Numerical Methods in TEM

Christoph T. Koch

Max Planck Institut für Metallforschung

http://hrem.mpi-stuttgart.mpg.de/koch/Vorlesung

DigitalMicrograph: What can it do … ?

- Function as a pocket calculator (process 0-dimensional variables)
- Operate on vectors (1-dimensional variables)
- Operate on images (2-dimensional variables)
- Operate on data cubes (3-dimensional variables)
- DM menus can be extended to include custom functions (e.g. your own scripts or compiled plug-ins)
- DM can record data from the microscope (controls CCD camera)
- DM can control many functions of the microscope (if implemented by TEM manufacturer)

DigitalMicrograph: What can't it do … ?

- Does not record macros (i.e. it does not automatically generate a script from command that you enter)

  - But: all the menu items can be accessed from scripts

- DM-scripts become slow, when excessive use of loops is being made (interpreted language: code is being compiled "on the fly")

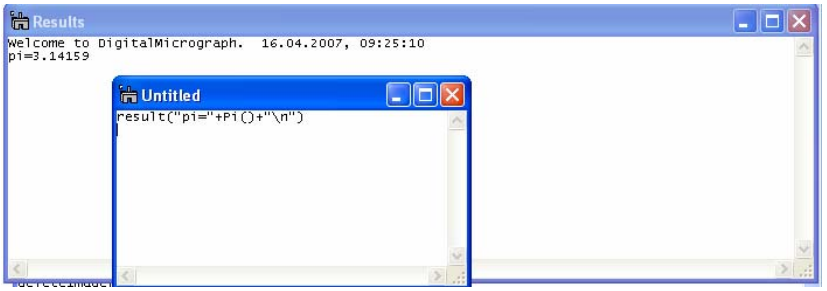
  - But: compiled code may be included as PLUG-IN

Outline of this first lecture

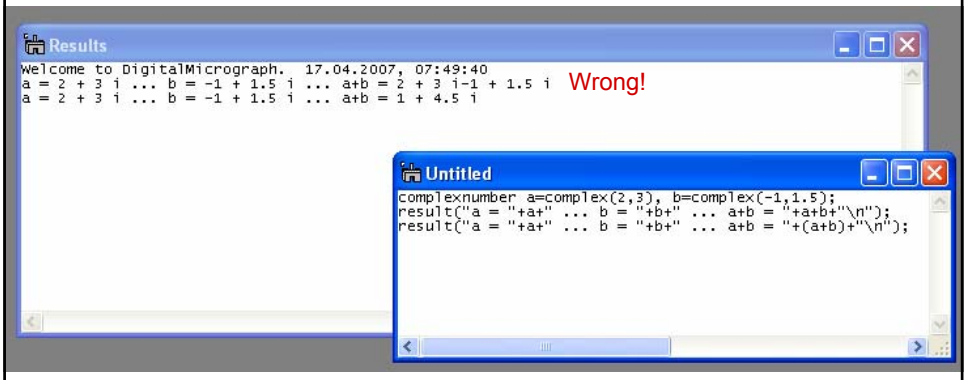
- Very quick introduction to writing scripts in DM (mostly complimentary to B. Schaffers DM scripting Intro at http://www.felmi-zfe.tugraz.at/dm_scripts/dm_scripts/AddInfo/DM-course-09-06-Wien/DM-basic-scripting_bs.pdf)
- FFTs of arbitrary arrays
- Spectra (1-dimensional arrays)
- Processing of wedge-shaped spectrum profiles

Displaying Text

- result("pi="+Pi()+"\n")



Work with complex numbers



Don't forget your parentheses!!!

