# Tempas Script Language

Tempas has a built in scripting C like scripting language with some C++ like member functions. Since many users of Tempas might be familiar with Digital Micrograph (DM) by Gatan Inc., there has been an effort made to make much of the scripting functionality of DM available in Tempas. Thus while many functionalities have alternate function names and arguments, in most cases there will be a DM syntax compatible function available. This is so that many DM scripts can be directly translated to Tempas with minimal effort. There is a basic difference in the syntax between DM and MT scripting when subroutines are used, but the difference is rather trivial and is explained below.

When using subroutines, the main entry point must have the routine main() declared as in

```
main() {
  number x = 10
  number y = 5
  number z = test(x,y)
  print(z)
}
int test(number x, number y) {
  return (x + y)
}
```

If there are no function / subroutine calls, then one can use either

```
main() {
  number x = 10
  number y = 5
  number z = x + y
  print(z)
}
```

or simply

```
number x = 10
number y = 5
number z = x + y
print(z)
```

without the main() { …. } declaration

This document parallels the DM scripting documentation such that differences and compatible syntax are clearly described.

There are some major differences between the scripting in DM and MT as far as the support of HRTEM simulation is concerned. MT allows the user to script the simulation.

At this point not all aspects of the simulation can be controlled, but this will change with further development. Functions marked with an * are DM functions that are not (yet) implemented in Tempas.

**EXECUTING THE SCRIPT**: Execution of the script is done by pressing the "Enter" key when the Script window has keyboard focus. This is fn+Return on a MacBook or MacBook Pro. *While the above is still true, the Script Window has now gained a "Run" button which starts the script. The "Run" button changes to a "Stop" button when the script is running.*

# Language Syntax

The Tempas scripting language is very similar to the syntax of the "C" programming language. The language is fairly simple, but is still quite powerful as in that it supports such types as "Image" , "ImageStack", "Microscope"… and so on as built in types. In many respects the types correspond to the type "Class" as used in C++ as they have built in "member functions" that operate on the type.
The language is case insensitive such that number, Number, NumBer etc. are all interpreted as the lower case type number

## Types

| | |
|---|---|
| int , short or bool | : Integer number |
| Number, float | : Real number |
| ComplexNumber or Cmplx | : Complex number {x,y} |
| String | : Holds a string like "This is a String" |
| Image | : a 2D Image, Height (or )Width can be 1 |
| ComplexImage | : a Complex Image of the above |
| Image3D | : a 3D Volume "Image" of (width,height,depth) |
| ComplexImage3D | : a Complex Image3D of the above |
| ImageStack | : a "stack" of Images of any number |
| Simulation | : an instance of a "Simulation" |
| Microscope | : an instance of a "Microscope" |
| Vector | : a Vector of real or complex numbers |
| Matrix | : a Matrix (ncols,nrows) of real or complex numbers |
| File | : a "File" that can be written to |

The language allows the use of the following type names
bool
short
int
float
"bool", "short" and "int" are all Integers (int) while the type "number" is a real number (float). The built in constants "True" , "False", "Yes" ,"No", "On" and "Off" (not case sensitive) correspond to the numbers 1 and 0 respectively. The constant π can be

expressed either as just “Pi” or as a function pi(). The constant “e” is expressed as a function exp(..).

Comments at the end of lines are specified using “//”

Multiple lines comments can be started and ended by a pair of /* and */
Take must be taken to not have other such pairs within the outer set.
Example

```
/*
number i
Image  testImage = NewImage(512,512)
Image  anotherImage(512,512,sin(2*pi*icol/128)+cos(2*pi*irow/64))
*/
```

The three lines in the middle will be ignored as if they were not present.

## Arrays

The scripting language supports arrays of Types
Hence one can write

| | |
|---|---|
| **number** x[100] | Declares x to hold 100 numbers. Each element is indexed as x[i] |
| **Image** img[5] | Declares 5 “Pointers” to Images. Each Image must be created separately, using NewImage or something similar |
| **String** str[40] | Declares 40 empty strings |

etc.

The syntax allows

```
x[0] = 5
x[1] = x[0]
int i
number x[100],y
image img[2],img2
for(int i=0; i < 100; i++) x[i] = sin(2*pi*i/64)
img[0] = NewImage(“name”,512,512)
img[1] = img[0]
str[0] = “Hello there”
str[1] = str[0]
y = img2[3,5] // y is assigned the value of the pixel at position [3,5]
y = img[1].getpixel(3,5)
```

and so on.

## Control loops key words

for , do , while , continue , break

Examples. **for:**

```
number x,y                // Declares the two variables x and y
number a = 10.2 , b = -3.5 // Declares a and b and sets a to 10.2 and b to -3.5
//
-------------------------------------------------------------------------------
--
// Initializes x to 0, executes the loop as long as x is less than 10
// and at the end of the loop increments x with 1
// The syntax x++ is equivalent to x = x + 1
//
-------------------------------------------------------------------------------
--
for ( x = 0; x < 10; x++) {
      y = a*x + b         // Defines a line segment
}


// Alternatively one could have done something like this
// Statements between  /*   …  */ are not executed. Treated as comments
number x[10],y[10]
for ( int i = 0;i < 10; i++) {
      /* y = slope * x + intercept */
      y[i] = a*x[i] + b
}
// or
for ( x = 0; x < 10;) {
      y = a*(x++) + b           // here x is incremented after evaluating the
expression
      // y = a*(++x) + b ;      here x is incremented before evaluating the
expression
}
```

**do** and **while:**

```
number x[10],y[10]        // Declares the two arrays x and y of real numbers
number a = 10.2 , b = -3.5 // Declares a and b and sets a to 10.2 and b to -3.5
int i = 0
do {
      y[i] = a*x[i] + b         // Defines a line segment
      i++
} while (i < 10)
```

**while:**

```
number x[10],y[10]        // Declares the two arrays x and y of real numbers
number a = 10.2 , b = -3.5 // Declares a and b and sets a to 10.2 and b to -3.5
int i = 0
while (i < 10) {
      y[i] = a*x[i] + b   // Defines a line segment
      i++
}
```

Use of “break” and “continue”

**break:**

```
number x,i
for ( i = 0; i < 10; i++) {
       x = i
       if (x == 5) break   // Break out of the loop if x is equal to 5
}
```

**continue:**

```
number x,i
for ( i = 0; i < 10; i++) {
       if (i == 5) continue       // Do not execute the next step(s) and go to
       x = 2*i                    // top of the loop ( next cycle)
}                                 // x = 2* i except for when i is equal to 5
```

## Library Functions

Library functions are a number of predefined functions that operate on numbers, strings and images and image-volumes(Image3D) .

Library functions operating on image]/image3D usually take the image(3D) of interest as an argument and returns an instance of the “result”, leaving the argument unchanged.

An example of the use of a predefined function is FFT. The function takes an image as an argument and returns the Fourier Transform of the argument as a new image.
Usage could be

```
Image ftimg = fft(img)     // Returns the Fourier Transform of the image “img”,
                           // “ftimg” is assigned to the return image
                           // The argument “img”is left unchanged

Image rotImg = Rotate(img,45)     // Makes a copy of “img” and rotates
                                  // the copy 45 deg. anti-clockwise
                                  // Returns the rotated image and
                                  // leaves “img” unchanged
```

## Member Functions

Member functions are functions that operate on an instance of a variable type.
An example of this would be the various functions that belong to the variable type “Image”. The member functions operate directly on “itself”. Thus while many library return a new image resulting from an operation on a copy of its argument, the member function will change itself and return nothing (except when specifically noted).
Thus if one wanted to take the Fourier transform of an image using a member function, this would be

```
img.fft()          // Take the Fourier Transform of itself
img.rotate(45)     // Rotate oneself 45 degrees anti-clockwise
```

## Built-in Implied loop keywords (follows the use in DM and clearly inspired by DM)

***icol*** // The index of the column in an implicit loop over entire image
***irow*** // The index of the row in an implicit loop over entire image
***iradius*** // The value of the radius in an implicit loop over entire image
***itheta*** // The value of the angle in an implicit loop over entire image
***iplane*** // The index of the plane in an implicit loop over entire image3D
***iwidth*** // The width of the image while within an implied loop
***iheight*** // The height of the image while within an implied loop
***idepth*** // The depth of the image while within an implied loop
***ipoints*** // The number of points of the image while within an implied loop
***ipixels*** // The number of points of the image while within an implied loop

The built in loop key words are incredibly powerful and save an enormous amount of computing time. They should always be used whenever possible.

Whenever one wants to do a loop over the entire image
such as

```
number i,j,value
number width = img.GetWidth()
number height = img.GetHeight()
for (j = 0; j < height; j++) {
       for (i = 0; i < width ; i++) {
              value = Some Expression
              img[i,j] = value
       }
}
```

and “Some Expression” can be expressed in terms of icol, irow, iradius etc.. , one should use these implied loops.
For instance the Expression

```
img = sin(2*pi*icol/32)+sin(2*pi*irow/32)
```

Is equivalent to evaluating the following.

```
number i,j,x,y
number width = img.GetWidth()
number height = img.GetHeight()
for (j = 0; j < height; j++) {
       y = sin(2*pi*j/32)
       for (i = 0; i < width ; i++) {
              x = sin(2*pi*i/32)
              img[i,j] = x+y
              // or one could write
```

```
            //     SetPixel(img,i,j,x+y)
            //     img.SetPixel(i,j,x+y)
      }
}
```

Similarly

```
img = exp(-iradius*iradius/64) or img = exp(-iradius**2/64)
```

Is equivalent to

```
number i,j,x,y,val
number width = img.GetWidth()
number height = img.GetHeight()
for (j = 0; j < height; j++) {
      y = j - height/2
      for (i = 0; i < width ; i++) {
            x = i - width/2
            val = exp(-(x**2+y**2)/64)  // iradius = sqrt(x*x+y*y)
            SetPixel(img,i,j,val)
      }
}
```

*The first expressions take a fraction of a second to compute, while the second approach can take minutes. Thus you should always use the expressions **icol** and **irow** whenever possible when you want to do a loop over the entire image pixel by pixel.*
*It will be possible if the pixel value at (i,j) is an expression of i and j (**icol** and **irow**).*
*The power of the use of **icol** and **irow** cannot be overestimated. When using the type Image3D, **iplane** takes on the third dimension (z)*

```
Image3D vol(32,32,32,iplane)     // Creates an image-volume of size 32*32*32
                                 //where each “plane” in the z-dimension is
                                 // set to the value of its z- index ( 0 - 31 )
```

# Graphing Options

1 Dimensional data can be visualized using the functions

```
graphxy(…)

plot(…)
```

and implicitly by displaying Images 1D (default) or 2D (setting the display style)

The code……

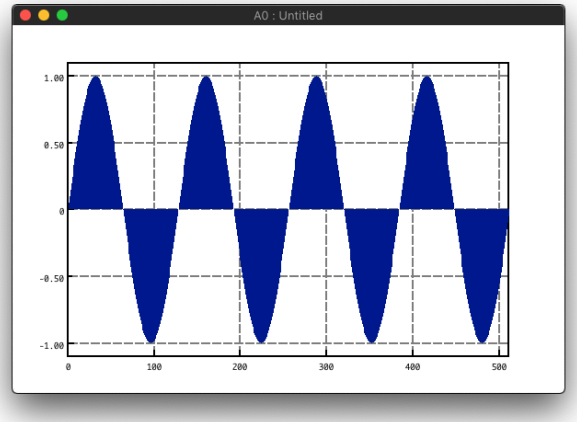
```
Image sineWave(512,1,sin(2*pi*icol/128))
sineWave.show()
…
```

will result in the following output

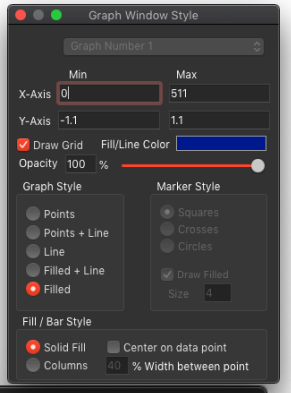
The same output could be produced by

```
…..
number y[512]
for(int i =0; i < 512; i++) {
	y[i] = sin(2*pi*i/128)
}
plot(y)
…..
```

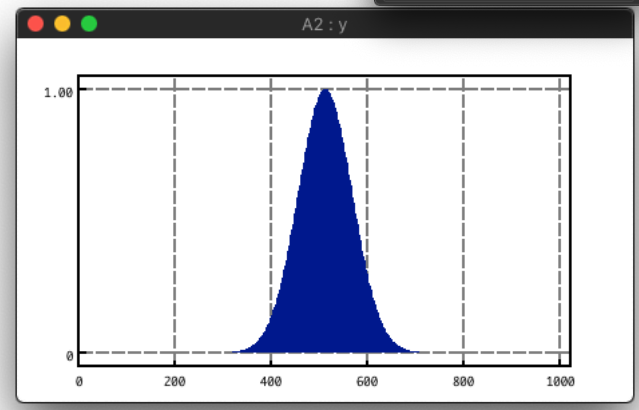
An example of plotting y vs x is the following…




Right Clicking in the Graph Window brings up a window for setting some further display options




```
number x[512],y[512],sigma = 40
int i
for(i = 0; i < 512; i++) {
	x[i] = 2*i
	y[i] = exp(-(x[i]/2-256)**2/sigma**2)
}
plot (x,y)
```

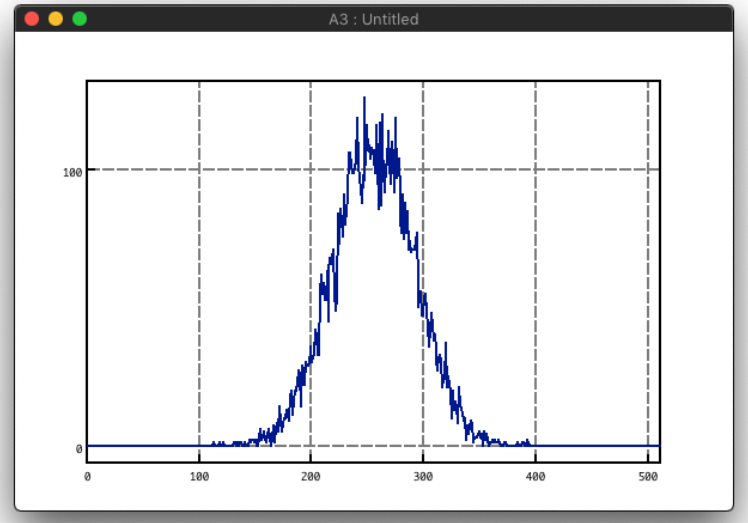



and…

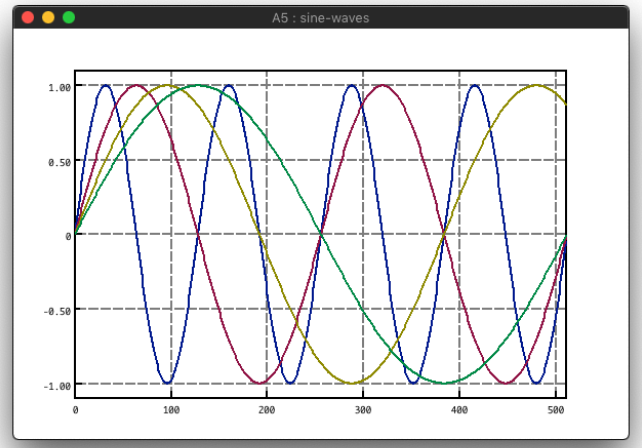
```
number y[512],x,sigma=30
int ndx,i
for(i = 0; i < 10000; i++) {
	x = GaussianRandom(sigma)
	ndx = x + 256
	if (ndx < 0) continue
	if (ndx > 511) continue
	y[ndx]++
}
graphxy (y)
```

and….

```
Image y(512,4)
number x[4]
```



```
for(int i = 0; i < 512;i++) {
	for(int j=0; j < 4; j++) {
		x[j] = sin(2*pi*i/((j+1)*128))
		y[i,j] = x[j]
	}
}
y.setName(“Sine-Waves”)
// 1 = rasterimage, 2 = RGB, 3 = SurfacePlot,
// 4 = Lineplot, 5 = Table, 6 = ArgandPlot,
// 7 =ComplexLinePlot
y.setDisplayType(4)
y.show()
```



# Scripting Reference

## Real Numbers / Integer Numbers

### Declaration

```
number x
number x(3.4)                Declares x and sets it to the
                             value 3.4
int    ix                    Declares ix and sets it to
int    ix(3.4)               the value 3
```

### Operators (operating on numbers), *read* Integer when appropriate

| Name | Summary |
|---|---|
| ! | Logical NOT operator for a real number |
| != | Inequality operator for real numbers |
| && | Logical AND operator for real numbers |
| * | Multiply operator for real numbers |
| ** | Exponentiation operator for real numbers |
| *= | Multiply and assign operator for real numbers |
| + | Addition operator for real numbers |
| ++ | Increment operator for a real number |
| += | Add and assign operator for real numbers |
| - | Negation operator for a real number |
| - | Subtraction operator for real numbers |
| -- | Decrement operator for real numbers |
| -= | Subtract and assign operator for real numbers |

| | |
|---|---|
| / | Division operator for real numbers |
| /= | Divide and assign operator for real number |
| < | Less than operator for real numbers |
| <= | Less than or equal operator for real numbers |
| = | Assignment operator for real numbers |
| == | Equality operator for real numbers |
| > | Greater than operator for real numbers |
| >= | Greater than or equal operator for real numbers |
| ? | Arithmetic if operator for real numbers |
| \|\| | Logical OR operator for real numbers |

## Functions (operating on numbers)

| Name | Summary |
|---|---|
| abs | Calculates absolute value of a real number |
| acos | Calculates the arccosine of a real number |
| acosh | Calculates the hyperbolic arccosine of a real number |
| AiryAi | Calculates the Airy Ai function |
| AiryBi | Calculates the Airy Bi function |
| asin | Calculates the arcsine of a real number |
| asinh | Calculates the hyperbolic arcsine of a real number |
| atan | Calculates the arctangent of a real number |
| atan2 | Calculates the arctangent of y/x for real numbers, real images or a complex image |
| atanh | Calculates the hyperbolic arctangent of a real number |
| BesselI | Calculates the general Bessel |

| | |
|---|---|
| | I function |
| BesselJ | Calculates the general Bessel J function ( 0 , 1 and N) |
| BesselK | Calculates the general Bessel K function |
| BesselY | Calculates the general Bessel Y function of orders (0,1 and n) |
| Beta | Calculates the beta function |
| BinomialCoefficient | Calculates the binomial coefficient ${}_nC_k$ |
| BinomialRandom* | Calculates a random number with binomial distribution |
| clip | Clip real number to be in a range |
| cos | Calculates the cosine of a real number |
| cosh | Calculates the hyperbolic cosine of a real number |
| distance | Calculates the pythagorean theorem |
| erf | Calculates the error function |
| erfc | Calculates the complement of the error function |
| exp | Calculates the exponential of a real number |
| exp1 | Calculates the exponential and of a real number and subtracts 1 |
| exp10 | Calculates 10 raised to a real number |
| exp2 | Calculates 2 raised to a real number |
| ExponentialRandom | Calculates a random number with exponential distribution |
| Factorial | Calculates the factorial of a real number |
| Gamma | Calculates the gamma of a real number |
| GammaP | Calculates the incomplete gamma function |
| GammaQ | Calculates the complement of the incomplete gamma function |
| GammaRandom* | Calculates a random number with gamma distribution |
| GaussianRandom | Calculates a random number |

| | |
|---|---|
| | with gaussian distribution |
| LegendrePolynomial | Calculates the Legendre polynomial function |
| log | Calculates the logarithm of a real number |
| log1 | Calculates the logarithm of a real number and adds 1 |
| log10 | Calculates the logarithm base 10 of a real number |
| log2 | Calculates the logarithm base 2 of a real number |
| LogGamma | Calculates the log gamma of a real number |
| max | Calculates the maximum of two real numbers |
| Maximum | Calculates the maximum of a given list of real numbers |
| Median | Calculates the median of a given list of real numbers |
| min | Calculates the minimum of two real numbers |
| Minimum | Calculates the minimum of a given list of real numbers |
| mod | Calculates the integer modulus for real numbers |
| Pi | Returns an approximation of $\pi$ |
| PoissonRandom | Calculates a random number with poisson distribution |
| Random | Calculates a random number with uniform distribution |
| Remainder | Calculates the integer remainder for real numbers |
| Round | Rounds a real number to the nearest integer |
| Sgn | Calculates the sign of a real number |
| sin | Calculates the sine of a real number |
| sinh | Calculates the hyperbolic sine of a real number |
| SphericalBesselJ | Calculates the spherical Bessel J function |
| SphericalBesselY | Calculates the spherical Bessel Y function |
| sqrt | Calculates the square root of a real number |

| | |
|---|---|
| tan | Calculates the tangent of a real number |
| tanh | Calculates the hyperbolic tangent |
| toDeg | Returns the value of the argument(radians) in Degrees |
| toRad | Returns the value of the argument(degrees) in radians |
| Trunc | Truncates a real number to an integer |
| UniformRandom | Calculates a random number with uniform distribution |

*Not yet implemented

### Pre Defined Constants

| Name | Summary |
|---|---|
| true | Evaluates to 1 |
| false | Evaluates to 0 |
| yes | Evaluates to 1 |
| no | Evaluates to 0 |
| on | Evaluates to 1 |
| off | Evaluates to 0 |
| pi | Evaluates to “pi” = 3.14… |

## Complex Numbers

### Declaration

| | |
|---|---|
| complexnumber z | Declares a complex number variable z set to (0,0) |
| cmplx z | cmplx is equivalent to complexnumber |
| cmplx z(1.0) | Declares z as a complex number and sets it equal to (1.0 + i0.0) |
| cmplx z(1.0,0.3) | Declares z as a complex number and sets it equal to |

| | |
|---|---|
| | (1.0 + i0.3) |
| cmplx z(cis(pi/4)) | Declares z as a complex number and sets it equal to exp(i*pi/4) ( cos(pi/4) + i*sin(pi/4) ) |

## Operators

| Name | Summary |
|---|---|
| != | Inequality operator for complex numbers |
| * | Multiply operator for complex numbers |
| ** | Exponentation operator for complex numbers |
| *= | Multiply and assign operator for complex numbers |
| + | Addition operator for complex numbers |
| += | Add and assign operator for complex numbers |
| - | Negation operator for a complex number |
| - | Subtraction operator for complex numbers |
| -= | Subtract and assign operator for complex numbers |
| / | Division operator for complex numbers |
| /= | Divide and assign operator for complex numbers |
| = | Assignment operator for complex numbers |
| == | Equality operator for complex numbers |
| ? | Arithmetic operator for complex numbers |

