MegaGraph V1

About the Author

DosaidSoft (aka Synaps3) is a one man programming company. I specialize in VB and Flash applications. If you would like to report a bug in this program please email: synaps3@hushmail.me. This program is released without a license as freeware (with no limitations), although I did put many weeks of work into this program. This program has no copy protection or other bloatware because it's tard to try to protect against it. If you appreciate the work I put into this program, buy it for $10.

Equations Section

This section has three panels: f(x), conditions, and color. The f(x) list contains all of your equations. To graph an equation, you first have to add one to the list by pressing the "Enter Equation" button. When you choose a custom equation, a list of functions (sin, cos, etc.) will appear to help you. If you are typing a more complex equation, use the reference in this help file to see the symbols, operators, and constants that can be used in your equation. When you add a new function to the list, a random color is automatically assigned. Click the color box afterward if you need to change it. This color represents the color of the line on the graph.

The conditions list contains if statements. For example, "if x < 3". This allows you to use certain functions only when the condition you specify is met. This is used for Piecewise Functions. You enter new conditions when you press the add button. **To edit a condition or equation, double click on the item in the list.**

Input & Output

This section allows you to plug numbers into your equation for either Y or X. First select the equation you want to use from the list of equations then just type your number in a box and press the Get X or Get Y button to get your answer. Getting the Y value is very quick and accurate, but getting the X value may take a few seconds and the result may not be accurate (when I say that I mean like 0.001 off, nothing major). If you are not getting all the answers, you can lower the sensitivity by clicking the bubble next to low. If it is a quadratic or something else with more than one result, it should give all of them. If you look under the tools menu, you may notice that there is no option to find zeros. Doing this is simple. Just put zero for Y (it should be zero by default) and press Get X. That will give you the zeros.

Use the View All button to see a list of all the points (x, y). **You can change the range and step of these points by changing the xmax, xmin, and xstep numbers. Xstep can be a decimal if you wish. Make sure you don't type something that will make the list have millions of answers because you will lock up the program.**

Graph Settings

This section contains the view settings for the graph. It allows you to change what part of the graph is focused on. It's like the window button on a graphing calculator. Pretty much all of this is self explanatory, so I am not going to go into too much detail. If you don't understand this section, then you shouldn't be using the program.

Info Log

This box displays information about the selected equation in the f(x) list. To see the information, you need to select one of the options in the tools menu. You will be able to select options such as domain, range, minimum, and maximum. After you choose, the results will be displayed in the box.

Tools Menu

Stat Plot/Regression - This allows you to enter a list of points and have a "best fit" equation generated. It's is similar to the Stat button on a graphing calculator. You add X and Y points to the list by typing in the boxes below each list. Press one of the buttons to generate an approximate equation. This equation can then be added to the equation list on the main window. Before pressing OK, remember to enable the Show Stat so you can see your data points on the graph along with your approximated equation.

Intersect - The intersect button allows you to approximate the intersection between two of your equations. Both equations have to be added to the list prior to using this feature. When you press the button a new window will appear. Select both of your equations there. You may also need to adjust the sensitivity of the algorithm. **If you are not getting all (or none) of the intersects, you should lower the sensitivity. If you are getting too many answers, you should raise it.** I did my best to make the results as concise as possible, so you should not get a million answers to read through. I put a lot of time into getting this algorithm right.

clsMathParser

Symbols, operators and functions

This version uses clsMathParser.

