Last updated: 10/17/08
Before leaving, Thalia also made sure all the MoaStudent versions were the same on her computer, linx2, and the new Nagle lab Windows computer ("fusion")

MATLAB WAXS ANALYSIS MANUAL

By Deren Guler and Thalia Mills

This manual is a step-by-step guide to using the WAXS analysis Matlab code written by Thalia Mills with help from Gilman Toombes. The WAXS analysis also uses the more general x-ray analysis package "MOA" (a set of Matlab functions) written by Gil Toombes. The basics of MOA are outlined in "Illustrative Introduction to Package" written by Gil Toombes (see the file MOA_User_Introduction.txt).

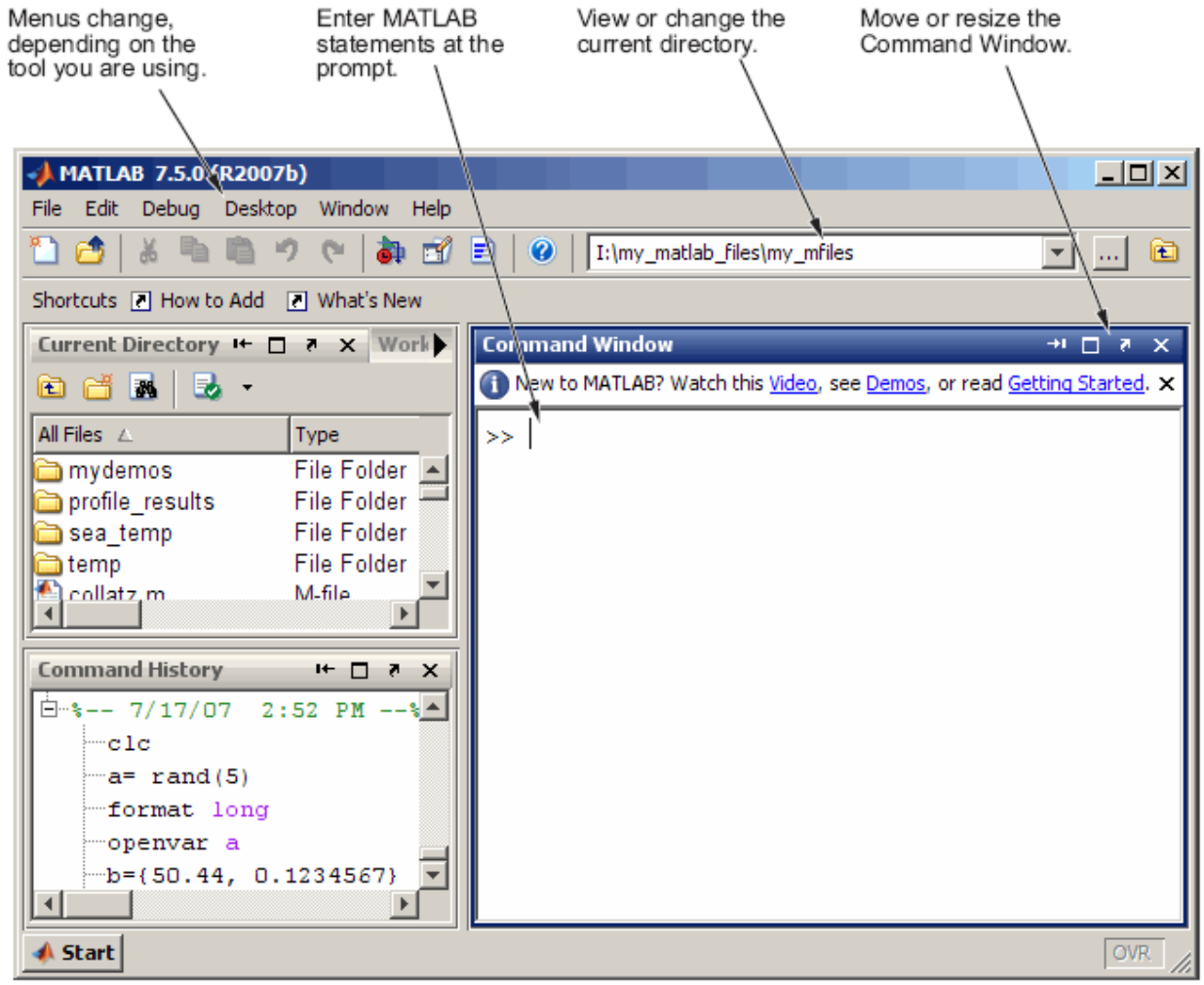
Although this manual is intended to work like a recipe book for a novice, it will be difficult to trouble-shoot without understanding the basics of x-ray image processing and some Matlab basics. Also, some functions (such as "mask") can be difficult to use for the first time without some help from a practiced user. Matlab is based on matrix manipulation. For learning Matlab basics, use the HELP feature or read a brief manual such as "An Introduction to Matlab" by David F. Griffiths. The student version also comes with a hard copy user manual.

NOTE (as of 1/3/08): The functions described in this package are the same as those used by TTM for her thesis with one exception: The fitting functions (NewFitPhi_MSG and NewFitPhi_MSG2, as well as several other functions which they call) have been modified to be compatible with the student version of Matlab, which has only a partial Symbolic Toolbox which does not include the Maltab function "mfun", which was used to evaluate Dawson's integral. The new fitting functions use an alternative function ("dawson") which was downloaded from the web. The old and new functions were checked against each other, and gave the same results to within error. The new student version-compatible versions of these functions have been installed onto computers in the Nagle lab.

Table of Contents

I. GETTING STARTED

1. To open Matlab click on the Matlab icon on your desktop or select it from program files. The gui interface will appear (see below). You can add or delete tiled windows by checking/unchecking them under the "Desktop" menu. The "Command History" is a useful window to have open, but all you really need to use Matlab is the "Command Window."



2. When you open Matlab, there will be a current directory button on the toolbar. Change the current directory to that which contains your data (all the 2D x-ray image files you want to analyze) and "startup.m".

3. The function startup tells Matlab the appropriate directories to look in for the x-ray analysis functions as well as your data. When a function is called in Matlab, it first looks in your current directory and then in the paths specified in startup and in the folder where Matlab was installed (you do not have to list the Matlab installation folder in your startup file). Select “Open” from the file menu and navigate to your new directory, open the startup.m file. It will open in the separate Matlab Editor window. Depending on where you have put the functions and your data, you will need to change the directories listed in startup.m.

For each image you analyze you will need to change some of the values in the startup.m file to match those of your experiment. These are the variables in startup.m that you will need to change (These were the values for the October 2006 CHESS run):

X_Lambda=1.274; % (energy=9.73 keV)
Spec_to_Phos=2174; % 151.7/0.06978=2714=sample to detector distance in pixels
(NOTE: In Matlab, the percentage signs denote that the text following is commented out. A semicolon prevents the contents of the variable (matrix) from being output onto the command prompt.)

Spec_to_Phos is the sample to detector distance in pixels. The S-distance of 151.7 mm was determined using AgBeh as a calibrant and using the Nagle lab program Tiffview. G. Toombes' Matlab MOA functions can also be used to find S-distances (see MOA_User_Introduction.txt). The pixel size of the FliCam detector was 0.06978 mm. After you make the appropriate changes, save startup.m (in your current directory).

If you are not looking at raw data, you do not have to worry about detector corrections. All the CHESS data and Carnegie Mellon data are already distortion and intensity corrected and de-zingered.

In the startup file the global variables (these are all matrices) are: Intensity _Correct, X_Distortion_Map, Y_Distortion_Map, X_Cen, Y_Cen, Spec_to_Phos, X_Lamba, MaskD, MaskI.

Running startup.m will load these variables into your Matlab "workspace." Once you start inputting commands into Matlab, you will add to your "workspace." The Matlab workspace is a set of Matlab variables (i.e. matrices). Note that when you exit Matlab, you will lose your workspace unless you save it. Remember to save your workspace regularly by selecting “Save Workspace” from the file menu in Matlab.

These global variables were defined by Gil in his MOA functions. They are automatically passed to functions. If you are only analyzing one image in a single workspace, you will only have to set your global variables one time. However, often it is convenient to analyze a set of data in a single workspace: for example, a series of DOPC/cholesterol images or temperature series. In that case, as you analyze each image, you may have to change the values of some of the global variables which are different for each image. In particular the following often change from image to image: the beam center (X_cen and Y_cen) and the integration mask (MaskI). On the other hand, X_Lambda and Spec_to_Phos are likely to be the same for a given set of images. This issue will become more clear later in the manual. Don't try to analyze all your data from a synchrotron run in one workspace: MATLAB will crash.

***Remember that Matlab is case-sensitive when re-defining the global variables.

Unlike global variables, local variables must be specifically passed into a function during the call to that function. Local variables (store manipulations of your x-ray data) do not require a specific name or name format. However, the name format for local variables used in the examples in this manual may be a convenient notation to follow until you develop greater proficiency in Matlab and WAXS analysis.

4. To load the MOA functions and variables click on the Command Window and type:

>>startup;

The double carrot signifies a new command that you are inputting and is already part of the command window. The semicolon at the end of startup is not necessary, but it is a good idea to be in the habit of ending commands with a semicolon. The semicolon suppresses Matlab from displaying the contents of any Matlab variables assigned during the command. Some variables (such as the detector images) are very large matrices, which take much space and time to display on screen. In some cases, you will want to display the contents of a Matlab variable. In those cases, leave out the semicolon (also discussed later).

5. As you go through your analysis, it is convenient to keep a text editor (like MS word) open for note recording. You should also keep a data table of all important values and graphs along the way, so you can refer back to them. In addition to data tables, as a reminder, you may want to copy and paste certain Matlab commands (especially commands you are using for the first time) into a text document. Matlab has a Command History window, but it only saves things for a certain amount of time, so you will want to make a file for your records.

II. USEFUL FUNCTIONS (used multiple times during analysis)

User-defined functions (The following functions were written by Gil Toombes as part of the MOA x-ray analysis package.)