## Functions

| Name | Summary |
|---|---|
| abs() | Calculates the absolute value of a complex number |

| | |
|---|---|
| cis() | Calculates a unit vector in the complex plane cis(arg) = (cos(arg), sin(arg)) |
| complex() | Creates a complex number from two real numbers |
| conjugate() | Calculates the conjugate of a complex number |
| cos() | Calculates the cosine of a complex number |
| cosh() | Calculates the hyperbolic cosine of a complex number |
| exp() | Calculates the exponential of a complex number |
| imaginary() | Returns the imaginary portion of a complex number as a real number |
| log() | Calculates the logarithm of a complex number |
| modulus() | Calculates the modulus of a complex number |
| norm() | Calculates the norm of a complex number |
| Phase() | Calculates the phase of a complex number |
| Polar() | Calculates the polar representation of a rectangular complex number |
| real() | Returns the real portion of a complex number |
| Rect() | Calculates the rectangular representation of a polar complex number |
| sin() | Calculates the sine of a complex number |
| sinh() | Calculates the hyperbolic sine of a complex number |
| sqrt() | Calculates the square root of a complex number |
| tan() | Calculates the tangent of a complex number |
| tanh() | Calculates the hyperbolic tangent of a complex number |

# Complex Number Member Functions

## Functions

| Name | Summary |
|---|---|
| set() | Sets the x,y pair of the complex number |
| real() | returns the real part of the complex number or sets the value of the real part if an argument is given. |
| imag() | returns the imaginary part of the complex number or sets the value of the imaginary part if an argument is given. |
| x() | returns the real part of the complex number or sets the value of the real part if an argument is given. |
| y() | returns the imaginary part of the complex number or sets the value of the imaginary part if an argument is given. |
| setX() | Equivalent to x(arg) |
| setY() | Equivalent to y(arg) |
| phase() | returns the phase in radians of the complex number |
| angle() | returns the phase in degrees of the complex number |
| modulus() | returns the modulus of the complex number |
| modsq() | returns the modulus square of the complex number |
| conjugate() | returns the complex conjugate of the complex number |

**Example:**

```
ComplexNumber c(2,3)       // Declares and initializes complex c
number x = c.x()           // x = 2
number y = c.y()           // y = 3

c.x(10)      // Sets the real part to 10
c.y(4)       // Sets the imaginary part to 4
c.setX(10)   // Sets the real part to 10
c.setY(4)    // Sets the imaginary part to 4

c.set(4,6)    // Sets the complex number equal to (4,6)

cmplx d = c.conjugate()
```

```
number phase = d.phase()
```

# Real Images

## Declaration

```
image img                       Declares a pointer to an
                                image which must be created
                                or assigned

image img(ncols,nrows)          Declares and creates a real
                                image of size ncols by nrows
                                Width=ncols , Height=nrows

image img(512,512,exp(-iradius**2/64))

                                Declares and creates a real
                                image of size 512 by 512,
                                assigning it to a Gaussian of
                                sigma 8 ( exp ( - (r/8)**2 )
```

## Operators

| Name | Summary |
|---|---|
| * | Multiply operator for real images |
| ** | Exponentiation operator for real images |
| *= | Multiply and assign operator for real images |
| + | Addition operator for real images |
| ++ | Increment operator for a real images |
| += | Add and assign operator for real images |
| - | Negation operator for a real images |
| - | Subtraction operator for real images |
| -= | Subtract and assign operator for real images |
| / | Division operator for real images |

| | |
|---|---|
| /= | Divide and assign operator for real images |
| < | Less than operator for real images |
| <= | Less than or equal operator for real images |
| = | Assignment operator for real images |
| == | Equality operator for real images |
| > | Greater than operator for real images |
| >= | Greater than or equal operator for real images |
| ? | Arithmetic if operator for real images |
| [] | Image region expression |

## Library Functions

| Name | Summary |
|---|---|
| abs | Returns a real image containing the absolute values of a real image |
| acos | Returns a real image containing the arccosine of a real image |
| acosh | Returns a real image containing the hyperbolic arccosine of a real image |
| asin | Returns a real image containing the arcsine of a real image |
| asinh | Returns a real image containing the hyperbolic arcsine of a real image |
| atanh | Returns a real image containing the hyperbolic arctangent of a real image |
| ceiling | Sets all values larger than a given value to the given value |
| clip | Sets all values smaller than a given value to the value and all values larger than a given value to the given value |
| cos | Returns a real image containing the hyperbolic cosine of a real image |

| | |
|---|---|
| cosh | Returns a real image containing the cosine of a real image |
| DotProduct | Calculates the dot product of two real image expressions |
| exp | Returns a real image containing the exponential of a real image |
| exp1 | Returns a real image containing the exponential of a real image and subtracts 1 |
| exp2 | Returns a real image containing 2**image |
| exp10 | Returns a real image containing 10**image |
| ExprSize | Sets the physical size of a real image expression |
| ExprSize | Sets the physical size of a real image expression |
| factorial | Returns the factorial of an image (values are rounded to integers) |
| floor | Sets all values smaller than a given value to the given value |
| log1 | Returns an image of the log of an image after subtracting 1. |
| log10 | Calculates log10 of an image |
| log2 | Calculates log2 of an image |
| log | Calculates the natural logarithm of an image |
| max | Finds the maximum of a real image expression |
| max | Finds the maximum value and position for a real image expression |
| mean | Calculates the mean of a real image expression |
| MeanSquare | Calculates the mean square of a real image expression |
| median | Calculates the median of a real image expression |
| min | Finds the minimum value and position for a real image expression |
| min | Finds the minimum of a real image expression |
| norm | Returns an image of the norms of an image (xi-squared) |

| | |
|---|---|
| Polynomial | Calculates a polynomial expansion using a real image expression |
| Pow | Returns a real image containing image**x |
| pow2 | Returns a real image containing 2**image |
| pow10 | Returns a real image containing 10**image |
| product* | Calculates the product of a real image expression |
| RMS | Calculates the RMS of a real image expression |
| Round | Rounds all values to the nearest integer |
| sum | Calculates the sum of a real image expression |
| sigma | Returns the standard deviation of an image |
| sin | Returns a real image containing the sine of a real image |
| sinh | Returns a real image containing the hyperbolic sine of a real image |
| sqrt | Returns a real image containing the square root of a real image |
| sq | Returns a real image containing the square of a real image |
| square | Returns a real image containing the square of a real image |
| stdv | Returns the standard deviation of an image |
| tan | Returns a real image containing the tangent of a real image |
| tanh | Returns a real image containing the hyperbolic tangent of a real image |
| TimeBar” | Displays a timebar while evaluating real image expression |
| Trunc | Truncates all real values to the integer part |
| Variance | Returns the variance of the image |
| Vectorlength | Returns the square root of the sum of all pixels squared |
| Warp | Calculates bilinear |

| | |
|---|---|
| | interpolated value within a real image |

*Not yet implemented

# Complex Images

## Declaration

| | |
|---|---|
| compleximage img | Declares a “Pointer” to a complex image which must be created or assigned |
| compleximage img(ncols,nrows) | Declares and creates a complex image of size ncols by nrows<br>Width=ncols , Height=nrows |

However it is not necessary in most instances to declare an image to be complex. One can also declare it as type Image.
The construct…

```
Image img(512,512)
Cmplx z = complex(3,2)
img = z      // Will convert the Image from being real to complex
```

## Operators

| Name | Summary |
|---|---|
| * | Multiply operator for complex images |
| ** | Exponentiation operator for complex images |
| *= | Multiply and assign operator for complex images |
| + | Addition operator for complex images |
| += | Add and assign operator for complex images |

| | |
|---|---|
| - | Negation operator for a complex images |
| - | Subtraction operator for complex images |
| -= | Subtract and assign operator for complex images |
| / | Division operator for complex images |
| /= | Divide and assign operator for complex images |
| = | Assignment operator for complex images |
| == | Equality operator for complex images |
| ? | Arithmetic if operator for complex images |

### Functions

| Name | Summary |
|---|---|
| ComplexConjugate | Returns the complex conjugate of an image |
| Conjugate | Returns the complex conjugate of an image |
| Real | Returns the real part of an image |
| Imaginary | Returns the imaginary of an image |
| Intensity | Returns the modulus square of a complex image |
| Phase | Returns the phase of a complex image |
| Modulus | Returns the modulus (amplitude) of a complex image |

## Built in Image Expressions

| Name | Summary |
|---|---|
| icol | When used in an expression involving an image, icol will refer to the index of the |

| | |
|---|---|
| | column in the image and there is an implied loop over all the elements of an image. Basically each pixel takes the value of the column number of the pixel. |
| irow | When used in an expression involving an image, irow will refer to the index of the row in the image and there is an implied loop over all the elements of an image. Each pixel takes the value of the row number of the pixel. |
| iplane | When used in an expression involving a 3D image, iplane will refer to the index of the depth in the 3D image and there is an implied loop over all the elements of an image. Each voxel has the value of the plane number of the voxel. |
| iradius | When used in an expression involving an image, iradius will refer to the value of sqrt((I-W/2)* (I-W/2)+(J-H/2)* (J-H/2)), where I and J are the column and row index of the image and W and H are the width and height of the image. There is an implied loop over all the elements of an image. Each pixel has the value of the radius of the pixel. |
| itheta | When used in an expression involving an image, itheta will refer to the value of atan((J-H/2)/(I-W/2)), where I and J are the column and row index of the image and W and H are the width and height of the image. There is an implied loop over all the elements of an image. Each pixel has the value of the angle with the x-axis of the pixel. |
| iwidth | When used in an expression involving an image, iwidth will refer to the width of the image. It's a constant. |
| iheight | When used in an expression |

| | |
|---|---|
| | involving an image, iheight will refer to the height of the image. It's a constant. |
| idepth | When used in an expression involving a 3D volume image, idepth will refer to the depth of the image. It's a constant. |
| ipoints / ipixels | When used in an expression involving an image, ipoints / ipixels will refer to the number of pixels in the image. It's a constant. |

# Image Stacks

## Declaration

| | |
|---|---|
| imagestack stack | Defines and Creates an empty image stack |

## Assignment

| | |
|---|---|
| imagestack stack<br>stack = existingStack | Defines the ImageStack<br>Sets the stack equal to an existing image stack |
| imagestack stack<br>stack = existing3DImage | Defines the ImageStack<br>Sets the stack equal to an existing 3D image volume |

## Member functions

| Name | Summary |
|---|---|
| AddImage(Image) | Adds an image to a stack |
| DeleteImage(Image) | Deletes an image from a stack |
| FFT(int num) | Performs a Fourier transform on an image on the stack |
| FFT | Performs a Fourier transform |

```
                                of every image on the stack
GetImage(int Num)               Returns an image from a stack
GetnumberOfImages()             returns the number of images
                                in a stack
IFFT(int num)                   Performs the inverse Fourier
                                transform on an image in the
                                stack
IFFT                            Performs the inverse Fourier
                                transform of every image on
                                the stack
Save(String)                    Saves an image stack as a MRC
                                file
```

# Volume Images

## Declaration / Creation

```
Image3D img3                    Declares a “Pointer” to a 3D
                                Volume / 3D Image

Image3D img3(32,32,32)          Declares and creates an Image
                                Volume of size 32 by 32 by 32

Image3D img3(32,32,32,exp(-iradius**2/8**2))
                                Declares and creates an Image
                                Volume of size 32x32x32 and
                                assigning it to a “spherical
                                Gaussian” of sigma 8
```

## Assignment

```
Image3D img3                    Defines the 3D Image
img3 = existingImage3D          Sets the Image Volume from an
                                existing 3D image volume
img3[x1:x2,y1:y2,z1:z2] =
existingImage3D[xx1:xx2,yy1:yy2,zz1:zz2]    where
                                (xx2-xx1) = (x2-x1)
                                (yy2-yy1) = (y2-y1)
                                (zz2-zz1) = (z2-z1)


Image3D img3                    Defines the 3D Image
img3= existingStack             Sets the Image Volume from an
                                existing image stack
```

## Creation

| Name | Summary |
|---|---|
| exprsize3 | Function for creating a 3D Image Volume |
| Image3D name(width,height,depth) or Image3D name(width,height,depth,ImageExpression) | Creates the 3D Volume and assigns the volume to the image expression |

**example**:

```
image3d img = exprsize3(256,256,256)        // Declares and creates a volume image
                                            // set to the initial value 0
image3d img1 = exprsize3(256,256,256,10)    // Declares and creates a volume image
                                            // set  to the initial value 10
image3d vol(256,256,256,exp(-iradius**2/20**2))
```

## Member Functions

| Name | Summary |
|---|---|
| BeginFill() | Starts a fill from projections |
| Depth() | Returns the depth (z dimension in pixels) of the volume image |
| Display() | Displays a 3D (volume) image |
| EndFill() | Ends a fill from projections |
| FFT() | Performs a 3D Fourier transform of a 3D (volume) image |
| FFT2() | Performs a 2D Fourier transform of each image (plane) of the 3D (volume) image |
| FillFromProjection*() | Filling the volume image from a 2D projection |
| GetImage(int z) | Returns a 2D image from a given position (z) in the volume image |
| GetName() | Returns the name of the image |
| GetSize(w,h,d) | Returns the width, height and depth of the volume image |
| GetVoxel(x,y,z) | Returns the value at position (x,y,z) |
| Height() | Returns the height (y |

| | |
|---|---|
| | dimension in pixels) of the volume image |
| IFFT() | Performs a 3D inverse Fourier transform of a 3D (volume) image |
| IFFT2() | Performs a 2D Fourier inverse transform of each image (plane) of the 3D (volume) image |
| Imaginary() | Replaces the volume image with its imaginary part |
| Modulus() | Transforms the image to the modulus |
| Phase() | Transforms the image to the phase |
| Real() | Replaces the volume image with its real part |
| Repeat(nx,ny,nz) | Repeat the volume image NX,NY,NZ times |
| RotateX(angle) | Rotate about x clockwise |
| RotateY(angle) | Rotate about y clockwise |
| RotateZ(angle) | Rotate about x clockwise |
| Save() | Saves the volume image as a MRC file |
| SetImage(i,image) | Sets a 2D image at a given position (z) in the volume image |
| SetName(“name”) | Sets the name of the 3D image |
| SetVoxel(x,y,z,value) | Sets the voxel value at position (x,y,z) |
| sq() | Replaces each pixel (voxel) with the square of its value |
| sqrt() | Replaces each pixel (voxel) with the squareroot of its value |
| Width() | Returns the width (x dimension in pixels) of the volume image |

*Not yet implemented

# Image Data Type

## Declaration

```
image ss
compleximage css
```

### Creating / initializing

| Name | Summary |
|---|---|
| Exprsize(width,height,..) | Allocates and initializes an image |
| realimage(width,height) | Creates a real image of a given size |
| newimage(width,height) | Creates a real image of a given size |
| createimage(width,height) | Creates a real image of a given size |
| createfloatimage(width,height) | Creates a real image of a given size |
| createcompleximage(width,height) | Creates a complex image of a given size |
| openimage() | opens an existing image file |
| Example:<br>A0 = exprsize(512,512,icol) | Creates an image with label a0 and displays it. The image contains a ramp where each pixel has the value of the column number. |
| Image ss = newimage(“real image”,512,512)<br>compleximage css = createcompleximage(“Complex TestImage”,512,512) | |
| Example:<br>image ss = openimage(“image.tif”) | |

# Image Member Functions

### Functions

| Name | Summary |
|---|---|
| ac | Replaces a real image with its autocorrelation |
| acos | Replaces a real with its arccosine |
| acosh | Replaces a real with its |

| | |
|---|---|
| | hyperbolic arccosine |
| AdjustAngle | Adjusts the image so that it has an angle of 90 degrees. This is applicable for images returned from a simulation. In this case the image represents a periodic object and the angle of the unit cell may be different from 90 deg. |
| AdjustSampling | Adjusts the image so that it has equal sampling in x and y. This is applicable for images returned from a simulation. In this case the image represents a periodic object and the sampling along the a and b axes may be different. |
| Amplitude | Replaces a complex with its amplitude |
| AnnularHighpassFilter | Applies a high pass filter to an image |
| AnnularLowpassFilter | Applies a low pass filter to an image |
| ApplyAnnularMask | Applies an annular mask to an image |
| ApplyCircularMask | Applies a circular mask to an image |
| ApplyCosineMask | Applies a cosine mask to an image |
| ApplyHanningMask | Applies a Hanning mask to an image |
| ApplyMasksFromImage | Applies masks belonging to a different image onto itself |
| asin | Replaces the image with its arcsine |
| asinh | Replaces the image with its hyperbolic arcsine |
| atan | Replaces the image with its arctan |
| atan2 | Replaces the image with its arctan |
| atanh | Replaces the image with its hyperbolic arctan |
| Autocorrelate | Replaces a real image with its autocorrelation (equivalent to ac) |
| bgs | Applies a Background Noise Subtraction Filter on a real image |
| cc | Replaces a complex image with its complex conjugate |
| ccd | Corrects for CCD detector bad |

| | |
|---|---|
| | pixels in a real image |
| ceiling | Sets all values greater than maxVal to maxVal |
| CenterOfMass | Returns the value of and position of the “Center of Mass” |
| clip | Sets all values greater smaller than minVal to minVal and all values greater than maxVal to maxVal |
| ConvertToRealSpaceStorage | Changes the data storage to regular real space storage (x,y) |
| ConvertToReciprocalSpaceStorage | Sets the storage to that of h,k in reciprocal space. (h=0,k=0) at position (0,0) |
| Complexconjugate | Replaces a complex image with its complex conjugate (equivalent to cc) |
| ComplexModulusSq | This replaces a complex image with the product of itself and its complex conjugate. It is the complex modulus square. Imaginary part is zero |
| Cmsq | Equivalent short for ComplexModulusSq |
| Conjugate | Replaces a complex image with its complex conjugate (equivalent to cc) |
| cos | Replaces a real with its cosine |
| cosh | Replaces a real with its hyperbolic cosine |
| Display | Displays the image |
| Displayonlogscale | Displays the image on a log scale |
| exp | Takes the exponential of an image |
| exp1 | Takes the exponential of a real image and subtracts the value 1 |
| exp10 | Calculates the 10**image |
| exp2 | Calculates the 2**image |
| factorial | Takes the factorial of each pixel of an image |
| fft | Takes the Fourier transform of an image |
| Fillfromprojection | Fills in a 2D image from 1D projections |
| Fliphorizontal | Flips an image horizontally (around the vertical axis) |
| Flipvertical | Flips an image vertically (around the horizontal axis) |

| | |
|---|---|
| floor | Sets all values smaller than minVal to minVal |
| GaussianLowpassFilter | Applies a Gaussian low pass filter |
| GaussianHighPassFilter | Applies a Gaussian high pass filter |
| GetCalibration | Returns the calibration of the image |
| GetCalibrationunit | Returns the calibration unit of the image |
| GetGamma | Returns the angle associated with the image |
| GetLattice | Returns the lattice (if defined) for the image |
| GetName | Return the name of the image |
| GetPeaklist | Returns the peaklist (if defined) for the image |
| GetPixel | Returns the pixel value for a given pixel |
| GetScale | Returns the scale/calibration |
| GetScaleX | Returns the scale/calibration in X |
| GetScaleY | Returns the scale/calibration in Y |
| GetSize | Returns the width and height of the image |
| HasLattice | Returns true(1)/false(0) if a lattice is defined on an image |
| HasPeaklist | Returns true(1)/false(0) if a peak list is defined on an image |
| Height | Returns the height (in pixels) |
| HighpassFilter | Applies a highpass filter to the image |
| Ifft | Replaces a complex image in reciprocal space with its inverse Fourier transform |
| Imaginary | Replaces a complex with its imaginary part |
| Intensity | Replaces an image with its modulus squared |
| Inverse | Sets the Image Values to 1/ Values |
| Invert | Sets the image equal to -Image |
| Laplacian | Takes the Laplacian of a real image |
| Log | Takes the natural log of a real image |
| Log1 | Takes the natural log of a real image after adding the value 1 |

| | |
|---|---|
| log10 | Takes the log10 of a real image |
| log2 | Takes the log2 of a real image |
| Max | Returns the maximum of a real image |
| Mean | Returns the mean of a real image |
| Min | Returns the minimum of a real image |
| Modulus | Replaces a complex image with its modulus |
| PadWithMean | Pads an image with its mean value to specified dimensions |
| PadWithZero | Pads an image with zero to specified dimensions |
| Phase | Replaces a complex image with its phase |
| pow | Replaces the image with image**factor |
| pow10 | Replaces the image with 10**(image) |
| pow2 | Replaces the image with 2**(image) |
| Powerspectrum | Calculates the Power Spectrum of an image |
| Ps | Calculates the Power Spectrum of an image |
| rccd | Corrects for CCD detector bad pixels in a real image (*ccd*) |
| Real | Replaces a complex image with its real part |
| Removeccddefects | Corrects for CCD detector bad pixels in a real image (*ccd*) |
| Repeat | Repeats an image by tiling |
| Resize | Resizes an image |
| RMS | Returns the RMS value of a real image |
| Rotate | Rotates the image by a given angle anti-clockwise |
| RotateLeft | Rotates anti-clockwise an image by 90 deg. |
| RotateRight | Rotates clockwise an image by 90 deg. |
| Round | Rounds all values to the nearest integer |
| SetBlackWhite | Sets the black and white display limits of an image |
| SetCalibration | Sets the calibration of an image |
| SetCalibrationUnit | Sets the calibration unit of an image |
| SetImageSpace | Sets the space (real/ |

| | |
|---|---|
| | reciprocal) of an image |
| SetName | Sets the name of an image |
| SetPixel | Sets a specified pixel to a given value |
| SetScale | Sets the scale of an image |
| Sharpen | Applies a Sharpening Filter to a real image |
| Shift | Shifts the position (0,0) to a new position (x,y) in the image |
| ShiftCenter | Shifts the position (0,0) to the position (W/2,H/2) in the image |
| ShiftOrigin | Shifts the position (0,0) to the position (W/2,H/2) in the image |
| show | Displays an image |
| sigma | Returns the standard deviation of a real image |
| sin | Replaces a real image with its sine |
| sinh | Replaces a real image with its hyperbolic sine |
| Smooth | Applies a Smoothing Filter to a real image |
| sobel | Applies a Sobel Filter to a real image |
| sq | Takes the square of an image |
| sqrt | Takes the square root of a real image |
| square | Takes the square of an image |
| stdv | Returns the standard deviation of a real image |
| tan | Replaces a real with its tangent |
| tanh | Replaces a real with its hyperbolic tangent |
| thf | Applies a Threshold Filter to a real image |
| Transpose | Transposes an image |
| ThresholdFilter | Applies a Threshold Filter to a real image |
| Trunc | Truncates the values to its integer part |
| Update | Updates an image |
| Variance | Returns the variance of the image |
| wf | Applies a Wiener Filter to a real image |
| Width | Returns the width (pixels) of an image |
| WienerFilter | Applies a Wiener Filter to a real image |

