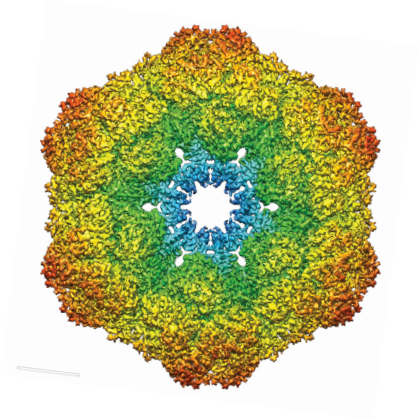
# THE 8TH BRAZIL SCHOOL FOR SINGLE PARTICLE CRYO-EM

## HANDS-ON
## Playing with IMAGIC
### Terminal Window



Version 12-June-2018
www.brazil-school.org





Purpose of this document is to play with IMAGIC in a terminal window and to learn what is averaging and what are Fourier transforms and filtering.

# 1. Remember: Some IMAGIC Basics

This chapter is on how to work with the **IMAGIC** software.

An "**IMAGIC** image file" consists of a header file (".hed") and the image density file (".img"):

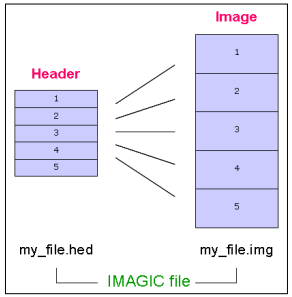



**Fig. 1:** IMAGIC file

The image file contains the actual image density values, while the header file contains information about the images ("meta data") as a set of records that can be accessed through different labels. For example:

| | |
|---|---|
| IMN | image location number (1,2,3,...) |
| IXLP | number of lines per image |
| IYLP | number of pixels per line |
| IZLP | number of sections if input is a 3-D volume |
| REF | multi-reference number |
| CLASSNO | class number |
| ALPHA | Euler alpha angle |
| BETA | Euler beta angle |
| GAMMA | Euler gamma angle |
| etc… | |

An additional PLT text file can be associated to an **IMAGIC** file to store further meta-data like:

coordinates of particles
contour of masks
image numbers
Euler angles
graphics (curves)
etc…

The PLT file may contain a maximum of five numerical values per line, separated by blanks or by commas.





A few other **IMAGIC** text (ASCII) files may be generated during processing:

CLS files are classification files containing classes and their members

LIS files contain information printed during execution of a program

LOG files contain output of programs when running as batch job (script)

DAT files contain data for various purposes

DFF (deFault Files) are used to store your last answers

The **IMAGIC** coordinate system is a right-handed system with its (1,1) origin in the top-left corner of the image. The length of the lines (number of rows/columns) is **NY** and the number of lines is **NX**:

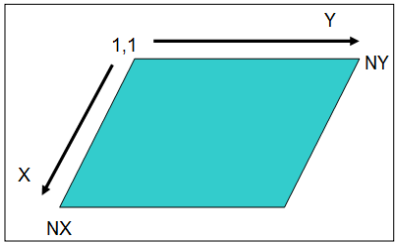



**Fig. 2:** IMAGIC 2-D coordinate system

The **IMAGIC** coordinates for a 3-D volume are the following:

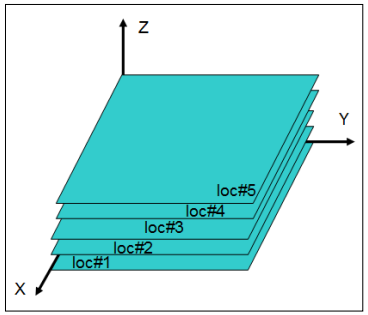


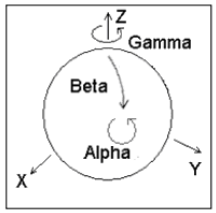

**Fig. 3:** IMAGIC 3-D coordinate system

Note that **Z** = **X** x **Y** as required for a right-handed co-ordinate system.

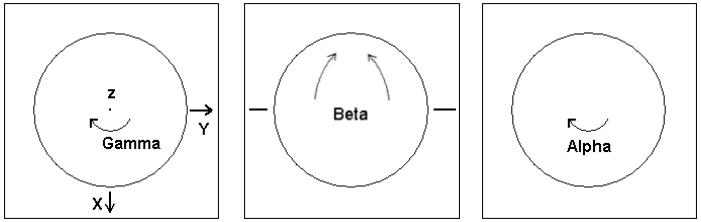
In **IMAGIC** 3-D orientations are defined by three Euler angles Alpha, Beta and Gamma.