1. **show**

Once the image has been converted from a .tif or .img into a Matlab matrix (described later), you can display the image (in this case we are looking at do_5c) use the **show** command:

>>show(do_5c, [0 1000]);

[0 1000] is the grayscale range; you can play around with this if you want to make it brighter or darker. "show" diplays your image with the display mask (the global variable MaskD). A nice feature of show is that it keeps the aspect ratio square, even if you resize the image by pulling down on the lower right-hand corner. Show does not change the variable do_5c (your image) in any way. Even if MaskD has certain pixels masked out, the image variable remains unmasked and can be redisplayed again using a different MaskD. This is also the case for functions which use the integration mask, MaskI: the mask is used for integration purposes, but the image variable remains the same.

Built-in Matlab functions (These come with the Matlab package)

1. **whos**

If you ever want to **see the variables** in your workspace use the **whos** command and press enter. The output will look something like this (this was done prior to analysis, so there are only global variables from startup.m).

Input:
>> whos
Output:

Name	Size	Bytes	Class
Intensity_Map	1024x1024	4194304	single array (global)
MaskD	1024x1024	1048576	uint8 array (global)
MaskI	1024x1024	1048576	uint8 array (global)
Spec_to_Phos	1x1	8	double array (global)
X_Distortion_Map	1024x1024	4194304	single array (global)
X_Lambda	1x1	8	double array (global)
X_cen	0x0	0	double array (global)
Y_Distortion_Map	1024x1024	4194304	single array (global)
Y_cen	0x0	0	double array (global)

Grand total is 5242882 elements using 14680080 bytes.

*You can also check any variable by just typing the variable name (without a semicolon) and pressing enter (ds_do_5 is the variable name for the d-spacing found in Step 6 in the "Steps for WAXS analysis" section):

>> ds_do_5
Output:
ds_do_5 =

5.0000	4.5292
10.0000	4.5159
15.0000	4.5069
20.0000	4.4967
25.0000	4.4864
30.0000	4.4731
35.0000	4.4577
40.0000	4.4393
45.0000	4.4215
50.0000	4.4053
55.0000	4.3918
60.0000	4.3762
65.0000	4.3575
70.0000	4.3419
75.0000	4.2892

2. **figure, clf**
When looking at an image using show or another function that creates an image (such as a graph) it is good habit to make a new window for each image. Otherwise, Matlab will overrirde an existing figure. To make a new figure window, use **figure** and a new window will appear:

>>figure;

figure will open up a figure numbered with the lowest number not already in use. If you want to open up a particular figure window (such as Figure 99), type:

>>figure(99);

If you want to clear a figure, make sure that figure window is active by clicking on it, and then type:

>>clf;

3. **save**
To save your entire workspace, you can either go to File-Save Workspaces As, or you can use save in the command prompt:

>>save Oct06_wdo

A file Oct06_wdo.mat will appear in your current directory which contains all of the Matlab variables in your current workspace.

Sometimes you may want to save only a single variable from the current workspace:

>> save junk do_5a1

A file junk.mat will appear in your current directory which only contains the variable do_5a1. Files with the .mat extension are only readable by Matlab. Usually, when you are saving particular variables, the purpose is to save in a more universal format that can be read by many programs. "ascii" is a particularly useful format. The following command outputs do_5a1.txt to your current directory (which contains two columns of phi and intensity values corresponding to the variable do_5a1):

>>save do_5a1.txt -ascii do_5a1

The general format for saving is (see also help for the save function):
save filename -format variable

You can save any variable as an ascii file, but you will probably want to save your integrated data (output of the functions described later such as SectplotNew, integrate_annulus, etc.). CCD image variables are large matrices (~1024 x 1024). There is no easy way of saving an image variable as a jpeg or tiff using the save function. However, you may want to export an image on which you have performed manipulations (background subtraction, rotation, etc.) as a jpeg or tiff. In order to do this, first display the image in a figure window using show, and then use the print command (described below).

The files will always be saved in your current working directory (you can change the directory before using save if you wish.)

4. **print**
You may want to save a figure you created in a different format. The figure windows have gui exporting options, but the print command is faster and easier and allows you to specify exactly the format you want.

Be sure to click on the figure (e.g Figure 2) you want to save or type the following tomake Figure 2 active:

>>figure(2);

To export the figure as a jpeg with 300 dpi resolution (junk.jpg will appear in the current directory):

>>print -djpeg -r300 'junk'

To export the figure as "junk.tif", a tiff with 300 dpi resolution (junk.tif will appear in the current driectory) :

>>print -dtiff -r300 'junk'

To export the figure as a color eps with an embedded tiff with 150 dpi resolution (junk.eps will appear in the current directory):

>>print -depsc -tiff -r150 'junk'

5. **load**
Sometimes you'll want to load a variable from another workspace into your current workspace. The following command loads the variable mdo_5c from wdo.mat, which contains many saved variables. The save worksapce wdo.mat must be in your current directory.

>> load wdo mdo_5c;

Often, you will want to continue adding to a saved workspace. To load all of the variables contained in wdo.mat, type:

>>load wdo

Alternatively, you can go to File-Open and double click on the desired workspace (.mat file). You will want to open the saved workspace after the startup command.

6. **help**
The functions mentioned above have additional options/features. For more documentation on functions (print in the following example), type in the command window:

>>help print

Or you can search for "print" in the gui Matlab Help window. The command line "help functionname" shows documentation even for user-defined functions (written in .m files).

III. STEPS FOR WAXS ANALYSIS: from detector CCD images to order parameters

Important to keep in mind:
1. When you are using any function and setting a variable to equal the output of the function the format is:
variable=function(input)

2. If you forget to type the semicolon at the end of a command, and Matlab starts showing the contents of a large matrix (e.g. a 1024 x 1024 image variable), type <Cntrl>C.

The following steps follow an example DOPC data set from the October 2006 CHESS run. It was the 5th DOPC data set analyzed in a single workspace (and the variables begin with do_5).

**Remember the startup command always comes first when you open Matlab. If you have a workspace saved and you want to add to it, open it up after the startup command and before you start more analysis.

Step 1: Load image and background files and check the subtraction.

The slurp function loads a CCD detector image (in tiff format) into Matlab. Matlab is case sensitive so make sure you type the file name correctly. The 'c' tells Matlab that the files have already been distortion and intensity corrected (this is done in the startup file), so Matlab leaves it alone. The following all use assign local variables.
Load the image:

```
>> do_5=slurp('wdo_117_cz.tif','c');
```

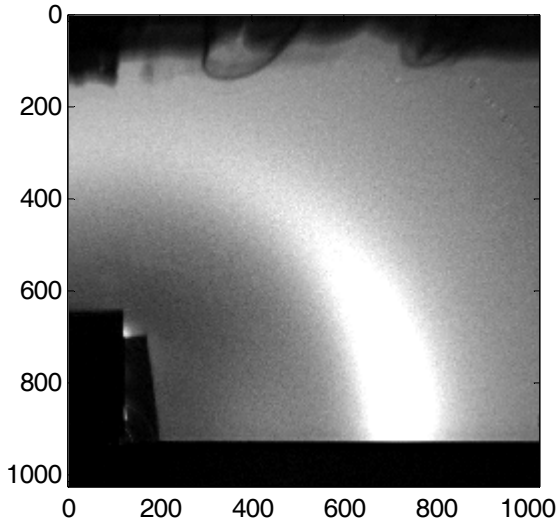
Load the background:

```
>> do_5b=slurp('wdo_118_cz.tif','c');
```

Subtract the image from the background to make the "corrected" image:

```
>> do_5c=do_5-do_5b;
```

All of these variables (do_5, do_5b, and do_5c) are 1024x1024 matrices. You should show the image (>>show(do_5c, [0 1000]) at this point to make sure you are subtracting correctly. If you see only black, you did something wrong; you may want to play around with the intensity scale to make sure you are seeing something.



The slurp function can import tiff files. The detector sofware Crystal Clear for the Nagle CMU detector outputs .img files. The function slurpNagle imports these files into Matlab:

>>junkimage=slurpNagle('10120002.img');

*For Matlab experts: The matrix containing the intensity values has rows and columns with the positions specified by 1 to image size (usually 1024): the numbers increase going down and across. The row number specifies the vertical position and and column number specifies the horizontal position. You can view the intensity of a particular element in a matrix as follows:
>> soch30_1c(127, 978)
The output is he intensity value at row=127 and column=978, which corresponds to the upper righthand corner of the image.

Step 2: Rotate (if necessary)

To rotate the figure, enter the degree of rotation from the horizontal axis (counterclockwise is the positive direction). To see the original display mask* use the show command (>>show(MaskD, [0 1000]). To check the alignment of the sample you can use the draw line tool in the figure window (on the tool bar or in the tools option) to draw a line between the bottom of the data and the substrate (you can adjust the color and thickness of the line by right clicking on the line). If the line is not straight on the substrate equator, you will have to rotate the image.
* A mask is a 2D array with some parts "masked out", see step 3 for an explanation of masking.

Because show uses the global MaskD, you must also rotate MaskD by the same amount as your image. (Your image and MaskD must be the same size matrices in order for the show function to work, and the rotation changes the size of the matrix slightly.) But first you should save the unrotated MaskD.

1. In this step you are setting MaskDold to MaskD as defined by the startup.m file (a white 1024 x 1024 array of 1's):
>>MaskDold=MaskD;
This is the unrotated, untouched MaskD: NEVER REDEFINE MaskDold! (Only do this step once.

2. Use the imrotate function (this is a Matlab function):
>>MaskD50=imrotate(MaskDold,-0.5);
This is a 0.5 degree clockwise rotation.

3. Now set MaskD to this (remember, MaskD is a global varibale that will need to be redefined again and again):
>>MaskD= MaskD50;

4. Rotate junk before changing the data images so that you can see what you have done. "junk" is your initial attempt.
>> junk=imrotate(do_5c,-0.5);

5. View the rotated image:
>>show(junk, [0 1000]);

Note: If you have forgotten to rotate MaskD and junk by the same amount, you will get the following error message:
??? Error using ==> times
Matrix dimensions must agree.

Error in ==> show at 37
  imagesc(double(x).* double(MaskD) + urange(1)*(1.0-double(MaskD)),urange);

You can always recover the unrotated MaskD by setting it to MaskDold.

6. Determine if this is correct by drawing a horizontal line. You may need to try more than one rotation angle, repeating steps 2-6 above.

7. Once you've decided the appropriate rotation you will need to rotate all of the images because these are the variables that are used throughout the analysis.
>> do_5=imrotate(do_5,-0.5);
>> do_5b=imrotate(do_5b,-0.5);
>> do_5c=imrotate(do_5c,-0.5);

Note: This particular image was found to require no rotation, so the above commands are just examples.