**Example:**

```
image img = exprsize(256,256,icol)
img.sin()
img.fft()
img.setname("test")
img.display()
```

**Example:**

```
// a# as in a0, a1, a10… are automatically assigned as
// images and are displayed by default
a10 = exprsize(256,256,sin(2*pi()*icol/8)*sin(2*pi()*irow/12))
a11 = a10
a11.fft()
a10.setname("test")
a11.setname("Fourier Transform of test")
a12 = a10[64,64,192,192]     // a12 is set to the top,left,bottom,right subregion of a10
```

# Image Creation

## Functions

| Name | Summary |
|---|---|
| ExprSize | Allocates and initializes an image |
| RealImage | Creates a real image of a given size |
| NewImage | Creates a real image of a given size |
| CreateImage | Creates a real image of a given size |
| CreateFloatImage | Creates a real image of a given size |
| CreateComplexImage | Creates a complex image of a given size |
| OpenImage | opens an existing image file |
| Image imag(width,height,…) | Declares and creates an image of specified dimensions and optionally assigns it to an image-expression |

```
Images can also be created and assigned from an array
numbers, a vector and a matrix

Matrix m(100,100)
Vector v(100)
number x[100]
..
..
Image i1,i2,i3
i1 = m // Creates an Image of size(100,100)
i2 = v // Creates an Image of size(100,1)
i3 = x // Creates an Image of size(100,1)

The images are filled with the content of m, v and x
If the images are already created, they must have the
dimensions of m, v and x
```

# Image Management

## Functions

| Name | Summary |
|---|---|
| cexp | Returns a complex image from two images x and y (real part = cos(x)) (imaginary part = sin(x)) |
| cis | Returns a complex image from two images x and y (real part = cos(x)) (imaginary part = sin(x)) |
| CloseImage | Closes an existing image |
| Complex | Returns a complex image from two images x (real part) and y (imaginary part) |
| CreateComplexImage | Creates a complex image of a given size |
| CreateFloatImage | Creates a real image of a given size |
| CreateImage | Creates a new image of a given type |
| CreateImageFromDisplay | Creates an image from the information in a given window |
| CreateNewImage | Creates a new image of a given type |
| CreateRealImage | Creates a real image of a given size |
| CreateTableFromImage | Creates a table from an image |

| | |
|---|---|
| Delete | Deletes an image |
| DeleteImage | Deletes an image |
| DoesImageExist | Returns true/false if a given named image exists |
| Extract | Returns an image by extracting a region of an existing image |
| get2dSize | Returns width and height of an image |
| getCalibration | Returns the calibration of an image |
| getCalibrationUnit | Returns the calibration unit of an image |
| getCalibrationUnitString | Returns the calibration unit of an image |
| getFrontImage | Returns the front image |
| getHeight | Returns the height of an image |
| getMagnification | Returns the zoom factor of an image |
| getNamedImage | Returns the image with a given name |
| getNumberedImage | Returns the image with a label A# |
| getScale | Returns the scale of an image |
| getSize | Returns the width and height of an image |
| getUnitString | Returns the calibration unit of an image |
| getWidth | Returns the width of an image |
| getZoom | Returns the zoom factor of an image |
| NewImage | Creates a new image |
| Open | Opens a named image file |
| OpenImage | Opens a named image file |
| OpenWithDialog | Opens an image file using a file selector dialog |
| PrintImage | Prints a given image |
| RealImage | Creates a real image |
| Resize | Resizes an image |
| saveImage | Saves an image |
| setCalibration | Sets the calibration of an image |
| setCalibrationUnit | Sets the calibration unit of an image |
| setMagnification | Returns the scale of an image |
| setName | Returns the name of an image |
| setScale | Sets the scale of an image |
| setUnitString | Sets the calibration unit of an image |
| setZoom | Sets the zoom factor of an image |

# Image Processing

## Functions

| Name | Summary |
|---|---|
| Ac | Returns the autocorrelation of a real image |
| Align | Aligns two images |
| AlignImages | Aligns two images |
| AlignTwoImages | Aligns two images |
| AnnularHighPassFilter | Returns a new image of a high pass filtered image |
| AnnularLowPassFilter | Returns a new image of a low pass filtered image |
| ApplyAnnularMask | Returns an image resulting from the application of an annular mask to an image |
| ApplyCircularMask | Returns an image resulting from the application of an annular mask to an image |
| ApplyCosineMask | Returns an image resulting from the application of a circular cosine mask to an image |
| ApplyHanningMask | Returns an image resulting from the application of a circular hanning mask to an image |
| AutoCorrelate | Returns an image resulting from the auto-correlation of two images |
| AutoCorrelation | Returns an image resulting from the auto-correlation of two images |
| CC | Returns an image resulting from the cross-correlation of two images |
| Convolute | Returns an image resulting from the convolution of two images |
| Convolve | Returns an image resulting from the convolution of two images |
| Correlate | Returns an image resulting from the cross-correlation of two images |

| | |
|---|---|
| CrossCorrelate | Returns an image resulting from the cross-correlation of two images |
| CrossCorrelation | Returns an image resulting from the cross-correlation of two images |
| DotProduct | Returns the dot-product (inner product) of two images |
| FFT | Returns the Fourier transforms of an image |
| FindPattern | Returns the position dependent cross-correlation coefficient between an image and a pattern |
| FlipHorizontal | Returns an image resulting from mirroring an image around the vertical axis |
| FlipVertical | Returns an image resulting from mirroring an image around the horizontal axis |
| GaussianHighPassFilter | Returns an image resulting from the application of a Gaussian High Pass filter to an image |
| GaussianLowPassFilter | Returns an image resulting from the application of an Gaussian Low Pass filter to an image |
| HighPass | Returns an image resulting from the application of a Annular High Pass filter to an image |
| HighPassFilter | Returns an image resulting from the application of a Annular High Pass filter to an image |
| IFFT | Returns the inverse Fourier transforms of an image |
| Invert | Returns the inverse of an image |
| Laplacian | Returns the Laplacian of an image |
| Lowpass | Returns an image resulting from the application of an Annular Low Pass filter to an image |
| LowpassFilter | Returns an image resulting from the application of an Annular Low Pass filter to an image |
| Negate | Returns the inverse of an image |
| PhaseCorrelate | Returns the phase correlation between two images |

| | |
|---|---|
| PhaseCorrelation | Returns the phase correlation between two images |
| PowerSpectrum | Returns the Power Spectrum of an image |
| Ps | Returns the Power Spectrum of an image |
| RadialAverage | Returns the radial average of an image |
| RealFFT | Returns the Fourier transforms of an image |
| RemoveCCDdefects | Returns an image by adjusting for ccd defects of a recorded image |
| Repeat | Returns an image by repeating in x and y an existing image |
| Rotate | Returns an image resulting from rotating an image x degrees anti-clockwise |
| RotateLeft | Returns an image resulting from rotating an image 90 deg. Anti-clockwise |
| RotateRight | Returns an image resulting from rotating an image 90 deg. clockwise |
| Scale | Returns an image resulting from scaling an image |
| Sharpen | Returns an image resulting from applying a sharpening operation to an image |
| Shift | Returns an image resulting from shifting the origin of an exiting image |
| ShiftCenter | Returns an image resulting from shifting the origin of an exiting image |
| ShiftOrigin | Returns an image resulting from shifting the origin of an exiting image |
| Smooth | Returns an image resulting from applying a smoothing operation to an image |
| Sobel | Returns an image resulting from applying a Sobel operation to an image |
| TemplateMatch | Returns the position dependent cross-correlation coefficient between an image and a pattern |
| Wf | Returns an image resulting from applying a Wiener Filter to an image |
| WienerFilter | Returns an image resulting from applying a Wiener Filter to an image |

# Image Data Access

### Functions

| Name | Summary |
|---|---|
| GetPixel | Gets the pixel value for a given pixel |
| GetPixelAmplitude | Gets the pixel amplitude for a given pixel in a complex image |
| GetPixelPhase | Gets the pixel phase for a given pixel in a complex image |
| SetPixel | Sets the pixel value for a given pixel |
| SetPixelAmplitude | Sets the pixel amplitude for a given pixel in a complex image |
| SetPixelPhase | Sets the pixel phase for a given pixel in a complex image |
| [col,row]<br>Image dmg(256,256)<br>img[10,10] = value | Indexing into an Image pixel<br><br>Sets the pixel at [10,10] to value |

# Peak Determination

### Functions

| Name | Summary |
|---|---|
| AddPeakList | Add a peaklist to an image. The peaklist gets merged with any other peaklists for the image. |
| CreateVectorMap | Creates a vector map from two images (displacements) |

| | |
|---|---|
| FindMaxima | Finds the maxima in an image |
| FindMinima | Finds the minima in an image |
| FindPeaks | Finds the peaks in an image |
| FitDoublePeaks | Fits a peak list to a set of overlapping Gaussian peaks (two peaks are close) |
| FitExponentials | Fits the peaks in a peak list to Exponential peaks. |
| FitGaussians | Fits the peaks in a peak list to Gaussian peaks. |
| FitParabolas | Fits the peaks in a peak list to Parabolic peaks. |
| FitPeaks | Fits the peaks in a peak list to Gaussian peaks (other shapes available) |
| GetPeakList | Returns the peak list defined for an image |
| HasPeakList | Return true/false if the image has/has not an associated peak list |
| ReadPeakList | Returns a peaklist (image) from a peak list file (tab-delimited text file) |
| SavePeaks | Save the peaks in a peak list to a file. |
| SavePeaksWithDialog | Save the peaks in a peak list to a file |
| SetPeakList | Creates a peaklist for an image replacing any existing peaklist. |
| VectorMap | Creates a vector map from two sets of displacements |

# Lattice Determination

### Functions

| Name | Summary |
|---|---|
| FitLattice | Fits an existing lattice to a Peaklist |
| GetLattice | Gets the lattice defined on an image |
| HasLattice | Return true/false if the image has/has not a lattice defined |

# Vector

## Operators

| | |
|---|---|
| * | The normal arithmetic operators apply on vectors of equal size |
| / | |
| - | |
| + | |
| *= | |
| /= | |
| -= | |
| += | |

## Declaration

| | |
|---|---|
| Vector v | Declares a vector, not yet created and assigned |
| Vector c(10) | Declares and creates a vector of size 10 ( 10 elements) initialized to 0 |
| ComplexVector v | Declares a complex vector, not yet created and assigned |
| ComplexVector c(10) | Declares and creates a complex vector of size 10 (10 elements) initialized to (0,0) |

## Member functions Summary

| | |
|---|---|
| angle() | Replaces each element of the complex vector with the angle (atan2(y/x)) in degrees |
| at(i) | Returns the value of the element of the vector at the position ‘i’ |

| | |
|---|---|
| conjugate() | Replaces the complex vector with its complex conjugate |
| create(len) | Creates the vector of size len |
| imag() | Replaces the complex vector with the imaginary component |
| length() | Returns the Square root of the sum of the squares |
| modulus() | Replaces the complex vector with a real vector containing the modulus of the complex elements |
| modsq() | Replaces the complex vector with a real vector containing the modulus square of the complex elements |
| Phase() | Replaces each element of the complex vector with the angle (atan(y/x)) in radians |
| Print() | Prints out all the values of the vector |
| Real() | Replaces the complex vector with a real vector containing the real part of the complex elements |
| Resize(len) | Resizes the vector to 'len' number of elements |
| Set(ndx,value) | Sets the element of the vector at index 'ndx' to the value 'value' |
| Size() | Returns the number of elements in the vector |
| Sort(order=0) | Sorts the vector in ascending(0) (default) or descending(1) order |

Creating an Image from a Vector

```
Vector V(100)
…
```

| | |
|---|---|
| Image imag = V | Creates an Image of size V.size(),1 |

# Image Display

## Functions

| Name | Summary |
|---|---|
| Display | Shows/Displays an image |
| DisplayAsTable | Displays the image as a table of numbers |
| DisplayAt | Displays the image in a window at the given position |
| DisplayOnLogscale | Displays the image on a log scale |
| GetSurveyMode | Gets the method of survey technique for setting black and white values |
| GetSurveyTechnique | Gets the method of survey technique for setting black and white values |
| GetWindowPosition | Returns the window position of an image |
| GetWindowSize | Returns the window size for a displayed image |
| SetDisplayType | Sets the type of display for an image |
| SetSurveyMode | Sets the method of survey technique for setting black and white values |
| SetSurveyTechnique | Sets the method of survey technique for setting black and white values |
| SetWindowPosition | Sets the window position of an image |
| SetWindowSize | Sets the window size for a displayed image |
| Show | Equivalent to Display |
| ShowImage | Equivalent to Display |
| UpdateImage | Updates the display for a modified image |

# Image Selections

### Functions

| Name | Summary |
|---|---|
| ExpandSelection | Expands a given selection |
| GetSelection | Gets the rectangle of the |

| | |
|---|---|
| | selection |
| SetSelection | Sets the rectangle of the selection |

# Annotations

## Functions

| Name | Summary |
|---|---|
| AnnotationType | Returns the type of a given annotation |
| CountAnnotations | Returns the number of annotations on the image |
| CreateArrowAnnotation | Creates an arrow annotation |
| CreateBoxAnnotation | Creates a rectangular annotation |
| CreateDoubleArrowAnnotation | Creates a double arrow annotation |
| CreateLineAnnotation | Creates a line annotation |
| CreateOvalAnnotation | Creates an oval annotation |
| CreateTextAnnotation | Creates an text annotation |
| DeleteAnnotation | Deletes a given annotation |
| DeselectAnnotation | Deselects an annotation |
| GetAnnotationRect | Gets the bounding rectangle of a given annotation |
| GetNthAnnotationID | Gets the ID of an annotation |
| IsAnnotationSelected | Determines if an annotation is selected |
| MoveAnnotation | Moves the annotation to a given position |
| OffsetAnnotation | Offsets the annotation with specified integer offsets |
| SelectAnnotation | Selects the specified annotation |
| SetAnnotationBackground* | Sets the background of an annotation |
| SetAnnotationColor | Sets the Color of an annotation |
| SetAnnotationFace* | Sets the text face of an annotation |
| SetAnnotationFont | Sets the text font of an annotation |
| SetAnnotationJustification* | Sets the text justification of an annotation |
| SetAnnotationRect | Sets the bounding rectangle |

| | |
|---|---|
| | of an annotation |
| SetAnnotationSize | Sets the size of an annotation |
| ShiftAnnotation | Shifts the position of an annotation. Equivalent to MoveAnnotation |
| ValidAnnotation | Is the annotation valid |

*Not yet implemented

# Strings

## Operators

| Name | Summary |
|---|---|
| != | Inequality operator for strings |
| + | Concatenate a string and a real number |
| + | Concatenate a string and a complex number |
| + | Concatenate a complex number and a string |
| + | Concatenate a real number and a string |
| + | Concatenate a string and a string |
| == | Equality operator for strings |

## Functions

| Name | Summary |
|---|---|
| Asc* | Returns numeric value in ascii |
| Chr* | Returns ascii equivalent of a number as a string |
| Left* | Returns the leftmost portion of a string |
| Len* | Returns the length of a string |
| mid* | Returns the middle portion of a string |
| right* | Returns the rightmost portion of a string |

| | |
|---|---|
| val* | Converts a string to a real number |

*Not yet implemented

# Persistent Notes (mostly not implemented)

| Name | Summary |
|---|---|
| DeletePersistentNote* | Deletes persistent note |
| GetPersistentComplexNumberNote | Gets the value of a persistent complex number note |
| GetPersistentNoteState* | Gets persistent note state |
| GetPersistentNumberNote | Gets the value of a persistent number note |
| GetPersistentRectNote | Gets the value of a persistent rect note |
| GetPersistentRGBNumberNote* | Gets the value of a persistent RGB number note |
| GetPersistentStringNote* | Gets the value of a persistent string note |
| GetPersistentStringNote* | Gets the value of a persistent string note |
| SetPersistentComplexNumberNote | Sets the value of a persistent complex number note |
| SetPersistentKeywordNote* | Adds a persistent keyword note |
| SetPersistentNoteState* | Sets persistent note state |
| SetPersistentNumberNote | Sets the value of a persistent number note |
| SetPersistentRectNote | Sets the value of a persistent rect note |
| SetPersistentRGBNumberNote* | Sets the value of a persistent RGB number note |
| SetPersistentStringNote* | Sets the value of a persistent string note |

*Not yet implemented

# Number Conversions

| Name | Summary |
|---|---|
| BaseN* | Convert a number to an arbitrary base string |
| BaseN* | Convert a number to an arbitrary base string with a fixed length |
| Binary* | Convert a number to a binary string with a fixed length |
| Binary* | Convert a number to a binary string |
| Decimal* | Convert a number to a decimal string |
| Decimal* | Convert a number to a decimal string with a fixed length |
| Hex* | Convert a number to a hex string with a fixed length |
| Hex* | Convert a number to a hex string |
| Octal* | Convert a number to an octal string with a fixed length |
| Octal* | Convert a number to an octal string |

*Not yet implemented

# Dialogs

| Name | Summary |
|---|---|
| ContinueCancelDialog | Puts up a dialog with the option to cancel or continue the script |
| ErrorDialog | Puts up a dialog with an error string |
| GetNumber | Prompts for a number to input |
| GetTwoImages | Prompts for two images to select |
| GetTwoImagesWithPrompt | Prompts for two images to |

| | |
|---|---|
| | select |
| OkCancelDialog | Puts up a dialog with the option to cancel or continue the script |
| OkDialog | Puts up a dialog with the option to accept or not a choice |
| TwoButtonDialog | Puts up a dialog with two buttons to choose from |

# Input/Output

| Name | Summary |
|---|---|
| OpenLogWindow | Opens the log/output window for scripts |
| OpenResultsWindow | Opens the log/output window for scripts (for DM compatibility) |
| Print | Prints (writes) an expression to the output window. By default adds a new line character at the end |
| Result | Prints (writes) an expression to the output window. DM compatible |

# Movies

| Name | Summary |
|---|---|
| AddImageToMovie | Adds an image to a movie |
| AddWindowToMovie | Adds a window (containing an image) to a movie |
| CloseMovie | Closes the movie |
| CreateNewMovie | Creates a new movie with a given name |

**Example:**

image img = exprsize(256,256,icol)

```
number i
createnewmovie("movie")
for(i=0; i < 256; i++) {
        addimagetomovie(img)
        img.shift(1,0) ;
}
closemovie()
```

# Miscellaneous

| Name | Summary |
|---|---|
| Catch | catch an exception thrown after a *try* statement |
| CloseProgressWindow* | *Not Yet Implemented* |
| CommandDown | Returns true/false depending on if the Command (Apple) key is down or not |
| DateStamp | Not Yet Implemented |
| Delay | Delay execution of the script x number of 1/60th of a second |
| DoEvents | Checks for input from the keyboard |
| Exit | Exit from the script |
| GetKey | Returns the key currently pressed |
| Help | Gets help on a given function |
| OpenAndSetProgressWindow* | *Not Yet Implemented* |
| OptionDown | Returns true/false depending on if the Option key is down or not |
| ShiftDown | Returns true/false depending on if the Shift key is down or not |
| SpaceDown | Returns true/false depending on if the Space bar is down or not |
| Throw | Throw an exception |
| ThrowString | Throw an exception with a string |
| Try | Try to execute the following bracketed statements. Check for an exception by using the *catch* statement |

*Not yet implemented

# Electron Microscopy Simulation Script Functions

### General Calculation Functions

| Name | Summary |
|---|---|
| CalculateAtomicScatteringFactors | Calculates the atomic scattering factors for a given atomic element and places them in a file |
| CalculatExitWave | Calculates the exit wave for the simulation currently open |
| CalculateImage | Calculates the image for the simulation currently open |
| CalculatePotential | Calculates the potential for the simulation currently open |
| CalculateImageFromWave | Calculates an image from a complex exit wave, given a microscope |
| CalculateLinearImageFromWave | Calculates a linear image from a complex exit wave, given a microscope |
| ApplyFocusPlate | Applies a focus plate (focus given in an image) to a complex wavefunction |
| ShiftImageFocus | Shifts the focus of a given complex wavefunction |
| PropagateWave | Calculates a 3D Complex Volume Image containing the electron wavefunction at each slice in the multislice calculation up to a given thickness |

# Microscope Data Type

### Declaration

```
Microscope mic
```

### Initializing

| Name | Summary |
|---|---|

```
Microscope mic                  Defines a default microscope
=                               Equates a microscope to
                                another microscope

Example:

Microscope mic
mic.setvoltage(300)


Example:

Simulation ss
ss = GetSimulation()            Get the Current Simulation
                                See data type below
ss = GetSimulation(String)      Get the named Simulation

Microscope mic                  Sets a default Microscope
mic = ss.GetMicroscope()        Assigns to simulation
                                Microscope
```

## Microscope Class Member Functions

| Name | Summary |
|---|---|
| GetAperture | Returns the Aperture of the objective lens in 1/Å |
| GetApertureH | Returns the horizontal Center of Objective Lens Aperture in units of h of reciprocal space |
| GetApertureHK | Returns the Center of Objective Lens Aperture in units of h and k of reciprocal space |
| GetApertureK | Returns the Vertical Center of Objective Lens Aperture in units of k of reciprocal space |
| GetCs | Returns the Cs in mm of the objective lens |
| GetCs5 | Returns the Cs5 in mm of the objective lens |
| GetDelta | Returns the Cs in mm of the objective lens |
| GetDivergence | Returns the Cs in mm of the objective lens |
| GetFocus | Returns the focus in Å of the objective lens |
| GetFocusSpread | Returns the spread in focus |

| | |
|---|---|
| | in Å of the objective lens |
| GetVoltage | Returns the Cs in mm of the objective lens |
| Print() | Prints out a summary of the microscope parameters |
| SetAperture | Sets the Aperture of the objective lens in 1/Å |
| SetApertureH | Sets the horizontal Center of Objective Lens Aperture in units of h of reciprocal space |
| SetApertureHK | Sets the Center of Objective Lens Aperture in units of h and k of reciprocal space |
| SetApertureK | Sets the Vertical Center of Objective Lens Aperture in units of k of reciprocal space |
| SetCs | Sets the Cs in mm of the objective lens |
| SetCs5 | Sets the Cs5 in mm of the objective lens |
| SetDelta | Sets the Cs in mm of the objective lens |
| SetDivergence | Sets the Cs in mm of the objective lens |
| SetFocus | Sets the focus in Å of the objective lens |
| SetFocusSpread | Sets the spread in focus in Å of the objective lens |
| SetVoltage | Sets the Cs in mm of the objective lens |

