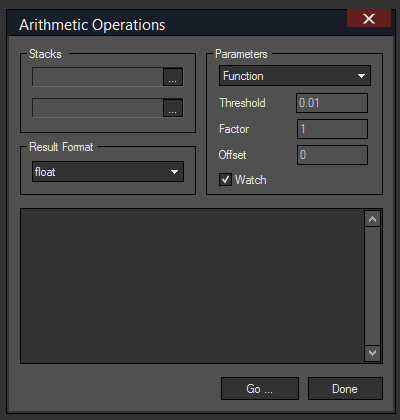
# The Arithmetics Plugin



The Analysis/Arithmetics plugin is used to create new data stacks from already existing data stacks and the relation between the old and the new data is specified by a function.

One or two Stacks can be selected (upper left panel of the Arithmetic Operations dialog). The format of the data values of the output stack is selected below.

On the right upper panel, you either select a pre-determined function (Scale, Add, Subtract, Multiply, Divide, Offset, Invert, Normalize) with obvious meanings of the parameters and requiring one (scale, offset, invert, normalize) or two (add, subtract, multiply, divide) input stacks.

The last option (Function) allows to input an arbitrary term (edit box in the lower part) which is then evaluated for every position and of the input stack(s) and determines the values of the output stack.

The function parser expression executes pointwise through the input stack and therefore has some restrictions compared to a general treatment of input data with a script containing loops and conditions (although simple conditions can be included).

The output stack can re-evaluate itself, should the data of the input stack change (Watch checkbox). And a convenient way to modify the function later on is Right-Click/Manage and Manipulate Data/Edit Stack (Ctrl.+Shift+E).

The Function parser which is used to evaluate the custom expressions is explained below.

## The Function Parser

The dimensions of the output stack will equal the dimensions of the input stack. The given parser expression is evaluated at every position and the resulting value is stored in the output stack.

A very simple expression is: r

which just replicates the input stack (because r denotes the actual value of the input stack).

Substitutions are possible, new variables can be assigned and are used in all remaining terms, the last term must then yield a value which determines the value of the output stack.

Example: a=5,r\*a

Creates a variable a with the value 5 and then computes r\*a which means the output stack will be the input stack scaled by five.

Values can be inserted in scientific notation, eg. 3.5e-6

### Built-in variables

|  |  |
| --- | --- |
| r | Pixel value of the input stack |
| R | Pixel value of the second input stack (if given) |
| s, u, v, w | Pixel positions of the actual position in the input stack in 1st, 2nd, 3rd, 4th dimension |
| x, y, z, t | Physical positions of the actual position |

### **Built-in Operators**

|  |  |
| --- | --- |
| +, -, \*, / | The basic algebraic operations |
| <, >, == | Logical operators. Result in a logical value (0 – false, 1 – true) |
| ^ | Power |
| % | Modulo operator. Also works for real input (example: r % (2 \* pi) wraps to [0,2Pi)) |
| // | Integer division (b\*(a//b)+(a%b) equals a) |

### Built-in Constants

|  |  |
| --- | --- |
| pi | Pi |
| S, U, V, W | Total number of pixels of the input stack in 1st, 2nd, 3rd, 4th dimension |
| X, Y, Z, T | Total physical length of input stack in 1st, 2nd, 3rd, 4th dimension |
| x0, y0, z0, t0 | Physical offset of input stack in 1st, 2nd, 3rd, 4th dimension |

### Built-in common functions

A list of built-in functions can be obtained by pressing F2. Functions are used with parenthesis and comma separated arguments.

|  |  |
| --- | --- |
| sqrt | Square root |
| sin, cos, tan | Sine, cosine, tangent |
| asin, acos, atan, atan2 | Inverse sine, cosine, tangent (atan2(y,x)) |
| min, max | Minimum/maximum of two values |
| abs | Absolute value |
| floor, ceil | Integer value rounded down, up |
| exp, ln | Exponential, logarithm |

Example: sigma=5e-7,exp(-((x-(x0+X/2))^2+(y-(y0+Y/2))^2)/(2\*sigma^2))

calculates a 2D centered Gaussian in a stack with the dimensions of the selected input stack but without regarding the current values in the input stack.

### Data Access

Access to the current value of the input stack is via the variable r (and R for the second input stack if a second stack has been selected).

However, it is also possible to access different values in the actual data stack or values in a different data stack via variables s,u,v,w or x,y,z,t. In this case, the stack name (can access any data stack open in Imspector) has to be printed followed by a dot and val(s,u,v,w) or func(x,y,z,t).

Example: "ExpControl #2 {6}".val(s,u,v,w)

Is the same as r if the selected stack has the name “ExpControl #2 {6}”. (Press F2 to get a list of all known objects.)

The arguments (s,u,v,w) do not have to be in this order and can be complex expressions themselves (see example Rotate a 2D stack).

Example: "ExpControl #2 {6}".val(u,s,v,w)

For a square 2D stack this exchanges the 1st and 2nd dimension.

Notes:

* func(x,y,z,t) will interpolate if pixel positions are not hit directly
* val and func will return 0 if the given arguments are outside of the current data stack

### Built-in advanced concepts

Conditionals: Condition ? expression 1 : expression 2

Condition is a logical expression (everything not zero is regarded as true). Depending on the outcome either expression 1 (true) or expression 2 (false) is executed.

Example: a > b ? a : b

Is equivalent to max.

Random number generation.

Random numbers can be generated at each pixel position and the parameter for the random number generation can depend on the value of the input stack or an expression containing the value of the input stack.

|  |  |
| --- | --- |
| rand(max, min) | Equally distributed random numbers in [min, max) |
| gaussdev(sigma) | Normally distributed random numbers with a certain standard deviation. |
| poidev(avrg) | Poissonian distributed random numbers with a certain mean value. |

## Examples

### Calculate a 2D Gaussian peak at the center

Example: sigma=5e-7,exp(-((x-(x0+X/2))^2+(y-(y0+Y/2))^2)/(2\*sigma^2))

Can be used to calculate a 2D Gaussian peak at the center of the current data stack.

A shortcut is: gaussian2D(x-(x0+X/2),y-(y0+Y/2),5e-7)

where the width is given as full width half maximum (FWHM) and you can use 1D/2D/3D versions and as well for a Lorentzian function (use lorentzian1/2/3D in this case).

Note: Often made mistake is applying this function to a data stack with an integer data type and not changing the output data type to a floating point type.

### Rotate a 2D stack by an angle

Assuming the rotation should be anti-clockwise in the XY plane (first two dimensions) and the rotation angle is given in deg.

arad=alpha/180\*pi, stack.func((x-X/2)\*cos(arad)-(y-Y/2)\*sin(arad)+X/2,(x-X/2)\*sin(arad)+(y-Y/2)\*cos(arad)+Y/2,z,t)

where stack is the name of the input data stack and alpha is the rotation angle in deg. Interpolation is applied.

## Summary

The Arithmetic Operations dialog can be used to create derived data stacks calculating functions depending on values of input stacks. The functions are calculated point wise which restricts the flexibility compared to for example running a custom script on the data with Python. Nevertheless, advanced features like conditionals or generation of random numbers make it a versatile tool.