***NOTE: You must figure out your rotation angle before proceeding, but you can reverse the order of steps 3 and 4 if you wish.**

Step 3: Masking

When you want to select, integrate, or view a fraction of an image you can create a mask. To create a mask, use the mask function. In this case, "image" is the name of the variable you are masking, newmask is a local variable you are creating. Remember, "image" contains a matrix of intensity values (that represent your WAXS data). The basic mask function is:

>>newmask= mask(image);

This will create a figure that you can change. When you make a mask, Matlab will output a list of things you can do in the command window and open the image in a figure window. First you will see the whole image, you can select certain parts of the image and eliminate (e), negate (n), reject (r), include (i) them to select the part of the image that you want. To get a crop tool type 'r' in the figure window, then left-click on your mouse to draw a shape. You can make a polygonal region using the left mouse button (if there is something else you need to do on your operating system Matlab will tell you when you create the mask). This creates lines and vertices. When you want to close the region, right-click on your mouse. You must type 'q' when you are done with the masking procedure to get to the next command line.

Example with DOPC data:
Input
>>mdo_5c=mask(do_5c,[0 1000]);

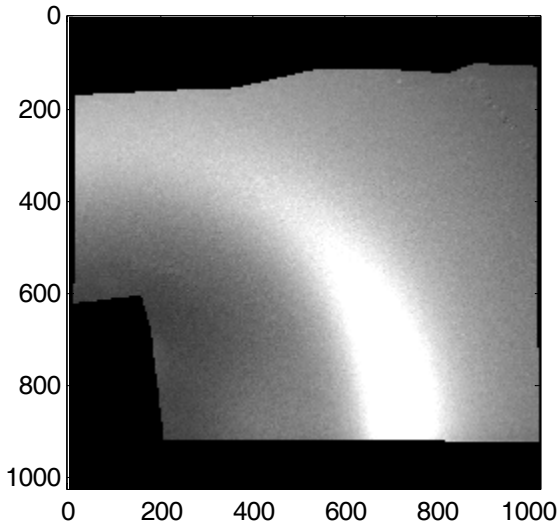
Output (in Command Window):
Mask.m --> Generate a Mask file
Select one of the following commands.
(r,R) - Select a region of interest.
- Mark polygon with left mouse button. Finish selection with right mouse button.
(e,E) - Eliminate data within current ROI.
(i,I) - Include data within current ROI.
(n,N) - Negate the current mask.
(f,F) - Start with a fresh mask where all points are included.
(z,Z) - Zoom

mdo_5c is a mask of do_5c. Your new mask is saved in the local variable mdo_5c. Use only light background corrected data for masking because you do not know what shadows subtract out from unsubtracted data.

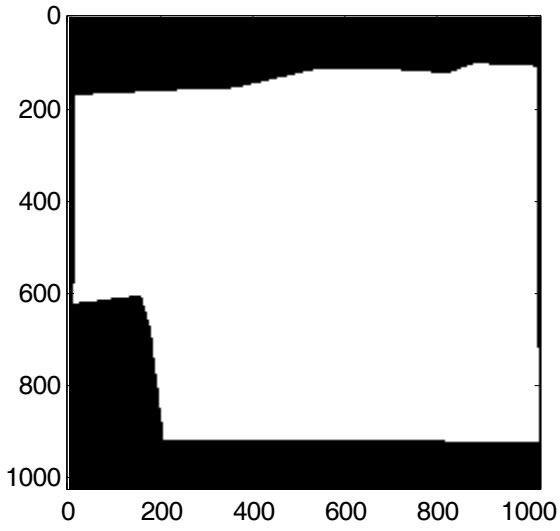
Suggestions for how to make the mask:
Two-step method: First, mask out top and edges: use eliminate (type 'e' in the figure box) and negate (type 'n' in the figure box) after you have selected the region you want to keep. If you cannot cut out a region type 'r' the figure and the cursor should appear. Next, mask out beamstop and below substrate (use eliminate feature). When you are finished type 'q' and then enter to go to next command line in Matlab.

One-step method: Choose the region you want to include. Type 'e' and then 'n'.

After exiting the mask function, you should view your mask.
The following commands let you view your mask on top of an image:
>> MaskD=mdo_5c;
>> show(do_5c,[0 1000]);



Or you can just look at your mask file:
>> show(mdo_5c);



It should be black where you want data excluded and white where you want the data included. If not, you did something wrong. For users familiar with Matlab: the mask is a matrix of 0's (for excluded data) and 1's (for included data).

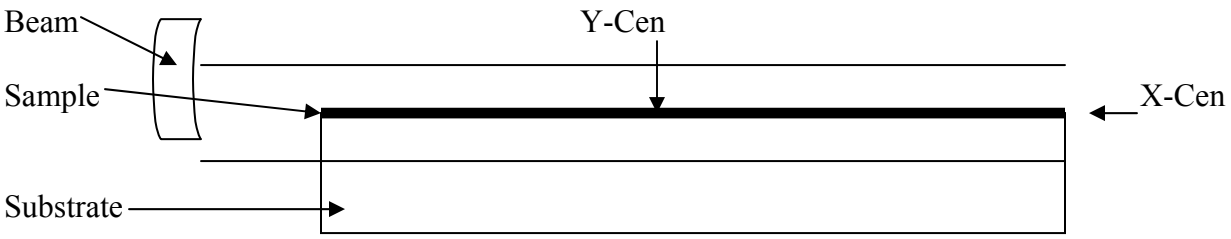
Now you can assign the mask to either the display mask (MaskD) or integration mask (MaskI).
>>MaskI= mdo_5c;
In general, there is no reason to mask out any part of MaskD (reset to MaskDold if no rotation necessary or MaskD50, MaskD75, etc. if rotations were necessary), but you will want to properly assign MaskI before doing any integrations of your images.

If you want to use a mask you have made before in a different workspace, you can use the load function:

>> load wdo mdo_5c;

wdo calls the workspace, mdo_5c loads the variable. If the workspace is in a different directory than your current directory, change the directory to where wdo is and then remember to change the current directory back after you're done with loading.

Step 4: Finding the beam center

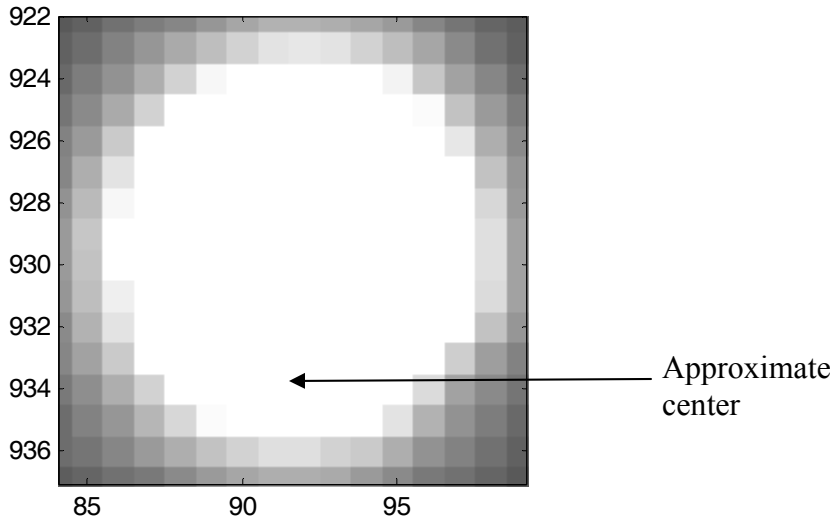


Since the beam goes through the sample as shown in the figure, the X_cen is on the bottom edge of the cut beam (not the center) and the Y_cen is in the middle as you would expect. **Note that the X-axis is vertical and the Y-axis is horizontal; this is Gil Toombes notation for MOA. So X_cen refers to the vertical pixel of the beam center and Y_cen refers to the horizontal pixel of the beam center. However, in Matlab (not MOA), the references to the X and Y-axes on a figure follow normal convention (x horizontal and y vertical).**

Use the unsubtracted data and a clean MaskD so you can see the beam center:

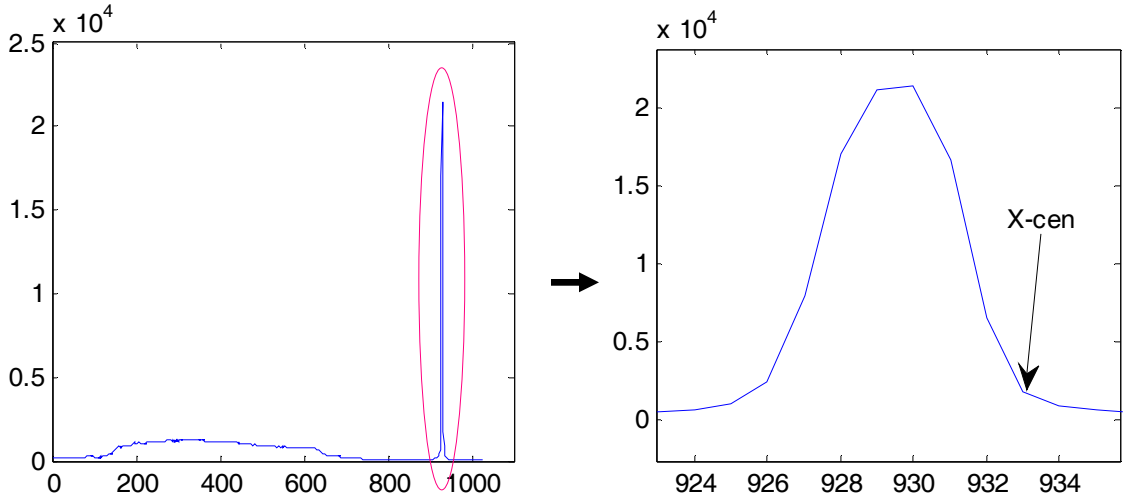
>> show(do_5,[0 500]);

Draw a box around the beam and enlarge it (see figure below). Remember Y is horizontal; X is vertical.



Make a plot of the intensity of the beam vs. pixels in the vertical direction (qz). Zoom in to find leading edge (closest to substrate) of beam peak- will be a larger pixel value. [91,93] specifies a range of pixels in the horizontal direction over which the intensity is averaged. The range should be chosen so that the vertical center Y_cen falls approximately in this range based on the image of the beam.