# Simulation Data Type

## Declaration

```
Simulation ss
```

## Initializing

| Name | Summary |
|---|---|
| GetSimulation | Gets the current simulation |
| OpenSimulation | Sets the current simulation from an existing structure file |

```
Example:
simulation ss
ss = GetSimulation()          Gets current simulation
ss = GetSimulation("bcsco")   Gets the open "bcsco"
                              simulation

Example:
simulation ss = opensimulation("bcsco.at")
```

## Simulation Class Member Functions

| Name | Summary |
|---|---|
| CalculateAll | (Re)Calculates the Potential(s), Exit Wave(s) and Image(s). |
| Calculate3DPotential | Calculates the 3D potential for the unit cell of the current simulation |
| CalculateExitWave | Calculates the Exit Wave |
| CalculateImage | Calculates the Image |
| CalculatePotential | Calculates the Potential |
| CreateFrequencyImage | Returns a square image of a simulated object in reciprocal space. |
| CreateImage | Returns a square image from a given calculated image of given size and sampling |
| DisplayExitWave | Displays a given calculated exit wave for the simulation |
| DisplayExitWaveModulus | Displays the modulus of the exit wave |
| DisplayExitWavePhase | Displays the phase of the exit wave for the simulation |
| DisplayImage | Displays a given image for the simulation |
| DisplayPotential | Displays the calculated potential for the simulation |
| Focus | Sets the focus of the simulation |
| GetAperture | Returns the outer objective lens aperture (1/Å) |
| GetApertureAngle | Returns the outer objective lens aperture in mradians |
| GetApertureCenter | Returns the center of the objective lens aperture |
| GetApertureCenterHK | Returns the center of the objective lens aperture in (H,K) of the reciprocal space |

| | |
|---|---|
| | of the unit cell |
| GetCs | Returns the Spherical Aberration Cs in mm |
| GetCs5 | Returns the 5$^{th}$ order Spherical Aberration Cs5 in mm |
| GetDeltaFocus | Returns the increment in focus for the calculation |
| GetDeltaThickness | Returns the increment in thickness for the calculation |
| GetDivergence | Returns the convergence angle (mrad) for the calculation |
| GetEndFocus | Returns the ending value for focus |
| GetEndThickness | Returns the ending value for thickness |
| GetExitWave | Returns an image containing a given number of unit cells of the exit wave of the calculation |
| GetExitWaveModulus | Returns an image containing a given number of unit cells of the modulus of the exit wave |
| GetExitWavePhase | Returns an image containing a given number of unit cells of the phase of the exit wave |
| GetFocus | Returns the focus (Å) for the calculation |
| GetFocusSpread | Returns the focus Spread (Å) for the calculation |
| GetImage | Returns an image containing the a given number of unit cells of calculated simulated image |
| GetInnerAperture | Returns the inner objective lens aperture (1/Å) |
| GetOpticAxis | Returns the center of the optic axis in tilt angle (mrad) and azimuthal angle (degrees) |
| GetOpticAxisHK | Returns the center of the optic axis in (H,K) of the reciprocal space of the unit cell |
| GetOuterAperture | Returns the outer objective lens aperture (1/Å) |
| GetPhaseShift | Returns the phase shift for the phase plate in units of $\pi$ |
| GetPhaseShiftRadius | Returns the radius for the phase plate in units of 1/Å |
| GetPhaseShiftRadius2 | Returns the outer radius for the phase plate in units of 1/Å. Beams are blocked |

| | |
|---|---|
| | between PhaseShiftRadius and PhaseShiftRadius2 if they are different |
| GetPotential | Returns an image containing a given number of unit cells of the calculated potential |
| GetStartFocus | Returns the starting focus (Å) for a thru-focus series |
| GetStartThickness | Returns the starting thickness (Å) for a thru-thickness series |
| GetThickness | Returns the thickness (Å) for the simulation |
| GetTilt | Returns the tilt angle of the specimen in mrad and the azimuthal angle of specimen tilt with respect to the horizontal axis in degrees |
| GetTiltAngle | Returns the tilt angle of the specimen in mrad |
| GetTiltDirection | Returns the azimuthal angle of specimen tilt with respect to the horizontal axis in degrees |
| GetTiltH | Gets the h value of the center of laue circle (specimen tilt) |
| GetTiltHK | Gets the h,k values of the center of laue circle (specimen tilt) |
| GetTiltK | Gets the k value of the center of laue circle (specimen tilt) |
| GetVibration | Gets the vibration of the “specimen” along x and y |
| GetVibrationX | Gets the vibration of the “specimen” along x |
| GetVibrationY | Gets the vibration of the “specimen” along y |
| GetVoltage | Returns the voltage of the microscope for the simulation (kV) |
| LoadExitWave | Loads a 1 by 1 unit cell of the Exit Wave as calculated and returns it as an image |
| LoadExitWaveModulus | Loads a 1 by 1 unit cell of the Exit Wave modulus as calculated and returns it as an image |
| LoadExitWavePhase | Loads a 1 by 1 unit cell of the Exit Wave Phase as calculated and returns it as an image |
| LoadImage | Loads a 1 by 1 unit cell of |

| | |
|---|---|
| | the Image as calculated and returns it as an image |
| LoadPotential | Loads a 1 by 1 unit cell of the Potential as calculated and returns it as an image |
| PropagateWave | Calculates a 3D Complex Volume Image containing the electron wavefunction at each slice in the multislice calculation up to a given thickness |
| SetAperture | Sets the outer objective lens aperture (1/Å) |
| SetApertureAngle | Sets the outer objective lens aperture in mradians |
| SetApertureCenter | Sets the center of the objective lens aperture |
| SetApertureHK | Sets the center of the objective lens aperture in (H,K) of the reciprocal space of the unit cell |
| SetCs | Sets the Spherical Aberration Cs in mm |
| SetCs5 | Sets the $5^{th}$ order Spherical Aberration Cs5 in mm |
| SetDeltaFocus | Sets the Incremental focus (Å) for a thru-focus series |
| SetDeltaThickness | Sets the incremental thickness (Å) for a thru-thickness series |
| SetDivergence | Sets the convergence angle (mrad) for the calculation |
| SetEndFocus | Sets the ending value for focus [Å] in a thru-focus series |
| SetEndThickness | Sets the ending value for thickness [Å] in a thru-thickness series |
| SetFocus | Sets the focus (Å) for the calculation |
| SetFocusSpread | Sets the focus Spread (Å) for the calculation |
| SetInnerAperture | Sets the inner objective lens aperture (1/Å) |
| SetOpticAxis | Sets the center of the optic axis in tilt angle (mrad) and azimuthal angle (degrees) |
| SetOpticAxisHK | Sets the center of the optic axis in (H,K) of the reciprocal space of the unit cell |
| SetOuterAperture | Sets the outer objective lens aperture (1/Å) |

| | |
|---|---|
| SetPhaseShift | Sets the phase shift for the phase plate in units of $\pi$ |
| GetPhaseShiftRadius | Sets the radius for the phase plate in units of 1/Å |
| GetPhaseShiftRadius2 | Sets the outer radius for the phase plate in units of 1/Å. Beams are blocked between PhaseShiftRadius and PhaseShiftRadius2 if they are different |
| SetStartFocus | Sets the starting focus (Å) for a thru-focus series |
| SetStartThickness | Sets the starting thickness for a thru-thickness series |
| SetThickness | Sets the thickness (Å) for the calculation |
| SetTiltAngle | Sets the tilt angle of the specimen in mrad |
| SetTiltDirection | Sets the azimuthal angle of specimen tilt with respect to the horizontal axis in degrees |
| SetTiltH | Sets the h value of the center of laue circle (specimen tilt) |
| SetTiltHK | Sets the h,k values of the center of laue circle (specimen tilt) |
| SetTiltK | Sets the k value of the center of laue circle (specimen tilt) |
| SetVibration | Sets the vibration of the "specimen" along x and y |
| SetVibrationX | Sets the vibration of the "specimen" along x |
| SetVibrationY | Gets the vibration of the "specimen" along y |
| SetVoltage | Sets the voltage of the microscope for the simulation (kV) |
| Thickness | Returns the thickness for the calculation |

**Example:**

```
Simulation ss = getsimulation()
Number cs = 0.5  // Cs in mm
Number voltage = 300       // voltage in kV
ss.setcs(cs)
ss.setvoltage(voltage)

// Assume that the potential has already been calculated
```

```
// as our changes only require the exit wave(s) (change in wavelength)
// and the image(s) to be recalculated

ss.showpotential(1,5,5)

ss.calculateexitwave()
ss.calculateimage()

// Display the exit wave. The first of whatever number calculated
// 5 by 5 unit cell
image xw = ss.getexitwave(1,5,5)  //
xw.phase()
xw.setname("Phase of exit wave")
xw.show()

// Show the image. The first of whatever number calculated
// 5 by 5 unit cell
image img = ss.getimage(1,5,5)
img.setname("Calculated Image")
img.show()
```

**Example:**

```
Simulation ss = getsimulation()          // Get current simulation
image3d test                             // Declare a 3D volume image
ss.calculate3dpotential(test)            // Calculate the 3D potential
test.display()                           // into test and display
                                         // Image sections are traversed
                                         // using the arrow keys
```

**Example:**

```
Simulation ss = getsimulation()          // Get current simulation
image tt = ss.loadimage()                // Load the image
tt.fft()                                 // Fourier transform
image dp = ss.createfrequencyimage(tt)   // Create image of fourier transform
                                         // with default size 512
dp.show()                                // Show the frequency image

tt = ss.loadexitwave()                   // Do the same for the diffraction
                                         // pattern
tt.fft()
image dp2 = ss.createfrequencyimage(tt)
dp2.show()
```

# Alphabetical description of general script functions and class member functions

## abs

SUMMARY Calculates the absolute value of a real/complex number or the absolute values of a real/complex image

SYNTAX number abs( number )

SYNTAX number abs( complexnumber )

SYNTAX image abs( image )

SYNTAX image abs( compleximage )

SYNTAX void image.abs() Image member function

DESCRIPTION Calculates the absolute value of a complex number or image. (also known as the modulus of a complex number) Calculates the absolute value(s) of a real number of real image

## ac

SUMMARY Calculates the autocorrelation function of a real image

SYNTAX image ac( image )

SYNTAX void image.ac() Image member function

## acos

SUMMARY Calculates the arccosine of a real number or a real image

SYNTAX number acos( number )

SYNTAX image acos( image )

SYNTAX void image.acos() Image member function

## acosh

| | |
|---|---|
| SUMMARY | Calculates the hyperbolic arccosine of a real number or a real image |
| SYNTAX | number acosh( number ) |
| SYNTAX | image acosh( image ) |
| SYNTAX | void image.acosh() Image member function |

## AddImage

| | |
|---|---|
| SUMMARY | Adds an image to an image stack |
| SYNTAX | void imagestack.addimage(image) Image stack member function |

## AddImageToMovie

| | |
|---|---|
| SUMMARY | Adds an image to an existing open movie |
| SYNTAX | void AddImageToMovie (image) |
| DESCRIPTION | Adds an image to an existing open movie. |

## AddPeakList

| | |
|---|---|
| SUMMARY | Add a peaklist to an image merging with an existing peaklist (if any). |
| SYNTAX | void AddPeakList(image theImage, image peaklist) |
| DESCRIPTION | After reading in a peaklist from a file or getting the peaklist from an image, this peaklist can be added to an existing image. The peaklist merges with any existing peaklist associated with the image. The dimensions of the image to be associated the peaklist must be of the same dimensions as the image from which the peaklist originated for this to make sense. |

## AddWindowToMovie

| | |
|---|---|
| SUMMARY | Adds an image to an existing open movie |

SYNTAX void AddWindowToMovie (image)

DESCRIPTION Adds a window (referenced by a displayed image) to an existing open movie.

## AdjustAngle

SUMMARY Adjusts the angle of the image if different from 90

SYNTAX void image.AdjustAngle() Image member function

DESCRIPTION Adjusts the image so that it has an angle of 90 degrees. This is applicable for images returned from a simulation. In this case the image represents a periodic object and the angle of the unit cell may be different from 90 deg.

## AdjustSampling

SUMMARY Adds an image to an image stack

SYNTAX void image.AdjustSampling(image) Image member function

DESCRIPTION Adjusts the image so that it has equal sampling in x and y. This is applicable for images returned from a simulation. In this case the image represents a periodic object and the sampling along the a and b axes may be different.

## AiryAi

SUMMARY Calculates the Airy Ai function

SYNTAX number AiryAi( number )

DESCRIPTION *Not Implemented

## AiryBi

SUMMARY Calculates the Airy Bi function

SYNTAX number AiryBi( number )

DESCRIPTION

## Align

SUMMARY Aligns two images

SYNTAX complexnumber align(image x, image y [, number method ] [, number freqCutoff [, number focusShift] [,number voltage] )

DESCRIPTION Aligns image y with image x using either crosscorrelation or phasecorrelation. Only argument 1 and 2 are required. The others are optional. Default values are method = 0 (crosscorrelation = 0, phasecorrelation =1), freqCutoff = 0.3*maxFrequency, focusShift = 0[Å], voltage = 300 [kV] Returns the shift used to translate image y in a complex number

## AlignImages

SUMMARY Aligns two images

SYNTAX AlignImages (image x, image y [, number method ] [, number freqCutoff [, number focusShift] [,number voltage] )

DESCRIPTION Equivalent to *Align* . Aligns image y with image x using either crosscorrelation or phasecorrelation. Only argument 1 and 2 are required. The others are optional. Default values are method = 0 (crosscorrelation = 0, phasecorrelation =1), freqCutoff = 0.3*maxFrequency, focusShift = 0[Å], voltage = 300 [kV]

## AlignTwoImages

SUMMARY Aligns two images

SYNTAX complexnumber AlignTwoImages(image x, image y [, number method ] [, number freqCutoff [, number focusShift] [,number voltage] )

DESCRIPTION Equivalent to *Align*. Aligns image y with image x using either crosscorrelation or phasecorrelation. Only argument 1 and 2 are required. The others are optional. Default values are method = 0 (crosscorrelation = 0, phasecorrelation =1), freqCutoff = 0.3*maxFrequency, focusShift = 0[Å],

voltage = 300 [kV]  Returns the shift used to translate image y in a complex number

## Amplitude

| | |
|---|---|
| SUMMARY | Returns the modulus of a complex number/image/ image3D as a real number/image |
| SYNTAX | number amplitude( complexnumber ) |
| SYNTAX | image amplitude( compleximage ) |
| SYNTAX | void image.amplitude() // Class Member Function |
| SYNTAX | void image3D.amplitude()// Class Member Function |

## AnalyzeDiffractogram*

| | |
|---|---|
| SUMMARY | Analyzes a diffractogram |
| SYNTAX | void AnalyzeDiffractogram( image , numbervariable defocus, numbervariable direction, numbervariable err) |
| DESCRIPTION | *Not Implemented  -- Analyze diffractogram in image. Returned defocus, astigmatism, and err are in nm |

## Angle

| | |
|---|---|
| SUMMARY | Returns the phase of a complex number in degrees |
| SYNTAX | number complexnumber.angle() |

## AnnotationType

| | |
|---|---|
| SUMMARY | Returns the type of an annotation |
| SYNTAX | number  AnnotationType ( image , number annotationID) |
| DESCRIPTION | Returns the type of the annotation specified in the given image with the given index. |

## AnnularHighpassFilter

SUMMARY — Applies a high pass filter to an image

SYNTAX — image AnnularHighPassFilter( image , number cutoff [, number edgewidth ] )

SYNTAX — void image.AnnularHighPassFilter(number cutoff [, number edgewidth ] ) // Image member function

If the image is calibrated in Å, the cutoff is in 1/Å

DESCRIPTION — edgewidth by default is set to 0 and represents a soft edge

## AnnularLowpassFilter

SUMMARY — Applies a low pass filter to an image

SYNTAX — image AnnularLowPassFilter( image , number cutoff [, number edgewidth ] )

SYNTAX — void image.AnnularLowPassFilter(number cutoff [, number edgewidth ] ) // Image member function

If the image is calibrated in Å, the cutoff is in 1/Å

DESCRIPTION — edgewidth by default is set to 0 and represents a soft edge

## ApplyAnnularMask

SUMMARY — Applies an annular mask to an image

SYNTAX — image ApplyAnnularMask(image , number r1, number r2 [, number edgewidth] [, number isopaque])

SYNTAX — void image.ApplyAnnularMask(number r1, number r2 [, number edgewidth] [, number isopaque]) // Image member function

If the image is calibrated in Å, the values r1 and r2 are in 1/Å

DESCRIPTION Annular mask of inner radius r1 and outer radius r2. By default the width of the edge = 0 and by default isopaque = false

## ApplyCircularMask

SUMMARY Returns an image resulting from the application of an annular mask to an image

SYNTAX void ApplyCircularMask (image , number r, [, number edgewidth] [, number isopaque]) // In place operation

If the image is calibrated in Å, the values r is in 1/Å

SYNTAX void image.ApplyCircularMask ([ number edgewidth] [, number isopaque]) // In place operation

DESCRIPTION Circular mask of radius r. By default the width of the edge = 0 and by default isopaque = false

## ApplyCosineMask

SUMMARY Returns an image resulting from the application of a circular cosine mask to an image

SYNTAX void ApplyCosineMask( image) // In place operation

SYNTAX void image.ApplyCosineMask() // Member function

## ApplyFocusPlate

SUMMARY Returns an image resulting from the application of a arbitrary focus plate to a complex image or wave function

SYNTAX ComplexImage ApplyFocusPlate( compleximage source, image focus [, number voltage = 300] [ , number sampling = 0.2])

SYNTAX void ApplyFocusPlate(image focus [, number voltage = 300] [ , number sampling = 0.2]) // Image member function

DESCRIPTION The focus variation (or constant) is given in the image focus. The complex image is propagated over the distance focus (which can vary as a function of

position). By default the voltage is 300kV. If the source is calibrated in Ångstrom or nanometer, the sampling is taken from the source. Otherwise the default is 0.2 Å/pixel and must be set if different.

## ApplyHanningMask

SUMMARY Returns an image resulting from the application of a circular hanning mask to an image

SYNTAX void ApplyHanningMask( image) // In place operation

SYNTAX void image.ApplyHanningMask() // Member function

## asin

SUMMARY Calculates the arcsine of a real number or a real image

SYNTAX number asin( number )

SYNTAX image asin( image )

SYNTAX void image.asin() Image member function

## asinh

SUMMARY Calculates the hyperbolic arcsine of a real number or a real image

SYNTAX number asinh( number )

SYNTAX image asinh( image )

SYNTAX void image.asinh() Image member function

## atan2

SUMMARY Calculates the arctangent of y/x for real numbers, real images of a complex image

SYNTAX number atan2( number x, number y) // atan(y/x)

| | |
|---|---|
| SYNTAX | image atan2( image x, image y) |
| SYNTAX | void image.atan2() Image member function |

## atanh

| | |
|---|---|
| SUMMARY | Calculates the hyperbolic arctangent of a real number or a real image |
| SYNTAX | number atanh( number ) |
| SYNTAX | image atanh( image ) |
| SYNTAX | void image.atanh() Image member function |

## Autocorrelate

| | |
|---|---|
| SUMMARY | Calculates the autocorrelation function of a real image |
| SYNTAX | image autocorrelate( image ) |
| SYNTAX | void image.autocorrelation() Image member function |

## BeginFill**

| | |
|---|---|
| SUMMARY | Starts a fill from projections |
| SYNTAX | void image.beginfill() // image class member function |
| SYNTAX | void image3D.beginfill() // image3d class member function |
| DESCRIPTION | BeginFill and EndFill must bracket the filling from projections. |

## BesselI

| | |
|---|---|
| SUMMARY | Calculates the general Bessel I function |
| SYNTAX | number BesselI( number, number ) |
| DESCRIPTION | |

## BesselJ

SUMMARY Calculates the general Bessel J function

SYNTAX number BesselJ( number, number )

DESCRIPTION

## BesselK

SUMMARY Calculates the general Bessel K function

SYNTAX number BesselK( number, number )

DESCRIPTION

## BesselY

SUMMARY Calculates the general Bessel Y function

SYNTAX number BesselY( number, number )

DESCRIPTION

## Beta

SUMMARY Calculates the beta function

SYNTAX number Beta ( number, number )

DESCRIPTION

## bgs

SUMMARY Applies a Background Noise Subtraction Filter on a real image

SYNTAX void image.bgs()

DESCRIPTION Attempts to subtract out an amorphous background from an image containing crystalline material. In general the WienerFilter is a safer filter from a statistical point.

## BinomialCoefficient

| | |
|---|---|
| SUMMARY | Calculates the binomial coefficient |
| SYNTAX | number BinomialCoefficient ( number, number ) |
| DESCRIPTION | |

## BinomialRandom*

| | |
|---|---|
| SUMMARY | Calculates a random number with binomial distribution |
| SYNTAX | number BinomialRandom ( number, number ) |
| DESCRIPTION | *Not Implemented |

## cc

| | |
|---|---|
| SUMMARY | Returns an image resulting from the cross-correlation of two images |
| SYNTAX | image cc( image x, image y) |

## ccd

| | |
|---|---|
| SUMMARY | Corrects for CCD detector bad pixels in a real image |
| SYNTAX | image ccd( image) |
| SYNTAX | void image.ccd() |
| DESCRIPTION | Attempts to locate pixels that correspond to bad pixels in the CCD camera. The pixel values fall out of the normal range and are substituted by mean values of the neighborhood |

## ceiling

| | |
|---|---|
| SUMMARY | Limits all values of a real image to a given maximum value |

SYNTAX image ceiling( image , number)

SYNTAX void image.ceiling(number) Image member function

## cis

SUMMARY Calculates a unit vector in the complex plane

SYNTAX compleximage cis( image x, image y)

DESCRIPTION Returns the complex image (cos(x) , sin(x))

## clip

SUMMARY Limits all values of a real image to given minimum and maximum values

SYNTAX image clip( image , number min, number max)

SYNTAX void image.clip(number min, number max) Image member function

## CloseMovie

SUMMARY Closes an open movie

SYNTAX void CloseMovie ()

## complex

SUMMARY Creates a complex number/image from two real numbers/images

SYNTAX complexnumber complex( number x, number y) // x +iy

SYNTAX compleximage complex( image x, image y) // x + iy

## complexconjugate

SUMMARY Returns the complex conjugate of a complex number/ image

SYNTAX complexnumber complexconjugate (complexnumber)

SYNTAX compleximage complexconjugate (compleximage)

## ComplexModulusSq / cmsq

SUMMARY Returns the complex modulus square of a complex image.