Built-in operations on complex numbers

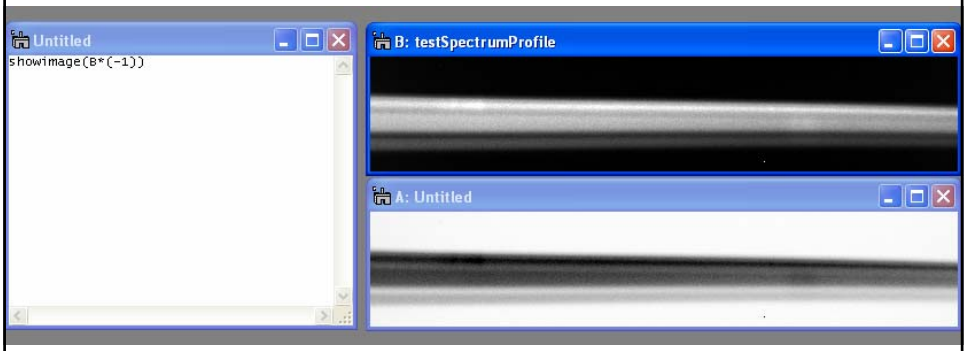
- abs
- cis
- complex
- conjugate
- cos
- cosh
- exp
- imaginary
- log
- modulus
- norm
- Phase
- Polar
- real
- Rect
- sin
- sin
- sqrt
- tan
- tanh

A selection of real-number functions

- sin, asin, sinh
- cos, acos, cosh
- tan, tanh, atan, atan2, atanh
- exp, exp2, exp10
- log, log2, log10
- exp1(x) = exp(x)-1
- log1(x) = log(x+1)
- AiryAi, AiryBi
- BesselI, (also J, K, Y)
- SphericalBesselJ (also Y)
- Beta
- erf, erfc
- Factorial
- Gamma, GammaP, GammaQ
- LegendrePolynomial
- PoissonRandom (also Binomial, Gaussian, Gamma, Uniform)
- BinomialCoefficient
- …

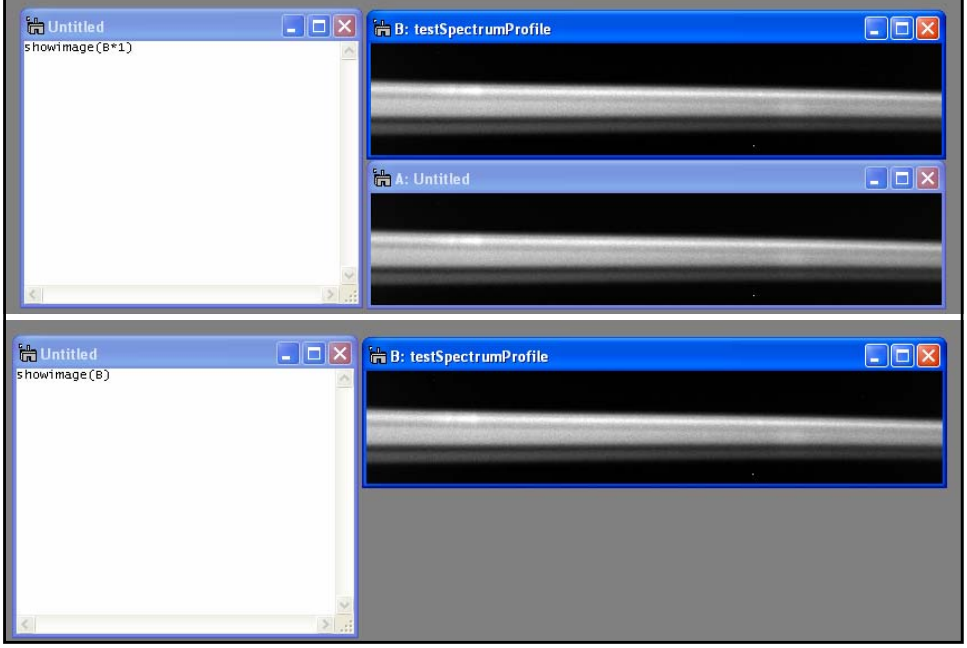
On the fly data manipulation

Makes sense if script is only used once !