Input:
>> qzplot(do_5,[91,93]);
Output:



The graph that appears will resemble the one on the left, use the zoom tool (magnifying glass on tool bar) to enlarge the region you want to analyze. This is the spike that is circled on the left graph. When you enlarge it a graph like that on the right will appear, and X_cen is at the end of the curve, like circled. Enter the value of X_cen (vertical), that you find.

>>X_cen=933;

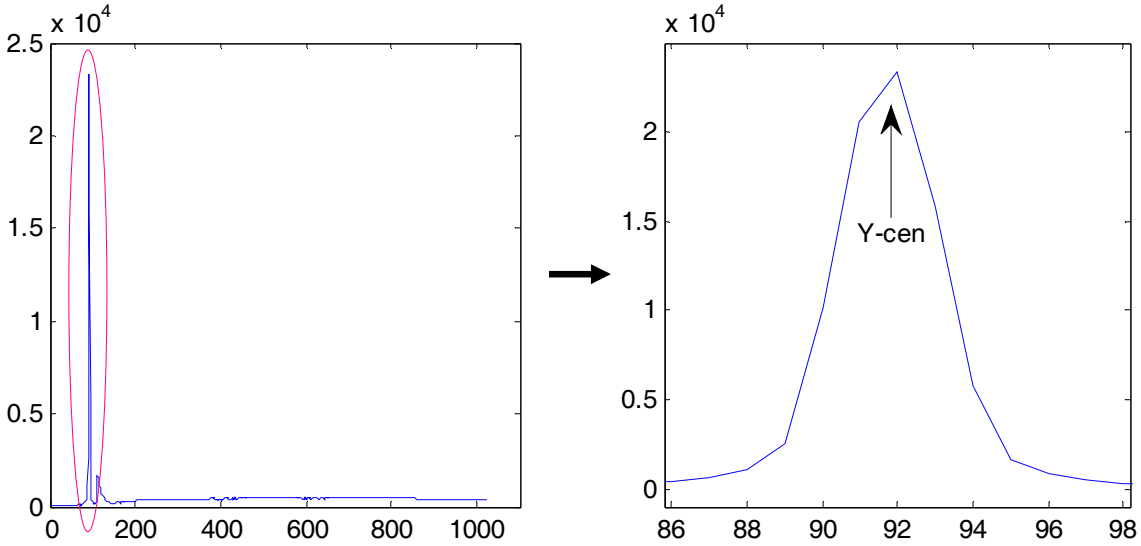
Now make a plot of the intensity of the beam vs. pixels in the horizontal direction (qr). Zoom in to find center of beam peak.

Input:

>> qrplot(do_5,[928,930]);

[928,930] specifies a range of pixels in the vertical direction over which the intensity is averaged. The range should be chosen so that the vertical center X_cen falls approximately in this range based on the image of the beam.

Ouput(left graph…):



Type in the value you find for Y_Cen (horizontal):

>>Y_cen=92;

Write down these values in your external data table. Remember X_Cen and Y_cen are global variables so you will have to define them (or make sure they are correct by checking) each time you use them, as each image will probably have a different X_cen and Y_cen.

Step 5: Sector plots

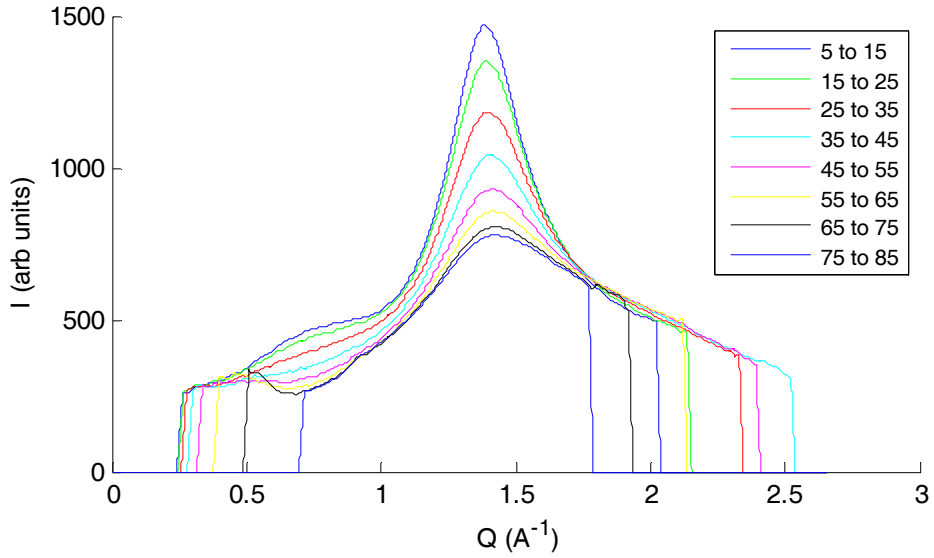
Set (or check to make sure they are correct) X_cen, Y_cen, and MaskI because these are global variables used in intregration functions:

>>X_cen=933; Y_cen=92;

>> MaskI=mdo_5c;

Make a "sector" plot smoothed with a 20-point span using "moving average". This is a series of I vs. q plots with the intensity averaged over different ϕ ranges. In the bracket [5:10:85], 5 is ϕ_{min}, $\Delta\phi$=10º is the integation span for each sector, and 85 is ϕ_{max}.

>> sect_do_5c=sectplotNew(do_5c,[5:10:85],20);



The function sectplotNew automatically makes the plot above. The results of the integration are stored in the variable sect_do_5c. This variable is a matrix with 9 columns. The first column specifies *q*, the second column specifies *I* averaged over phi=5-15˚, the third column specifies *I* for phi=15-25˚, etc. You can use "save" to make an ascii text file for sect_ do_5c.

Step 6: HWHM (half width at half maximum)

***Check that X_cen, Y_cen, and MaskI are correct. If not, redefine them.**

Use the hwhm function to get the following values, which are all stored as local variables in the ouput of the function:
qmax is the *q* value of maximum intensity,
ds is the *d*-spacing (2π/qmax)
hwhm is the half width at half maximum
cl is the correlation length (=1/hwhm)

Input:
>> [qmax_do_5,ds_do_5,hwhm_do_5,cl_do_5]=hwhm(do_5c,[0.8 1.8],[5:5:80],20);

In the call to the function:
*[0.8 1.8] represents the *q* range over which the peak is defined. But note that the function uses only the lefthand (lower *q*) portion of the peak to calculate HWHM because of the water peak near *q*=2.0 Å^{-1} (see Thalia Mills' thesis or paper "Order parameters and areas in fluid-phase oriented lipid membranes using wide angle x-ray scattering."). *q*=1.8 Å^{-1} is the maximum *q* value measured in the q_z direction, which is more limited than the q_r direction in the NIH sample chamber.
* [5:5:80] specifies the ϕ ranges over which *I*(*q*) plots are created. ϕ=5 is the starting angle, 5 is the degree span over which each *I*(*q*) plot is averaged, and 80 is the end angle.
* The last entry (20) is the smoothing span used. If a smoothing is not used, the function can get confused looking for local minima and maxima.

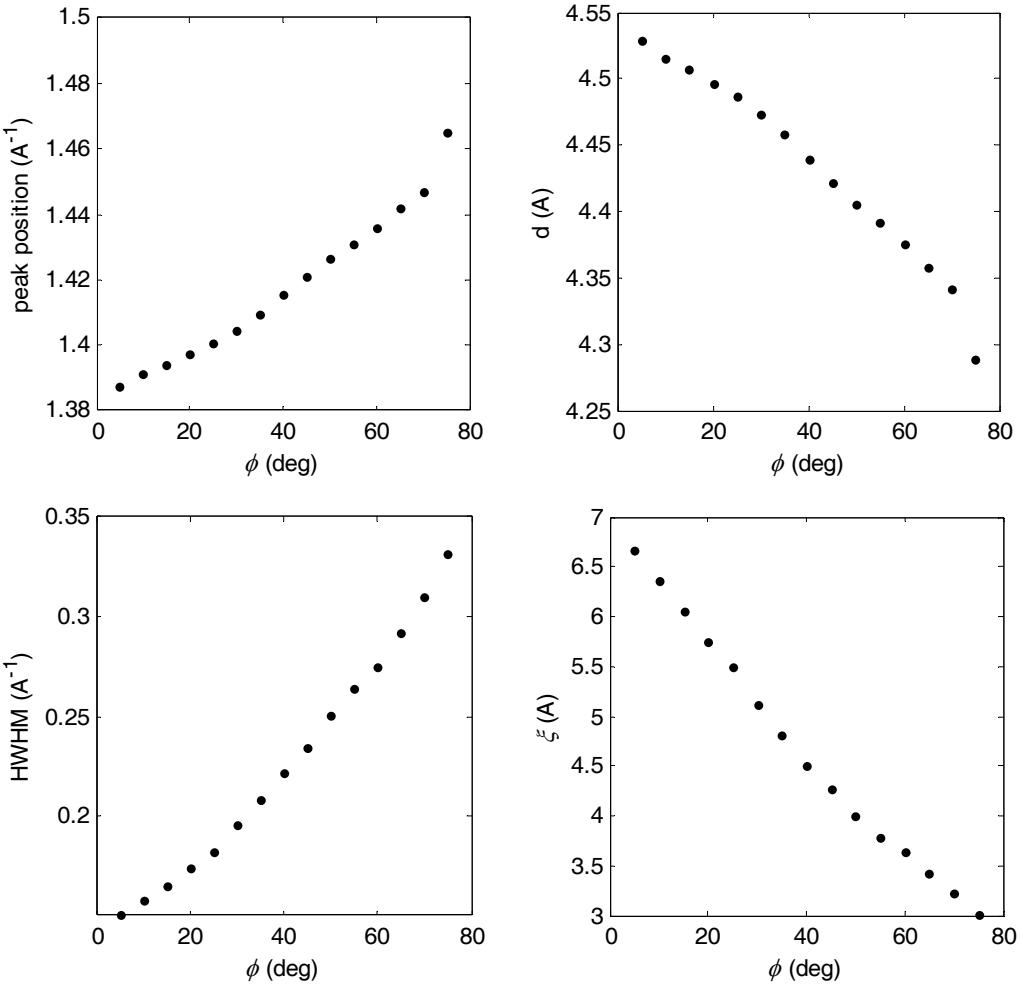
***If you see this after pressing enter:**
??? Error using ==> interp1
There should be at least two data points.