SYNTAX compleximage ComplexModulusSq(compleximage)

SYNTAX void image.ComplexModulusSq() // Member Function

SYNTAX void image.cmsq() // Member Function

DESCRIPTION This replaces a complex image with the product of itself and its complex conjugate. It is the complex modulus square. Imaginary part is zero

## conjugate

SUMMARY Returns the complex conjugate of a complex number/ image

SYNTAX complexnumber conjugate(complexnumber)

SYNTAX compleximage conjugate(compleximage)

SYNTAX void image.conjugate(number) Image member function

SYNTAX number complexnumber.conjugate() complex number member function

## ContinueCancelDialog

SUMMARY Continue cancel dialog

SYNTAX Boolean ContinueCancelDialog( String prompt )

DESCRIPTION Puts up a dialog with both a Continue button and Cancel button. Returns true for Continue and false for Cancel.

## Convolve

SUMMARY Returns the convolution of two images

SYNTAX image convolve(image x, image y)

DESCRIPTION Equivalent to Convolute

## Convolute

SUMMARY Returns the convolution function of two images

SYNTAX image convolve(image x, image y)

DESCRIPTION Equivalent to Convolve

## Correlate

SUMMARY Returns the correlation function of two images

SYNTAX image correlate(image x, image y)

DESCRIPTION

## cos

SUMMARY Calculates the cosine of a real number or a real image

SYNTAX number cos( number )

SYNTAX image cos( image )

SYNTAX void image.cos() Image member function

## cosh

SUMMARY Calculates the hyperbolic cosine of a real number or a real image

SYNTAX number cosh( number )

SYNTAX image cosh( image )

SYNTAX void image.cosh() Image member function

## CountAnnotations

SUMMARY Returns the number of annotations in an image

SYNTAX Number CountAnnotations( Image )

DESCRIPTION Returns the number of annotations contained in the image as a number.

## CreateArrowAnnotation

SUMMARY Creates an arrow annotation

SYNTAX Number CreateArrowAnnotation( Image, Number top, Number left, Number bottom, Number right )

DESCRIPTION Creates an arrow annotation in the given image with the given endpoints. Returns the ID to the new annotation as a number.

## CreateDoubleArrowAnnotation

SUMMARY Creates a double arrow annotation

SYNTAX Number CreateDoubleArrowAnnotation( Image, Number top, Number left, Number bottom, Number right )

DESCRIPTION Creates a double arrow annotation in the given image with the given endpoints. Returns the ID to the new annotation as a number.

## CreateComplexImage

SUMMARY Creates a complex image of a given size

SYNTAX image CreateComplexImage(string name, number width, number height)

DESCRIPTION Returns a Complex image with the given name and dimensions

## CreateFloatImage

| | |
|---|---|
| SUMMARY | Creates a real image of a given size |
| SYNTAX | image CreateFloatImage(string name, number width, number height) |
| SYNTAX | image CreateFloatImage(number width, number height) |
| DESCRIPTION | Returns a real image with the given name and dimensions. Equivalent to “CreateImage”, “RealImage” and “NewImage” |

## CreateImage

| | |
|---|---|
| SUMMARY | Creates a real image of a given size |
| SYNTAX | image CreateImage(string name, number width, number height) |
| SYNTAX | image CreateImage(number width, number height) |
| DESCRIPTION | Returns a real image with the given name and dimensions. Equivalent to “CreateFloatImage”, “RealImage” and “NewImage” |

## CreateLineAnnotation

| | |
|---|---|
| SUMMARY | Creates a line annotation |
| SYNTAX | Number CreateLineAnnotation( Image, Number top, Number left, Number bottom, Number right ) |
| DESCRIPTION | Creates a line annotation in the given image with the given endpoints. Returns the ID to the new annotation as a number. |

## CreateNewMovie

| | |
|---|---|
| SUMMARY | Creates and opens a movie |
| SYNTAX | void CreateNewMovie (string movieName) |

## CreateOvalAnnotation

SUMMARY — Creates an oval annotation

SYNTAX — Number CreateOvalAnnotation( Image, Number top, Number left, Number bottom, Number right )

DESCRIPTION — Creates an oval annotation in the given image with the given coordinates. Returns the ID to the new annotation as a number.

## CreateTableFromImage

SUMMARY — Displays the image as a table of numbers

SYNTAX — void CreateTableFromImage (image)

DESCRIPTION — Will create a table representing the content of an image. Equivalent to “DisplayAsTable”

## CreateTextAnnotation

SUMMARY — Creates a text annotation

SYNTAX — Number CreateTextAnnotation( Image, Number top, Number left, String text )

DESCRIPTION — Creates a text annotation in the given image in the box specified by the coordinates. Returns the ID to the new annotation as a number.

## CreateVectorMap

SUMMARY — Creates a vector map from two images

SYNTAX — void CreateVectorMap(image x, image y [, number samplingX ] [, number samplingY] [, number scale])

DESCRIPTION — Creates and displays a vector map from two images x and y which correspond to the x and y components of the vectors. Vectors will be created every samplingX (default=16) pixels and samplingY (default=16) pixels. Vectors are drawn with the magnification factor: scale (default=10)

## CrossCorrelate

| | |
|---|---|
| SUMMARY | Returns the correlation function of two images |
| SYNTAX | image crosscorrelate(image x, image y) |
| DESCRIPTION | Equivalent to cc, correlate and crossscorrelation |

## CrossCorrelation

| | |
|---|---|
| SUMMARY | Returns the correlation function of two images |
| SYNTAX | image crosscorrelate(image x, image y) |
| DESCRIPTION | Equivalent to cc, correlate and crossscorrelate |

## CrossProduct

| | |
|---|---|
| SUMMARY | Calculates the cross product |
| SYNTAX | RealImage CrossProduct( RealImage a, RealImage b ) |
| DESCRIPTION | Calculates the cross product of two 3 element images. |

## DateStamp

| | |
|---|---|
| SUMMARY | Returns date and time |
| SYNTAX | String DateStamp( void ) |
| DESCRIPTION | Returns a string representing the current date and time. |

## Delay

| | |
|---|---|
| SUMMARY | Delay execution of the script x number of 1/60$^{th}$ of a second |
| SYNTAX | void Delay ( number ) |

## DeleteAnnotation

| | |
|---|---|
| SUMMARY | Deletes an annotation |
| SYNTAX | void DeleteAnnotation( Image, Number annotationID ) |
| DESCRIPTION | Deletes the annotation specified by the annotation ID in the given image. |

## DeleteImage

| | |
|---|---|
| SUMMARY | Deletes an image. Deletes an image in an image stack |
| SYNTAX | void DeleteImage( image ) |
| SYNTAX | void imagestack.deleteimage(image) Image member function |

## DeselectAnnotation

| | |
|---|---|
| SUMMARY | Deselects an annotation |
| SYNTAX | void DeselectAnnotation( Image, Number annotationID ) |
| DESCRIPTION | Deselects the annotation specified by the annotation ID in the given image. |

## Display

| | |
|---|---|
| SUMMARY | Displays an image |
| SYNTAX | void Display(image) |
| SYNTAX | void image.Display() // Class member |
| SYNTAX | void image3D.Display() // Class member |
| SYNTAX | void imagestack.Display() // Class member |

## DisplayAsTable

| | |
|---|---|
| SUMMARY | Displays the image as a table of numbers |
| SYNTAX | void DisplayAsTable(image) |
| DESCRIPTION | Will create a table representation of the image. Currently the image does not change its representation as in DM, but rather creates a separate table. Equivalent to “CreateTableFromImage” |

## DisplayAt

| | |
|---|---|
| SUMMARY | Displays the image in a window at the given position |
| SYNTAX | void DisplayAt(image, number x, number y) |
| DESCRIPTION | x and y are the top left coordinates of the window |

## DisplayOnLogScale

| | |
|---|---|
| SUMMARY | Determines if an image is to be displayed on a log scale |
| SYNTAX | void DisplayOnLogScale(image, number log) |
| SYNTAX | void image.DisplayOnLogScale(number log) // Class Member |
| DESCRIPTION | Sets if the image is to be displayed on a logscale “log” set to 1 (true) or 0 (false) |

## distance

| | |
|---|---|
| SUMMARY | Calculates the pythagorean theorem |
| SYNTAX | number distance(number x, number y ) |
| DESCRIPTION | Returns sqrt(x*x + y*y). |

## Doevents

| | |
|---|---|
| SUMMARY | Checks for input from the keyboard. Useful to check for interrupts in a loop or for control |

SYNTAX void Doevents ()

## DotProduct

SUMMARY Calculates the inner product (dot-product) between to real images (vectors)

SYNTAX number dotproduct( image img1, image img2 )

## EndFill**

SUMMARY Ends a fill from projections

SYNTAX

DESCRIPTION

## erf

SUMMARY Calculates the error function

SYNTAX number erf( number )

DESCRIPTION

## erfc

SUMMARY Calculates the complement of the error function

SYNTAX number erfc( number )

DESCRIPTION

## ErrorDialog

SUMMARY Puts up a dialog with an error number

SYNTAX void ErrorDialog( number )

## Exit

SUMMARY Exit from the script

SYNTAX void Exit ()

## exp

SUMMARY Calculates the exponential of a real/complex number or a real/complex image

SYNTAX number exp( number )

SYNTAX complexnumber exp( complexnumber )

SYNTAX image exp( image )

SYNTAX compleximage exp( compleximage )

SYNTAX void image.exp() Image member function

## exp1

SUMMARY Calculates the exponential of a real number or a real image and subtracts 1

SYNTAX number exp1( number )

SYNTAX image exp1( image )

SYNTAX void image.exp1() Image member function

## exp2

SUMMARY Calculates 2 raised to the power of a real number or a real image

SYNTAX number exp2( number )

SYNTAX image exp2( image )

SYNTAX void image.exp2() Image member function

## exp10

SUMMARY Calculates 10 raised to the power of a real number or a real image

SYNTAX number exp10( number )

SYNTAX image exp10( image )

SYNTAX void image.exp10() Image member function

## ExpandSelection

SUMMARY Expands the selection of an image

SYNTAX void ExpandSelection( Image )

DESCRIPTION Expands the selection in the given image to fit the entire image.

## ExponentialRandom

SUMMARY Calculates a random number with exponential distribution

SYNTAX number ExponentialRandom()

DESCRIPTION

## exprsize

SUMMARY Allocates an image of a given size and optionally assigns it to an expression

SYNTAX image exprsize( number width, number height )

SYNTAX compleximage exprsize( number width, number height )

SYNTAX image exprsize( number width, number height,realimageexpression )

SYNTAX compleximage exprsize( number width, number height,realimageexpression )

## exprsize3

| | |
|---|---|
| SUMMARY | Allocates a volume (3D) image of a given size and optionally assigns it to an expression |
| SYNTAX | image exprsize3( number width, number height, number height ) |
| SYNTAX | compleximage exprsize3( number width, number height , number height) |
| SYNTAX | image exprsize( number width, number height, number height ,realimageexpression ) |
| SYNTAX | compleximage exprsize( number width, number height, number height ,realimageexpression ) |

## extract

| | |
|---|---|
| SUMMARY | Returns an image from a region of another image |
| SYNTAX | image extract( image, number centerX, number centerY , number width, number height) |

## factorial

| | |
|---|---|
| SUMMARY | Calculates the factorial of a real number or a real image |
| SYNTAX | number factorial( number ) |
| SYNTAX | image factorial( image ) |
| SYNTAX | void image.factorial()  Image member function |
| DESCRIPTION | The values are rounded to the nearest integer. The factorial of values less than 1 are returned as 0 |

## FFT

| | |
|---|---|
| SUMMARY | Takes the forward Fourier transform of an image, a volume image or an image within an imagestack or the entire imagestack |
| SYNTAX | image fft( image ) |
| SYNTAX | image3D fft( image3D ) |

| | | |
|---|---|---|
| SYNTAX | void image.fft() | Image member function |
| SYNTAX | void image3D.fft() | Image member function |
| SYNTAX | void imagestack.fft(number) | Image member function |
| SYNTAX | void imagestack.fft() | Image member function |

DESCRIPTION

## FillFromProjection**

SUMMARY Fills in a 2D image from 1D projections

SYNTAX

DESCRIPTION

## FindMaxima

SUMMARY Finds the minima in an image

SYNTAX void FindMaxima(Image, [number minValueForPeak], [number minPeakDistance], [number cmRadius], [number distanceFromEdgde])

SYNTAX void image.FindMaxima([number minValueForPeak] , [number minPeakDistance], [number cmRadius], [number distanceFromEdgde])

DESCRIPTION Look for maxima in the image. minValueForPeak is the smallest value in the image to be considered to be a peak. CmRadius is the center of mass radius used for defining the peak. minPeakDistance is the smallest distance allowed between peaks. distanceFromEdgde is the closest proximity to the edge of the image that is searched for peaks. Default values if not specified are: minValueForPeak = imageMax - 0.2*imageRange, minPeakDistance = 0, cmRadius = 0 , distanceFromEdgde = 0

## FindMinima

SUMMARY Finds the minima in an image

SYNTAX void FindMinima (Image, [number maxValueForPeak], [number minPeakDistance], [number cmRadius], [number distanceFromEdgde])

SYNTAX void image.FindMinima([number maxValueForPeak] , [number minPeakDistance], [number cmRadius], [number distanceFromEdgde])

DESCRIPTION Look for minima in the image. maxValueForPeak is the largest value in the image to be considered to be a peak. CmRadius is the center of mass radius used for defining the peak. minPeakDistance is the smallest distance allowed between peaks. distanceFromEdgde is the closest proximity to the edge of the image that is searched for peaks. Default values if not specified are: maxValueForPeak = imageMin + 0.2*imageRange, minPeakDistance = 0, cmRadius = 0 , distanceFromEdgde = 0

## FindPattern

SUMMARY Returns the position dependent cross-correlation coefficient between an image and a pattern for each position of the pattern within the image

SYNTAX image FindPattern(image sourceImage [. Image template] [,number normalize])

DESCRIPTION This function performs a cross correlation between the sourceimage and the template for each possible position of the template within the image. If the sourceImage has a selection, the template needs not be specified as the selection is used as the template. The argument normalize is set to true/ false (default = false) to set if the source and template are normalized to zero mean before the cross correlation is taken. Equivalent to “TemplateMatch”

## FindPeaks

SUMMARY Finds peaks in an image

SYNTAX void FindPeaks(Image, [number minValueForPeak], [number minPeakDistance], [number cmRadius], [number distanceFromEdgde])

SYNTAX void image.FindPeaks([number minValueForPeak] , [number minPeakDistance], [number cmRadius], [number distanceFromEdgde])

DESCRIPTION Look for maxima in the image. minValueForPeak is the smallest value in the image to be considered to be a

peak. CmRadius is the center of mass radius used for defining the peak. minPeakDistance is the smallest distance allowed between peaks. distanceFromEdgde is the closest proximity to the edge of the image that is searched for peaks. Default values if not specified are: minValueForPeak = imageMax - 0.2*imageRange, minPeakDistance = 0, cmRadius = 0 , distanceFromEdgde = 0

## FitDoublePeaks

SUMMARY — Fits a peak list to a set of double peaks (two peaks are close)

SYNTAX — void FitDoublePeaks(image [number maxPeakSeparation] [, number outputTable])

DESCRIPTION — Fits the peaks found in an image to a set of Gaussian peaks. Peaks within a given distance maxPeakSeparation (default = 10 pixels) are considered to be closely spaced Gaussian peaks. Optionally the parameters for the peaks can be output as a table (outputTable = false by default). The peaks in the image peaklist are updated to reflect the Gaussian fit.

## FitExponentials

SUMMARY — Fits the peaks in a peak list to Exponential peaks

SYNTAX — void FitExponentials(image [, number output = 0] [,number pixelsAcrossPeak = 26] [,number minNumberofPixelsInPeak = 100] )

DESCRIPTION — Fits the peaks found in an image to a set of Exponential peaks. Optionally the parameters for the peaks can be output as a table [1] or written to the log window[2]. pixelsAcrossPeak is an estimate of the numbers of pixels across the entire peak. minNumberofPixelsInPeak represents a minimum number of pixels that must be in a peak. The peaks in the image peaklist are updated to reflect the fit.

## FitGaussians

SUMMARY — Fits the peaks in a peak list to Gaussian peaks

SYNTAX
void FitGaussians(image [, number output = 0] [,number pixelsAcrossPeak = 26] [,number minNumberofPixelsInPeak = 100] )

DESCRIPTION
Fits the peaks found in an image to a set of Gaussian peaks. Optionally the parameters for the peaks can be output as a table [1] or written to the log window[2]. pixelsAcrossPeak is an estimate of the numbers of pixels across the entire peak. minNumberofPixelsInPeak represents a minimum number of pixels that must be in a peak. The peaks in the image peaklist are updated to reflect the fit.

## FitLattice

SUMMARY
Fits an existing lattice to a peaklist

SYNTAX
void FitLattice(image [, number maxDeviation])

DESCRIPTION
The lattice defined on the image will be refined to minimize the sum squared distance from the lattice to the peaks in the peak list. Only peaks lying within the distance maxDeviation (fraction of a lattice vector) will be used in the fitting routine.

## FitParabolas

SUMMARY
Fits the peaks in a peak list to Parabolic peaks

SYNTAX
void FitParabolas(image [, number output = 0] [,number pixelsAcrossPeak = 26] [,number minNumberofPixelsInPeak = 100] )

DESCRIPTION
Fits the peaks found in an image to a set of Parabolic peaks. Optionally the parameters for the peaks can be output as a table [1] or written to the log window[2]. pixelsAcrossPeak is an estimate of the numbers of pixels across the entire peak. minNumberofPixelsInPeak represents a minimum number of pixels that must be in a peak. The peaks in the image peaklist are updated to reflect the fit.

## FitPeaks

SUMMARY
Fits the peaks in a peak list to Gaussian/ Exponential peaks

SYNTAX void FitPeaks(image [, number output = 0] [, number peakShape = 0 ] [,number pixelsAcrossPeak = 26] [,number minNumberofPixelsInPeak = 100] )

DESCRIPTION Fits the peaks found in an image to a set of Gaussian peaks. Optionally the parameters for the peaks can be output as a table [1] or written to the log window[2]. peakShape (0=Gaussian), (1=Exponential). pixelsAcrossPeak is an estimate of the numbers of pixels across the entire peak. minNumberofPixelsInPeak represents a minimum number of pixels that must be in a peak. The peaks in the image peaklist are updated to reflect the fit.

## FlipHorizontal

SUMMARY Flips an image horizontally (around the vertical axis)

SYNTAX void FlipHorizontal( image)

SYNTAX void image.FlipHorizontal() // Class member

## FlipVertical

SUMMARY Flips an image vertically (around the horizontal axis)

SYNTAX void FlipVertical( image)

SYNTAX void image.FlipVertical() // Class member

## floor

SUMMARY Limits all values of a real image to a given minimum value

SYNTAX image floor( image , number)

SYNTAX void image.floor(number) Image member function

DESCRIPTION Sets all values < minVal to minVal

## Gamma

| | |
|---|---|
| SUMMARY | Calculates the gamma of a real number |
| SYNTAX | number Gamma(number) |
| DESCRIPTION | |

## GammaP

| | |
|---|---|
| SUMMARY | Calculates the incomplete gamma function |
| SYNTAX | number GammaP(number , number) |
| DESCRIPTION | |

## GammaQ

| | |
|---|---|
| SUMMARY | Calculates the complement of the incomplete gamma function |
| SYNTAX | number GammaQ(number , number) |
| DESCRIPTION | |

## GammaRandom*

| | |
|---|---|
| SUMMARY | Calculates a random number with gamma distribution |
| SYNTAX | number GammaRandom() |
| DESCRIPTION | *Not Implemented |

## GaussianLowPassFilter

| | |
|---|---|
| SUMMARY | Applies a Gaussian low pass filter |
| SYNTAX | image GaussianLowPassFilter(image, number sigma) |
| DESCRIPTION | Applies a Gaussian low pass filter of sigma in the units of calibration unit of the image |

## GaussianHighPassFilter

| | |
|---|---|
| SUMMARY | Applies a Gaussian high pass filter |
| SYNTAX | image GaussianHighPassFilter (image, number sigma) |
| DESCRIPTION | Applies a Gaussian High pass filter of sigma in the units of calibration unit of the image |

## GaussianRandom*

| | |
|---|---|
| SUMMARY | Calculates a random number with gaussian distribution |
| SYNTAX | number GaussianRandom() |
| DESCRIPTION | |

## GetAnnotationRect

| | |
|---|---|
| SUMMARY | Gets the rectangle of the annotation |
| SYNTAX | void GetAnnotationRect( Image, Number annotationID, NumberVariable top, NumberVariable left, NumberVariable bottom, NumberVariable right ) |

## GetCalibration

| | |
|---|---|
| SUMMARY | Returns the calibration of the image |
| SYNTAX | void GetCalibration( image, numbervariable scalex, numbervariable scaley) |
| SYNTAX | void GetCalibration( image, numbervariable scale) |
| SYNTAX | number GetCalibration( image) |
| SYNTAX | number image.GetCalibration() // Image Class Member Function |
| DESCRIPTION | In general the scale return for a single number is the value stored in scaleX, which is normally the same as scaleY |

## GetCalibrationUnit

SUMMARY — Returns the calibration unit of the image

SYNTAX — number GetCalibrationUnit( image)

SYNTAX — String image.GetCalibrationUnit () // Image Class Member Function

DESCRIPTION — The non-class function returns a number. The index numbers for the calibration units are: 0 – Pixels, 1 – Å, 2 – nanometer, 3 – 1/Pixels, 4 – 1/Å, 5 – 1/nm. The class member function returns a string representation of the unit

## GetHeight

SUMMARY — Returns the height in pixels of an image

SYNTAX — number GetHeight(image)

## GetImage

SUMMARY — Returns a 2D image from a given position (z) in a volume image

SYNTAX — image image3D.GetImage( number whichImage)

DESCRIPTION — Returns an image which is a copy of the image at the given depth in the volume image. The range for whichImage is 0 – (Depth-1)

## GetKey

SUMMARY — Returns key

SYNTAX — Number GetKey( void )

DESCRIPTION — Returns the key that was last pressed as a number.