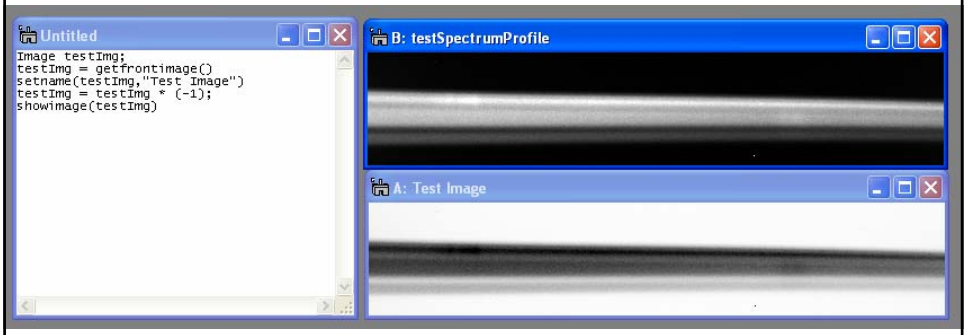
- Images may be addressed in a script by the letter assigned to the window they are shown in.
- If a new image is generated (e.g. by performing some operation on the image), this image will receive a new letter (image variable)

Automatic generation of new images



Make script independent of image name

This makes scripts more general !



Img = GetFrontImage(): obtain the image which is currently active.

Note: if no name is assigned to an image variable, the name of the image remains “Untitled”. It helps to assign names to image windows using SetName

DM Built-in commands

Script Reference 2.5

See list of script functions in alphabetical order.

Images
Image Processing
Image Data Types
Real Images
Complex Images
RGB Images
Image Management
Image Display
Image Scrap

Numbers and Strings
Real Number Functions
Complex Number Functions
RGB Number Functions
Number Conversion
Strings

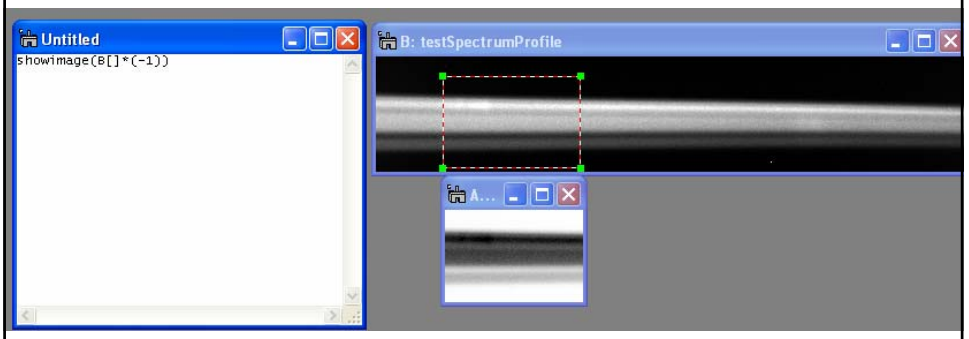
Annotations, Selections, Tags, I/O
Annotations
Selections
Tags (aka Notes)
Dialogs
Input/Output

Other
Miscellaneous

This list of commands used to be available online but has been discontinued by Gatan.

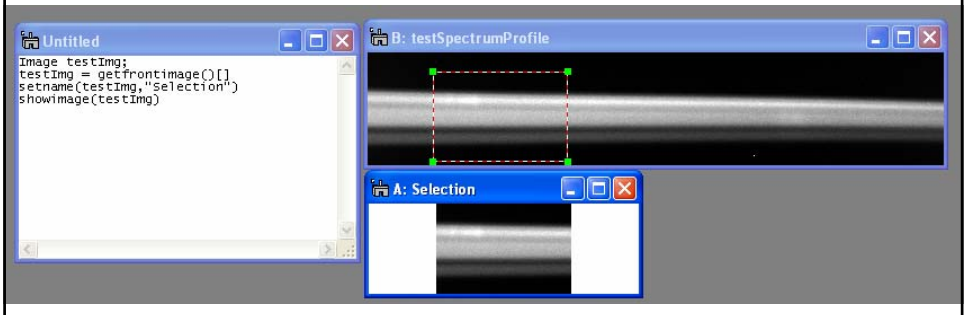
I have a local copy of this website and can make it available to whoever wants it. …

Addressing user-marked regions of interest (ROI)



On the fly: simply use square brackets behind the image letter

Addressing ROIs in re-usable scripts



One may append square brackets also to “GetFrontImage()”