Or this:
Error in ==> hwhm at 72
qhalf=interp1(y([1:nelement]),x([1:nelement]),Ihalf);

Fix like so:
Narrow the q range (e.g. to [0.8 1.5]) or narrow the ϕ range (e.g. change to [5:5:70]). The following example does both:

>> [qmax_do_5,ds_do_5,hwhm_do_5,cl_do_5]=hwhm(do_5c,[0.8 1.5],[5:5:70],20);

These errors usually happen when the sample has a lot of high-q water scatter, and the function has problems picking a peak.
When there are no errors the function automatically generates 4 graphs:



The increase in qmax as a function of ϕ is due to the isotropic water peak at q~2.0 $Å^{-1}$, which begins to dominate the scattering at larger ϕ angles. Spikes in the graph (noise) could be due to zingers (check the image you are integrating).

You may want to save the graphs as ascii files to import them into another program. To do this use the save function. This is saving ds_do_5 (the d-spacing variable) as an ascii in the file ds_do_5.txt:

>>save ds_do_5.txt -ascii ds_do_5

To see any of these variables type the following and press enter:
Input:
>> ds_do_5
Output:
ds_do_5 =

5.0000	4.5292
10.0000	4.5159
15.0000	4.5069
20.0000	4.4967
25.0000	4.4864
30.0000	4.4731
35.0000	4.4577
40.0000	4.4393
45.0000	4.4215
50.0000	4.4053
55.0000	4.3918
60.0000	4.3762
65.0000	4.3575
70.0000	4.3419
75.0000	4.2892

Write down the value of qmax, ds, and hwhm for ϕ=5-10˚. (corresponds to the first entry). The value of ds will be used later in a calculation of lipid areas. Also, check that these values are nearly the same for the first 2-3 entries (ϕ=5-10˚ and ϕ=10-15˚). If not, something may be wrong with the image or the analysis.

Step 7: integrate_annulus: creates I(ϕ) data

Remember to change MaskI and beam center for each sample if necessary.

>> MaskI=mdo_5c;
>> X_cen=933; Y_cen=92;

7/22/08: The Gil Matlab function integrate_annulus_basic should now be used. It no longer uses C code

Integrate Input: SEE NOTE ABOVE

>> do_5a1=integrate_annulus(do_5c,[0.8,1.8]);

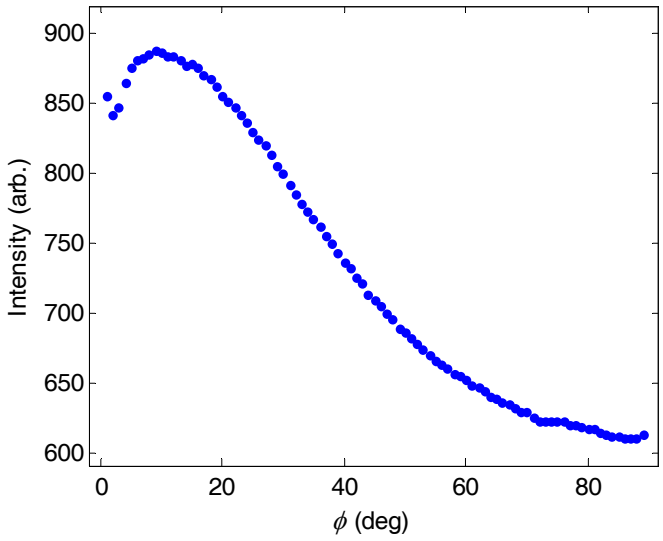
7/22/08 This function was used for all the data in Thalia's thesis, and it has been left in the MoaStudent folders. However, the new function integrate_annulus_basic should now be used. It no longer uses C code. Problems with the old integrate_annulus code is documented by Gilman Toombes and Thalia Mills in the folder ReadMe_IntegrateAnnulusProblems. The changes between the new and old version are minor. The old version made the small angle approximation, which is not valid in the WAXS region.

>> do_5a1=integrate_annulus_basic(do_5c,[0.8,1.8]);

[0.8,1.8] is the q range over which the intensity is averaged. The function automatically bins over a 1˚ ϕ range. The outputted variable do_5a1 is a column vector containing the I(ϕ) data. The first column has the ϕ angle and the second column has the intensity.

Nothing will appear: you will see the data when you do fits. To see it now use plottera (a user-defined plotting function which uses the built-in Matlab function "plot"). Note: 0 specifies the offset, 'b.' means blue dots (for the curve on the graph). The plotting options are the same as those for "plot". For more Matlab-defined plotting options, type "help plot."

>>plottera(soch30_1a1,0,'b.');



Note the zoom tool was used on the above plot.

SMOOTHING is unnecessary for the $I(\phi)$ plots. To see numbers which characterize the above plot, use the peak_phi function. [0, 85] is the ϕ range over which the function determines the plot statistics:

Input:
>> peak_phi(do_5a1,[0,85])

Output:
ans =

```
    phimax: 9.0000
      Imax: 887.7854
    phimin: 85.0000
      Imin: 612.0967
   phihalf: 37.9365
     Ihalf: 749.9411
```

If phimax=0, pick a different ϕ range (say [2,85]) over which to calculate the above quantities. Always look at the graph as well, to make sure that the output of peak_phi makes sense. Record phimax and phihalf in your editor file. You will use phimax to determine lower end of fitting range in the next step.

Step 8: Fitting I(ϕ) plots to obtain $S_{x\text{-}ray}$

In this step, you fit the $I(\phi)$ plot to obtain the x-ray order parameter (see T. T. Mills, et al., "Order parameters and areas in fluid-phase oriented lipid membranes using wide angle x-ray scattering").
The single order parameter fit is defined as (use function NewFitPhi_MSG):

$$I(\phi_L) = \frac{A}{8} \times \frac{\sqrt{m}}{\exp(m) D\left(\sqrt{m}\right)} \times \exp\left(\frac{m \cos^2 \phi_L}{2}\right) \times I_0\left(\frac{m \cos^2 \phi_L}{2}\right)$$

where I_0 is a modified Bessel function of the first kind, D is Dawson's integral and C can be adjusted to fit different amounts of sample with different exposure times. The parameter m is the most important because it describes the width of the Maier-Saupe distribution, from which the order parameter $S_{x\text{-}ray}$ is calculated.
The double order parameter fit is (use function NewFitPhi_MSG2):

$$I(\phi) = I_{back} + \frac{A_1}{8} \times \frac{\sqrt{m_1}}{\exp(m_1) D\left(\sqrt{m_1}\right)} \times \exp\left(\frac{m_1 \cos^2 \phi}{2}\right) \times I_0\left(\frac{m_1 \cos^2 \phi}{2}\right) + \frac{A_2}{8} \times \frac{\sqrt{m_2}}{\exp(m_2) D\left(\sqrt{m_2}\right)} \times \exp\left(\frac{m_2 \cos^2 \phi}{2}\right) \times I_0\left(\frac{m_2 \cos^2 \phi}{2}\right),$$

where the five fitting parameters are the constant background I_{back}, C_1 and m_1 for phase 1, and C_2 and m_2 for phase 2. (Note in the Mills et al. papers, C was used in place of A to stand for the normalization coefficient.)

Example input and output for NewFitPhi_MSG:
Input (what's sent into the function on the right-hand sign of the "=" sign):

```
>> [d1_do_5a1,f1_do_5a1,r1_do_5a1] = NewFitPhi_MSG(do_5a1, [9,80], [100,1,1],0,'do-5a1');
```

[9, 80] is the phi range over which the data is fit. The first number (9 in this case) should be the value for phimax that was found from peak_phi in step 7 (you will need to change this for

different samples). The last number (80 in this case) should be the largest ϕ angle for which the data is reasonable (it should not change).

[100,1,1] are the initial guesses for the 3 fitting parameters for single order parameter fit (corresponds to I_{back}, A, and m). These do not have to be changed from fit to fit.

0 is the zero-offset for the plot generated. It can be set to any value and does not affect the fitting results or output in any way. It is simply for plotting purposes.

'do-5a1' gives the name of the $I(\phi)$ data that your are fitting. Again, it is only for plotting purposes. Since you probably will want to re-plot the data and fit, the last two inputs to the function do not matter (they can always be set to 0 and 'junk', if you wish).

Output (what comes out of the function):

Output printed to screen:

```
Optimization terminated: relative function value
 changing by less than OPTIONS.TolFun.
df=69  SSE=932.239  RMSE=3.67569  adjR2=0.998439  R2=0.998483
Iback=433.954  A=2804.35  m=1.66342
errIback=20.1514  errA=159.859  errm=0.0967616
S=0.24561  errS=0.0148536
Fchains_030=0.289382  Fchains_6090=0.291168
Fscatt_030=0.479009  Fscatt_6090=0.209548
```

The program prints a number of results to the screen (as shown above). None of these are saved as Matlab variables. Typically, it's a good idea to record all of these parameters in your editor file (you never know if you might need them, and you'd have to repeat the fit to recover them.)

The first line gives general fit results (see Ch. 3, Section 3.3.5.3 in Thalia Mills' thesis for further information on these parameters):
df: degrees or freedom
SSE: sum of squares of the residuals
RMSE: root mean square
R2: coefficient of determination (R^2)
adjR2:degrees of freedom adjusted R^2
Iback, **A**, and **m** (see the fitting equations)
errIback, **errA**, and **errm**: These are half of the size of the 95% confidence intervals for the fit, as detemined by the Matlab function "nlparci." For example, the value of *m* should be recorded as "1.66 ± 0.10".
S: The x-ray order parameter $S_{x\text{-}ray}$, which is calculated from *m*. The 95% confidence interval (errS) is calculated from propagation of the error in m (see Mills thesis, Ch. 3). When recording the result for the order parameter, for the above example, write down "0.25 ± 0.01".
Fchains is the fraction of chains tilted between the specified angle β in real space (in this case 0 to 30, and then 60 to 90). β is the angle between the chain and the membrane normal.
Fscatt is the fraction of scattering which appears on the detector in the specified ϕ range.