## GetLattice

| | |
|---|---|
| SUMMARY | Returns the lattice (if defined) for the image |
| SYNTAX | image GetLattice( image) |
| SYNTAX | image image.GetLattice() |
| DESCRIPTION | The lattice is returned in a 2 by 3 image. OriginX is Lattice(0,0). OriginY is Lattice(1,0). UX is Lattice(0,1). UY is Lattice(1,1). VX is Lattice(0,2). VY is Lattice(1,2). |

## GetName

| | |
|---|---|
| SUMMARY | Return the name of the image |
| SYNTAX | string GetName( image) |
| SYNTAX | void GetName( image, stringvariable name) |
| SYNTAX | string image.GetName() // Image Class member function |

## GetNamedImage

| | |
|---|---|
| SUMMARY | Return the image with a given name |
| SYNTAX | image GetNamedImage(string name) |
| SYNTAX | void GetNamedImage(image , string name) |

## GetNumber

| | |
|---|---|
| SUMMARY | Prompt for a number using an OkCancelDialog. Returns 0 (False) if cancel is pressed. 1 (True) otherwise. |
| SYNTAX | number GetNumber ( string prompt, NumberVariable val) |
| SYNTAX | number GetNumber ( string prompt, number default, NumberVariable val) |

## GetNumberedImage

| | |
|---|---|
| SUMMARY | Return the image with a given name |
| SYNTAX | image GetNumberedImage (number num) |
| SYNTAX | void GetNumberedImage (image destImage, number num) |
| DESCRIPTION | Returns the image with the label/tag A# as in A0, A1, A2 etc… |

## GetNthAnnotationID

| | |
|---|---|
| SUMMARY | Get the ID of an annotation |
| SYNTAX | Number GetNthAnnotationID( Image, Number index ) |
| DESCRIPTION | Returns the ID of the index'th annotation in the image. |

## GetPeakList

| | |
|---|---|
| SUMMARY | Returns the peaklist (if defined) for the image |
| SYNTAX | image GetPeakList( image) |
| SYNTAX | image image.GetPeakList() // class member |
| DESCRIPTION | Reaturns the peak list in the form of an image of size 3 by numberPeaks. Column 0 – xposition, Column 1 – yposition, Column 2 - peakValue |

## GetPixel

| | |
|---|---|
| SUMMARY | Returns the pixel value for a given pixel |
| SYNTAX | number GetPixel( image, number x, number y) |
| SYNTAX | complexnumber GetPixel(compleximage, number x, number y) |
| SYNTAX | number image.GetPixel(number x, number y) |
| SYNTAX | complexnumber compleximage.GetPixel(number x, number y) |

## GetPixelAmplitude

SUMMARY Returns the pixel amplitude for a given pixel in a complex image

SYNTAX number GetPixelAmplitude( compleximage, number x, number y) )

## GetPixelPhase

SUMMARY Returns the pixel phase for a given pixel in a complex image

SYNTAX number GetPixelPhase( compleximage, number x, number y) )

## GetScale

SUMMARY Returns the scale/calibration of an image

SYNTAX void GetScale( image, numbervariable scalex, numbervariable scaley)

SYNTAX void GetScale( image, numbervariable scale)

SYNTAX number GetScale( image)

SYNTAX number image.GetScale() // Image Class Member Function

DESCRIPTION In general the scale return for a single number is the value stored in scaleX, which is normally the same as scaleY

## GetSelection

SUMMARY Gets the selection rectangle of an image

SYNTAX Boolean GetSelection( Image, NumberVariable top, NumberVariable left, NumberVariable bottom, NumberVariable right )

DESCRIPTION Sets the given coordinate variables to the coordinates of the current selection in the given image. If the image has no selection, the coordinates are set to the size of the image. Returns true if there was a selection, false if not.

## GetSize

| | |
|---|---|
| SUMMARY | Returns the size of an image |
| SYNTAX | void GetSize(image, numbervariable width, numbervariable height) |
| SYNTAX | void image.GetSize(numbervariable width, numbervariable height) // Class member function |
| SYNTAX | void image3D.GetSize(numbervariable width, numbervariable height, numbervariable depth) // Class member function |

## GetSurveyMode

| | |
|---|---|
| SUMMARY | Gets the method of survey technique for setting black and white values |
| SYNTAX | Number  mode = GetSurveyMode( Image ) |
| DESCRIPTION | mode = 0 CrossHair . mode = 1 Entire Image, Equivalent to GetSurveyTechnique |

## GetSurveyTechnique

| | |
|---|---|
| SUMMARY | Sets the method of survey technique for setting black and white values |
| SYNTAX | Number  mode = GetSurveyTechnique ( Image ) |
| DESCRIPTION | mode = 0 CrossHair . mode = 1 Entire Image, Equivalent to GetSurveyMode |

## GetTwoImages

| | |
|---|---|
| SUMMARY | Two image dialog |
| SYNTAX | Boolean GetTwoImages( String title, ImageVariable image1, ImageVariable image2 ) |

DESCRIPTION Puts up an Ok-Cancel dialog box and allows the user to choose two images. Returns true for Ok and false for Cancel.

## GetTwoImagesWithPrompt

SUMMARY Two image dialog with prompt

SYNTAX Boolean GetTwoImagesWithPrompt( String prompt, String title, ImageVariable image1, ImageVariable image2 )

DESCRIPTION Puts up an Ok-Cancel dialog box and allows the user to choose two images. Returns true for Ok and false for Cancel.

## GetVoxel

SUMMARY Gets the voxel value at position (x,y,z)

SYNTAX number image3D.GetVoxel(number x,number y,number z)

SYNTAX complexnumber compleximage3D.GetVoxel(number x,number y,number z)

## GetWidth

SUMMARY Returns the width in pixels of an image

SYNTAX number GetWidth(image)

## GetWindowPosition

SUMMARY Returns the window position of an image

SYNTAX void GetWindowPosition(image numbervariable left, numbervariable top)

## GetWindowSize

| | |
|---|---|
| SUMMARY | Returns the window size for a displayed image |
| SYNTAX | void GetWindowSize (image numbervariable width, numbervariable height) |

## HasLattice

| | |
|---|---|
| SUMMARY | Returns true/false if a lattice is defined on an image |
| SYNTAX | number HasLattice(image) |
| SYNTAX | number image.HasLattice() // class member |

## HasPeaklist

| | |
|---|---|
| SUMMARY | Returns true/false if a peak list is defined on an image |
| SYNTAX | number HasPeaklist(image) |
| SYNTAX | number image.HasPeaklist() // class member |

## Height

| | |
|---|---|
| SUMMARY | Returns the height (in pixels) of an image |
| SYNTAX | number image.height()  // Image Member Function |
| SYNTAX | number image3D.height()  // Image3D Member Function |

## Highpass

| | |
|---|---|
| SUMMARY | Returns an image resulting from the application of a Annular High Pass filter to an image |
| SYNTAX | image Highpass( image , number cutoff [, number edgewidth ] ) |
| DESCRIPTION | Equivalent to AnnularHighPassFilter. edgewidth by default is set to 0 and represents a soft edge |

## Highpassfilter

| | |
|---|---|
| SUMMARY | Returns an image resulting from the application of a Annular High Pass filter to an image |
| SYNTAX | image Highpassfilter( image , number cutoff [, number edgewidth ] ) |
| DESCRIPTION | Equivalent to AnnularHighPassFilter. edgewidth by default is set to 0 and represents a soft edge |

## HorizontalProjection

| | |
|---|---|
| SUMMARY | Returns an image resulting projecting the pixels (summed) onto the horizontal (x) axis |
| SYNTAX | image HorizontalProjection(image) |

## IFFT

| | | |
|---|---|---|
| SUMMARY | Takes the inverse Fourier transform of a complex image, a complex volume image or a complex image within an imagestack or the entire imagestack | |
| SYNTAX | image ifft( complexImage ) | |
| SYNTAX | image3D ifft( image3D ) | |
| SYNTAX | void image.ifft() | Image member function |
| SYNTAX | void image3D.ifft() | Image member function |
| SYNTAX | void imagestack.ifft(number) | Image member function |
| SYNTAX | void imagestack.ifft() | Image member function |

## imaginary / imag

| | |
|---|---|
| SUMMARY | Returns the imaginary portion of a complex number/ image as a real number/image |
| SYNTAX | number imaginary( complexnumber ) |
| SYNTAX | image imaginary( complexImage ) |

| | | |
|---|---|---|
| SYNTAX | void image.imaginary() | Image member function |
| SYNTAX | number complexnumber.imag() | complex number member function |

## intensity

| | |
|---|---|
| SUMMARY | Returns the modulus square of a complex number/image as a real number/image |
| SYNTAX | number intensity( complexnumber ) |
| SYNTAX | image intensity( complexiamge ) |
| SYNTAX | void image.intensity() Image member function |

## Inverse

| | |
|---|---|
| SUMMARY | Inverts an image |
| SYNTAX | void image.Inverse() // Member function |

## Invert

| | |
|---|---|
| SUMMARY | Inverts an image |
| SYNTAX | image Inverse (image ) |
| SYNTAX | void image.Inverse() // Member function |

## IsAnnotationSelected

| | |
|---|---|
| SUMMARY | Checks if an annotation is selected |
| SYNTAX | Boolean IsAnnotationSelected( Image, Number annotationID ) |
| DESCRIPTION | Returns true if the annotation specified by the annotation ID in the given image is selected; returns true otherwise. |

## Laplacian

SUMMARY Takes the Laplacian of a real image

SYNTAX image Laplacian(image )

SYNTAX void image.Laplacian() // Member function

## LegendrePolynomial

SUMMARY Calculates the Legendre polynomial function

SYNTAX number LegendrePolynomial(number, number, number)

DESCRIPTION

## log

SUMMARY Calculates the natural logarithm of a real number or a real image

SYNTAX number log( number )

SYNTAX image log( image )

SYNTAX void image.log() Image member function

## log1

SUMMARY Calculates the logarithm of a real number or a real image after first adding 1

SYNTAX number log1( number )

SYNTAX image log1( image )

SYNTAX void image.log1() Image member function

DESCRIPTION First the argument is changed by adding 1 (useful when the image contains 0's) and then the logarithm is taken

## log2

SUMMARY Calculates the logarithm base 2 of a real number or a real image

SYNTAX number log2( number )

SYNTAX image log2( image )

SYNTAX void image.log2() Image member function

## log10

SUMMARY Calculates the logarithm base 10 of a real number or a real image

SYNTAX number log10( number )

SYNTAX image log10( image )

SYNTAX void image.log10() Image member function

## Lowpass

SUMMARY Returns an image resulting from the application of an Annular Low Pass filter to an image

SYNTAX image AnnularLowPassFilter( image , number cutoff [, number edgewidth ] )

DESCRIPTION Equivalent to AnnularLowPassFilter. edgewidth by default is set to 0 and represents a soft edge

## Lowpassfilter

SUMMARY Returns an image resulting from the application of an Annular Low Pass filter to an image

SYNTAX image Lowpassfilter( image , number cutoff [, number edgewidth ] )

DESCRIPTION Equivalent to AnnularLowPassFilter. edgewidth by default is set to 0 and represents a soft edge

## MatrixDeterminant*

SUMMARY Returns the determinant of a matrix

SYNTAX number MatrixDeterminant ( image )

DESCRIPTION *Not Implemented

## MatrixInverse*

| | |
|---|---|
| SUMMARY | Inverts a real matrix |
| SYNTAX | image MatrixInverse ( image ) |
| DESCRIPTION | *Not Implemented |

## MatrixMultiply

| | |
|---|---|
| SUMMARY | Does a matrix multiplication of two real images |
| SYNTAX | image MatrixMultiply ( image, image ) |

## MatrixPrint*

| | |
|---|---|
| SUMMARY | Prints out the values of a matrix / image |
| SYNTAX | void MatrixPrint(image) |
| DESCRIPTION | *Not Implemented |

## MatrixTranspose

| | |
|---|---|
| SUMMARY | Transposes the matrix representation of a real image |
| SYNTAX | image MatrixTranspose ( image ) |

## max

| | |
|---|---|
| SUMMARY | Returns the maximum value of a real image. Can also return the positions of the maximum. Calculate the min of two real number expressions or two images |
| SYNTAX | number max( image ) |
| SYNTAX | number max( image, number xpos, number ypos ) |
| SYNTAX | number max( number, number ) |
| SYNTAX | void max( number, number , numbervariable result ) |
| SYNTAX | image max( image, image ) |

SYNTAX void max( image, image, imagevariable result )

SYNTAX number image.max() Image member function

SYNTAX number image.max(number xpos, number ypos) Image member function

## Maximum*

SUMMARY Calculates the maximum of a given list of real numbers

SYNTAX number minimum ( number, number, ... ) up to a maximum of 16 arguments

DESCRIPTION *Not yet implemented

## mean

SUMMARY Returns the mean value of a real image.

SYNTAX number max( image)

SYNTAX number image.mean() Image member function

## meansquare

SUMMARY Returns the mean square value of a real image.

SYNTAX number meansquare( image)

SYNTAX number image.meansquare() Image member function

## median

SUMMARY Returns the median value of a real image or a list of numbers.

SYNTAX number median(image)

SYNTAX number median(number x1, number x2, number x3…) up to a maximum of 16 arguments

## min

SUMMARY Returns the minimum value of a real image. Can also return the positions of the minimum. Calculate the min of two real number expressions or two images

SYNTAX number min( image )

SYNTAX number min( image, number xpos, number ypos )

SYNTAX number min( number, number )

SYNTAX void min( number, number, numbervariable result )

SYNTAX image min( image, image )

SYNTAX void min( image, image, imagevariable result )

SYNTAX number image.min() Image member function

SYNTAX number image.min(number xpos, number ypos) Image member function

## Minimum*

SUMMARY Calculates the minimum of a given list of real numbers

SYNTAX number minimum ( number, number, ... ) up to a maximum of 16 arguments

DESCRIPTION *Not yet implemented

## modsq

SUMMARY Returns the modulus squareof a complex number

SYNTAX number complexnumber.modsq() // complex number member function

## modulus

SUMMARY Returns the modulus of a complex number/image/ image3D as a real number/image

SYNTAX number modulus( complexnumber )

SYNTAX image modulus( compleximage )

| | |
|---|---|
| SYNTAX | number complexnumber.modulus() // complex number member function |
| SYNTAX | void image.modulus() // Class Member Function |
| SYNTAX | void image3D.modulus()// Class Member Function |

## MoveAnnotation

| | |
|---|---|
| SUMMARY | Moves an annotation |
| SYNTAX | void MoveAnnotation( Image, Number annotationID, Number top, Number left, Number bottom, Number right ) |
| DESCRIPTION | Moves the annotation specified by annotation ID in the given image to the specified coordinates. |

## Negate

| | |
|---|---|
| SUMMARY | Returns the inverse of an image |
| SYNTAX | image Negate(image) |

## NewImage

| | |
|---|---|
| SUMMARY | Creates a real image of a given size |
| SYNTAX | image NewImage(string title, number width, number height) |
| SYNTAX | image NewImage(number width, number height) |

## norm

| | |
|---|---|
| SUMMARY | Calculates the norm of a real/complex number or a real/complex image. |
| SYNTAX | number norm( number ) |
| SYNTAX | number norm( complexnumber ) |
| SYNTAX | realimage norm( image ) |
| SYNTAX | realimage norm( compleximage ) |

SYNTAX void image.norm() Image member function

DESCRIPTION The norm of the real number is its square. The norm of a complex number is its modulus square.

## OffsetAnnotation

SUMMARY Offsets an annotation

SYNTAX void OffsetAnnotation( Image, Number annotationID, Number deltax, Number deltay )

DESCRIPTION Offsets the annotation specified by annotation ID in the given image by the specified offsets.

## OkDialog

SUMMARY Ok dialog

SYNTAX void OkDialog( String prompt )

DESCRIPTION Puts up a dialog with an Ok button

## OpenAndSetProgressWindow

SUMMARY Opens and sets the progress window

SYNTAX void OpenAndSetProgressWindow( String line1, String line2, String line3 )

## OpenImage

SUMMARY Creates an image from an existing image file

SYNTAX image open( string filename)

SYNTAX void open( string filename, number width, number height [number type = 7 (real)] [, number byteOffset = 0] [, number swapBytes = 0])

DESCRIPTION Opens an existing image. If the image is fully specified by its internal structure and is supported, only the filename is needed as long as the path is set properly beforehand. If the image

file contains raw image data, then image width, height and optionally type, offset and swapbytes are needed.

## OpenLogWindow

SUMMARY Opens the output window

SYNTAX void OpenLogWindow ( void )

## OpenResultsWindow

SUMMARY Opens the results window

SYNTAX void OpenResultsWindow( void )

DESCRIPTION Equivalent to OpenLogWindow. DM compatibility function

## OpenWithDialog

SUMMARY Creates an image from an existing image file chosen through a file dialog

SYNTAX image OpenWithDialog()

## OptionDown

SUMMARY Returns true/false depending on if the Option key is down or not

SYNTAX Boolean OptionDown( void )

DESCRIPTION Returns 1 if the option key is down and 0 otherwise.

## PadWithMean

SUMMARY Pads an image with its mean value to specified dimensions

SYNTAX void image.PadWithMean(number newWidth, number newHeight) // class member

## PadWithZero

SUMMARY Pads an image with zero to specified dimensions

SYNTAX void image.PadWithZero(number newWidth, number newHeight) // class member

## Pi

SUMMARY Returns an approximation of $\pi$ One can also just write Pi which is a predefined constant

SYNTAX number pi()

## phase

SUMMARY Returns the phase of a complex number/image/image3D as a real number/image

SYNTAX number phase( complexnumber )

SYNTAX image phase( compleximage )

SYNTAX number complexnumber.phase() // complex number member function

SYNTAX void image.phase() // Image member function

SYNTAX void image3D.phase() // Image3D member function

## PhaseCorrelate

SUMMARY Returns the phase correlation between two images

SYNTAX image PhaseCorrelate (image x, image y [, number freqCutoff])

DESCRIPTION Calculate the phase correlation between two images but using frequencies up to a maximum frequency cut off “freqCutoff” default freqCutoff = 0.3*maxFrequency

## PhaseCorrelation

| | |
|---|---|
| SUMMARY | Returns the phase correlation between two images |
| SYNTAX | image PhaseCorrelation(image x, image y [, number freqCutoff]) |
| DESCRIPTION | Calculate the phase correlation between two images but using frequencies up to a maximum frequency cut off “freqCutoff” default freqCutoff = 0.3*maxFrequency |

## PoissonRandom*

| | |
|---|---|
| SUMMARY | Calculates a random number with poisson distribution |
| SYNTAX | number PoissonRandom() |
| DESCRIPTION | |

## Polar

| | |
|---|---|
| SUMMARY | Calculates the polar representation of a rectangular complex number/image |
| SYNTAX | complexnumber polar( complexnumber ) |
| SYNTAX | complexnumber polar( compleximage ) |
| SYNTAX | void image.polar()  Image member function |
| DESCRIPTION | Amplitude stored in real part. Phase stored in imaginary part |

## Polynomial*

| | |
|---|---|
| SUMMARY | Calculates a polynomial expansion using a real image expression |
| DESCRIPTION | *Currently not implemented |

## pow

SUMMARY Calculates the exponential of a real/complex number or a real/complex image

SYNTAX `number pow( number x, number y)    // x**y`

SYNTAX `image exp( image x, number y )     // x**y`

SYNTAX `void image.pow(number x)      Image member function`

## pow2

SUMMARY Calculates 2 raised to the power of a real number or a real image

SYNTAX `number pow2( number )`

SYNTAX `image pow2( image )`

SYNTAX `void image.pow2()             Image member function`

## pow10

SUMMARY Calculates 10 raised to the power of a real number or a real image

SYNTAX `number pow10( number )`

SYNTAX `image pow10( image )`

SYNTAX `void image.pow10()            Image member function`

## PowerSpectrum

SUMMARY Calculates the power spectrum of a real image

SYNTAX `image PowerSpectrum( image )`

SYNTAX `void image.PowerSpectrum() // Image Member Function`

## product*

| | |
|---|---|
| SUMMARY | Calculates the product of a real/complex image expression |
| SYNTAX | Number product( RealImageExpression ) |
| SYNTAX | ComplexNumber product( ComplexImageExpression ) |
| DESCRIPTION | *Currently not implemented |

## PropagateWave

| | |
|---|---|
| SUMMARY | Calculates a 3D complex volume containing the wave function at each slice for the current simulation up to a given thickness. |
| SYNTAX | Image3D PropagateWave(number thickness) |
| SYNTAX | Image3D simulation.PropagateWave(number thickness) // member function of the simulation object |
| DESCRIPTION | Implemented as both a standalone function for an open simulation or as a member function of the class simulation. |

## ps

| | |
|---|---|
| SUMMARY | Calculates the power spectrum of a real image |
| SYNTAX | image ps( image ) |
| SYNTAX | void image.ps() // Image Member Function |

## RadialAverage

| | |
|---|---|
| SUMMARY | Returns the radial average of an image |
| SYNTAX | image RadialAverage(image sourceImage [, number mode]) |
| DESCRIPTION | Creates a new image of type mode that is a representation of the radial average of sourceImage. The optional argument mode represents: Mode = 1D , Mode = 1 2D Split Plane , Mode = 2 2D |

## ReadPeakList

SUMMARY Returns the peak list from a file

SYNTAX image ReadPeakList(string filename)

DESCRIPTION Returns the peak list in an image by reading a file containg the list of peaks (saved by the program as a tab-delimited text file). The peaklist can then be associated with an existing image through the SetPeakList function.

## real

SUMMARY Returns the real part of a complex number/image as a real number/image

SYNTAX number real( complexnumber )

SYNTAX image real( compleximage )

SYNTAX number complexnumber.real() // complex number member function

SYNTAX void image.real() // Image member function

## RealImage

SUMMARY Creates a real image of a given size

SYNTAX image RealImage(string title, number numBytes, number width, number height)

SYNTAX image RealImage(number width, number height)

DESCRIPTION Creates a real floating point image. Only 4 byte real numbers are supported

## RealFFT

SUMMARY Takes the forward Fourier transform of an image

SYNTAX image Realfft( image )

DESCRIPTION This does not compute a packed Fourier transform as in DM, but is present so that there is an equivalent syntax to DM scripting

## Remainder

SUMMARY Calculates the integer remainder for real numbers or real images

SYNTAX number remainder( number )

SYNTAX image remainder( image )

DESCRIPTION

## Rect

SUMMARY Calculates the rectangular representation of a polar complex number/image

SYNTAX complexnumber rect( complexnumber )

SYNTAX complexnumber rect( compleximage )

SYNTAX void image.rect() Image member function

## RemoveCCDDefects

SUMMARY Corrects for CCD detector bad pixels in a real image (*ccd*)

SYNTAX image RemoveCCDDefects(image)

SYNTAX void image.RemoveCCDDefects() //Member function

DESCRIPTION Equivalent to “ccd”

## repeat

SUMMARY Repeats an image/image3D in the 2 or 3 dimensions

SYNTAX image repeat( image, number nx, number ny )

SYNTAX void image.repeat(number nx, number ny) Image member function

SYNTAX void image3D.repeat(number nx, number ny, number nz)
Image3D member function

## Resize

SUMMARY Resizes an image

SYNTAX image Resize(image , number width, number height)

SYNTAX void image.Resize(number width, number height)

DESCRIPTION Resizes the image using interpolation

## RMS

SUMMARY Calculates the root mean square value of a real image

SYNTAX number rms( image )

SYNTAX number image.rms() Image member function

## Rotate

SUMMARY Rotates a 2D image clockwise by a given angle

SYNTAX image rotate( image, number angle )

SYNTAX void image.rotate(number angle )

## RotateLeft

SUMMARY Rotates an image anti-clockwise an image by 90 deg.

