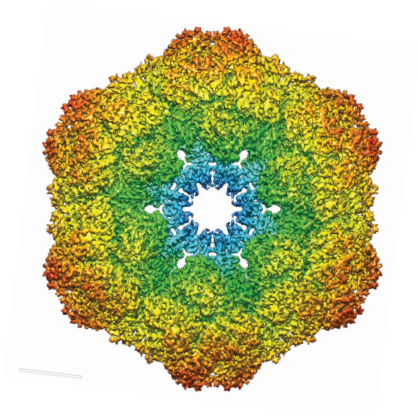


THE 8TH BRAZIL SCHOOL FOR SINGLE PARTICLE CRYO-EM

HANDS-ON Playing with IMAGIC GISP



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

© Michael Schatz (Image Science)

Purpose of this document is to play with IMAGIC within the GISP program and to learn what is averaging and what are Fourier transforms and filtering.

PART 1: [Remember: Some IMAGIC Basics](#)

PART 2: [Playing with IMAGIC](#)

PART 3: [The Fourier Transform](#)

[Content](#)

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"):

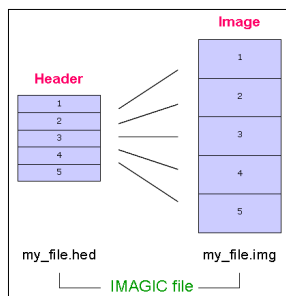


Fig. 1: IMAGIC file

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**:

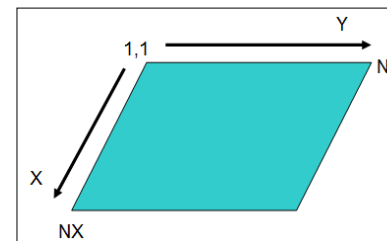


Fig. 2: IMAGIC 2-D coordinate system

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

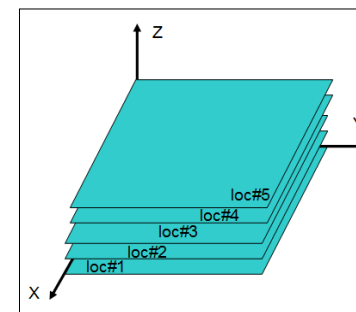


Fig. 3: IMAGIC 3-D coordinate system

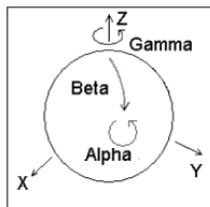
Note that $\mathbf{Z} = \mathbf{X} \times \mathbf{Y}$ as required for a right-handed co-ordinate system.

Brazil-School for Single Particles Cryo-EM: Hands On

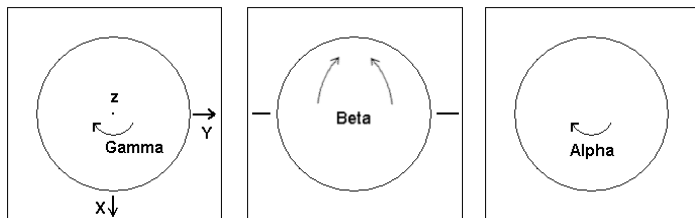
In **IMAGIC** 3-D orientations are defined by three Euler angles Alpha, Beta and Gamma.

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":



- Look at the particle along the Z-axis ("north pole")
- Rotate the particle clockwise by Gamma
- Rotate the particle into the plane clockwise by Beta
- 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.

Brazil-School for Single Particles Cryo-EM: Hands On

IMAGIC command questions will often have default values which appear in the text/value boxes.

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.

IMAGIC question always has an associated help, which can be accessed by clicking the "?" buttons.

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. Note that in the GISP single particles workflow the names of the output files are created automatically.

MPI refers to parallel processing. In the beginning always first run on a single core (non-parallel mode) which will give you more feedback (click **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.

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:

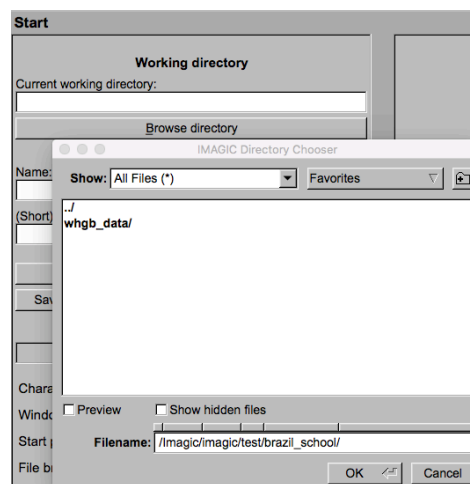
2. Playing with IMAGIC

2.1. Start GISP

Click the **GISP** icon to start the "GUI IMAGIC Single Particles" program.

Before doing any calculations, you have to define some project parameters on the "Start" page:

1. Specify your working directory. You can type the name into the text box or use the "Browse directory" button.

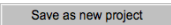


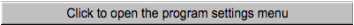
NOTE: You can store your directory in "Favorites".