http://digilander.libero.it/foxes/mathparser/MathExpressionsParser.htm

<table>
<tr><th>Function</th><th>Description</th><th>Note</th></tr>
<tr><td>+</td><td>addition</td><td></td></tr>
<tr><td>-</td><td>subtraction</td><td></td></tr>
<tr><td>*</td><td>multiplication</td><td></td></tr>
<tr><td>/</td><td>division</td><td>35/4 = 8.75</td></tr>
<tr><td>%</td><td>percentage</td><td>35% = 0.35</td></tr>
<tr><td>\</td><td>integer division</td><td>35\4 = 8</td></tr>
<tr><td>^</td><td>raise to power</td><td>3^1.8 = 7.22467405584208 (°)</td></tr>
<tr><td>| |</td><td>absolute value</td><td>|-5|=5 (the same as abs)</td></tr>
<tr><td>!</td><td>factorial</td><td>5!=120 (the same as fact)</td></tr>
<tr><td>abs(x)</td><td>absolute value</td><td>abs(-5)= 5</td></tr>
<tr><td>atn(x), atan(x)</td><td>inverse tangent</td><td>atn(pi/4) = 1</td></tr>
<tr><td>cos(x)</td><td>cosine</td><td>argument in radiant</td></tr>
<tr><td>sin(x)</td><td>sin</td><td>argument in radiant</td></tr>
<tr><td>exp(x)</td><td>exponential</td><td>exp(1) = 2.71828182845905</td></tr>
<tr><td>fix(x)</td><td>integer part</td><td>fix(-3.8) = 3</td></tr>
<tr><td>int(x)</td><td>integer part</td><td>int(-3.8) = −4</td></tr>
<tr><td>dec(x)</td><td>decimal part</td><td>dec(-3.8) = -0.8</td></tr>
<tr><td>ln(x), log(x)</td><td>logarithm natural</td><td>argument x>0</td></tr>
<tr><td>logN(x,n)</td><td>N-base logarithm</td><td>logN(16,2) = 4</td></tr>
<tr><td>rnd(x)</td><td>random</td><td>returns a random number between x and 0</td></tr>
<tr><td>sgn(x)</td><td>sign</td><td>returns 1 if x >0 , 0 if x=0, -1 if x<0</td></tr>
<tr><td>sqr(x)</td><td>square root</td><td>sqr(2) =1.4142135623731, also 2^(1/2)</td></tr>
<tr><td>cbr(x)</td><td>cube root</td><td>→x, example cbr(2) = 1.2599, cbr(-2) = -1.2599</td></tr>
<tr><td>tan(x)</td><td>tangent</td><td>argument (in radian) x← k*▲/2 with k = ← 1, ← 2…</td></tr>
<tr><td>acos(x)</td><td>inverse cosine</td><td>argument -1 ↑ x ↑ 1</td></tr>
<tr><td>asin(x)</td><td>Inverse sine</td><td>argument -1 ↑ x ↑ 1</td></tr>
<tr><td>cosh(x)</td><td>hyperbolic cosine</td><td>→ x</td></tr>
<tr><td>sinh(x)</td><td>hyperbolic sine</td><td>→ x</td></tr>
<tr><td>tanh(x)</td><td>hyperbolic tangent</td><td>→ x</td></tr>
<tr><td>acosh(x)</td><td>Inverse hyperbolic cosine</td><td>argument x ← 1</td></tr>
<tr><td>asinh(x)</td><td>Inverse hyperbolic sine</td><td>→ x</td></tr>
<tr><td>atanh(x)</td><td>Inverse hyperbolic tangent</td><td>-1 < x < 1</td></tr>
<tr><td>root(x,n)</td><td>n-th root (the same as x^(1/n)</td><td>argument n ← 0 , x ← 0 if n even , →x if n odd</td></tr>
<tr><td>mod(a,b)</td><td>Division remainder</td><td>mod(29,6) = 5 mod(-29 ,6) = -5</td></tr>
</table>