SYNTAX image rotateleft( image, number angle )

SYNTAX void image.rotateleft(number angle )

## RotateRight

SUMMARY Rotates an image clockwise an image by 90 deg.

SYNTAX image rotateright( image, number angle )

SYNTAX void image. rotateright(number angle )

## RotateX

SUMMARY Rotates a volume image (image3D) clockwise about x

SYNTAX void image3D.rotatex(number angle )

## RotateY

SUMMARY Rotates a volume image (image3D) clockwise about y

SYNTAX void image3D.rotatey(number angle )

## RotateZ

SUMMARY Rotates a volume image (image3D) clockwise about z

SYNTAX void image3D.rotatez(number angle )

## round

SUMMARY rounds to the nearest integer a real number or a real image

SYNTAX number round( number )

SYNTAX image round( image )

SYNTAX void image.round() Image member function

## SaveImage

SUMMARY Save the image

SYNTAX void SaveImage (image theImage, string fileName [, number type] )

DESCRIPTION Saves the peak data to a given file, as the specified file type. Default type = current type. Type = 1 (ascii file), type = 2 (binary) , type = 3 (tiff)

## SavePeaks

SUMMARY Save the peaks in a peak list to a file

SYNTAX void SavePeaks(image theImage, string fileName [, number type] )

DESCRIPTION Saves the peak data to a given file, as the specified file type. Type = 1 (default, text file), type = 2 — Tempas file

## SavePeaksWithDialog

SUMMARY Save the peaks in a peak list to a file after prompting for filename and location

SYNTAX void SavePeaks(image theImage[, number type])

DESCRIPTION Saves the peak data to a given file, as the specified file type. Type = 1 (default, text file), type = 2 — Tempas file

## SelectAnnotation

SUMMARY Selects an annotation

SYNTAX void SelectAnnotation( Image, Number annotationID )

DESCRIPTION Selects the annotation specified by the annotation ID in the given image.

## Set

SUMMARY Sets the real and imaginary part of a complex number

SYNTAX void complexnumber.set(number x,number y) // complex number member function

## SetAnnotationBackground*

SUMMARY Sets the background of an annotation

SYNTAX void SetAnnotationBackground( Image, Number annotationID, Number background )

## SetAnnotationColor

SUMMARY Sets the RGB Color of the Annotation

SYNTAX void SetAnnotationColor( Image, Number annotationID, Number red, Number green, Number blue)

## SetAnnotationFace*

SUMMARY Sets the text face of an annotation

SYNTAX Sets the type face of the annotation specified by the annotation ID in the given image.

## SetAnnotationFont

SUMMARY Sets the text justification of an annotation

SYNTAX void SetAnnotationJustification( Image, Number annotationID, Number justification )

DESCRIPTION Sets the justification of the text annotation specified by the annotation ID in the given image.

## SetAnnotationJustification*

SUMMARY Sets the text justification of an annotation

SYNTAX void SetAnnotationJustification( Image, Number annotationID, Number justification )

## SetAnnotationRect

SUMMARY Sets the rect of an annotation

SYNTAX
void SetAnnotationRect( Image, Number annotationID, Number top, Number left, Number bottom, Number right )

DESCRIPTION
Moves the annotation specified by annotation ID in the given image to the specified coordinates.

## SetAnnotationSize

SUMMARY
Sets the text size of an annotation

SYNTAX
void SetAnnotationSize( Image, Number annotationID, Number textSize )

DESCRIPTION
Sets the size of text of the annotation specified by the annotation ID in the given image.

## SetBlackWhite

SUMMARY
Sets the black and white display limits of an image

SYNTAX
void image.SetBlackWhite( number black, number white) // Member function

DESCRIPTION
Sets the limits for what is to be displayed as black and white. Values <= black are all displayed as black. Values >= white are all displayed as white.

## SetCalibration

SUMMARY
Sets the calibration and possibly calibration unit of an image

SYNTAX
void SetCalibration( image , number calibration)

SYNTAX
void SetCalibration( image , number calibration, number calibrationunit)

DESCRIPTION
The index numbers for the calibration units are: 0 — Pixels, 1 — Å, 2 — nanometer, 3 — 1/Pixels, 4 — 1/Å, 5 — 1/nm

## SetCalibrationUnit

SUMMARY — Sets the calibration unit of an image

SYNTAX — void SetCalibrationUnit( image , number calibrationunit)

DESCRIPTION — The index numbers for the calibration units are: 0 = Pixels, 1 = Å, 2 = nanometer, 3 = 1/Pixels, 4 = 1/Å, 5 = 1/nm

## SetImage

SUMMARY — Sets a 2D image at a given position (z) in the volume image

SYNTAX — void image3D.SetImage(image, number whichposition)  // Member function

DESCRIPTION — Copies an existing image into the depth “whichposition” (0 — (depth-1)) in a volume image

## SetDisplayType

SUMMARY — Sets the type of display of an image

SYNTAX — SetDisplayType(image img, number type)

SetDisplayType(Image, string type) type = “raster”,” surface”,”rgb”,”line”,”table”,”argand”,”complex”. String is case insensitive

SYNTAX — void image.SetDisplayType(number type // Member function

void image.SetDisplayType(string type) type = “raster”,”surface”,”rgb”,”line”,”table”,”argand”,” complex”. String is case insensitive

DESCRIPTION — types : 1=Raster Image , 2=Surface Plot , 3=RGB, 4 Line Plot , 5-Table , Types 3 is not implemented

## SetImageSpace

SUMMARY — Sets the space (real/reciprocal) of an image

SYNTAX — void image.SetImageSpace(number space) // Member function

SYNTAX void image3D.SetImageSpace(string space) // Member function

DESCRIPTION space = 0 Real space , space = 1 Reciprocal Space

space = “real”, space = “reciprocal”

## SetLimits

SUMMARY Sets the black and white display limits of an image

SYNTAX SetLimits(image, number black, number white)

SYNTAX void image.SetLimits ( number black, number white)
// Member function

Equivalent to the member function SetBlackWhite

DESCRIPTION Sets the limits for what is to be displayed as black and white. Values <= black are all displayed as black. Values >= white are all displayed as white.

## SetName

SUMMARY Sets the name of an image

SYNTAX void SetName( image , string)

SYNTAX void image.SetName(string) // Image Member Function

## SetPeakList

SUMMARY Associated an image with an existing peaklist.

SYNTAX void SetPeakList(image theImage, image peaklist)

DESCRIPTION After reading in a peaklist from a file or getting the peaklist from an image, this peaklist can be associated with a desired existing image. The dimensions of the image to be associated the peaklist must be of the same dimensions as the image from which the peaklist originated for this to make sense.

## SetPixel

SUMMARY Sets a specified pixel to a given value

SYNTAX void SetPixel(image , number x, number y, number val)

SYNTAX void SetPixel(compleximage ,number x, number y, number val)

SYNTAX void SetPixel(compleximage ,number x, number y, complexnumber val)

SYNTAX void image.SetPixel(number x, number y, number val) // Member function

SYNTAX void compleximage.SetPixel(number x, number y, number val) // Member function

SYNTAX void compleximage.SetPixel(number x, number y, complexnumber val) // Member function

## SetPixelAmplitude*

SUMMARY Sets the pixel amplitude for a given pixel in a complex image

SYNTAX void SetPixelAmplitude( image, number x, number y, number amplitude )

SYNTAX void image.SetPixelAmplitude(number x, number y, number amplitude) Image member function

DESCRIPTION

## SetPixelPhase

SUMMARY Sets the pixel phase for a given pixel in a complex image

SYNTAX void SetPixelPhase(image, number x, number y, number phase)

SYNTAX void image.SetPixelPhase(number x, number y, number phase) Image member function

DESCRIPTION

## SetSelection

| | |
|---|---|
| SUMMARY | Sets the selection rectangle of an image |
| SYNTAX | void SetSelection( Image, Number top, Number left, Number bottom, Number right ) |
| DESCRIPTION | Sets the selection of the given image to the coordinates. |

## SetScale

| | |
|---|---|
| SUMMARY | Sets the scale/calibration of an image |
| SYNTAX | void SetScale( image , number scale) |
| SYNTAX | void SetScale( image , number scaleX, number scaleY) |
| DESCRIPTION | Sets the x and y scale, the number of units per pixel in x and y |

## SetSurveyMode

| | |
|---|---|
| SUMMARY | Sets the method of survey technique for setting black and white values |
| SYNTAX | void SetSurveyMode( Image, Number mode ) |
| DESCRIPTION | mode = 0 CrossHair . mode = 1 Entire Image, Equivalent to SetSurveyTechnique |

## SetSurveyTechnique

| | |
|---|---|
| SUMMARY | Sets the method of survey technique for setting black and white values |
| SYNTAX | void SetSurveyTechnique( Image, Number mode ) |
| DESCRIPTION | mode = 0 CrossHair . mode = 1 Entire Image, Equivalent to SetSurveyMode |

## SetVoxel

| | |
|---|---|
| SUMMARY | Sets the voxel value at position (x,y,z) |

| | |
|---|---|
| SYNTAX | image3D.SetVoxel(number x, number y, number z, number value) |
| SYNTAX | compleximage3D.SetVoxel(number x, number y, number z, complexnumber value) |

## SetWindowPosition

| | |
|---|---|
| SUMMARY | Sets the window position of an image |
| SYNTAX | void SetWindowPosition(image, number left, number top) |
| DESCRIPTION | |

## SetWindowSize

| | |
|---|---|
| SUMMARY | Sets the window size for a displayed image |
| SYNTAX | void SetWindowSize(image, number width, number height) |

## Sharpen

| | |
|---|---|
| SUMMARY | Applies a Sharpening Filter to a real image |
| SYNTAX | void Sharpen( image ) // In place operation |
| DESCRIPTION | Does a sharpening operation on real image in place |

## Shift

| | |
|---|---|
| SUMMARY | Shifts the position (0,0) to a new position (sx,sy) in the image |
| SYNTAX | image Shift( image, number sx, number sy) |
| SYNTAX | void Shift( image)  // sx = W/2 , sy = H/2   In place operation |
| SYNTAX | void image.Shift(number sx, number sy) |

## ShiftAnnotation

SUMMARY Shifts the position of an annotation

SYNTAX void SetAnnotationRect( Image, Number annotationID, Number shiftX, Number shiftY)

## ShiftCenter

SUMMARY Shifts the position (0,0) to the position (W/2,H/2) in the image

SYNTAX void ShiftCenter( Image) // In place operation

SYNTAX void Image.ShiftCenter() // Member function

DESCRIPTION Shifts each dimension of an image by half. For two dimensional images it will swap quadrants.

## ShiftDown

SUMMARY Returns true/false depending on if the Shift key is down or not

SYNTAX Boolean ShiftDown( void )

DESCRIPTION Returns 1 if the shift key is down and 0 otherwise.

## ShiftImageFocus

SUMMARY Propagates a complex image or wave function by a distance focus

SYNTAX void ShiftImageFocus( compleximage source, number focus [, number voltage = 300] [ , number sampling = 0.2])

DESCRIPTION The focus variation (or constant) is given in the image focus. The complex image is propagated over the distance focus. By default the voltage is 300kV. If the source is calibrated in Ångstrom or nanometer, the sampling is taken from the source. Otherwise the default is 0.2 Å/pixel and must be set if different.

## ShiftOrigin

| | |
|---|---|
| SUMMARY | Shifts the position (0,0) to the position (sx,sy) in the image |
| SYNTAX | image ShiftOrigin( Image, number sx, number sy) |
| SYNTAX | void ShiftOrigin( Image)  // sx = W/2 , sy = H/2 ( in place) |
| SYNTAX | void image.ShiftOrigin(number sx, number sy) |

## show

| | |
|---|---|
| SUMMARY | Displays an image. Equivalent to Display |
| SYNTAX | void Show(image) |
| SYNTAX | void image.Show()  // Member function |
| SYNTAX | void image3D.Show()  // Member function |
| DESCRIPTION | Equivalent to Display |

## ShowImage

| | |
|---|---|
| SUMMARY | Displays an image. |
| SYNTAX | void ShowImage(image) |
| SYNTAX | void image.ShowImage()  // Member function |
| DESCRIPTION | Equivalent to Display |

## sgn

| | |
|---|---|
| SUMMARY | Calculates the sign of a real number |
| SYNTAX | RealNumberExpression sgn( RealNumberExpression ) |
| DESCRIPTION | Returns 1 if the number is equal or greater than 0 otherwise returns -1 |

## sigma

SUMMARY Calculates the standard deviation of a real image

SYNTAX number sigma( image )

SYNTAX number image.sigma() Image member function

## sin

SUMMARY Calculates the sine of a real number or a real image

SYNTAX number sin( number )

SYNTAX image sin( image )

SYNTAX void image.sin() Image member function

## sinh

SUMMARY Calculates the hyperbolic sine of a real number or a real image

SYNTAX number sinh( number )

SYNTAX image sinh( image )

SYNTAX void image.sinh() Image member function

## Smooth

SUMMARY Applies a Smoothing Filter to a real image

SYNTAX image Smooth(image)

SYNTAX void image.Smooth() // Member function

## Sobel

SUMMARY Applies a Sobel Filter to a real image

SYNTAX image sobel(image)

SYNTAX void image.sobel() // Member function

## SpaceDown

SUMMARY Returns true/false depending on if the Space bar is down or not

SYNTAX Boolean SpaceDown( void )

DESCRIPTION Returns 1 if the space key is down and 0 otherwise.

## SphericalBesselJ

SUMMARY Calculates the spherical Bessel J function

SYNTAX number SphericalBesselJ( number, number )

DESCRIPTION

## SphericalBesselY

SUMMARY Calculates the general Bessel Y function

SYNTAX number SphericalBesselY( number, number )

DESCRIPTION

## sqrt

SUMMARY Calculates the square root of a real number or a real image

SYNTAX number sqrt( number )

SYNTAX image sqrt( image )

SYNTAX void image.sqrt() Image member function

## sq

SUMMARY Calculates the square of a real number or a real image

SYNTAX number sq( number )

SYNTAX image sq( image )

SYNTAX void image.sq() Image member function

## square

SUMMARY Calculates the square of a real number or a real image

SYNTAX number square( number )

SYNTAX image square( image )

SYNTAX void image.square() Image member function

## stdv

SUMMARY Calculates the standard deviation of a real image

SYNTAX number stdv( image )

SYNTAX number image.stdv() Image member function

## sum

SUMMARY Calculates the sum of a real image

SYNTAX number sum( image )

SYNTAX number image.sum() Image member function

## tan

SUMMARY Calculates the tangent of a real number or a real image

SYNTAX number tan( number )

SYNTAX image tan( image )

SYNTAX void image.tan() Image member function

## tanh

SUMMARY — Calculates the hyperbolic sine of a real number or a real image

SYNTAX — number tanh( number )

SYNTAX — image tanh( image )

SYNTAX — void image.tanh() Image member function

## Templatematch

SUMMARY — Returns the position dependent cross-correlation coefficient between an image and a pattern for each position of the pattern within the image

SYNTAX — image TemplateMatch(image sourceImage [. Image template] [,number normalize])

DESCRIPTION — This function performs a cross correlation between the sourceimage and the template for each possible position of the template within the image. If the sourceImage has a selection, the template needs not be specified as the selection is used as the template. The argument normalize is set to true/ false (default = false) to set if the source and template are normalized to zero mean before the cross correlation is taken. Equivalent to “FindPattern”

## TimeBar*

SUMMARY — Displays a timebar while evaluating real image expression

SYNTAX — RealImageExpression TimeBar( String title, RealImageExpression expression )

DESCRIPTION — *Not Implemented - Puts up a timebar with the string as a title for the real expression.

## thf

SUMMARY — Applies a Threshold Filter to a real image

SYNTAX — void image.thf() // Class Member function

DESCRIPTION Equivalent to ThresholdFilter

## throw

SUMMARY throws an exception that can be caught by a try statement

SYNTAX throw( number)

SYNTAX throw( string)

## throwstring

SUMMARY throws an exception that can be caught by a try statement

SYNTAX throwstring(string )

## ThresholdFilter

SUMMARY Applies a Threshold Filter to a real image

SYNTAX void image.ThresholdFilter() // Class Member function

DESCRIPTION

## Transpose

SUMMARY Transposes an image

SYNTAX image Transpose(image)

SYNTAX void image.Transpose() // Class Member function

## trunc

SUMMARY Truncates a real number to an integer or a real image to integer values

SYNTAX number trunc( number )

SYNTAX image trunc( image )

SYNTAX void image.trunc() Image member function

## TwoButtonDialog

SUMMARY Two button dialog

SYNTAX Boolean TwoButtonDialog( String prompt, String, rejectLabel, String acceptLabel )

DESCRIPTION Puts up a two button dialog with the accepting and rejecting buttons labeled according to the parameters 'acceptLabel' and 'rejectLabel'. Returns true for accept and false for reject.

## UniformRandom*

SUMMARY Calculates a random number with uniform distribution

SYNTAX number UniformRandom()

DESCRIPTION

## update

SUMMARY Updates an image that has been modified

SYNTAX void image.update() // Image member function

DESCRIPTION To ensure that an image that has been modified gets its display representation and other statistics reset

## UpdateImage

SUMMARY Updates an image that has been modified

SYNTAX void UpdateImage(image)

DESCRIPTION To ensure that an image that has been modified gets its display representation and other statistics reset

## ValidAnnotation

SUMMARY — Checks if specified annotation exists

SYNTAX — Boolean ValidAnnotation( Image, Number annotationID )

DESCRIPTION — Returns true if the annotation specified by the annotation ID in the given image is valid; returns false otherwise.

## variance

SUMMARY — Returns the variance of a real image

SYNTAX — number variance( image )

SYNTAX — number image.variance() Image member function

## VectorLength

SUMMARY — Returns the Length of a real image as a vector

SYNTAX — number VectorLength( image )

SYNTAX — number image.VectorLength() Image member function

DESCRIPTION — Returns the square root of the sum of the squares

## VectorMap

SUMMARY — Creates a vector map from two images

SYNTAX — void VectorMap(image x, image y [, number samplingX ] [, number samplingY] [, number scale])

DESCRIPTION — Creates and displays a vector map from two images x and y which correspond to the x and y components of the vectors. Vectors will be created every samplingX (default=16) pixels and samplingY (default=16) pixels. Vectors are drawn with the magnification factor: scale (default=10)

## VerticalProjection

SUMMARY Returns an image resulting projecting the pixels (summed) onto the vertical (y) axis

SYNTAX image VerticalProjection(image)

## Warp

SUMMARY Calculates bilinear interpolated value within a real image

SYNTAX image warp(RealImage source, RealImageExpression sourceX, RealImageExpression sourceY)

DESCRIPTION Transforms the source into a new image based on a transformation of the x and y values

## wf

SUMMARY Returns an image resulting from applying a Wiener Filter to an image

SYNTAX image wf( image )

SYNTAX void image.wf()  // Image Member Function

DESCRIPTION Attempts to reduce random noise in the image of a crystalline object. Equivalent to “wienerfilter”

## width

SUMMARY Returns the width of an image

SYNTAX number image.width()  // Image Member Function

SYNTAX number image3D.width()  // Image3D Member Function

## WienerFilter

SUMMARY Returns an image resulting from applying a Wiener Filter to an image

SYNTAX image WienerFilter( image )

SYNTAX void image.wienerfilter() // Image Member Function

DESCRIPTION Attempts to reduce random noise in the image of a crystalline object. Equivalent to “wf”

### x

SUMMARY Returns or sets the real part of a complex number

SYNTAX number complexnumber.x() // returns the real part

SYNTAX voi complexnumber.x(number) // sets the real part

### y

SUMMARY Returns or sets the imaginary part of a complex number

SYNTAX number complexnumber.y() // returns the imaginary part

SYNTAX void complexnumber.y(number) // sets the imaginary part

# Alphabetical description of simulation script functions

Non Member Functions

### CalculateAtomicScatteringFactors

SUMMARY Calculates the atomic scattering factors for a given atomic element and places them in a file

SYNTAX void CalculateAtomicScatteringFactors(number Z [, number debyeWaller ] [, number voltage ] [, number gMax] [, number deltaG] )

DESCRIPTION Calculates the full Atomic Scatering Factors for the element with atomic number Z for all [h,k,l] out to gMax. Default values are: debyeWaller = 0.5 , voltage = 300 kV , gMax = 4.0 1/Å , deltaG — 0.1 1/Å

---

## CalculateExitWave

SUMMARY Calculates the Exit WaveFunction(s) for the current simulation. Optionally can use a starting wave different from a uniform plane wave of value 1 everywhere.

SYNTAX void CalculateExitWave ()

or

CalculateExitWave(ComplexImage entranceWave)

## CalculateImage

SUMMARY Calculates the simulated Images(s) for the current simulation

SYNTAX void CalculateImage()

## CalculatePotential

SUMMARY Calculates the 2D Projected Potential(s) for the current simulation

SYNTAX void CalculatePotential()

## PropagateWave

SUMMARY Calculates a 3D complex volume containing the wave function at each slice for the current simulation up to a given thickness.

SYNTAX Image3D PropagateWave(number thickness)

DESCRIPTION Implemented as both a standalone function for an open simulation or as a member function of the class simulation.

Simulation Class Member Functions

The syntax simulation.functionname() would be used as in the following example
Example:

```
simulation sim = getsimulation()
sumber focus = sim.getfocus()
print(focus)
```

Any brackets [] within the functions argument list represents optional arguments which have default values if not specified. If any optional argument needs to be specified, all othe optional arguments preceding it must be specified.

## Calculate3DPotential

| | |
|---|---|
| SUMMARY | Calculates the 3D potential for the unit cell of the current simulation |
| SYNTAX | void simulation.Calculate3Dpotential (image3D potential) |
| SYNTAX | Image3d potential=simulation.Calculate3Dpotential() |
| DESCRIPTION | Calculates the full 3D potential for the specimen unit cell out to 2*gmax for all [h,k,l]. Stores the 3D complex potential in the volume image “potential” which can be displayed using the command “potential.display()” The volume image is created in the process. |

## CalculateExitWave

| | |
|---|---|
| SUMMARY | Calculates the Exit WaveFunction(s) for the current simulation |
| SYNTAX | void simulation.CalculateExitWave () |

## CalculateImage

| | |
|---|---|
| SUMMARY | Calculates the simulated Images(s) for the current simulation |
| SYNTAX | void simulation.CalculateImage() |

## CalculatePotential

SUMMARY — Calculates the 2D Projected Potential(s) for the current simulation

SYNTAX — void simulation.CalculatePotential()

## CreateFrequencyImage

SUMMARY — Returns a square image of a simulated object in reciprocal space

SYNTAX — image simulation.CreateFrequencyImage ( image [, number imageSize ] [, number divergenceAngle ] [, number gMax ] [, number minIntensity] [, number h, number k, number l])

DESCRIPTION — Creates and returns a square image of size *imageSize*imageSize* of the specified image representing the Fourier transform of one of the calculated types in the simulation (potential, exit wave, image) using a sampling given by the value of gMax (sampling = imageSize/(2*gMax)). The minimum intensity in the pattern (the valye of black) is 10**(- minIntensity). Gaussian peaks of sigma given by the divergenceAngle are placed on the diffraction spots. Default values are: imageSize = 512, gMax = gMax for the current simulation, divergenceAngle is the value for the microscope for the simulation. MinIntensity = 6. The optional values h,k,l are the indicies of the desired reflection along the positive x-axis in the diffraction pattern image.