2. Type in the name of your project and, if wanted, some information. Note that this is only for your information.

Project	
Name:	Brazil School Hands-On
(Short) Prefix to be used for all file names:	whgb

3. Also specify a SHORT prefix. Output file names on the **GISP** workflow pages will be created automatically using this prefix. When using "Commands" the output file names have to be specified.
4. Save the information given by clicking the  button.
5. You can also change some **GISP** program settings:



May be, your computer window/monitor is too small and you want to reduce the **GISP** window:

Window size: x

In this case you normally also have to adjust the font size:

Character/font size:

Save the settings and note that **GISP** will re-start.

6. Click the  button to start working.

YOUR NOTES:

2.2. GISP and the Single Particles Analysis Workflow

In **GISP** you can follow a suggested single particles analysis workflow. Check and, if wanted, change the parameters on each page and "Run" the suggested calculations. When finished click the "Next" button to continue with the next page.

2.3. GISP and Commands

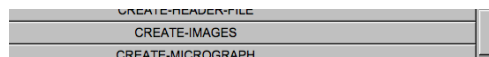
To get used to **IMAGIC** as well as to get used to images and to image analysis we do not yet start with the Single Particles Analysis Workflow but are playing with some IMAGIC commands.

Remember the meaning of some buttons in the toolbar (refer to "Hands-On: Introduction to GISP IMAGIC"):

	Open a list to run any IMAGIC command wanted
	Open a DISPLAY page to show IMAGIC image
	Open a PLOT page to show IMAGIC curves
	Open a MOVIE page (display a stack of images in an endless loop)

2.4. Command CREATE-IMAGES and DISPLAY

- To start, click the button to get the list of IMAGIC commands.
- First, run a command to create an image. The command name is **CREATE-IMAGES** (its synonym is **TEST-IMAGE-CREATION**). You can choose the command from the list and run it by clicking the related button.



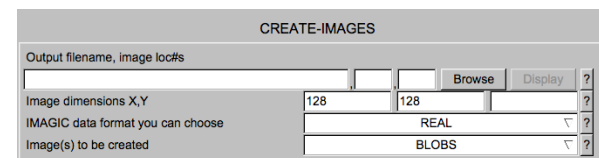
- If you know the command name you can also type in the command:



Note that the program tries to automatically fill in the full command name so that you do not have to write long command names.

Double click the button or hit the ENTER button of your computer to start the command.

- Remember that all **IMAGIC** commands first ask for all parameters and all file names needed before the actual calculations are started:



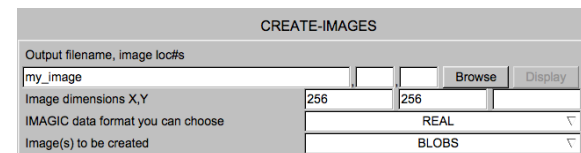
Note that most of the parameters are already given. These so-called "default" values are parameters suggest by the programmer. When calling this command the next time the values shown are the parameters which you had chosen the last time when running this command.

- Press all question marks and read the related help.

NOTE:

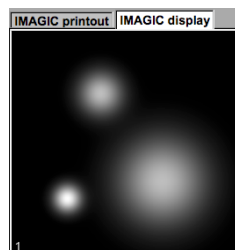
All **IMAGIC** pages provide context related detailed help. Move the cursor to any item wanted and wait until the help window pops up.

- Specify the output file name and create a slightly larger image than is suggested by the default values:



- Click the button to run the command. (No more mentioned in the following.)
- Read the program print-out on the right-hand side. (No more mentioned in the following.)
- Note that you can always "Zoom" and/or "Save" this print-out. Click the related buttons and see what happens. (No more mentioned in the following.)

10. Click the **Display** button or click the "IMAGIC display" tab to visualise the created image. (No more mentioned in the following.)



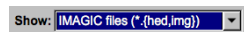
Check the image created ([my_image](#)). It contains one single image with three so-called blobs.

Refer to the "Hands-On: Introduction to GISP" and its chapter "DISPLAY". Play around with the **Contrast** and the **Zoom** button.

2.5. DISPLAY

Instead of using this simple display tab, you can also use the "Display" button on the toolbar.

This display offers a number of additional options to adjust the displayed images. Note that in the "IMAGIC file chooser" you must set the filter to "IMAGIC files" to be able to browse the created file:



Refer to the DISPLAY chapter in ""Hands-On: Introduction to GISP IMAGIC""

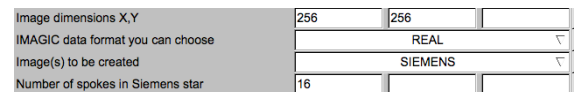
2.6. Command CREATE-IMAGE and Stacks of Images

1. **IMAGIC** can hold stacks of images within one **IMAGIC** image file. So, create a second image with command **CREATE-IMAGES** and store this image in the same file ([my_image](#)). Note that you now have to give location numbers (a **2** here, of course)



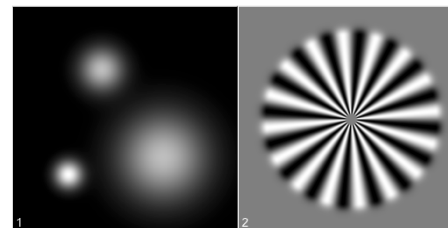
Note that image size and image type MUST be the same.