Function	Description	Note
fact(x)	factorial	argument 0 ↑ x ↑ 170
comb(n,k)	combinations	comb(6,3) = 20 , comb(6,6) = 1
perm(n,k)	permutations	perm(8,4) = 1680 ,
min(a,b,...)	minimum	min(13,24) = 13
max(a,b,...)	maximum	max(13,24) = 24
mcd(a,b,...)	maximum common divisor	mcd(4346,174) = 2
mcm(a,b,...)	minimum common multiple	mcm(1440,378,1560,72,1650) = 21621600
gcd(a,b,...)	greatest common divisor	The same as mcd
lcm(a,b,...)	lowest common multiple	The same as mcm
csc(x)	cosecant	argument (in radiant) x← k*▲ with k = 0, ← 1, ← 2…
sec(x)	secant	argument (in radiant) x← k*▲/2 with k = ← 1, ← 2…
cot(x)	cotangent	argument (in radiant) x← k*▲ with k = 0, ← 1, ← 2…
acsc(x)	inverse cosecant	
asec(x)	inverse secant	
acot(x)	inverse cotangent	
csch(x)	hyperbolic cosecant	argument x>0
sech(x)	hyperbolic secant	argument x>1
coth(x)	hyperbolic cotangent	argument x>2
acsch(x)	inverse hyperbolic cosecant	
asech(x)	inverse hyperbolic secant	argument 0 ↑ x ↑ 1
acoth(x)	inverse hyperbolic cotangent	argument x<-1 or x>1
rad(x)	radiant conversion	converts radiant into current unit of angle
deg(x)	degree sess. conversion	converst sess. degree into current unit of angle
grad(x)	degree cent. conversion	converts cent. degree into current unit of angle
round(x,d)	round a number with d decimal	round(1.35712, 2) = 1.36
>	greater than	return 1 (true) 0 (false)
>=	equal or greater than	return 1 (true) 0 (false)
<	less than	return 1 (true) 0 (false)
<=	equal or less than	return 1 (true) 0 (false)
=	equal	return 1 (true) 0 (false)
<>	not equal	return 1 (true) 0 (false)
and	logic and	and(a,b) = return 0 (false) if a=0 or b=0
or	logic or	or(a,b) = return 0 (false) only if a=0 and b=0
not	logic not	not(a) = return 0 (false) if a ← 0 , else 1
xor	logic exclusive-or	xor(a,b) = return 1 (true) only if a ← b
nand	logic nand	nand(a,b) = return 1 (true) if a=1 or b=1
nor	logic nor	nor(a,b) = return 1 (true) only if a=0 and b=0
nxor	logic exclusive-nor	nxor(a,b) = return 1 (true) only if a=b
Psi(x)	Function psi	
DNorm(x,μ,σ)	Normal density function	→x, μ > 0 , σ > 0
CNorm(x,m,d)	Normal cumulative function	→x, μ > 0 , σ > 1
DPoisson(x,k)	Poisson density function	x >0, k = 1, 2, 3 ...
CPoisson(x,k)	Poisson cumulative function k = 1, 2,3 ...	x >0, k = 1, 2, 3 ...
DBinom(k,n,x)	Binomial density for k successes for n trials	k , n = 1, 2, 3…, k < n , x ↑1
CBinom(k,n,x)	Binomial cumulative for k successes for n trials	k , n = 1, 2, 3…, k < n , x ↑1
Si(x)	Sine integral	→ x
Ci(x)	Cosine integral	x >0
FresnelS(x)	Fresnel's sine integral	→ x