Output saved as Matlab variables

Before the = sign in the call to the function, 3 local variables are assigned. These record the output of the fits in terms of $I(\phi)$ data (all normalized- see below). All are column vectors, with the first column being ϕ. The "1" in "d1", "f1", and "r1" means that these are the outputs of the single order parameter fit.

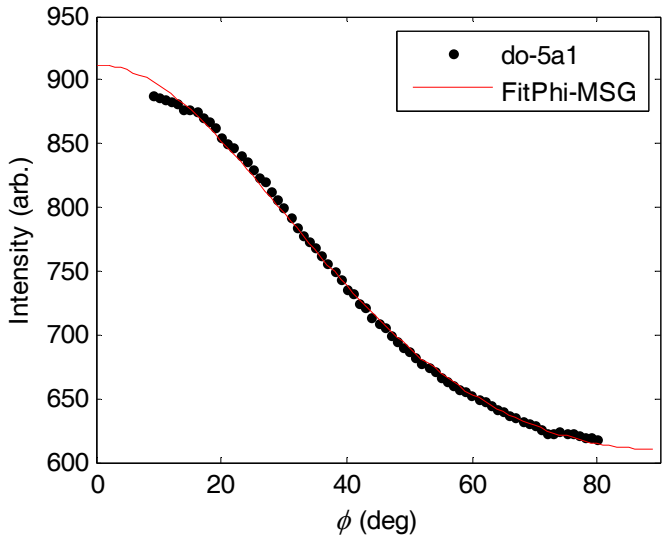
d1_ do_5a1: normalized $I(\phi)$ data

f1_ do_5a1: single order parameter fit to the data

r1_ do_5a1: residual (fit-data)

For the the data "d" and fit "f", Iback is subtracted before dividing by A for normalization. The residual variables "r" are divided by A. Note that although the outputted variables are normalized in this way according to the results of the fit, the fit is performed on the unnormalized $I(\phi)$ data.

The following graph is automatically generated (it shows the unnormalized data and fit):



Example input and output for NewFitPhi_MSG2:
*****This is for samples with 2 different phases**

Input:

```
>> [d2_do_5a1,f2_do_5a1,r2_do_5a1] = NewFitPhi_MSG2(do_5a1, [9,80], [100,1,1,1,10],0,'do-5a1');
```

Output:

```
Optimization terminated: relative function value
 changing by less than OPTIONS.TolFun.

df=67  SSE=930.778  RMSE=3.72723  adjR2=0.998395  R2=0.998485
Iback=432.132  A1=1420.72  m1=1.65877  A2=1398.1  m2=1.65083
errIback=199.218  errA1=2.71709e+008  errm1=818.597  errA2=2.71709e+008  errm2=753.227
S1=0.244895  S2=0.243677  errS1=125.677  errS2=115.666
P1=0.504011  P2=0.495989  errP1=68160.9  errP2=68160.9
AreaF1=0.503836  AreaF2=0.496164
Fchains_030=0.288429  Fchains_6090=0.29218
Fscatt_030=0.478123  Fscatt_6090=0.210197
```

The explanations for the input and ouptut follow those of the single order parameter fit, except there are two m's and two A's. Additional explanations specific to double order parmater fit:

[100,1,1,1,10] are the inital guesses for the 5 fitting parameters for double order parameter fit (corresponds to Iback, A1, m1, A2, and m2). Note that the results of the double order parameter fit are fairly robust and independent of the input parameters as long as the initial guess for m1 is not equal to the initial guess for m2.

The "2" in d2_do_5a1, etc. refers to double order parameter fit. The outputted data is normalized by first subtracting Iback and then dividing by A1+A2. The residual is normalized by dividing by A1+A2.

P1 and P2 are "phase fractions", although we are cautious in intepreting them as such. P1=A1/(A1+A2) and P2=A2/(A1+A2). errP1 and errP2 are the 95% confidence intervals propagated from errA1 and errA2 (see Mills thesis).

AreaF1 and AreaF2 are area fractions (percent of scattering observed on the detector due to each phase). Note that P1 (refers to a real-space quantity) is not the same as AreaF1 (a reciprocal space quantity), just as Fchains_030 is not the same as Fscatt_030.

***Note that in this example, S1=S2 (to within error) and the errors are all very large for the double order parameter fit. This is expected, as we are just looking at DOPC, which should be in a single phase.

The following compares the input and output for a sample known to have coexisting Ld and Lo phases (1:1 DOPC/DPPC + 15% cholesterol at 15˚C)

Single order paramter fit

```
>> [d1_dodpch15_15_3a1,f1_dodpch15_15_3a1,r1_dodpch15_15_3a1] =
NewFitPhi_MSG(dodpch15_15_3a1, [3,80], [100,1,1],0,'dodpch15-15-3a1');

Optimization terminated: relative function value
 changing by less than OPTIONS.TolFun.

df=75  SSE=355944  RMSE=68.8907  adjR2=0.969462  R2=0.970255
Iback=809.692  A=2917.17  m=11.5626
errIback=20.5821  errA=144.771  errm=1.1187
S=0.862687  errS=0.0143499
Fchains_030=0.934469  Fchains_6090=0.000413572
Fscatt_030=0.976159  Fscatt_6090=0.000233151
```

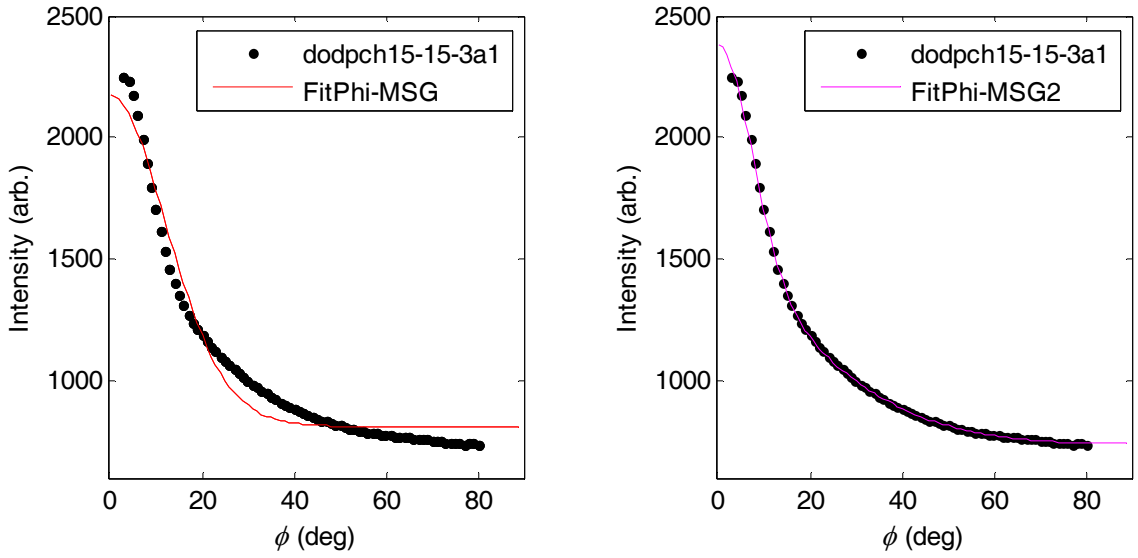
Double order parameter fit

```
>> [d2_dodpch15_15_3a1,f2_dodpch15_15_3a1,r2_dodpch15_15_3a1] =
NewFitPhi_MSG2(dodpch15_15_3a1, [3,80], [100,1,1,1,10],0,'dodpch15-15-3a1');

Optimization terminated: relative function value
 changing by less than OPTIONS.TolFun.

df=73  SSE=5350.04  RMSE=8.56085  adjR2=0.999528  R2=0.999553
Iback=690.077  A1=1338.78  m1=30.6183  A2=2677.67  m2=3.69999
errIback=12.5158  errA1=50.1772  errm1=1.27319  errA2=73.3677  errm2=0.22608
S1=0.950135  S2=0.524635  errS1=0.00211348  errS2=0.025327
P1=0.333325  P2=0.666675  errP1=0.010317  errP2=0.010317
AreaF1=0.301797  AreaF2=0.698203
Fchains_030=0.682094  Fchains_6090=0.0707235
Fscatt_030=0.785348  Fscatt_6090=0.056369
```

Comparison of fits (Left is single order parmaeter fit and right is double order parameter fit):



IMPORTANT NOTE: To make sure the fit worked, you should look for the following on the output screen:

Optimization terminated: relative function value
changing by less than OPTIONS.TolFun.

Sometimes, instead you might get the following error messages:

Maximum number of function evaluations exceeded;
increase options.MaxFunEvals

OR

Maximum number of iterations exceeded;
increase options.MaxIter

The best way to fix this problem is by using the outputted fitting parameters (Iback, A, and m) as the new inital guesses and doing another iteration. (T. Mills has never found it necessary to do more than 2 iterations.)

Step 9: Calculate Areas

See TTM's thesis or Mills et al., "Order parmaters in areas in fluid-phase oriented lipid membranes using wide angle x-ray scattering" for explanation of the equation used to calculate lipid areas:

$$A_{\mathrm{L}} = \frac{4}{\sqrt{3}} d^2 \langle \sec \beta \rangle .$$

The function you use is CalcArea(m,d). The m value is found from fitting, the d value is obtained from ds_do_5 (from the hwhm function). Use the first value in ds_do_5 (for ϕ=5-10˚), as long as it is similar to the 2nd and 3rd.

Input:
>>CalcArea(1.66,4.53)
Output:
Area1=73.3367 Area2=73.8395 Area3=87.2344 sec1=1.54748 sec2=1.55809 sec3=1.84074

Area1 is with $\langle \sec\beta \rangle \approx \langle \cos\beta \rangle^{-1}$
Area2 is with $\langle \sec\beta \rangle \approx 3 - 3\langle \cos\beta \rangle + \langle \cos^2\beta \rangle$
Area2 is usually best. Ignore Area3.

IV. QUICK REFERENCE: STEPS FOR WAXS ANALYSIS

Step 1: Load image and background files and check the subtraction.

Load the image:
>> do_5=slurp('wdo_117_cz.tif','c');

Load the background:
>> do_5b=slurp('wdo_118_cz.tif','c');

Subtract the image from the background to make the "corrected" image:
>> do_5c=do_5-do_5b;

If using CMU rotating anode:
>>junkimage=slurpNagle('10120002.img');

Step 2: Rotate (if necessary) and test by drawing line on subtracted image.