Linear Transformations

- Single variable x → y:

$$y = a \cdot x + b$$

- Vector variable x=[x_1,x_2,…x_n] → y=[y_1,y_2,…y_n]:

$$\vec{y} = A \cdot \vec{x} + \vec{b}$$

$$y_i = \sum_{j=1}^{n} A_{i,j} \cdot x_j + b_i$$

Example: Discrete Fourier Transform

Continuous: $$F(k) = \frac{1}{\sqrt{2\pi}} \int_0^a f(x) \cdot \exp\left[-2\pi i \cdot x \cdot k\right] \cdot dx$$

Discrete grid: $$F(k_i) = \frac{1}{\sqrt{2\pi}} \cdot \sum_{j=1}^{n} f(x_j) \cdot \exp\left[-2\pi i \cdot x_j \cdot k_i\right]$$

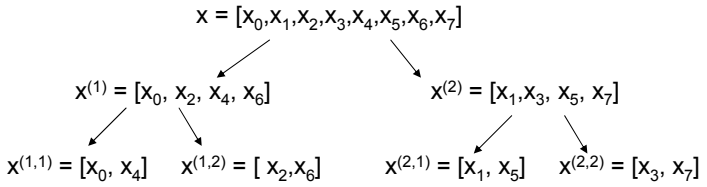
Matrix formulation: $$A_{k,l} = \frac{1}{\sqrt{2\pi}} \exp\left[-2\pi i \cdot l \cdot \frac{k}{n}\right], \qquad \vec{b} = 0$$

$$y_k = \sum_{l=0}^{n-1} A_{k,l} \cdot x_l + b_k$$

Fast Fourier Transform (FFT)

- 1805: First use by C.F. Gauss
- 1965: Official publication by Cooley and Tukey

Radix 2 – Algorithm as implemented by DM as fft (and ifft):
(use ALT-key to make selections of size 2^m)

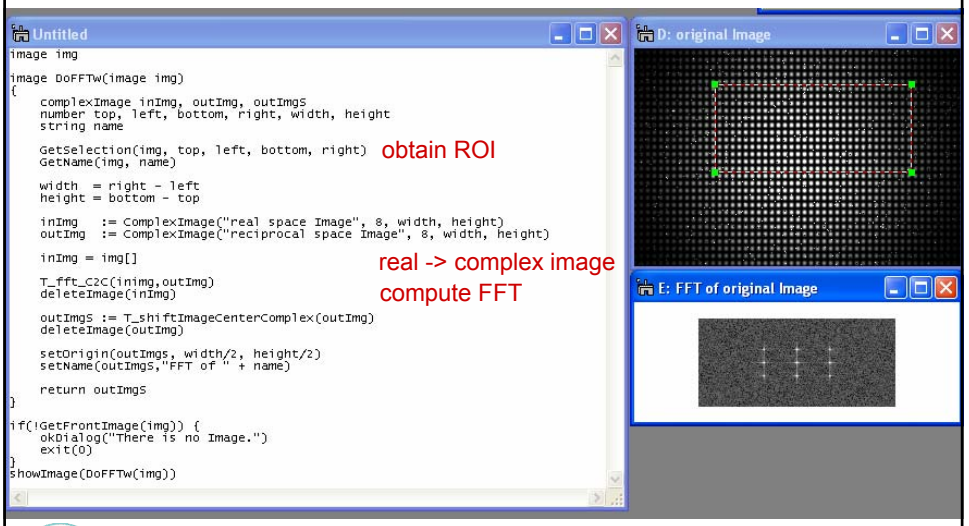


The matrix multiplication y=Ax requires n·n multiplications
The FFT splits this big multiplication into several smaller ones, so that only $n \cdot \log_2(n)$ multiplications are necessary.

FFTW Fastest Fourier Transform in the West

- FFTW is a software library that also allows every 3rd, 4th, fifth, etc. element to be used. It can therefore handle any size of array.
- Implemented for DM in plugin Transforms.dll (http://hrem.mpi-stuttgart.mpg.de/koch/DM-Plugin/index.html)
- Implemented functions (do not allocate memory for new image):
  - T_fft_c2c(compl_ImgIn,compl_ImgOut)
  - T_ifft_c2c(compl_ImgIn,compl_ImgOut)
  - compl_Img =T_shiftImageCenterComplex(compl_Img) (shifts k=(0,0) of FFT to center of image for complex images – DM function does not work)
  - Additional functions for computing different correlations

Usage of FFTW within DM



Such scripts (written by Bernd Kraus) are available for FFT and IFFT, for both real- and complex input.