You are allowed, of course, to create another motif:



Run the command and check the print-out.

2. Check the displayed output images.



3. You can play with this **CREATE-IMAGE** command and create other **REAL** images and stacks of images, but of different sizes, and using different options. Always have a look at the created images.
4. When finished go back to the commands listing by clicking **New command** (No more mentioned in the following.)

2.7. Play around with some other commands

1. Call **CREATE-IMAGES** again. Create a new image with a **SIEMENS** star.

CREATE-IMAGES			
Output filename, image loc#s			
my_img			Browse Display ?
Image dimensions X,Y	256	256	?
IMAGIC data format you can choose	REAL ?		
Image(s) to be created	SIEMENS ?		
Number of spokes in Siemens star	16		?

2. Click the **New command** button and call the new command **COARSEN-IMAGES**:

COARSEN-IMAGES
COARSEN-IMAGES
COARSEN-SAMPLING

3. Resize the created image (**my_img**). Type in the output file name (**my_img_coarse2**) and first choose a summing (binning) factor of **2**.

COARSEN-IMAGES			
Input file, image loc#s			
my_img			Browse Display ?
Output file, image loc#s			
my_img_coarse2			Browse Display ?
Summing parameter (2,3,4,etc.)	2		?

4. Have a look at the input image (**my_img**) as well as at the resized image (**my_img_coarse2**) using the **Display** buttons.
5. Now also use summing factors of **4** (or **6**) to coarsen the images. Check the displayed images and see the effects of sampling size on your resolution. What is the size of your image now? What detail can you still see?
6. Use the **BLOW-UP-IMAGE** command to blow up the most resized image (**my_img_coarse6** or the like) back to the original size:

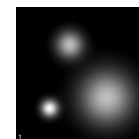
BLOW-UP-IMAGE			
Mode of operation BLOWUP			
Input file, image loc#s			
my_img_coarse6			Browse Display ?
Output file, image loc#s			
my_img_blowup			Browse Display ?
Size of output image(s) (X,Y)	256	256	?

7. To better compare the originally created file (**my_img**) with this blown-up image (**my_img_blowup**) we append the later one to the original image (**my_img**). Call command **APPEND-IMAGE-FILE**:

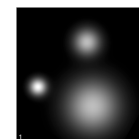
APPEND-IMAGE-FILE			
Input file to be appended to output file			
my_img_blowup			Browse Display ?
Output file, NO loc#s			
my_img	2	2	Browse Display ?

8. In the "IMAGIC display" tab compare the originally created Siemens star (first image: **my_img,1**) with the resized & blown-up one (second image: **my_img,2**).
9. Use the command **CREATE-IMAGE** to create a new test image using the option **BLOBS**.
10. Use **MOVE-IMAGE** to **ROTATE**, **SHIFT** and **COMBINE** (= **ROTATE** plus **SHIFT**) the image. Have a look at the images.
11. Use these shifted and rotated images to remember the **IMAGIC** coordinate system (chapter 1).

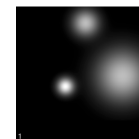
Original image:



Rotation by 30°:



Shift by -40,30:



2.8. 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**.
2. Remember: To create a file with multiple images you specify the start and final location numbers, like **my_images,1,256** where **my_images** is the file name, **1** is the start location and **256** is the final location.

CREATE-IMAGES			
Output filename, image loc#s			
my_images	1	256	<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
Image dimensions X,Y	128	128	<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
IMAGIC data format you can choose	REAL <input type="button" value="Browse"/> ?		
Image(s) to be created	CHECKERS <input type="button" value="Browse"/> ?		
Checker size	16		<input type="button" value="Browse"/> <input type="button" value="Display"/> ?

3. Use command **ADD-NOISE** to add noise to these images:

ADD-NOISE			
Mode of operation <input type="button" value="ADD_NOISE"/>			
Input file, image loc#s			
my_images			<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
Output file, image loc#s			
my_images_noise			<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
Mean, sigma of Gaussian noise	0.0	10.0	<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
Random number seed (0:random)	0		<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
The program can run in MPI parallel mode, but for the chosen option parallel processing usually is NOT very helpful. But it is your choice, of course			
Use MPI parallelisation	<input type="radio"/> Yes <input checked="" type="radio"/> No <input type="button" value="Display"/> ?		

4. In the "IMAGIC display" tab check the input (**my_images**) and the noisy images (**my_images_noise**). Use "Global" contrast.

2.9. Noise Reduction by Image Averaging

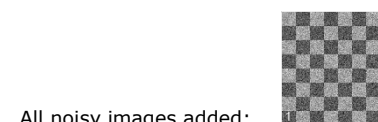
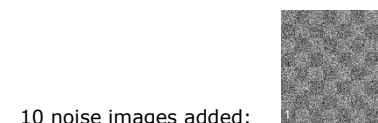
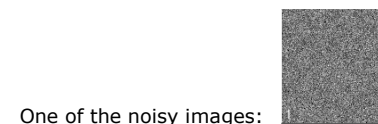
You will get some impression what image averaging means and why image processing can enhance the resolution of noisy images.