1. In this step you are setting MaskDold to MaskD as defined by the startup.m file (a white 1024 X 1024 array of 1's):
>>MaskDold=MaskD;
This is the unrotated, untouched MaskD: NEVER REDEFINE MaskDold! (Only do this step once.

2. Use the imrotate function (this is a Matlab function):
>>MaskD50=imrotate(MaskDold,-0.5);
This is a 0.5 degree clockwise rotation.

3. Now set MaskD to this
>>MaskD= MaskD50;

4. Rotate junk
>> junk=imrotate(do_5c,-0.5);

5. View the rotated image:
>>show(junk, [0 1000]);

6. Determine if this is correct by drawing a horizontal line. You may need to try more than on rotation angle, repeating steps 2-6.

7. Once you've decided the appropriate rotation you will need to rotate all of the images because these are the variables that are used throughout the analysis.
>> do_5=imrotate(do_5,-0.5);
>> do_5b=imrotate(do_5b,-0.5);
>> do_5c=imrotate(do_5c,-0.5);

Step 3: Masking

Follow directions printed to screen by Matlab function in order to make the mask:
>>mdo_5c=mask(do_5c,[0 1000]);

View the mask to make sure it is correct:
>> show(mdo_5c);

Step 4: Finding the beam center

Use the unsubtracted data and a clean MaskD so you can see the beam center:

Estimate the approximate center by displaying the image and zooming in on the beam:
>> show(do_5,[0 500]);

Make a plot of intensity vs. pixels in the vertical direction to find the vertical center (X_cen):
>> qzplot(do_5,[91,93]);
[91,93] specifies a range of pixels in the horizontal direction over which the intensity is averaged. The range should be chosen so that the vertical center Y_cen falls approximately in this range based on the image of the beam. Zoom in to find leading edge (closest to substrate) of beam peak- will be a larger pixel value and assign to X_cen:
>>X_cen=933;

Make a plot of intensity vs. pixels in the horizontal direction to find the horizontal center (Y_cen):
>> qrplot(do_5,[928,930]);
[928,930] specifies a range of pixels in the vertical direction over which the intensity is averaged. The range should be chosen so that the vertical center X_cen falls approximately in this range based on the image of the beam.
Assign Y_cen to the middle of the peak:
>>Y_cen=92;

Step 5: Sector plots

*First set (or check to make sure they are correct) X_cen, Y_cen, and MaskI because these are global variables used in intregration functions:
>>X_cen=933; Y_cen=92;
>> MaskI=mdo_5c;

>> sect_do_5c=sectplotNew(do_5c,[5:10:85],20);

Step 6: HWHM (half width at half maximum)

*Check that X_cen, Y_cen, and MaskI are correct. If not, redefine them.

>> [qmax_do_5,ds_do_5,hwhm_do_5,cl_do_5]=hwhm(do_5c,[0.8 1.8],[5:5:80],20);

Write down the value of qmax, ds, and hwhm for phi=5-10 deg. (corresponds to the first line in each variable).

Step 7: integrate_annulus: creates I(ϕ) data

*Remember to change MaskI and beam center for each sample if necessary.

>> do_5a1=integrate_annulus(do_5c,[0.8,1.8]);

The above was used for all the analysis in the Nagle lab as of 10/17/08. See the documentation in the ReadMe_IntegrateAnnulusProblems folder.
Should use istead: >> do_5a1=integrate_annulus_basic(do_5c,[0.8,1.8]);

To view:
>>plottera(soch30_1a1,0,'b.');

To view statistics of the $I(\phi)$ data, use the peak_phi function:
>> peak_phi(do_5a1,[0,85])
Record phimax and phihalf in your editor file. You will use the phimax to determine lower end of fitting range.

Step 8: Fitting I(ϕ) plots to obtain $S_{x\text{-}ray}$

Single order parameter fit:
>> [d1_do_5a1,f1_do_5a1,r1_do_5a1] = NewFitPhi_MSG(do_5a1, [9,80], [100,1,1],0,'do-5a1');

Double order parameter fit:
>> [d2_do_5a1,f2_do_5a1,r2_do_5a1] = NewFitPhi_MSG2(do_5a1, [9,80], [100,1,1,1,10],0,'do-5a1');

Step 9: Calculate areas

The function you use is CalcArea(m,d). The m value is found from fitting, the d value is obtained from ds_do_5 (from the hwhm function). Use the first value in ds_do_5 (for ϕ=5-10˚).

>>CalcArea(1.66,4.53)
Output:
Area1=73.3367 Area2=73.8395 Area3=87.2344 sec1=1.54748 sec2=1.55809 sec3=1.84074

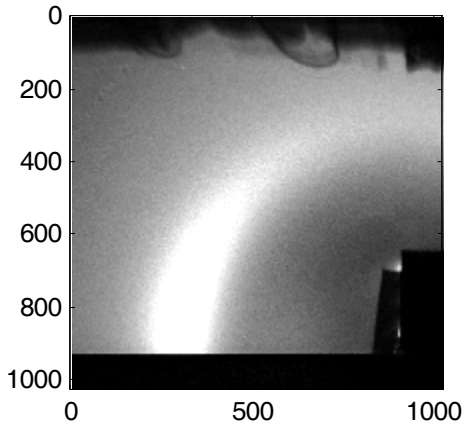
Area1 is with $\langle\sec\beta\rangle \approx \langle\cos\beta\rangle^{-1}$
Area2 is with $\langle\sec\beta\rangle \approx 3 - 3\langle\cos\beta\rangle + \langle\cos^2\beta\rangle$
Area2 is usually best. Ignore Area3.

V. OTHER USEFUL FUNCTIONS

1. fliplr

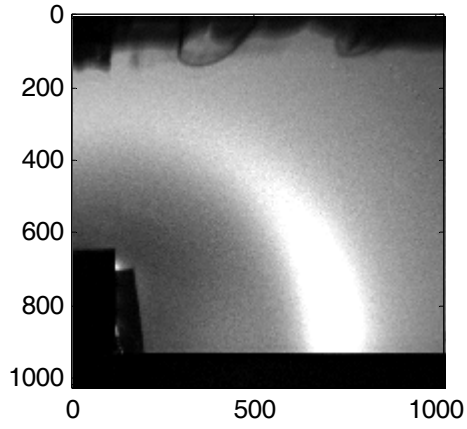
This is a built-in Matlab function which flips matrices left to right.

You may collect x-ray data that looks like this, and will be incompatible with the WAXS analysis functions:
>>show(testimage, [0 1000]);



Flip like so:
>> testimage=fliplr(testimage);

And view to make sure it was flipped correctly:
>> show(junk,[0 1000]);

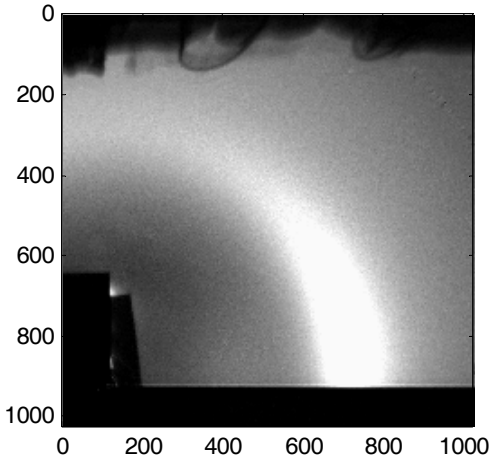


2. qlabel_WAXS

This is is a user-defined function (written by TTM) which labels an image with q values instead of pixels. It assumes a typical WAXS image with q_r and q_z running from 0 to ~2.0 $Å^{-1}$. (If you are experienced with Matlab, you can open up the .m file and change the labeling range.)

First display the image

>> show(do_5c, [0 1000]);



Now label the axes:

>>qlabel_WAXS(933,92,1.2740,2174)

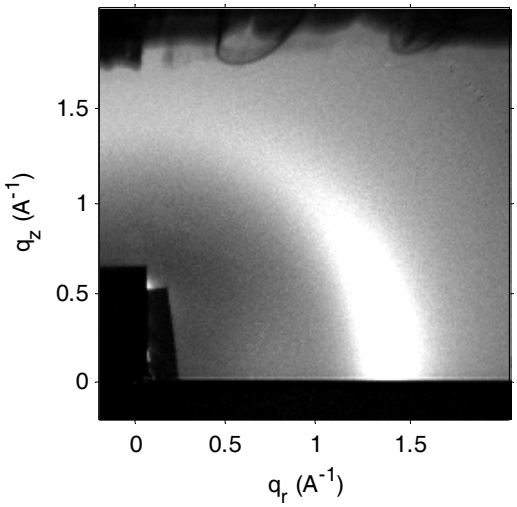
Description of input:

933=X_cen

92=Y_cen

X_Lambda=1.2740

Spec_to_Phos=2174



3. qlabel_LAXS

For their low angle scattering data, the Nagle lab uses Tiffview and other programs. However, these programs do not provide The following example shows how to label LAXS data for an SOPC LAXS image taken during the Nagle July 2007 CHESS run (goes from loading image to saving the figure). These were the steps used to create the figures for the Greenwood et al. CRAC paper:

1. Load file

>>sopc_064=slurp('sopc_064_cz.tif','c');

2. Find beam center using qrplot and qzplot, as for the WAXS data.

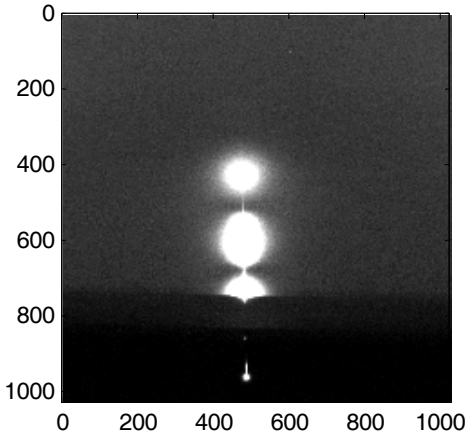
>> qzplot(sopc_064,[484 485]); >> X_cen=958;
>> qrplot(sopc_064,[954 955]); >> Y_cen=484;

3. Subtract constant background

>> sopc_064_a=sopc_064-100;

