Table of Contents

Formula Editing

Formula editor is used in the following cases:

- Setting Column Formula
- Custom Fit Curve
- Cell editing in Table Editing

MagicPlot uses standard IEEE 754 double precision floating-point arithmetic. Double precision floating point takes 8 bytes per number and provides a relative precision of about 16 decimal digits and magnitude range from about 10^{-308} to about 10^{+308}.

Syntax Highlighting

MagicPlot formula editor highlights expression syntax. It also marks matching brackets:

```
x = $A;
a = 1.4499;
x0 = -1.232;
dx = 0.6936;
(a * exp(-ln(2) * (x-x0)^2 / dx^2))
+ (a * exp(-ln(2) * (x-(x0*-1))^2 / dx^2))
```

General Rules

Case Sensitivity

MagicPlot formula translator is generally **case sensitive**, i.e. you can write `sin` but not `Sin`.

Note that x and X are different variables. You can use this feature when naming curve parameters.

Entering Numbers

You can use only dot (`.`) as decimal separator, e.g. `12.45`. Comma (`,`) is used as function arguments separator, e.g. `max(3.56,17.865)`.

You can use e or E for scientific notation: `1.45e-3` or `1.45E-3`.

Using Spaces and Line Breaks

You can freely insert space characters and line breaks in formula, but do not break function names, numbers, operators. You do not need to enter special characters to indicate line break.

Cycle Variable

You can use a cycle variable in expression. Such variables are:

- `i` (row number) in *Set Column Formula* window
- `x` in curve equation

Using Table Data

Only in *Set Column Formula* window.

There are two functions to obtain current table cell values in formula:

- `col(A)` – returns the value of cell in column 'A' in the current (i-th) row. Equivalent to `cell(A, i)`.
- `cell(A, 3)` – returns the value in column A and row 3.

You can use either upper-case letters (A…Z, e.g. `col(B)`) or numbers (`1, 2, 3,..`, e.g. `col(1)`) in columns numeration in arguments of `col` and `cell` functions.

Example

- `col(A) + 15 + cell(B, i+1)`

What is "Not-a-Number returned at row #" Warning?

Some mathematical functions can be defined only on a certain interval. For example, square root (`sqrt(x)`) is not defined for negative numbers (all calculations in MagicPlot are made in real numbers, not complex). Hence if the argument of `sqrt` is negative, a Not-a-Number (NaN) is returned. If a NaN value occurs in some part of formula, the result of calculation will also be a NaN, and corresponding table cells will be empty.

The calculations are not terminated if NaN value occurs in some row(s).

In some cases you may want to check if a NaN values occurs in calculations. MagicPlot shows the warning "*Not-a-Number returned at row #*". This row number is the first row in which NaN value was returned. MagicPlot also highlights the function or operator which first produces NaN value.

Functions

You can see a list of all available functions and their descriptions in *Functions* tab in *Set Column Formula* window.

MagicPlot uses functions of Java programming language library `StrictMath` to evaluate `sin`, `cos`,

exp, etc. These functions are available from the well-known network library `netlib` as a “Freely Distributable Math Library”, `fdlibm` package. The same library is widely used in many scientific computing applications.

Trigonometric Functions

MagicPlot supports all standard trigonometric functions (`sin`, `cos`, etc.). All angles are always measured in radians for clarity.

You can use the following functions to convert angles units:

- `deg(a)` – converts angles input in radians to an equivalent measure in degrees.
- `rad(a)` – converts angles input in degrees to an equivalent measure in radians.

Examples

- `sin(rad(90))`
- `deg(asin(1))`

Constants

The predefined constants are:

- `pi, Pi, PI` – $\boldsymbol{\pi}$ = 3.1416... value (the ratio of circumference of a circle to its diameter).
- e – $\boldsymbol{e}$ = 2.7183... value (the base of the natural logarithms). **Note:** expression e^a is evaluated as `exp(a)`.
- `nan`, `NaN`, `NAN` – Not-a-Number value.
- `inf`, `Inf`, `infinity`, `Infinity` – positive infinity value which may be used in some calculations. **Note:** Write `-inf` for negative infinity.
- `eps` – machine epsilon, gives an upper bound of the relative error due to rounding in floating point arithmetic. **Note:** `eps = ulp(1) = 2^(-52) = 2,2204E-16`. (52 is the number of bits used to store fractional part of a number).