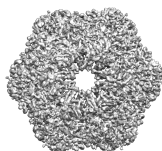
1. Use **SUM-IMAGES** with option **SOME_SUM** to make sums of **2, 8, 64**, and **256** of the images from the file you added noise to.

Input file is **my_image_noise**. Suggested output file names are: **my_sum2**, **my_sum8**... and **my_sum256**. In "Location number(s) wanted:" you should specify **1-2**, or **1-8**, or **1-64**, or **1-256** accordingly:

SUM-IMAGES			
Mode of summing <input type="button" value="SOME_SUM"/> <input type="button" value="Browse"/> <input type="button" value="Display"/> ?			
Input file, NO loc#s			
my_images_noise			<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
Output file, ONE loc#			
my_sum2			<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
Variance file, ONE loc#			
NONE			<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
Location number(s) wanted			
1-2			<input type="button" value="Browse"/> <input type="button" value="Display"/> ?
Numbers wanted			
ALL			<input type="button" value="Browse"/> <input type="button" value="Display"/> ?

2. Have a look at each result and see the effects of image averaging on the signal to noise ratio.





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

- Go back to the commands listing and create a curve (a 1-D image) using the command **CREATE-CURVE**. Create an **IMAGE** with amplitude **1** and wave length **0.1**:

CREATE-CURVE			
Mode of output	BOTH		
Output file, curve loc#s	my_curve		
Output plot filename (NO loc#s)	my_curve.plt		
Length of curve	512		
Choose curve option	SINE		
Amplitude of the curve	1.0		
Wave length of periodic signal	0.1		

- Create additional curves into the same file, using sequential locations. Always choose the same curve length (512) but with **Curve options**, **Amplitudes of the curve** and other curve parameters:

CREATE-CURVE			
Mode of output	IMAGE		
Output file, curve loc#s	my_curve		
Length of curve	512		
Choose curve option	SINC		
Amplitude of the curve			
Width of signal			

Values: your choice

- To visualise the curves, click the **Plot** button.
- 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 scaling**: For example:

Vertical scaling	
-10.00	10.00

- If you want to display all curves at the same time use option **Multiple plots**
- To exit the plot program, click the **Exit Plot** button.

3.2. Curves and their Fourier Transforms

Now create various curves and check how their Fourier transforms look like.

The usual workflow consists of the following three steps ((1) to (3)):

- Use command **CREATE-CURVE** to create a curve (**my_curve**).
- After the curve was created use the command **CURVE-FORWARD-FT** to calculate the related Fourier transform (**my_curve_ft**).

CURVE-FORWARD-FT			
Input file, curve loc#s	my_curve		
Output file, curve loc#s	my_curve_ft		
Options you can choose	FORWARD_FT		

- Have a look at the curve (**my_curve**) and its Fourier transform (**my_curve_ft**).

Maybe, you want to set **Multiple plots** back to 1.

You can also use option **Vertical scaling** to use the same vertical scaling for all curves.

- Use as many **Curve options** to create curves (**my_curve**) according to (1) and calculate the Fourier transforms (**my_curve_ft**) according to (2). Check the curves (according to (3)).
- Create various **SINE** curves using different amplitudes **2**, **4**, **6**, and **8**. Always use the wave length **0.1** and calculate the Fourier transforms.

- Have a look at the Fourier transforms.

Look at the vertical scaling of the curves and notice how the amplitudes relate to the height of the peaks in the Fourier transforms.

- Continue using **SINE** curves now with fixed amplitudes but changing the periodicity. Use amplitude **1** and wave length of periodic signal **0.1**, **0.25** etc.

Calculate the Fourier transforms.

Have a look at the curves and 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.

3.3. Relationship between Image Space and Fourier Space

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

Mode of output	IMAGE			?
Output file, curve loc#s				
my_curve			Browse	Display ?
Length of curve	512			?
Choose curve option	SINE			?
Amplitude of the curve				?
Wave length of periodic signal				?

Parameters: your choice

...

CREATE-CURVE				
Mode of output	IMAGE			?
Output file, curve loc#s				
my_curve	20	20	Browse	Display ?
Length of curve	512			?
Choose curve option	SINE			?
Amplitude of the curve				?
Wave length of periodic signal				?

Parameters: your choice

- As usual, calculate the Fourier transforms of the new curves with **CURVE-FORWARD-FT** and store them in file **my_curve_ft**.

- Now, sum all curves (**my_curve**) with command **SUM-CURVES**:

SUM-CURVES				
Mode of summing	TOTAL_SUM			?
Input file, NO loc#s				
my_images_noise			Browse	Display ?
Output file, ONE loc#				
my_curve_sum			Browse	Display ?
Variance file, ONE loc#				
NONE			Browse	Display ?

- Have a look at 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.

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

- Have a look at the result (**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.