4. Display the image in the pixel range you want.

>> show(sopc_064_a,[0 250]);

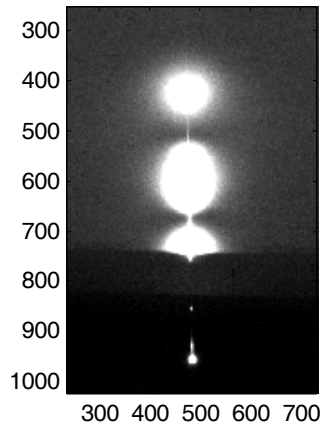


Now specify the axis range using the Matlab built-in function 'axis' (remember in Matlab)

>> axis([Y_cen-250 Y_cen+250 250 1024]);

Y_cen-250 is the minimum horizontal pixel to display
Y_cen+250 is the maximum horizontal pixel to display
250 is the minimum vertical pixel (remember starting at the top of the image)
1024 is the maximum vertical pixel

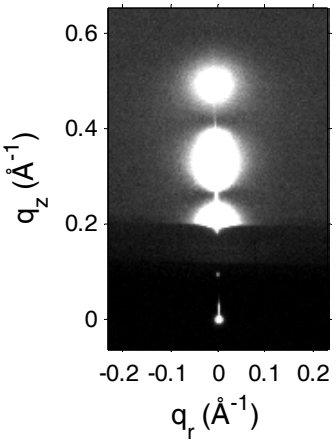
Note the min. and max. horizontal pixels were chosen so that the image is symmetric about the horizontal center (Y_cen). You will need to adjust these numbers depending on the image. As long as you last clicked on the figure above, the picture will be cropped like so and will overwrite the previous figure:



5. label the axes in reciprocal angstroms

>> qlabel_LAXS(958,484,1.1797,5728.0);

958 is X_cen
484 is Y_cen
1.1797 is X_Lamda (for the July 2007 CHESS data)
5728.0 is Spec_to_Phos (for the July 2007 LAXS *S* distance)



Of course all of these parameters will need to be adjusted depending on the image.

6. Now save the figure in the format you wish using the "print" command:

>>print -dtiff -r300 'sopc_064fig'

This saves a figure as a 300 dpi tiff with the following file name: sopc_064fig.tif

VI. MATLAB/MOA LOCATIONS IN NAGLE LAB

In the Nagle lab, linx2 (linux computer) has a Matlab version installed by Florin (in charge of the Math/Phys computer cluster. You must login as jianjunp in order to use Matlab (ask Jianjun for the password). This is not a stand alone license (the computer must be connected to the internet, as it checks for the license from the Carnegie Mellon server.)

The new windows computer (Biophys) has the student version of Matlab installed (Thalia's laptop also has the student version installed). The student version license is stand alone. You can use the student version with any login. The student version can be installed on 2 computers (can be linux/windows mixture) at once. If you wish to transfer the license, first run the uninstaller. Then run the installer CD on the computer you wish to transfer the license to. When you get to the activation part of the installation process, stop and call Mathworks technical assistance (phone # 508-647-7000) and tell them what you want to do. They will send you a new license.dat file (either by email or downloadable from internet). You will need to replace the existing license.dat file in the bin\win32 directory in the folder where you installed Matlab. You will need the following pieces of information:

1. Mathworks Account Info:
    email: jianjunp@andrew.cmu.edu
    password: tview314
    Case Number: 1-5IVIK6

2. Physical address of computer you wish to transfer the license to. Get this by going to: start→Run→open "cmd". In the command prompt, type:
>ipconfig /all
write down the physical address. For example, for the Nagle lab "Biophys" windows computer, it is: 00-1D-60-6D-B1-7D

A folder (with subfolders) called MoaStudent has been put on linx2 and on Biophys. It is the same version as on the CD (in the Matlab/WAXS folder in the Nagle lab).
Contents of MoaStudent (3 folders):
→WAXSmanual:
    MOA_User_Introduction.txt: Gil Toombes' introduction to the general x-ray analysis user package
    WAXS_Manual.doc, WAXS_Manual.pdf (This manual)
    MatlabGeneralIntro.pdf: a general introduction to Matlab written by David F. Griffiths

→MoaWindows: contains Moa files and WAXS analysis files used for windows computers
→MoaLinux: contains Moa files and WAXS analysis files used for linux computers

MoaWindows and MoaLinux contain example startup.m files which can be copied and pasted into your appropriate current working directory.

MoaWindows and MoaLinux have several subfolders: the folders you need to access are shown in the paths in the example startup files and are listed below (for the windows version as an example):

```
\MoaStudent\MoaWindows\bin_windows_9x
\MoaStudent\MoaWindows\EXISTING_CORRECTION_FILES
\MoaStudent\MoaWindows\New_Ideas
\MoaStudent\MoaWindows\New_Ideas\WAXS_Student
\MoaStudent\MoaWindows\New_Ideas\WAXS_Student\OtherUsefulFunctions
```

The WAXS_Student folder contains the functions written Thalia to add to the Moa package specifically for WAXS analysis.

*** Important: If you change any of the x-ray analysis functions, please keep a record and notify the other MOA users. If you make a change, make sure all the computers have the same version of the MOA functions loaded (This was last done by Thalia 10/17/08).

VII. EXAMPLE STARTUP FILES

Example startup.m file for Nagle lab windows computer (named "Biophys")

```
% Startup file for MOA softare.
% Edit to fit.
% Step 1. Adds MOA routines to the matlab search path.
% Step 2. Declares Intensity_Map, X_Distortion_Map, Y_distortion_Map,
%  Spec_to_Phos, X_lambda, X_cen, Y_cen, MaskD and MaskI as global variables
for use by
% MOA Routines.
% Step 3. Initializes Intensity_Map, X_Distortion_Map, Y_Distortion_Map to
match a
% particular detector.  These are usually stored in a .mat file.
% Look in src/image_correction/EXISTING_CORRECTION_FILES for examples
%  For instance Correction_2K.mat is the correction file for the 2K detector.
% Step 4.  Print reminder to set X_lambda, Spec_to_Phos, X_cen and Y_cen.
% Step 5.  Set default MaskD and MaskI as all on.  This means images are
unmasked.
% ************************************************
%  Note you must edit this file to match your system setup.
% ************************************************

% Step 1.
% Where are the MOA programs?
addpath 'F:\Program Files\MoaStudent\MoaWindows\bin_windows_9x'
addpath 'F:\Program Files\MoaStudent\MoaWindows\EXISTING_CORRECTION_FILES'
addpath 'F:\Program Files\MoaStudent\MoaWindows\New_Ideas'
addpath 'F:\Program Files\MoaStudent\MoaWindows\New_Ideas\WAXS_Student'
addpath 'F:\Program
Files\MoaStudent\MoaWindows\New_Ideas\WAXS_Student\OtherUsefulFunctions'
% This is where data is
addpath 'G:\Thalia\testWAXS'
% eg. a=pwd; eval(sprintf('addpath %s',a));

% Step 2.
global Intensity_Map; global X_Distortion_Map; global Y_Distortion_Map;
global Spec_to_Phos; global X_Lambda; global X_cen; global Y_cen;
global MaskD; global MaskI;

% Step 3.
load Correct_FliCam2.mat


% Step 4.
%fprintf(1,'\n Please initialize X_Lambda, Spec_to_Phos, X_cen and
Y_cen.\n');
X_Lambda = 1.274;  %October 06 CHESS run
Spec_to_Phos = 2174 ;  %October 06 CHESS run
%X_cen = 858.8;
%Y_cen = 934.7;

% Step 5.
MaskD = uint8(ones(size(Intensity_Map)));
MaskI = uint8(ones(size(Intensity_Map)));
```

Example startup.m file for Nagle lab linux computer (named "linx2")

```
% Startup file for MOA softare.
% Edit to fit.
% Step 1. Adds MOA routines to the matlab search path.
% Step 2. Declares Intensity_Map, X_Distortion_Map, Y_distortion_Map,
%  Spec_to_Phos, X_lambda, X_cen, Y_cen, MaskD and MaskI as global variables
for use by
% MOA Routines.
% Step 3. Initializes Intensity_Map, X_Distortion_Map, Y_Distortion_Map to
match a
% particular detector.  These are usually stored in a .mat file.
% Look in src/image_correction/EXISTING_CORRECTION_FILES for examples
%  For instance Correction_2K.mat is the correction file for the 2K detector.
% Step 4.  Print reminder to set X_lambda, Spec_to_Phos, X_cen and Y_cen.
% Step 5.  Set default MaskD and MaskI as all on.  This means images are
unmasked.
% ************************************************
%  Note you must edit this file to match your system setup.
% ************************************************

% Step 1.
%addpath MOA/bin_linux
% eg. a=pwd; eval(sprintf('addpath %s',a));
addpath '/home/jianjunp/Desktop/MoaStudent/MoaLinux/bin_linux'
addpath
'/home/jianjunp/Desktop/MoaStudent/MoaLinux/EXISTING_CORRECTION_FILES'
addpath '/home/jianjunp/Desktop/MoaStudent/MoaLinux/New_Ideas'
addpath '/home/jianjunp/Desktop/MoaStudent/MoaLinux/New_Ideas/WAXS_Student'
addpath
'/home/jianjunp/Desktop/MoaStudent/MoaLinux/New_Ideas/WAXS_Student/OtherUsefu
lFunctions'
% This is where data is
addpath '/home/jianjunp/Desktop/Thalia/testWAXS'

% Step 2.
global Intensity_Map; global X_Distortion_Map; global Y_Distortion_Map;
global Spec_to_Phos; global X_Lambda; global X_cen; global Y_cen;
global MaskD; global MaskI;

% Step 3.
%load
MOA/src/Image_Processing/EXISTING_CORRECTION_FILES/Correction_1KBin2.mat
   % Set up for 1K detector running in Bin 2.
load Correct_FliCam2.mat

% Step 4.
%fprintf(1,'\n Please initialize X_Lambda, Spec_to_Phos, X_cen and
Y_cen.\n');
X_Lambda = 1.274;   % value for Oct06 CHESS run
Spec_to_Phos = 2174 ;  % value for Oct06 CHESS run

% Step 5.
MaskD = uint8(ones(size(Intensity_Map)));
MaskI = uint8(ones(size(Intensity_Map)));
```