themselves.

A few notes about these programs:

- For the programs relating to image analysis to yield the correct results they must be done in a particular order and for this reason, the programs are numbered accordingly.

- Most programs only require the user to change the input and output directory as well as the file names. To aid in this process there are comment lines (‘%%%%%%’) signifying where a change should be made.

  ? Please consult my reference to subfolder orders and there should not be a problem with the directories.

- Often you will see loops that appear as the following:

  ```matlab
  i= 10:10:3610
  t=i-10;
  j=1:3
  ```

  - the i-loop signifies looping for through all times (up to one hour or 3600 sec) with steps of every 10 seconds. The t=i-10 lines is simply ensures that we will look at even time zero since a loop cannot with a zero index. This index may be changed depending on what times you are interested in.

  - secondly, the j-loop loops through the three smaller sections of the image. If, however, you have cropped the image into 6 sections you can simply change the second number in the loop to be “6” in programs 5 through 9.

**no1_testimages.m**

This program is to ensure that the cropping dimensions are correct so that the raw images from the camera can be properly cropped into a quarter circle of the tool. By testing one of the images (preferably an image where the center of the tool is very clear), the same cropping conditions can be used for all images assuming of course that the camera has not shifted during experimentation. After inputting the image, the program will do all the rotating and cropping and display the image for you. You may keep adjusting the parameters until the image looks how you want it to look.
The first item to check is the angle of rotation. By adjusting the number in this argument, you adjust the angle with which the image is rotated. Remember, we want the image to a quarter circle of the tool with the center in the top right corner.

```
rotimage=imrotate(image,18);
```

Secondly, we want to make sure the square we are cropping puts the center in the top right corner and the edges of the tool in the top left and bottom right corners. In the following line of code, the first number argument is the xmin and the second is the ymin. By playing around with these parameters, you should be able to come up with an image you are happy with. DO NOT change the two other dimensions as they are crucial the section cropping later on.

```
cropimage=imcrop(rotimage,[140 410 881 881]);
```

**no2_imagecropping.m**

This program is used to input all the raw images taken from the camera during testing and outputs them as cropped tif files. If program 1 was done correctly, it should be easy to change the cropping parameters to those which you chose previously.

Another thing to note about this program is that when the files are saved from the camera they are saved with the frame number at the end. This program uses the fps (frame per second) rate to convert frame numbers to time in seconds. If for some reason the frame rate is different than what was used originally (10.7 fps), then a necessary change must be made to the time conversion and looping parameters.

**no3a_testthreshold_3.m**

This program is designed to test the thresholding that will be used when the images are cropped into three smaller sections. The minthresh1, 2, or 3 variable signify the minimum tolerated threshold values for sections 1, 2, and 3 respectively (edge, middle, center). The value must be between 0 and 1. For my experiments I set these three values to be the same thing but this is not necessary.

```
minthresh1=.146;
minthresh2=.146;
```
The rule of thumb I used was to find the smallest thresholding with which the first image (image at time zero) appears completely black. This holds true since at time zero there should be no structures that have formed. It is best to work with a low number and rerun the program several times to work up until this condition is satisfied. I found the best thresholdind value to be .146. Again, you may change the bounds of the i-loops to looks at more images than just the one at time zero.

no3b_testthreshold_6.m

This program does the same thing as program 3a except that it does it if you are dividing your image into 6 sections.

no4a_sectionimage_3.m

This program inputs all your images of the quarter circle of the tools and crops to make three smaller sections corresponding to the edge of the tool, the middle, and the center. The program outputs these three sections into the “No Threshold” folder and then thresholds these images and outputs the thresholded images to the “Threshold” folder. To do this, you must adjust the minthresh variables according to those that were chosen in program 3.

no4b_sectionimage_6.m

This program does the same thing as program 4a but for 6 sections.

no5_fft_unwrap.m

This program reads in all the thresholded section images and performs a fourier transform of them (specifically a log fourier transform). The program saves these as “logfft” images. The program then unwraps these images so they are in a 2d array of q and chi and then outputs these as “unwrapped” images.

no6_q_chi_plot.m
This program takes the unwrapped images and creates two dimensional q and chi plots. The q plots are bounded around the chi of 90 and 270 degrees since this is where the signals occur (midchi1-chibound to midchi1+chibound). Two chi plots are produced: one is for the first order region (midq-qbound) and one is for the second order region (midq+qbound). The following variables signify the mid values for these bounds and then the range of these bounds. These variables may be changed if desired. If they are changed, ensure that this change is implemented in future programs.

```matlab
midchi1=90;
midchi2=270;
chibound=20;

midq=55;
qbound=40;
```

This program also displays the image from that section in the graph as a point of reference. Comment lines also signify when a change in a graph title should be made.

**no7a_q_3d.m**

This program takes all the 2d plots and puts them together in a 3d plot with the third dimension being time. Two graphs are produced. The first shows 2d plots of every 30 seconds up to 5 min and the second one shows plots of every 5 min up to 1 hour. This is done because in most cases structures form in the first 5 minutes so this is to zoom in on this region.

**no7b_q_2d.m**

This program does the same thing as program 7a but instead of taking the 2d plots and putting them together in 3d it overlays several 2d plots together on one overall 2d plot. I found that this method was not as good as the 3d plots since the data seems to jump around a lot and you don’t always see the trend you want. For this reason, this program is not 100% perfect and may need some tweaking before it should be used.
**no8a_chi_3d.m**

This program does the same thing as program 7a except that it produces 4 chi integration plots: 2 for first order and two for second order.

**no8b_chi_2d.m**

This program does the same thing as program 7b except that it produces 4 chi integration plots: 2 for first order and two for second order.

**no9_time_plot.m**

This program bounds both the q and the chi integrations and produces a graph showing the change in intensity of the box (one for first order and one for second order) over time. These plots do not show the distinct trend we were hoping to see of the change in intensity but the program is here for future tweaking if that is desired.

**plot_rheo.m**

This program plots the data from the rheometer into plots for sample 1 and sample 2.

**plot_video.m**

This program creates a series of images showing the progression of data points in the rheology plots and along with the images of the quarter of the tool. These images can then be strung together using imagej to create a movie depicting the change in structure as the viscosity changes.