## CreateImage

SUMMARY — Returns a square image from a given calculated image of given size and sampling

SYNTAX — image simulation.CreateImage( image [, number imageSize ] [, number sampling ] )

DESCRIPTION — Creates and returns a square image of size *imageSize*imageSize* of the specified image representing one of the calculated types in the simulation (potential, exit wave, image) using a sampling of *sampling*. Default values are: whichImage = 1 , imageSize = 512 , sampling = 0.1Å

## DisplayExitWave

SUMMARY — Displays a calculated exit wave

SYNTAX — void simulation.DisplayExitWave( [number whichExitWave] [, number nX ] [, number nY ] [, number zoom ] )

DESCRIPTION — Creates and displays the specified exit wave for nX by nY unit cells, using a zoom factor. The image will be resampled to make dx and dy the same and to make the angle 90 degrees if necessary. Defaults are: whichExitWave = 1, nX = 1, nY = 1, zoom = 1

## DisplayExitWaveModulus

SUMMARY — Displays the modulus of a calculated exit wave

SYNTAX — void simulation.DisplayExitWaveModulus( [number whichExitWave] [, number nX ] [, number nY ] [, number zoom ] )

DESCRIPTION — Creates and displays the modulus of the specified exit wave for nX by nY unit cells, using a zoom factor. The image will be resampled to make dx and dy the same and to make the angle 90 degrees if necessary. Defaults are: whichExitWave = 1, nX = 1, nY = 1, zoom = 1

## DisplayExitWavePhase

SUMMARY — Displays the phase of a calculated exit wave

SYNTAX — void simulation.DisplayExitWavePhase( [number whichExitWave ] [, number nX ] [, number nY ] [, number zoom ] )

DESCRIPTION — Creates and displays the phase of the specified exit wave for nX by nY unit cells, using a zoom factor. The image will be resampled to make dx and dy the same and to make the angle 90 degrees if necessary. Defaults are: whichExitWave = 1, nX = 1, nY = 1, zoom = 1

## DisplayImage

SUMMARY — Displays a calculated image

SYNTAX — void simulation.DisplayImage( [number whichImage ] [, number nX ] [, number nY ] [, number zoom ] )

DESCRIPTION — Creates and displays the specified image for nX by nY unit cells, using a zoom factor. The image will be resampled to make dx and dy the same and to make the angle 90 degrees if necessary. Defaults are: whichImage = 1, nX = 1, nY = 1, zoom = 1

## DisplayPotential

SUMMARY — Displays a calculated 2D projected potential

SYNTAX — void simulation.DisplayPotential( [number whichPotential ] [, number nX ] [, number nY ] [, number zoom ] )

DESCRIPTION — Creates and displays the specified exit wave for nX by nY unit cells, using a zoom factor. The image will be resampled to make dx and dy the same and to make the angle 90 degrees if necessary. Defaults are: whichPotential = 1, nX = 1, nY = 1, zoom = 1

## Focus

SUMMARY — Sets the focus of the simulation

SYNTAX — void simulation.Focus(number focus)

DESCRIPTION — Sets the focus [Å] for the current simulation

## GetAperture

SUMMARY — Returns the radius of the outer objective lens aperture (1/Å)

SYNTAX — number simulation.GetAperture()

DESCRIPTION — Equivalent to GetOuterAperture

## GetApertureAngle

SUMMARY Returns the angle of the outer objective lens aperture (mrad)

SYNTAX `number simulation.GetApertureAngle()`

## GetApertureCenter

SUMMARY Returns the center of the objective lens aperture in “tilt” angle (mrad) and azimuthal angle (degrees)

SYNTAX `number simulation.GetApertureCenter( number theta, number phi)`

## GetApertureCenterHK

SUMMARY Returns the center of the objective lens aperture in (H,K) of the reciprocal space of the unit cell

SYNTAX `void simulation.GetApertureCenterHK( number cH, number cK)`

## GetCs

SUMMARY Returns the Spherical Aberration Cs in mm le of the outer objective lens aperture (mrad)

SYNTAX `number simulation.GetCs()`

## GetCs5

SUMMARY Returns the $5^{th}$ order Spherical Aberration Cs5 in mm

SYNTAX `number simulation.GetCs5()`

## GetDeltaFocus

SUMMARY Returns the increment in focus [Å] for a simulation of a thru-focus series

SYNTAX `number simulation.GetDeltaFocus()`

## GetDeltaThickness

SUMMARY Returns the increment in thickness [Å] for a thru-thickness calculation

SYNTAX number simulation.GetDeltaThickness()

## GetDivergence

SUMMARY Returns the convergence angle (mrad) for the calculation

SYNTAX number simulation.GetDivergence()

## GetEndFocus

SUMMARY Returns the last focus [Å] for a simulation of a thru-focus series

SYNTAX number simulation.GetEndFocus()

## GetEndThickness

SUMMARY Returns the last thickness [Å] for a thru-thickness calculation

SYNTAX number simulation.GetEndThickness()

## GetExitWave

SUMMARY Returns an image containing the exit wave of the calculation

SYNTAX image simulation.GetExitWave( [number whichExitWave] [, number nX ] [, number nY ] [, number zoom ] )

DESCRIPTION Creates and returns an image of the specified exit wave for nX by nY unit cells, using a zoom factor, Defaults are: whichExitWave = 1, nX = 1, nY = 1, zoom = 1

## GetExitWaveModulus

SUMMARY Returns an image containing the modulus of the exit wave

SYNTAX image simulation.GetExitWaveModulus( [number whichExitWave] [, number nX ] [, number nY ] [, number zoom ] )

DESCRIPTION Creates and returns an image of the specified exit wave modulus for nX by nY unit cells, using a zoom factor, Defaults are: whichExitWave = 1, nX = 1, nY = 1, zoom = 1

## GetExitWavePhase

SUMMARY Returns an image containing the phase of the exit wave

SYNTAX image simulation.GetExitWavePhase( [number whichExitWave])

DESCRIPTION Creates and returns an image of the specified exit wave phase for nX by nY unit cells, using a zoom factor, Defaults are: whichExitWave = 1, nX = 1, nY = 1, zoom = 1

## GetFocus

SUMMARY Returns the focus [Å] for the simulation

SYNTAX number simulation.GetFocus()

## GetFocusSpread

SUMMARY Returns the focus [Å] for the simulation

SYNTAX number simulation.GetFocusSpread()

DESCRIPTION The focus spread refers to the effect of the chromatic aberration of the objective lens and contributes to the damping of the contrast transfer function

## GetImage

SUMMARY Returns an image containing the calculated simulated image

SYNTAX `image simulation.GetImage( [number whichImage] [, number nX ] [, number nY ] [, number zoom ] )`

DESCRIPTION Creates and returns an image of the specified simulated image for nX by nY unit cells, using a zoom factor, Defaults are: whichImage = 1, nX = 1, nY = 1, zoom = 1

## GetInnerAperture

SUMMARY Returns the inner radius of the objective lens aperture (1/Å)

SYNTAX `number simulation.GetInnerAperture()`

## GetOpticAxis

SUMMARY Returns the center of the optic axis in tilt angle (mrad) and azimuthal angle (degrees)

SYNTAX `number simulation.GetApertureCenter( number theta, number phi)`

## GetOpticAxisHK

SUMMARY Returns the center of the optic axis in (H,K) of the reciprocal space of the unit cell

SYNTAX `void simulation.GetOpticAxisHK( number cH, number cK)`

## GetOuterAperture

SUMMARY Returns the radius of the outer objective lens aperture (1/Å)

SYNTAX `number simulation.GetOuterAperture()`

DESCRIPTION Equivalent to GetAperture

## GetPhaseShift

SUMMARY Returns the phase shift for the phase plate in units of $\pi$

SYNTAX number simulation.GetPhaseShift()

## GetPhaseShiftRadius

SUMMARY Returns the radius for the phase plate in units of 1/Å

SYNTAX number simulation.GetPhaseShiftRadius()

## GetPhaseShiftRadius2

SUMMARY Returns the outer radius for the phase plate in units of 1/Å

SYNTAX number simulation.GetPhaseShiftRadius2()

## GetPotential

SUMMARY Returns an image containing the calculated 2D projected potential

SYNTAX image simulation.GetPotential( [number whichPotential] [, number nX ] [, number nY ] [, number zoom ] )

DESCRIPTION Creates and returns an image of the specified potential for nX by nY unit cells, using a zoom factor, Defaults are: whichPotential = 1, nX = 1, nY = 1, zoom = 1

## GetStartFocus

SUMMARY Returns the starting focus (Å) for a thru-focus series

SYNTAX number simulation.GetStartFocus()

## GetStartThickness

SUMMARY Returns the starting thickness (Å) for a thru-thickness series

SYNTAX `number simulation.GetStartThickness()`

## GetThickness

SUMMARY Returns the thickness (Å) for the simulation

SYNTAX `number simulation.GetThickness()`

## GetTilt

SUMMARY Returns the tilt angle of the specimen in mrad and the azimuthal angle of specimen tilt with respect to the horizontal axis in degrees

SYNTAX `void simulation.GetTilt(number theta, number phi)`

## GetTiltAngle

SUMMARY Returns the tilt angle of the specimen in mrad

SYNTAX `number simulation.GetTiltAngle()`

## GetTiltDirection

SUMMARY Returns the azimuthal angle of specimen tilt with respect to the horizontal axis in degrees

SYNTAX `number simulation.GetTiltDirection()`

## GetTiltH

SUMMARY Gets the h value of the center of laue circle (specimen tilt)

SYNTAX `number simulation. GetTiltH()`

## GetTiltHK

SUMMARY Returns the center of Laue circle in (H,K) of the reciprocal space of the unit cell

SYNTAX `void simulation.GetTiltHK( number cH, number cK)`

## GetTiltK

SUMMARY Gets the k value of the center of laue circle (specimen tilt)

SYNTAX `number simulation.GetTiltK()`

## GetVibration

SUMMARY Gets the vibration of the “specimen” along x and y

SYNTAX `void simulation.GetVibration(number variable vX, numberVariable vY)`

## GetVibrationX

SUMMARY Gets the vibration of the “specimen” along x

SYNTAX `number simulation.GetVibrationX()`

## GetVibrationY

SUMMARY Gets the vibration of the “specimen” along y

SYNTAX `number simulation.GetVibrationY()`

## GetVoltage

SUMMARY Returns the voltage of the microscope for the simulation (kV)

SYNTAX `number simulation.GetVoltage()`

## LoadExitWave

SUMMARY Returns an image containing the exit wave of the calculation

SYNTAX image simulation.LoadExitWave( [number whichExitWave])

DESCRIPTION Returns the specified exit wave as an image. Default value for which exit wave if not specified is 1. The image will have the sampling of the simulation and the angle of the unit cell.

## LoadExitWaveModulus

SUMMARY Returns an image containing the modulus of the exit wave

SYNTAX image simulation.LoadExitWaveModulus( [number whichExitWave] )

DESCRIPTION Returns the specified exit wave modulus as an image. Default value for which exit wave if not specified is 1. The image will have the sampling of the simulation and the angle of the unit cell.

## LoadExitWavePhase

SUMMARY Returns an image containing the phase of the exit wave

SYNTAX image simulation.LoadExitWavePhase( [number whichExitWave])

DESCRIPTION Returns the specified exit wave phase. Default value for which exit wave if not specified is 1. The exit wave phase image will have the sampling of the simulation and the angle of the unit cell.

## LoadImage

SUMMARY Returns an image containing the calculated simulated image

SYNTAX image simulation.LoadImage( [number whichImage])

DESCRIPTION Returns the specified image. Default value for which image if not specified is 1. The image will have the sampling of the simulation and the angle of the unit cell.

---

## LoadPotential

SUMMARY Returns an image containing the calculated 2D projected potential

SYNTAX image simulation.LoadPotential( [number whichPotential] )

DESCRIPTION Returns the specified potential as an image. Default value for which potential if not specified is 1. The image will have the sampling of the simulation and the angle of the unit cell.

## PropagateWave

SUMMARY Calculates a 3D complex volume containing the wave function at each slice for the current simulation up to a given thickness.

SYNTAX Image3D simulation.PropagateWave(number thickness)

DESCRIPTION Returns a 3D complex volume containing the wave function up to a given thickness. In order to see the wave, the volume image must be displayed in the script, such as Image3D wave = simulation.PropagateWave(200) ; wave.show() ;

## SetAperture

SUMMARY Sets the outer objective lens aperture (1/Å)

SYNTAX void simulation.SetAperture( number )

## SetApertureAngle

SUMMARY Sets the outer objective lens aperture in mradians

SYNTAX void simulation.SetApertureAngle( number )

## SetApertureCenter

SUMMARY Sets the center of the objective lens aperture

SYNTAX `void simulation.SetApertureCenter( number theta, number phi)`

## SetApertureHK

SUMMARY Sets the center of the objective lens aperture in (H,K) of the reciprocal space of the unit cell

SYNTAX `void simulation.SetApertureHK(number cH,Number cK )`

## SetCs

SUMMARY Sets the Spherical Aberration Cs in mm

SYNTAX `void simulation.SetCs( number )`

## SetCs5

SUMMARY Sets the $5^{th}$ order Spherical Aberration Cs5 in mm

SYNTAX `void simulation.SetCs5( number )`

## SetDeltaFocus

SUMMARY Sets the Incremental focus (Å) for a thru-focus series

SYNTAX `void simulation.SetDeltaFocus( number )`

## SetDeltaThickness

SUMMARY Sets the incremental thickness (Å) for a thru-thickness series

SYNTAX `void simulation.SetDeltaThickness( number )`

## SetDivergence

SUMMARY Sets the convergence angle (mrad) for the calculation

SYNTAX `void simulation.SetDivergence( number )`

## SetEndFocus

SUMMARY Sets the ending value for focus [Å] in a thru-focus series

SYNTAX `void simulation.SetEndFocus( number )`

## SetEndThickness

SUMMARY Sets the ending value for thickness [Å] in a thru-thickness series

SYNTAX `void simulation.SetEndFocus( number )`

## SetFocus

SUMMARY Sets the focus (Å) for the calculation

SYNTAX `void simulation.SetFocus( number )`

## SetFocusSpread

SUMMARY Sets the focus Spread (Å) associated with the chromatic aberration of the objective lens for the calculation

SYNTAX `void simulation.SetFocusSpread( number )`

## SetInnerAperture

SUMMARY Sets the inner objective lens aperture (1/Å)

SYNTAX `void simulation.SetInnerAperture( number )`

## SetOpticAxis

SUMMARY Sets the center of the optic axis in tilt angle (mrad) and azimuthal angle (degrees)

SYNTAX void simulation.SetOpticAxis( number theta , number phi)

## SetOpticAxisHK

SUMMARY Sets the center of the optic axis in (H,K) of the reciprocal space of the unit cell real

SYNTAX void simulation.SetOpticAxisHK( number cH, number cK)

## SetOuterAperture

SUMMARY Sets the outer objective lens aperture (1/Å)

SYNTAX void simulation.SetOuterAperture( number )

## SetPhaseShift

SUMMARY Sets the phase shift for the phase plate in units of $\pi$

SYNTAX void simulation.SetPhaseShift( number )

## SetPhaseShiftRadius

SUMMARY Sets the radius for the phase plate in units of 1/Å

SYNTAX void simulation.SetPhaseShiftRadius( number )

## SetPhaseShiftRadius2

SUMMARY Sets the outer radius for the phase plate in units of 1/Å

SYNTAX void simulation.SetPhaseShiftRadius( number )

DESCRIPTION If the second radius is set greater than the PhaseShiftRadius, the beams are blocked between PhaseShiftRadius and PhaseShiftRadius2

## SetStartFocus

SUMMARY Sets the starting focus (Å) for a thru-focus series

SYNTAX void simulation.SetStartFocus( number )

## SetStartThickness

SUMMARY Sets the starting thickness for a thru-thickness series

SYNTAX void simulation.SetStartThickness( number )

## SetThickness

SUMMARY Sets the thickness (Å) for the calculation

SYNTAX void simulation.SetThickness( number )

## SetTiltAngle

SUMMARY Sets the tilt angle of the specimen in mrad

SYNTAX void simulation.SetTiltAngle( number )

## SetTiltDirection

SUMMARY Sets the azimuthal angle of specimen tilt with respect to the horizontal axis in degrees

SYNTAX void simulation.SetTiltDirection( number )

## SetTiltH

SUMMARY Sets the h value of the center of laue circle (specimen tilt)

SYNTAX void simulation.SetTiltH( number )

## SetTiltHK

SUMMARY Sets the h,k values of the center of laue circle (specimen tilt)

SYNTAX void simulation.SetTiltHK( number h,number k)

## SetTiltK

SUMMARY Sets the k value of the center of laue circle (specimen tilt)

SYNTAX void simulation.SetTiltK( number )

## SetVibration

SUMMARY Sets the vibration of the “specimen” along x and y

SYNTAX void simulation.SetVibration( number vibX, number vibY)

## SetVibrationX

SUMMARY Sets the vibration of the “specimen” along x

SYNTAX void simulation.SetVibrationX( number )

## SetVibrationY

SUMMARY Sets the vibration of the “specimen” along y

SYNTAX void simulation.SetVibrationY( number )

## SetVoltage

SUMMARY Sets the voltage of the microscope for the simulation (kV)

SYNTAX `void simulation.SetVoltage( number )`

## ShowExitWave

SUMMARY Displays a calculated exit wave

SYNTAX `void simulation.ShowExitWave( [number whichExitWave] [, number nX ] [, number nY ] [, number zoom ] )`

DESCRIPTION Creates and displays the specified exit wave for nX by nY unit cells, using a zoom factor, Defaults are: whichExitWave = 1, nX = 1, nY = 1, zoom = 1

## ShowExitWaveModulus

SUMMARY Displays the modulus of a calculated exit wave

SYNTAX `void simulation.ShowExitWaveModulus( [number whichExitWave] [, number nX ] [, number nY ] [, number zoom ] )`

DESCRIPTION Creates and displays the modulus of the specified exit wave for nX by nY unit cells, using a zoom factor, Defaults are: whichExitWave = 1, nX = 1, nY = 1, zoom = 1

## ShowExitWavePhase

SUMMARY Displays the phase of a calculated exit wave

SYNTAX `void simulation.ShowExitWavePhase( [number whichExitWave ] [, number nX ] [, number nY ] [, number zoom ] )`

DESCRIPTION Creates and displays the phase of the specified exit wave for nX by nY unit cells, using a zoom factor, Defaults are: whichExitWave = 1, nX = 1, nY = 1, zoom = 1

## ShowImage

| | |
|---|---|
| SUMMARY | Displays a calculated image |
| SYNTAX | void simulation.ShowImage( [number whichImage ] [, number nX ] [, number nY ] [, number zoom ] ) |
| DESCRIPTION | Creates and displays the specified image for nX by nY unit cells, using a zoom factor, Defaults are: whichImage = 1, nX = 1, nY = 1, zoom = 1 |

## ShowPotential

| | |
|---|---|
| SUMMARY | Displays a calculated 2D projected potential |
| SYNTAX | void simulation.ShowPotential( [number whichPotential ] [, number nX ] [, number nY ] [, number zoom ] ) |
| DESCRIPTION | Creates and displays the specified exit wave for nX by nY unit cells, using a zoom factor, Defaults are: whichPotential = 1, nX = 1, nY = 1, zoom = 1 |

**Example:**

```
// Precession Tilt series
// This is summing over the power-spectrum of the exit wave function
// by spinning the beam in a circle. The beam tilt is theta (30 mrad).
// The increment in the azimuthal angle is dphi (6 degrees)
// A table of HKL values for different thicknesses is shown
// For illustration purposes, a precession image is also calculated


number theta = 30                 // The tilt angle in mrad
number phi = 0                    // Tilt angle (degrees) with respect to a-axis
number dphi = 6                   // increments in tilt angle (degrees)

simulation sim = getsimulation()  // Get the simulation

// We are making sure that everything has been calculated and is current
sim.calculateall()

image xw = sim.loadexitwave()     // Declare and load the exit wave
image im = sim. loadimage()       // Declare and load the image
image sumim = im ; sumim = 0 ;    // Declare the sum for the images and zero
```

```
image sumps = xw ; sumps = 0 ;        // Declare the sum for the powerspectrum
                                      // and zero

OpenResultsWindow()

for(number thickness = 10; thickness <= 100; thickness += 10) {
      sim.setthickness(thickness)
      number i = 0                                // declare and  initialize our counter
      for(phi = 0 ; phi < 360; phi += dphi) {     // loop over the azimuthal angle
            sim.settilt(theta,phi)                // set the tilt of the specimen
                                                  // this is equivalent to the tilting the beam
            sim.calculateexitwave()               // Calculate the new exit wave
            sim.calculateimage()                  // Calculate the new image
            sumim += sim. loadimage()             // Add the image to the sum
            xw = sim. loadexitwave()              // Load the exit wave
            xw.fft()                              // Fourier transform to get the frequency
                                                  // complex coefficients
            xw *= conjugate(xw)                   // Set the complex PowerSpectrum
                                                  // If we had  used xw.ps() to get the
                                                  // power spectrum we would have had a real
                                                  // image in “real” space
            sumps += xw                           // Add the powerspectrum to the sum
            i++                                   // Keep track of the count
            print("phi = "+phi)                   // Just to know where we are in the loop
      }
      sumim /= i                                  // Divide by the number of terms in the sum

      // Create a rectangular image of size 1024 by 1024 of sampling 0.1 Å (default)
      image precessionImage = sim.createimage(sumim,1024)

      precessionImage.setname("Image Precession")
      precessionImage.show()                      // Show the summed images



      sumps /= i                                  // Divide by the number of terms in the sum
      sumps.sqrt()                                // To compare with the Scattering factors

      // Create a rectangular image of size 1024 by 1024 out to gMax = 4 1/Å
      // with a convergence angle of 0.2 mrad
      image precessionPS = sim.createfrequencyimage(sumps,1024,0.2,4)
      precessionPS.setname("Power Spectrum Precession Thickness "+ sim.getthickness())

      precessionPS.show()                         // Show the summed power spectrum
      sumps.setname("thickness " + sim.getthickness())
      sim.createhkltable(sumps)
}
```