Boolean Logic

MagicPlot can interpret boolean logic expressions. Zero and negative values (`<=0`) are interpreted as `false` and positive values (`>0`) are interpreted as `true` similarly to C programming language. You can use simple logical operators which are described below. Use `1` as `true` and `0` as `false`.

'if' Function

The basic logical function is `if(condition, a, b)`. If `condition` argument is `true` (greater than 0) it returns the second argument (`a`), else returns the third argument (`b`).

Examples

- `if(col(A) >= 0, col(A), -col(A))` – evaluates absolute value of column A (certainly, you can use `abs(col(A))` for that).
- `if(col(B) >= 0, col(B), NaN)` – returns only positive values from column B. Negative values are replaced with `NaN` value. You can use this expression to filter negative values if you do not want to use them in future calculations. Note that "*Not-a-Number returned at row #*" warning can be shown for such expression.
- `if(col(A) > 0 & col(B) > 0, max(col(A), col(B)), NaN)`

Equality Checking

You have to be careful if you need to check equality of two values. Due to inaccuracy of computer calculations the result of evaluation is always approximate. For example, result of `sqrt(3)^2` is number `2.9999999999999996`, not exactly 3. The expression `sqrt(3)^2 == 3` is `false` (it returns `0`). Keep in mind that for convenience MagicPlot rounds numbers when showing on the screen, so this value will be shown as 3 in table if the number of showed fractional digits in MagicPlot preferences is not big enough.

Generally, if you want to check equality of two values you need to use some equality threshold for relative difference. That is, you should compare the modulus of relative difference of two values a and b with threshold `t`: `if(abs((a-b)/a) < t, …, …)`.

Examples

- `sqrt(3)^2 - 3` results something about `-4,4409E-16`
- `if(abs(sqrt(3)^2 - 3) / 3 < 1e-10, …, …)` – checks equality of `sqrt(3)^2` and 3 with a threshold of `1e-10`.

Operators

Operator	Description	Operator	Description
+	addition	==	equal to
-	subtraction	!=	not equal to
*	multiplication	<	less than
/	division	>	greater than
^	power	<=	less than or equal to
|	or	>=	greater than or equal to
&	and		

Operations Priority

Operators with **lower** precedence value are evaluated **earlier**. You can use brackets to change calculation sequence.

Expression is evaluated left-to-right, excluding repeated exponentiation operator ^. The ^ operator is

right-associative like in Fortran language (evaluated right-to-left; note that in general case `a^(b^c) ≠ (a^b)^c`). Hence `a^b^c` is evaluated as `a^(b^c)`.

The reason exponentiation is right-associative is that a repeated left-associative exponentiation operation would be less useful. Multiple appearances could (and would) be rewritten with multiplication: `(a^b)^c = a^(b*c)`.

Operations	Precedence	Associativity
( ) *(function call)*	1	
^	2	Right-to-left
- *(unary minus)*	3	
*, /	4	Left-to-right
+, -	5	Left-to-right
<, >, <=, >=	6	Left-to-right
==, !=	7	Left-to-right
&	8	Left-to-right
|	9	Left-to-right
= *(assignment)*	10	Left-to-right

Examples

- `1 + 2 * 3` returns 7.
- `(1 + 2) * 3` returns 9.
- `2*-3` returns `-6`.
- `-3^2` is equal to `-(3^2)`, because ^ priority is higher than that of unary minus. The result is `-9`.
- `(-3)^2` returns 9.
- `2^2^3` is equal to `2^(2^3)`, because ^ is right-associative operator. The result is 256.

Table Rows Evaluation Order

Rows are always evaluated one after another from the first to the last in range which was specified in 'Set Column Formula' window. Accordingly the row number `i` is incremented on each step.

Example

You can use this behaviour to calculate factorial: set 1 in the first row of column A and after that set formula `cell(A, i-1) * i` and rows interval from 2 to `100`. Note that formula is to be set for rows beginning from the second, and not from the first. You will get the factorial of row number (`i`).