Difference between '==', '=', and ':='

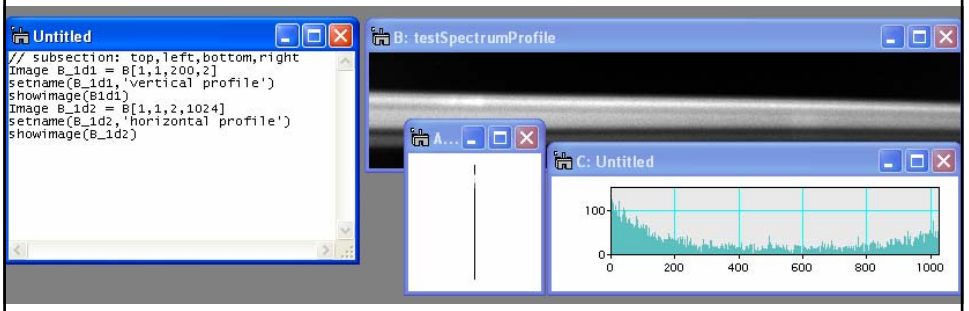
A==B comparison (1, if left=right, 0 otherwise)

A = B copies variable content

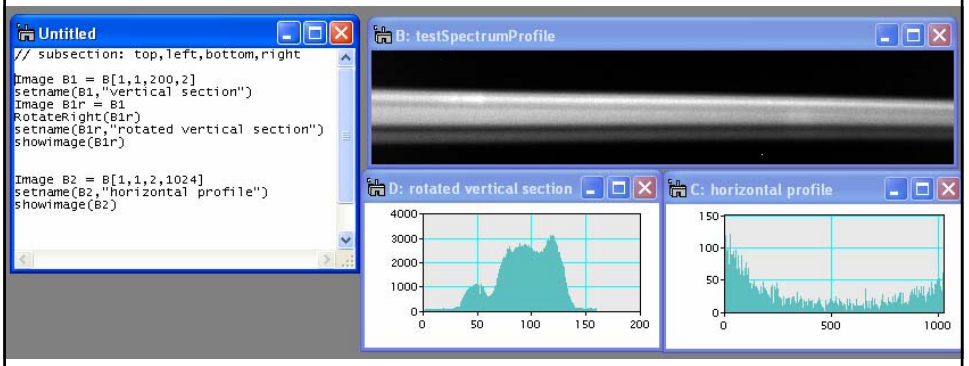
A:= B assigns variable A to the image given by B
(should always be used when creating images,
although it also works with '=')

(see more examples in B. Schaffer's tutorial, slides 33 & 34)

Spectra: 1-dimensional images



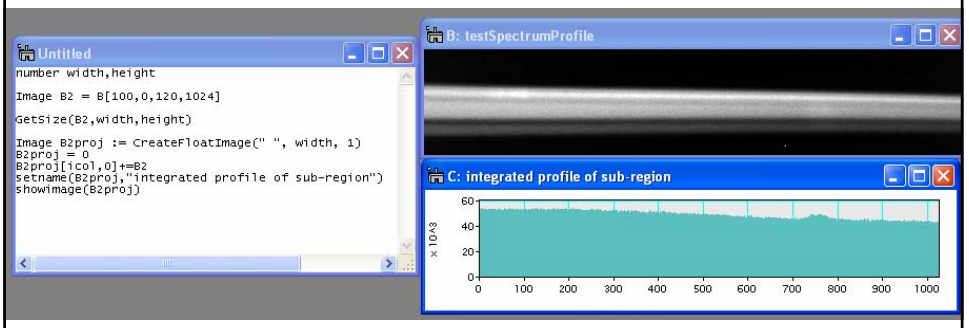
Vertical 1D-image -> spectrum



Allowed values for selection of sub-region:

top: 0	…(bottom-1)	bottom:	(top+1)	… height
left: 0	… (right-1)	right:	(left+1)	… width

Integrated linescans



This script sums horizontal linescans fromvertical position 100 to 119

icol, irow, iradius, ...