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

CURVE-REVERSE-FT				
Input file, curve loc#s				
my_curve_sum_ft			Browse	Display ?
Output file, curve loc#s				
my_curve_sum_reverse_ft			Browse	Display ?
Options you can choose	REVERSE_FT			

8. Have a look at the result transform (`my_curve_sum_reverse_ft`).

Notice 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:

3.4. Fourier Space and Filtering

- 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`.
- 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").

- 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
- 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.
- 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`.
- Have a look at the sum (`my_curve_ramp`). Notice how the signal (block-wave) is disturbed by the ramp.
- 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`.

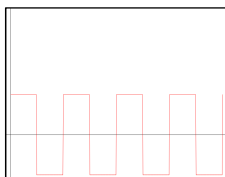
CURVE-FILTER			
Input file, curve loc#s			
my_curve_ramp		Browse	Display ?
Output file, curve loc#s			
my_curve_ramp_hp		Browse	Display ?
Options you can choose		HIGH_PASS	
Low-frequency cut-off	0.01		?
Remaining LF transmission	0.0		?

8. Have a look at the high-pass filtered curve ([my_curve_ramp_hp](#)).

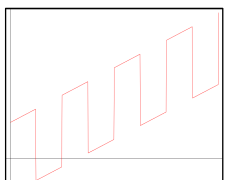
NOTE:

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.

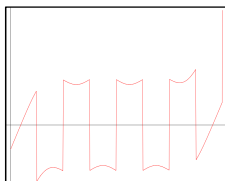
Original curve:



Ramp added:



High-pass filtered



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**. Note that the curves to be added have to specified like this: **1;3**

SUM-CURVES			
Mode of summing		SOME_SUM	
Input file, NO loc#s			
my_curve		Browse	Display ?
Output file, ONE loc#			
my_curve_noise		Browse	Display ?
Variance file, ONE loc#			
NONE		Browse	Display ?
Location number(s) wanted		1;3	

Have a look at the sum ([my_curve_noise](#)) and notice that the signal is disturbed by noise.

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

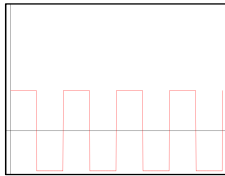
CURVE-FILTER			
Input file, curve loc#s			
my_curve_noise		Browse	Display ?
Output file, curve loc#s			
my_curve_noise_lp		Browse	Display ?
Options you can choose		LOW_PASS	
High-frequency cut-off	0.1		?

12. Have a look at the resulting low-pass filtered curve ([my_curve_noise_lp](#)).

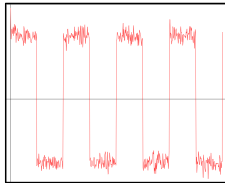
NOTE:

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.

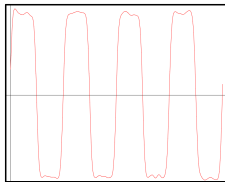
Original curve:



Noise added:



Low-pass filtered

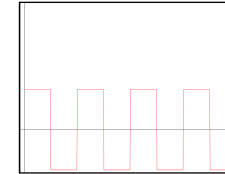


13. Finally, create a curve with disturbing low (ramp) and high frequencies (noise). Add all three curves (`my_curve`) with `CURVE-SUM`, using the option `TOTAL_SUM` (to get `my_curve_ramp_noise`).
14. `PLOT` the curve (`curve_ramp_noise`) to see how the curve is disturbed by the ramp and by Gaussian noise.
15. 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:

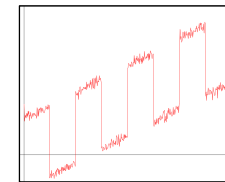
CURVE-FILTER			
Input file, curve loc#s			
my_curve_ramp_noise		Browse	Display ?
Output file, curve loc#s			
my_curve_ramp_noise_bp		Browse	Display ?
Options you can choose		BAND_PASS	
Low-frequency cut-off	0.01		?
Remaining LF transmission	0.0		?
High-frequency cut-off	0.1		?

16. As usual, visualise the resulting filtered curve (`my_curve_ramp_noise_bp`) and see how a band-pass can remove unwanted information.

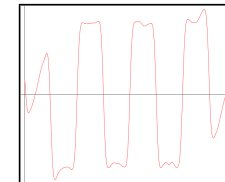
Original curve:



Ramp and Noise added:



Band-pass filtered



YOUR NOTES:

3.5. 2-D Images and Fourier Transforms - First Steps

Now you are going to play with 2-D images and their Fourier transforms.

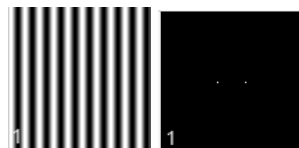
1. Start by using **CREATE-IMAGE** to create a **SINE**-wave image (**my_image**). Use the option **HORIZONTAL**. When asked for the wave length choose **0.1**:

CREATE-IMAGES			
Output filename, image loc#s			
my_image		Browse	Display ?
Image dimensions X,Y	512	512	?
IMAGIC data format you can choose	REAL		
Image(s) to be created	SINE		
Wave length of periodic signal	0.1		
Direction of the wave	HORIZONTAL		

2. Calculate the 2-D Fourier transform with command **FOURIER-TRANSFORM-IMAGE**. Output will be (**my_image_ft**). Use the option **FT2D_FORWARD**.