From the perspective of an external viewer (like every IMAGIC image used/created in commands ANGULAR-RECONSTITUTION, THREED-SURFACE, THREED-FORWARD, etc.) the Euler angles are defined as follows:

The first rotation is a rotation around the Z-axis by GAMMA, followed by a rotation BETA around the new Y-axis and a rotation ALPHA around the new Z-axis.




But normally a user does not think in this way but tries to imagine how the particle would look like "in his hands":




a) Look at the particle along the Z-axis ("north pole")
b) Rotate the particle clockwise by Gamma
c) Rotate the particle into the plane clockwise by Beta
d) Rotate the particle clockwise by Alpha

PLEASE NOTE:

The important angles to define a 3-D orientation are Beta and Gamma. Alpha is only the final in-plane rotation.





**IMAGIC** is started in one (or more) command window(s). The commands are interactive and are followed by specific questions. Every question also has an associated help, which can be accessed by typing "?"

**IMAGIC** command questions will often have a default value which appears in brackets [default]. You can use the default value by just hitting ENTER/CR.

**IMAGIC** remembers the last values you have entered for a specific command. These values (stored in the DFF files) become the default values the next time the command is started in that working directory. In general, if you do not know how to answer a question, the default values serve as an intelligent first guess.

File names are only suggested. You are free to choose whatever names you wish. However, bear in mind you will have to remember what you've chosen for the next commands.

MPI refers to parallel processing. If your notebook computer (or any other type of computer) has multiple cores commands which are using parallel processing will ask you if you want to run the command in parallel or not. In the beginning always first run on a single core (non-parallel mode) which will give you more feedback (answer NO). Later when using many images your answer may be YES. Note that the number of processors to be used should be at least the number of nodes PLUS 1:

```
Use MPI parallelisation [YES]                    : yes
Number of processors to be used                  : 3
```

Throughout this hands-on, words that appear in GREEN refer to **IMAGIC** commands. Words in red are required/suggested input values. Suggested file names are in blue.

YOUR NOTES:

# 2. Playing with IMAGIC

## 2.1. Commands CREATE-IMAGE and DISPLAY

1. To start, open a command window and run **IMAGIC** by typing i or imagic.

```
my_notebook> imagic

IMAGIC-COMMAND:
```

2. Use CREATE-IMAGE to create a (test) image. First use the default options, i.e., just hit the ENTER button.

```
IMAGIC-COMMAND: create-image

** TESTIM welcomes you **

Output filename, image loc#s           : my_image
Image dimensions X,Y                   : 256,256
IMAGIC data formats you can choose     : real
Currently you can choose               : blobs      you choose
```

3. Use a separate command window to DISPLAY the image on the screen:

```
my_notebook> imagic

IMAGIC-COMMAND: display

Input image file, image loc#s          :
```

4. The first question that will appear on the screen asks for the file you wish to display. Use the test image (my_image), which you just created. If you have forgotten the names of the images type:

```
Input image file, image loc#s        : $dir        MS Windows
```

or





```
Input image file, image loc#s        : $ls         Linux
```

These are direct operating system calls to get a list files in your directory. Look for files with the extension ".img".

> NOTE:
>
> You can always launch operation system commands using a $ .

Now specify the name of the image file you want to display:

```
Input image file, image loc#s     : my_image
```

DISPLAY first shows the current settings:

```
Current DISPLAY settings:

Input image FILE name             : my_image
LOCATION numbers                  : 1,1
Output DEVICE                     : XWINDOWS
DEVICE window size                : 800,1024
SCALE factor                      : 1.0
MINX, MAXX                        : 1,256
MINY, MAXY                        : 1,256
GREYVALUES                        : 2D local survey
ERASE screen before display       : no
STARTING point (top left)         : 1,1
Display of NAME & information     : file name and location
Video lookup table (VLT)          : linear black/white
...
Parameters to be changed:
NO_CHANGES(=DISPLAY), SETTINGS, OPTIONS [NO]:
```

5. Hit the ENTER key, which means that the default NO CHANGES is used and the image will be displayed using the current settings.

6. If you want to change certain settings go for the words written in capitals. For example, to change the scaling factor:

```
Parameters to be changed:
NO_CHANGES(=DISPLAY), SETTINGS, OPTIONS [NO] : scale
```

```
   Image size is: 128 x 128
Give scale factor for display                : 2
```

7. Hit ENTER to apply the changes and to go back to the DISPLAY parameter settings.

8. Then use option GREYVALUES to display the image(s) with different grey levels. Start with option INTERACTIVE and black, white levels -10,10 and display. Use other black, white levels and display to see how brightness and contrast of the displayed image changes. Also play around with the options SURVEY, 2D_LOCAL.