- There are several *intrinsic* variables which can be used in calculations of images. Their value depends on the position within the image.

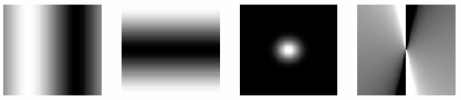
  (e.g.: `icol` becomes 5 for all points in an image, which have x=5 as coordinate. It becomes 6 for x=6 and so on..)

Name	Description
icol	column of the image
iheight	height of the image
ipoints	number of points in the image
iradius	distance from the center of the image
irow	row of the image
itheta	angle with respect to the center of the image
iwidth	width of the image
iplane	plane of the image (3D images)

- The following script creates some examples:

  (The function `Pi()` returns the value of Pi.)

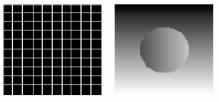
```
image  TestImage
TestImage := RealImage("Test",4,100,100)
ShowImage(TestImage)

TestImage = sin(2*Pi()/iwidth*icol)
TestImage = cos(2*Pi()/iheight*irow)
TestImage = exp(-iradius**2/(iheight/10)**2)
TestImage = tan(itheta)
```



- Often, the intrinsic variables are used in the `tert()` command:

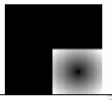
  (The function `mod(a,b)` returns the modulo, e.g. `mod(14,3)=2` as 14 = 4*3 + 2)

```
TestImage = tert( mod(icol,10)==0 || mod(irow,10)==0,1,0)
TestImage = tert( iradius<iwidth/4 , icol , irow)
```

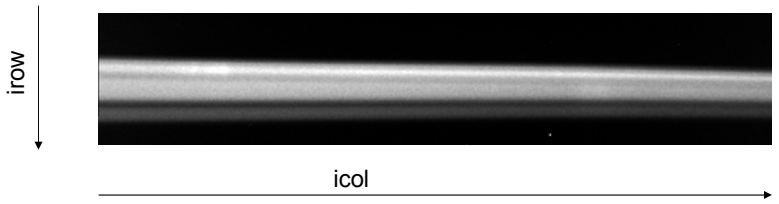


- Note that the variables check the actual image expression, not the image itself. If an area of an image is used, the top-left pixel of this area is (0/0):

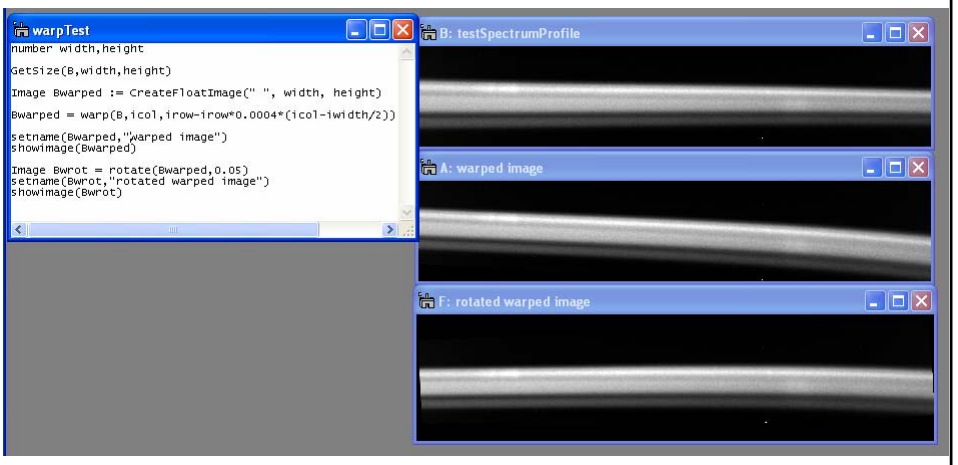
```
TestImage = 0
TestImage[50,50,100,100] = iradius  // the center is now at 75/75!
```



Icol, irow ….



The WARP function



The warp function is extremely powerful in removing (linear and non-linear) image distortions!