FOURIER-TRANSFORM-IMAGE			
Mode of operation	FOURIER		
Mode of Fourier operation	FT2D_FORWARD		
Input file, image loc#s			
my_image		Browse	Display ?
Output file, image loc#s			
my_image_ft		Browse	Display ?
Use MPI parallelisation	<input type="radio"/> Yes <input checked="" type="radio"/> No		

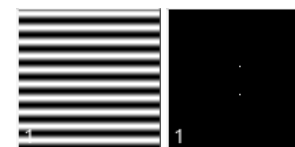
3. When finished have a look at the images using the **Display** button. Adjust the contrast (Interactive: -0.3,0.8 or the like).



Horizontal sine wave image and its FT:

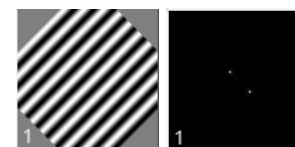
4. You should now see two "points" in the Fourier transform of the input image. This is the Fourier space representation of a sine wave, with the periodicity you have specified (**0.1** was suggested). In Fourier space, high frequency information (i.e. corresponding to short wave lengths in real space) 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.

5. Since you are examining 2-D images, note the sine waves also have a direction. In this case the sine wave travels horizontally.
6. To demonstrate this effect, use **CREATE-IMAGE** and create the same image again but now using option **VERTICAL**.
7. As before calculate the Fourier transform with **FOURIER-TRANSFORM-IMAGE**.



Vertical sine wave image and its FT:

8. Again, have a look at the images and notice how the direction has changed in both, the image and the related Fourier transform.
9. Once more, use **CREATE-IMAGE** and create the same image again but now using the direction option **ANGLE**. Use **45** (degrees).
10. As before calculate the Fourier transform **FOURIER-TRANSFORM-IMAGE**.
11. Again, have a look at the images and notice that the direction of the sine wave as well as of the frequency space points has rotated by 45 degrees.



Rotated sine wave image and its FT:

12. 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.
13. Play around with other images: use the commands **CREATE-IMAGE** to create 2-D images with various motifs (**my_image**), which you may or may not sum with the command **SUM-IMAGES** (**my_image_sum**) and finally Fourier transform with the command **FOURIER-TRANSFORM-IMAGE** (option **FT2D_FORWARD**). Learn how the related Fourier transforms look like.

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.

3.6. 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. To visualise the effect of masking first create a new **SINE** image with command **CREATE-IMAGE**:

CREATE-IMAGES			
Output filename, image loc#s			
my_image3			
Image dimensions X,Y	512	512	
IMAGIC data format you can choose	REAL		
Image(s) to be created	SINE		
Wave length of periodic signal	0.1		
Direction of the wave	HORIZONTAL		

3. Rotate the **SINE** image (**my_image3**) by 30 degrees with command **ROTATE-IMAGE**:

ROTATE-IMAGE			
Mode of operation			
ALL_ROTATE			
Mode of rotation			
ROTATE			
Also blow-up images			
<input type="radio"/> Yes <input checked="" type="radio"/> No			
Where to get the move parameters			
INTERACTIVE			
Input file, image loc#s			
my_image3			
Output file, image loc#s			
my_image3_rot			
Angle in degrees (+ve = clockwise)	30		
Mode of operation			
BILINEAR			

4. Use command **FOURIER-TRANSFORM-IMAGE** to calculate the Fourier transform of the rotated **SINE** image (**my_image3_rot**):

FOURIER-TRANSFORM-IMAGE			
Mode of operation			
FOURIER			
Mode of Fourier operation			
FT2D_FORWARD			
Input file, image loc#s			
my_image3_rot			
Output file, image loc#s			
my_image3_rot_ft			
Use MPI parallelisation			
<input type="radio"/> Yes <input checked="" type="radio"/> No			

Look at the created Fourier transforms (**my_image3_rot_ft**). Adjust the contrast (Interactive: -0.3,0.8 or the like).

5. Subsequently run command **MASK-IMAGE** to mask out the centre of the image (**my_image3_rot**) with a soft drop off circular mask

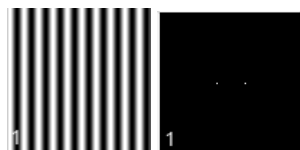
MASK-IMAGE			
Mode of operation			
MASK_IMAGE			
Mode of mask			
SOFT_CIRCULAR			
Input file, image loc#s			
my_image3_rot	1	7	
Output file, image loc#s			
my_image3_rot_masked			
Mask centre (0.0: image centre)			
0.0 0.0			
The image will be masked by a circle. Please specify the mask radius (pixels or fraction of inner radius). If you specify a drop-off it will be a soft mask.			
Mask radius, drop-off (0: no mask)			
0.7 0.1			
The program can run in MPI parallel mode, but for the chosen option parallel processing usually is NOT very helpful. But it is your choice, of course			
Use MPI parallelisation			
<input type="radio"/> Yes <input checked="" type="radio"/> No			

before calculating the Fourier transform with **FOURIER-TRANSFORM-IMAGE**:

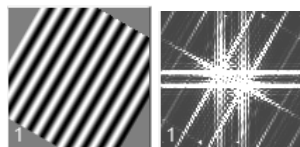
FOURIER-TRANSFORM-IMAGE			
Mode of operation			
FOURIER			
Mode of Fourier operation			
FT2D_FORWARD			
Input file, image loc#s			
my_image3_rot_masked			
Output file, image loc#s			
my_image3_rot_masked_ft			
Use MPI parallelisation			
<input type="radio"/> Yes <input checked="" type="radio"/> No			

Have a look at the created images.

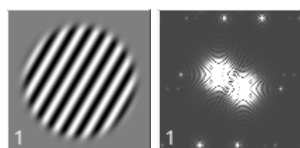
Horizontal sine wave image and its FT:



Rotated sine wave image and its FT:



Masked rotated sine wave image and its FT:

**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.

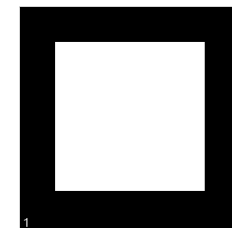
3.7. 2-D Images and Fourier Filters

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 (**CREATE-IMAGE**) showing a **RECTANGLE**

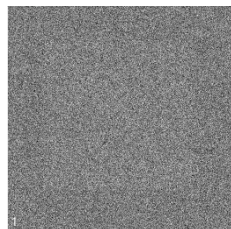
CREATE-IMAGES			
Output filename, image loc#s			
my_rectangle		Browse	Display ?
Image dimensions X,Y	256	256	?
IMAGIC data format you can choose	REAL		
Image(s) to be created	RECTANGLE		
Upper left (X,Y) coordinates	41	41	?
Upper left (X,Y) coordinates	210	210	?
Inside, outside densities	1.0	0.0	?

and add some noise to it (**ADD-NOISE**):

ADD-NOISE			
Mode of operation			
ADD_NOISE			
Input file, image loc#s			
my_rectangle		Browse	Display ?
Output file, image loc#s			
my_rectangle_noise		Browse	Display ?
Mean, sigma of Gaussian noise	0.0	5	?
Random number seed (0:random)	0		?
The program can run in MPI parallel mode, but for the chosen option parallel processing usually is NOT very helpful. But it is your choice, of course			
Use MPI parallelisation	<input type="radio"/> Yes	<input checked="" type="radio"/> No	?



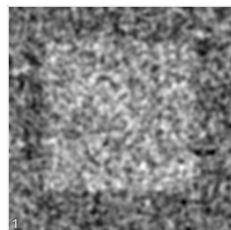
The image with the rectangle



The image with the noisy rectangle

3. Apply low-pass filters to the images ([my_rectangle_noise](#)) with the command **LOW-PASS-FILTER-IMAGE**:

LOW-PASS-FILTER-IMAGE			
Mode of operation	LOWPASS		
Input file, image loc#s	my_rectangle_noise 1 500 Browse Display ?		
Output file, image loc#s	my_rectangle_noise_lp Browse Display ?		
The image will be filtered. Please specify:			
High frequency cut	0.2		?
Use MPI parallelisation	<input type="radio"/> Yes <input checked="" type="radio"/> No ?		

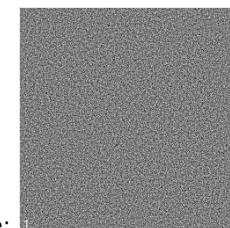


The low-pass filtered noisy rectangle image:

4. Play with different values for "High frequency cut-off" and always have a look at both, the original image ([my_rectangle_noise](#)) and its low-pass filtered version ([my_rectangle_noise_lp](#)).

5. Also apply high-pass filters onto the images ([my_rectangle_noise](#)). Use command **HIGH-PASS-FILTER-IMAGE**:

HIGH-PASS-FILTER-IMAGE			
Mode of operation	HIGHPASS		
Input file, image loc#s	my_rectangle_noise Browse Display ?		
Output file, image loc#s	my_rectangle_noise_hp Browse Display ?		
The image will be filtered. Please specify:			
Low frequency cut	0.2		?
Remaining transm. below cut off	0		?
Use MPI parallelisation	<input type="radio"/> Yes <input checked="" type="radio"/> No ?		



The high-pass filtered noisy rectangle image:

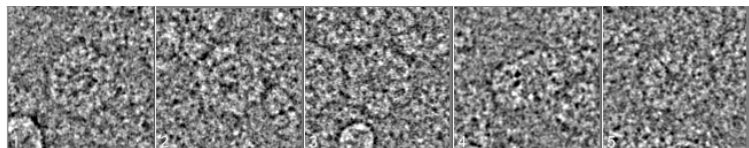
6. Play with different values for the "Low frequency cut-off" and have a look at both, the original image ([my_rectangle_noise](#)) and its high-pass filtered version ([my_rectangle_noise_hp](#)).