> NOTE:
>
> When you are displaying a gallery of images (aligned images, class averages, 3-D sections etc. you should preferably use the same grey values given interactively or using the GREYVALUES options SURVEY, 3D or GLOBAL.

9. Play with CREATE-IMAGE (or its synonym TEST-IMAGE) again. Create REAL images, but of different sizes, and using different options. DISPLAY the images.

10. After this, create a 256,256 image of a SIEMENS star.

11. Use the COARSE-IMAGE command with factors of 4 (or 6) to coarsen the images. DISPLAY to see the effects of sampling size on your resolution. What is the size of your image now? What detail can you still see?

12. Use the BLOW-UP-IMAGE command (option BLOWUP) to blow up the coarsened images back to the original size. DISPLAY the results and compare those with the original and the coarsened images. Compare the output images of COARSE-IMAGE and BLOW-UP-IMAGE.

> TIP:
>
> You can open multiple DISPLAY windows from different command/terminal windows.

13. Use CREATE-IMAGE to create a new test image using the option BLOBS. Use MOVE-IMAGE to ROTATE, SHIFT and COMBINE (ROT&SHIFT) the image. DISPLAY the results. Remember the IMAGIC coordinate system (chapter 1).





## 2.2. Noise

Create a sequence of test images and add noise to them.

1. Use CREATE-IMAGE to create 256 images of CHECKERS. Make the images REAL and of size 128,128. Note: To create a file with multiple images you specify the start and final location numbers, like my_image,1,256 where my_image is the file name, 1 is the start location and 256 is the final location.

```
IMAGIC-COMMAND: create-image

Output filename, image loc#s             : my_image,1,256
Image dimensions X,Y                     : 128,128
IMAGIC data formats you can choose       : real
Currently, you can choose                : checkers
Checker size                             : 16
```

2. Use ADD-NOISE to add noise to the images:

```
IMAGIC-COMMAND: add-noise

Option used                              : ADD_NOISE
Input filename, image loc#s              : my_image
Output filename, image loc#s             : my_image_noise
Mode of operation                        : noise
Mean, sigma of Gaussian noise            : 0,10
Random number seed                       : 0
```

3. DISPLAY the results.

## 2.3. Noise Reduction by Image Averaging

You will get some impression what image averaging means and why image processing can enhance the resolution of noisy images.

1. Use SUM-IMAGES (option SOME_SUM) to make sums of 2, 8, 64, and 256 of the images from the file you added noise to. Input file is my_image_noise. Suggested output file names are: my_sum_2, my_sum_8... and my_sum_256. In "Location number(s) wanted:" you should specify 1-2, or 1-8, or 1-64, or 1-256 accordingly:

```
IMAGIC-COMMAND: sum-images

Mode of summing                          : some_sum
Input file, NO loc#s                     : my_image_noise
Output file, ONE loc#                    : my_sum_2      etc.
Variance file, ONE loc#                  : none
Location number(s) wanted                : 1-2           etc.
Numbers wanted                           : all
```

2. DISPLAY each result and see the effects of image averaging on the signal to noise ratio.

YOUR NOTES:





## 2.4. MODE Commands

MODE commands allow the creation of batch/script files with a collection of commands that can be run from the command window.

1. Use command MODE-ACCUMULATE:

```
IMAGIC-COMMAND: mode-acc

IMAGIC-COMMAND (ACC.) :
```

2. Now use commands CREATE-IMAGE and SURVEY-DENSITIES. The input file for SURVEY-DENSITIES is the output file of CREATE-IMAGE.

   Remember: You can find out about a command using HELP, and about a question using ? .

   Stop accumulating commands with MODE-STOP. You can run the file with the accumulated commands in the command, e.g. bigjob.b (Linux) or bigjob.bat (MS Windows), respectively:

```
IMAGIC-COMMAND (ACC) : mode-stop

Filename for batch/script file [bigjob] : bigjob

   Command (batch/script) file bigjob.b

   is available now

   To run the job on the monitor please use

   ...
```

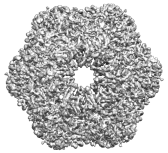
3. Use command MODE-ACCUMULATE again and accumulate command TEST-IMAGE and some other commands like ARITHM-WITH-IMAGE, SURVEY...

4. Stop accumulating commands and run the script file with MODE-SEND.

5. Use MODE-PROTOCOL to create a protocol file.

   Again use command CREATE-IMAGE with some other commands.

   Stop the protocol mode with command MODE-STOP. Edit the protocol file using a text editor of your choice.



# 3. The Fourier Transform

This is an exercise that will provide some basic insight into the Fourier transform (FT). Fourier transforms will be covered in the lectures; the aim of this exercise is to familiarize you with the principles of the Fourier transform and the associated **IMAGIC** commands.

Start with one-dimensional (1-D) Fourier transforms and later play around with 2-D images.

## 3.1. Test Curves

1. Open a command window. Create a curve (a 1-D image) using command CREATE-CURVE. Create an IMAGE with amplitude 1 and wavelength 0.1:

```
IMAGIC-COMMAND: create-curve

Mode of output                          : image
Output file, curve loc#s                : my_curve
Length of curve                         : 512
Curve option                            : sine
Amplitude of signal                     : 1
Wavelength of periodic signal           : 0.1
```

2. Create additional curves into the same file, using sequential locations:

```
IMAGIC-COMMAND: create-curve

Mode of output                          : image
Output file, curve loc#s                : my_curve,2
Length of curve                         : 512       as before
Curve option                            : sinc      you choose
Amplitude of signal                     :           you choose
Wavelength of periodic signal           :           you choose
```





3. Open a second command window and use the command PLOT to display the curves. Like the command DISPLAY, the PLOT command first displays the current settings, which you can change by typing the names in capitals. Giving ENTER means "NO CHANGES" and the curve is displayed. You want to compare all curves so it is a good idea to fix the vertical scaling of the plot according to the chosen amplitudes with the option VERTICAL:

```
Change settings (MULT,HOR,VER,SURVEY...) [NO]: vert
Minimum, maximum for vertical scaling        : -10,10
```

Plot the curves.

4. If you want to display all curves at the same time use option MULTIPLE:

```
Change settings (MULT,HOR,VER,SURVEY...) [NO]: mult
Number of curves per plot                    :     you choose
```

## 3.2. Fourier Transform

1. Now calculate the Fourier transforms of the curves (my_curve) with the command CURVE-FORWARD-FT. The suggested output file name is my_curve_ft.

```
IMAGIC-COMMAND: curve-forw

Input file, curve loc#s                 : my_curve
Output file, curve loc#s                : my_curve_ft
Options you can choose                  : FORWARD_FT
```

2. PLOT the Fourier transforms (May be, you want to set MULTIPLE back to 1. You can also use option VERTICAL (values like -1000,1000) to use the same vertical scaling for all curves.

## 3.3. Curves and their Fourier Transforms

1. There is an interactive command to create images/curves and to display the related Fourier transforms: PLAY-WITH-FOURIER-TRANSFORMS.

```
IMAGIC-COMMAND: play-with-fourier

Play with                              : 1d_image
Mode of input                          : create
Choose curve                           : triangle
Length of curve                        : 512
Amplitude of the curve                 : 1
Width of signal                        : 0.25
```

Both, the curve and the related Fourier transform will be displayed.

You can create the next curve by typing NEXT_IMAGE:

```
How to continue                        : next
...
```

or leave the command by giving STOP_PLAYING.

```
How to continue                        : stop
```

NOTE:

In command PLAY-WITH-FOURIER-TRANSFORMS you can also use a curve/image from an input file. Use Mode of input : FILE





2. Now create various sine curves using different amplitudes 2, 4, 6, and 8. Always use wavelength 0.1:

```
IMAGIC-COMMAND: play-with-f

Play with                              : 1d_image
Mode of input                          : create
Choose curve                           : sine
Length of curve                        : 512
Amplitude of the curve                 :          you choose
Wavelength of periodic signal          : 0.1
```

Look at the vertical scaling of the plots and notice how the amplitudes are related to the height of the peaks in the Fourier transforms.

3. Continue using SINE curves now with fixed amplitude but changing the periodicity. Use amplitude 1 and wavelength of periodic signal 0.1, 0.25 etc.

```
IMAGIC-COMMAND: play-with-f

Play with                              : 1d_image
Mode of input                          : create
Choose curve                           : sine
Length of curve                        : 512
Amplitude of the curve                 : 1
Wavelength of periodic signal          :          you choose
```

Notice how the periodicity changes the Fourier transforms.

NOTE:

The sine (or cosine) curve in the image space corresponds to a peak in Fourier space. The amplitude and wavelength of the sine (or cosine) curve are "related" to the height and the position in Fourier space.