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.

7. 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.



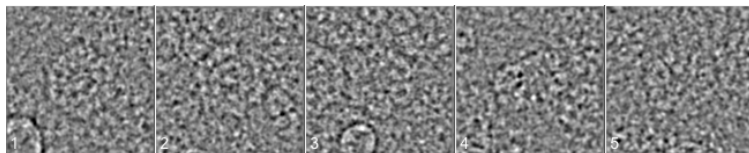
The test images

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.

8. Apply a band-pass filter with command **BAND-PASS-FILTER**

BAND-PASS-FILTER-IMAGE			
Mode of operation	BANDPASS		
Input file, image loc#s	test_images		
Output file, image loc#s	test_images_bp		
The image will be filtered. Please specify:			
Low frequency cut (0:low-pass)	0.2		?
Remaining low-freq. transmission	0.001		?
High frequency cut (0: high-pass)	0.8		?
Also ASQ filter the images	<input type="radio"/> Yes	<input checked="" type="radio"/> No	?
Use MPI parallelisation	<input type="radio"/> Yes	<input checked="" type="radio"/> No	?

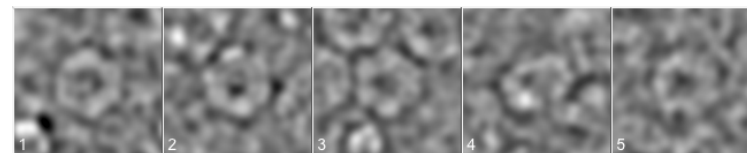
And have a look at both, the original ([test_images](#)) and the band-pass filtered images ([test_images_bp](#)).



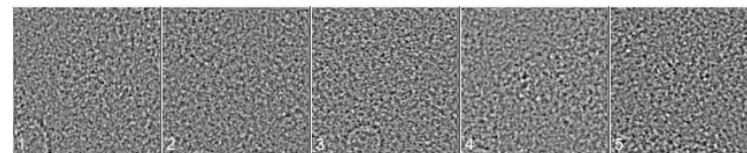
The test images – band-pass filtered

9. Now play with different values of “Low frequency cut-off” and “High-frequency cut-off). Have a look at the filtered images.

Use “extreme” band-pass parameters so that only high frequencies ([0.2,0.9](#), for example) or only low frequencies ([0.05,0.005,0.1](#), for example) are retained.



Extremely low-pass filtered test images



The extremely high-pass filtered test images

YOUR NOTES:

Brazil-School for Single Particles Cryo-EM: Hands On

10. 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:

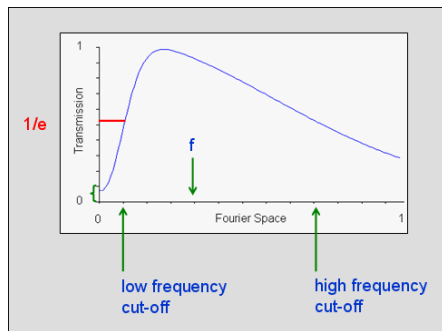


Fig. 4: Band-Pass Filter

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

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

$2 \times \text{pixel size}$

Which corresponds to the maximum spatial frequency

$$\frac{1}{2 \times \text{pixel size}}$$

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

$$\frac{f}{2 \times \text{pixel size}}$$

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

$$\frac{2 \times \text{pixel size}}{\text{particle size}}$$

Brazil-School for Single Particles Cryo-EM: Hands On

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:

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

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:

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

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

So you can use:

LF cut-off: 0.05

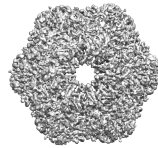
HF cut-off: 0.75

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

NOTE:

$2 \times \text{pixel size}$ 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

1.	REMEMBER: SOME IMAGIC BASICS	3
2.	PLAYING WITH IMAGIC	7
2.1.	START GISP	7
2.2.	GISP AND THE SINGLE PARTICLES ANALYSIS WORKFLOW	9
2.3.	GISP AND COMMANDS	9
2.4.	COMMAND CREATE-IMAGES AND DISPLAY	9
2.5.	DISPLAY	11
2.6.	COMMAND CREATE-IMAGE AND STACKS OF IMAGES	12
2.7.	PLAY AROUND WITH SOME OTHER COMMANDS	13
2.8.	NOISE	15
2.9.	NOISE REDUCTION BY IMAGE AVERAGING	16
3.	THE FOURIER TRANSFORM	17
3.1.	TEST CURVES	17
3.2.	CURVES AND THEIR FOURIER TRANSFORMS	18
3.3.	RELATIONSHIP BETWEEN IMAGE SPACE AND FOURIER SPACE	19
3.4.	FOURIER SPACE AND FILTERING	22
3.5.	2-D IMAGES AND FOURIER TRANSFORMS - FIRST STEPS	27
3.6.	2-D IMAGES AND FOURIER TRANSFORMS - MASKS	29
3.7.	2-D IMAGES AND FOURIER FILTERS	32
	CONTENT	40