## 3.4. Relationship between Image Space and Fourier Space

1. Create a new curve file (my_curve) with a number of SINE waves with various amplitudes and wavelengths. Create these curves into the same file, using sequential locations (my_curve,1 , my_curve,2 , … my_curve,20):

```
IMAGIC-COMMAND: create-curve

Mode of output: image
Output file, curve loc#s                : my_curve
Length of curve                         : 512
Curve option                            : sine
Amplitude of the curve                  :          you choose
Wave length of periodic signal          :          you choose

IMAGIC-COMMAND: create-curve

Mode of output: image
Output file, curve loc#s                : my_curve,2
Length of curve                         : 512
Curve option                            : sine
Amplitude of the curve                  :          you choose
Wave length of periodic signal          :          you choose

IMAGIC-COMMAND: create-curve

...

IMAGIC-COMMAND: create-curve

Mode of output: image
Output file, curve loc#s                : my_curve,20
Length of curve                         : 512
Curve option                            : sine
Amplitude of the curve                  :          you choose
Wave length of periodic signal          :          you choose
```

2. Calculate the Fourier transforms of the new curves with CURVE-FORWARD-FT and store them in file my_curve_ft.





```
IMAGIC-COMMAND: curve-forw

Input file, curve loc#s                 : my_curve
Output file, curve loc#s                : my_curve_ft
Option used for current command         : FORWARD_FT
```

3. As before PLOT both, the curves (my_curve) and the related Fourier transforms (my_curve_ft).

4. Now, sum all curves (my_curve) with command SUM-CURVE:

```
IMAGIC-COMMAND: sum-curve

Choose summing option                   : total_sum
Input file, NO loc#s                    : my_curve
Output file, ONE loc#                   : my_curve_sum
Variance file, ONE loc#                 : none
```

5. PLOT the result (my_curve_sum).

**NOTE:**

a) Summing a huge number of sine (and cosine) curves with different amplitudes and wavelengths creates a non-periodic curve.

b) And even more: one can say that any (real) curve can be constructed by a combination of sine and cosine waves of different wavelengths and amplitudes.

6. Calculate the Fourier transform (my_curve_sum_ft) of the new curve (my_curve_sum) using command CURVE-FORWARD-FT. PLOT the Fourier transform (my_curve_sum_ft).

**NOTE:**

The sum of sine (or cosine) curves in the image space relates to the sum of the sine (or cosine) peaks in Fourier space.

7. Now calculate a reverse Fourier transform with command CURVE-REVERSE-FT. Input is the Fourier transformed curve (my_curve_sum_ft).

8. PLOT the reverse Fourier transform. Note that it is the same as the original curve (my_curve_sum) before the Fourier transformation.

NOTE:

The information in a curve/image ("Image Space" or "Real space") and in the Fourier transform ("Fourier Space") is equivalent. The curves/images in both spaces contain all curve/image information.

This means that in image processing it is possible to go from one space to the other without losing any image information!

NOTE:

Mode of Fourier transforms: FORWARD means going from a curve/image ("Image Space" or "Real space") to its Fourier transform ("Fourier Space"). The transformation going from the Fourier transform to the curve/image is called REVERSE (sometimes also called "inverse").

YOUR NOTES:





## 3.5. Fourier Space and Filtering

1. Again, create sine curves: one with a long wavelength and another with a short wavelength (0.5 and 0.002) using CREATE-CURVE. Remember to save these two curves to the same file using the same file name and different location numbers, like in my_curve and my_curve,2 .

2. Calculate the Fourier transforms with CURVE-FORWARD-FT and compare the results using command PLOT (use option MULTIPLE).

NOTE:

(a) The first sine curve shows "large" details, which in Fourier space are represented by densities close to the centre of the Fourier transform.

(b) The second sine curve shows "small" details, which in Fourier space are located far away from the centre.

(c) Usually the very "large" details (density ramps, for example) and the very "small" details (mostly noise) are hiding the motif, which you are interested in.

(d) As seen in your two test curves Fourier space offers a nice possibility to remove this unwanted information: filter the Fourier transform close to the centre ("low frequencies") and at the borders ("high frequencies").

3. Create a new curve (my_curve) with CREATE-CURVE, with BLOCKWAVE for the curve option, 500 for the amplitude and 0.25 for the wave length

4. Next create a ramp in location #2 (my_curve,2). Run CREATE-CURVE with option RAMP and use 1 for the amplitude and 3 for the inclination.

5. Now add the block-wave and the ramp curves with SUM-CURVE using summing option TOTAL_SUM. Do not calculate a standard deviation (give none). Use the output name my_curve_ramp.

6. PLOT the sum (my_curve_ramp). Notice how the signal (block-wave) is disturbed by the ramp.

7. Remove these "large" unwanted details (low frequencies) with a high-pass filter in Fourier space using command CURVE-FILTER and option HIGH_PASS and low-frequency cut-off 0.01.

```
IMAGIC-COMMAND: curve-filt

Input file, curve loc#s                 : my_curve_ramp
Output file, curve loc#                 : my_curve_ramp_hp
Filter option                           : high_pass
Low frequency cut-off                   : 0.01
Remaining LF transmission               : 0,0
```

8. PLOT the high-pass filtered curve (my_curve_ramp_hp).

NOTE:

Large details can be suppressed by high-pass filtering in Fourier space. But, be aware that the low-frequency component of the original curve can also be affected.

Errors/artefacts can occur at the edges.

9. Next create a curve with Gaussian noise in location #3 (my_curve,3). Run CREATE-CURVE with option NOISE, and use 0, 40 for the MEAN and SIGMA.

10. Add the noise to the block-wave signal with the command CURVE-SUM.

```
IMAGIC-COMMAND: sum-curve

Choose summing option                   : some_sum
Input file, NO loc#s                    : my_curve
Output file, curve loc#                 : my_curve_noise
Output standard deviation file          : none
Location number(s) wanted               : 1;3
```

11. PLOT the sum (my_curve_noise). Notice that the signal is disturbed by noise.





12. Remove these "small" unwanted details (high frequencies) by low-pass filtering in Fourier space with command CURVE-FILT using option LOW_PASS and a high-frequency cut-off of 0.1.

```
IMAGIC-COMMAND: curve-filt

Input file, curve loc#s                 : cmy_curve_noise
Output file, curve loc#                 : my_curve_noise_lp
Filter option                           : low_pass
High frequency cut-off                  : 0.1
```

13. PLOT the low-pass filtered curve (my_curve_noise_lp).

NOTE:

Noise (small details / high frequencies) can be removed by a (Fourier space) low-pass filter. But, of course, also fine details of the original curve are affected.

14. Finally, create a curve with disturbing low (ramp) and high frequencies (noise). Add all three curves (my_curve) with CURVE-SUM, using the option TOTAL_SUM (to get my_curve_ramp_noise).

15. PLOT the curve (curve_ramp_noise) to see how the curve is disturbed by the ramp and by Gaussian noise.

16. To remove both unwanted information call CURVE-FILTER again now using a BAND_PASS filter, which is a combination of a high-pass and a low-pass filter:

```
IMAGIC-COMMAND: curve-filt

Input file, curve loc#s                 : my_curve_ramp_noise
Output file, curve loc#                 : my_curve_ramp_noise_bp
Option to choose                        : band
Low frequency cut-off                   : 0.01
Remaining LF transmission               : 0
High frequency cut-off                  : 0.1
```

17. As usual PLOT the filtered curve (my_curve_ramp_noise_bp) and see how a band-pass can remove unwanted information.

TIP:

During image processing, it is a good idea to have your own naming convention, so that in a list of files you can easily understand what each file is from its name. For example, in this case "sine" is your input file containing a sine wave and "sine_mask" is your output file with the masked sine wave.

YOUR NOTES:





## 3.6. 2-D Images and Fourier Transforms - First Steps

We now want to play with 2-D images and their Fourier transforms.

1. Start by using PLAY-WITH-FOURIER-TRANSFORMS using 2-D images (option 2D_IMAGE). First, create a SINE-wave image (my_sine). When asked for a PERIODICITY choose 0.1:

```
IMAGIC-COMMAND: play-with-f

Play with                                  : 2d_image
Mode of input                              : create
Choose curve                               : my_sine
Image dimensions X,Y                       : 512,512
Wavelength of periodic signal              : 0.1
Direction of wave                          : horizontal
Mask radius, drop-off (0: no mask)         : 0          no mask
Grey values to scale display (0: auto)     : 0          automatic
```

   PLAY-WITH-FOURIER-TRANSFORMS will display the create images in the first display window and the related Fourier transform in the second display window.

2. You should now simply see two "points" in the Fourier transform of the input image. This is the Fourier space representation of a sine wave, with the periodicity you have specified (0.1 was suggested). In Fourier space, high frequency information (i.e. corresponding to short wave lengths in real space) is found far from the origin, close to the edge of the Fourier transform image, whilst low frequency information is found around the centre of the Fourier transform.

3. Since you are examining 2-D images, note the sine waves also have a direction. In this case the sine wave travels horizontally.

4. To demonstrate this effect, continue with NEXT_IMAGE and create the same image again but now using option VERTICAL. Notice how the direction has changed in both, the image and the related Fourier transform.

5. Finally continue with NEXT_IMAGE now using the direction option ANGLE. Use 45 (degrees). As before you should find that the direction of the frequency space points has rotated by 45 degrees.

```
IMAGIC-COMMAND: play-with-four

Play with                                 : 2d_image
Mode of input                             : create
Choose curve                              : sine
Image dimensions X,Y                      : 512,512
Wavelength of periodic signal             : 0.1
Direction of wave                         : angle
Rotatation angle                          : 45
Mask radius, drop-off (0: no mask)        : 0          no mask
Grey values to scale display (0: auto)    : 0          automatic
```

6. However, when using angles, which are not multiples of 45 (let's say: 30) there will no longer be just points. This is because the rotated sine waves are no longer continuous, and also have interpolation effects from the rotation. In other words, the images are no longer pure sine waves

7. Use command PLAY-WITH-FOURIER-TRANSFORMS to play around with other images. Learn how the related Fourier transforms look like.

8. You can use command CREATE-IMAGE to create 2-D image files (my_image), which you can sum with command SUM-IMAGES (my_image_sum). Use command PLAY-WITH-FOURIER-TRANSFORMS to visualise images and Fourier transforms.

NOTE:

Like in the 1-D case for curves, any (real) 2-D image can be seen as a combination of sine and cosine waves of different frequencies, and in different directions. A Fourier transform decomposes a real image into its constituent sine / cosine waves. Only sine waves that fit perfectly on the sampling grid, have perfect diffraction peaks in Fourier space.





## 3.7. 2-D Images and Fourier Transforms - Masks

1. The motif in your images is normally close to the centre of the frame and you are not interested in the information near the frame edge. To minimize the influence of background you may want to mask out the edges.

2. The effect of masking can also be visualised using PLAY-WITH-FOURIER-TRANSFORMS. Create a SINE image rotated by 30 degrees. First use no mask. To better visualise the result adapt the display grey-value scale:

```
IMAGIC-COMMAND: play-with-f

Play with                                 : 2d_image
Mode of input                             : create
Choose curve                              : sine
Image dimensions X,Y                      : 512,512
Wavelength of periodic signal             : 0.1
Direction of wave                         : angle
Rotatation angle                          : 30
Mask radius, drop-off (0: no mask)        : 0          no mask
Grey values to scale display (0: auto)    : 1,8
```

3. Subsequently mask out the centre of the image with a soft drop off circular mask. You should see the peaks more clearly now.

```
IMAGIC-COMMAND: play-with-f

Play with                                 : 2d_image
Mode of input                             : create
Choose curve                              : sine
Image dimensions X,Y                      : 512,512
Wavelength of periodic signal             : 0.1
Direction of wave                         : angle
Rotatation angle                          : 30
Mask radius, drop-off (0: no mask)        : 0.7,0.1    soft mask
Grey values to scale display (0: auto)    : 1,8        as before
```

NOTE:

You may have noticed that after applying the soft-circle, as well as the points becoming clearer, they also become larger. This demonstrates a very important image space / Fourier space relationship. A multiplication in image space (the application of a soft-circle is effectively a multiplication) leads to a convolution in Fourier space. This relationship occurs in both directions i.e. if you were to multiply the Fourier space image by a circular mask, you would get a convolution in image space (this is what filtering is), and similarly if you were to convolute in one space, you will get a multiplication in the other.

## 3.8. 2-D Images and Fourier Filters

1. As was done for the 1-D curves you can also use filters in Fourier space to remove unwanted information in 2-D images.

2. Create a new test-image showing a RECTANGLE and add some noise to it:

```
IMAGIC-COMMAND: create-im

Putput filename                  : my_rectangle
Image dimension                  : 256,256
...

IMAGIC-COMMAND: add-noise

Mode of operation                : ADD_NOISE
Input file                       : my_rectangle
Output file                      : my_rectangle_noise
Mwan, sigma of Gaussian noise    : 0,5
Random number seed               : 0            your choice
```





3. Apply low-pass filters to the images (my_rectangle_noise) with the command LOW-PASS-FILTER:

```
IMAGIC-COMMAND: low-pass

Mode of operation                : LOWPASS
Input file                       : my_rectangle_noise
Output file                      : my_rectangle_noise_lp
High frequency cut-off           : 0.2          your choice
```

Play with different values for "High frequency cut-off" and always DISPLAY both, the original image (my_rectangle_noise) and its low-pass filtered version (my_rectangle_noise_lp).

4. Also apply high-pass filters onto the images (my_rectangle_noise):

```
IMAGIC-COMMAND: high-pass

Mode of operation                : HIGHPASS
Input file                       : my_rectangle_noise
Output file                      : my_rectangle_noise_hp
Low frequency cut-off            : 0.2          your choice
Remaining transmission           : 0
```

Play with different values for the "Low frequency cut-off" and DISPLAY both, the original image (my_rectangle_noise) and its high-pass filtered version (my_rectangle_noise_hp).

NOTE:

(a) "Large" details are represented by low frequencies.

(b) "Small" details are represented by high frequencies

(c) Usually the very "large" details (density ramps, for example) and the very "small" details (mostly noise) are hiding the motif, which you are interested in.

(d) Fourier filters offers a nice possibility to remove this unwanted information: filter the Fourier transform close to the centre ("low frequencies") and at the borders ("high frequencies"). Such a filter is called a BAND-PASS FILTER.

5. Now apply band-pass filters to "real" images.

In the data directory **whgb_data** on the Brazil School network drive you can find an IMAGIC image file with five "worm hemoglobin" particles called test_images. Copy both files (test_images.hed and test_images.img) to your working directory.

You can actually use any test images you generated so far too. But at this stage of the hands-on it can be helpful to play around with "real science" images.

6. Apply the band-pass filter with command BAND-PASS-FILTER:

```
IMAGIC-COMMAND: band-pass

Mode of operation                    : BANDPASS
Input file                           : test_images
Output file                          : test_images_bp
Low frequency cut-off                : 0.2          your choice
Remaining transmission               : 0
High frequency cut-off               : 0.8          your choice
```

7. Open a second terminal window and DISPLAY the output (test_images_bp) to check the result. Do NOT exit DISPLAY but call option WATCHDOG.

8. Now play with different values of "Low frequency cut-off" and "High-frequency cut-off). Always use the same output file (test_images_bp). DISPLAY/WATCHDOG will automatically display the new images.

   Use "extreme" band-pass parameters so that only high frequencies (0.2,0,0.9, for example) or only low frequencies (0.05,0.005,0.1, for example) are retained.

9. Next, adjust the low and high frequency cut-offs to the particle size. Remember, a band-pass filter is combination of low- and a high-pass filter:





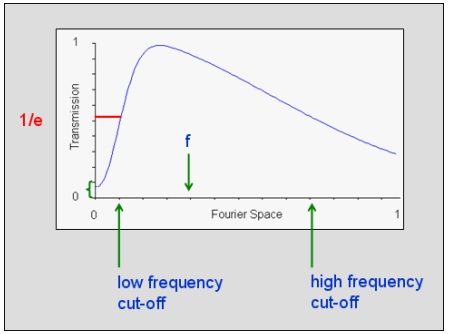



**Fig. 4:** Band-Pass Filter

Say that the image is scanned such that each pixel is of size

$$\text{pixel size}$$

The best resolution, which can (theoretically) be achieved for this sampling is given by the right edge of the Fourier transform image. It is the so-called *Nyquist* frequency

$$2\,x\,\text{pixel size}$$

Which corresponds to the maximum spatial frequency

$$\frac{1}{2\,x\,\text{pixel size}}$$

Remember that the centre of the transform is zero spatial frequency.

Any cut-off value asked by **IMAGIC** filtering commands is a fraction f between 0.0 and 1.0 and corresponds to a spatial frequency

$$\frac{\text{f}}{2\,x\,\text{pixel size}}$$

The low-frequency cut-off: to remove all those low frequencies, which contain information larger than the size of your particle. These could be density ramps or other low-frequency information coming from the background of the images. You can adapt the low-frequency cut-off ("large patterns") to the size of the particle

$$\frac{2\,x\,\text{pixel size}}{\text{particle size}}$$

High frequency cut-off: to remove high frequencies containing mostly noise and little signal, thus increasing the overall SNR (signal to noise ratio) of the

images. Adapt the high frequency cut-off ("small patterns, noise") to the expected resolution:

$$\frac{2\ x\ \text{pixel size}}{\text{expected resolution}}$$

The size of the test particle is 200 Angstrom and the pixel size is 4.4 Angstrom. Expecting a resolution of 15 Angstrom you get:

$$\text{LF cut-off} = \frac{2\ x\ \text{pixel size}}{\text{particle size}} = \frac{2 \cdot 4.4}{200} = 0.044$$

$$\text{HF cut-off} = \frac{2\ x\ \text{pixel size}}{\text{exp. resolution}} = \frac{2 \cdot 4.4}{14} = 0.63$$

So you can use:

LF cut - off: 0.05
HF cut - off: 0.75

The **IMAGIC** filter "cut-off" parameters are very gradual Gaussian drop-off values and do not correspond to sharp masks in Fourier space!

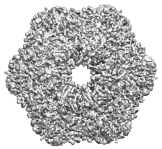
NOTE:

2 x pixel size is the Nyquist frequency which is the theoretical limit to the resolution that can be achieved.

YOUR NOTES:







## ERROR HINTS:

We tried to find and correct all errors and typos before, during and after the Brazil School. If you still find some mistakes please send your error hints to michael@ImageScience.de so that we can improve this tutorial. Thank you very much.

## CONTENT