Function	Description	Note
FresnelC(x)	Fresnel's cosine integral	→ x
J0(x)	Bessel's function of 1st kind	x ⬅0
Y0(x)	Bessel's function of 2st kind	x ⬅0
I0(x)	Bessel's function of 1st kind, modified	x ↓0
K0(x)	Bessel's function of 2st kind, modified	x ↓0
BesselJ(x,n)	Bessel's function of 1st kind, nth order	x ⬅0 , n = 0, 1, 2, 3…
BesselY(x,n)	Bessel's function of 2st kind, nth order	x ⬅0 , n = 0,1, 2, 3…
BesselI(x,n)	Bessel's function of 1st kind, nth order, modified	x ↓0 , n = 0,1, 2, 3…
BesselK(x,n)	Bessel's function of 2st kind, nth order, modified	x ↓0 , n = 0,1, 2, 3…
HypGeom(x,a,b,c)	Hypergeometric function	-1 < x <1 a,b >0 c ⬅ 0, −1, −2…
PolyCh(x,n)	Chebycev's polynomials	→x , orthog. for -1 ⬆ x ⬆1
PolyLe(x,n)	Legendre's polynomials	→ x , orthog. for -1 ⬆ x ⬆1
PolyLa(x,n)	Laguerre's polynomials	→ x , orthog. for 0 ⬆ x ⬆1
PolyHe(x,n)	Hermite's polynomials	→ x , orthog. for −∞ ⬆ x ⬆ +∞
AiryA(x)	Airy function Ai(x)	→ x
AiryB(x)	Airy function Bi(x)	→ x
Elli1(x)	Elliptic integral of 1st kind	→ ← , 0 < k < 1
Elli2(x)	Elliptic integral of 2st kind	→ ← , 0 < k < 1
Erf(x)	Error Gauss's function	x >0
gamma(x)	Gamma function	→ x, x ⬅ 0, −1, −2, −3… (x > 172 overflow error)
gammaln(x)	Logarithm Gamma function	x >0
gammai(a,x)	Gamma Incomplete function	→ x a > 0
digamma(x) psi(x)	Digamma function	x ⬅ 0, −1, −2, −3…
beta(a,b)	Beta function	a >0 b >0
betaI(x,a,b)	Beta Incomplete function	x >0 , a >0 , b >0
Ei(x)	Exponential integral	x ⬅0
Ein(x,n)	Exponential integral of n order	x >0 , n = 1, 2, 3…
zeta(x)	zeta Riemman's function	x <-1 or x >1
Clip(x,a,b)	Clipping function	return a if x<a , return b if x>b, otherwise return x.
WTRI(t,p)	Triangular wave	t = time, p = period
WSQR(t,p)	Square wave	t = time, p = period
WRECT(t,p,d)	Rectangular wave	t = time, p = period, d= duty-cycle
WTRAPEZ(t,p,d)	Trapez. wave	t = time, p = period, d= duty-cycle
WSAW(t,p)	Saw wave	t = time, p = period
WRAISE(t,p)	Rampa wave	t = time, p = period
WLIN(t,p,d)	Linear wave	t = time, p = period, d= duty-cycle
WPULSE(t,p,d)	Rectangular pulse wave	t = time, p = period, d= duty-cycle
WSTEPS(t,p,n)	Steps wave	t = time, p = period, n = steps number
WEXP(t,p,a)	Exponential pulse wave	t = time, p = period, a= dumping factor
WEXPB(t,p,a)	Exponential bipolar pulse wave	t = time, p = period, a= dumping factor
WPULSEF(t,p,a)	Filtered pulse wave	t = time, p = period, a= dumping factor
WRING(t,p,a,fm)	Ringing wave	t = time, p = period, a= dumping factor, fm = frequency
WPARAB(t,p)	Parabolic pulse wave	t = time, p = period
WRIPPLE(t,p,a)	Ripple wave	t = time, p = period, a= dumping factor
WAM(t,fo,fm,m)	Amplitude modulation	t = time, p = period, fo = carrier freq., fm = modulation freq., m = modulation factor

Function	Description	Note
WFM(t,fo,fm,m)	Frequecy modulation	t = time, p = period, fo = carrier freq., fm = modulation freq., m = modulation factor
Year(d)	year	d = dateserial
Month(d)	month	d = dateserial
Day(d)	day	d = dateserial
Hour(d)	hour	d = dateserial
Minute(d)	minute	d = dateserial
Second(d)	second	d = dateserial
DateSerial(a,m,d)	Dateserial from date	a = year, m = month, d = day
TimeSerial(h,m,s)	Timeserial from time	h = hour, m = minute, s = second
time#	system time	
date#	system date	
now#	system timestamp	
Sum(a,b,...)	Sum	sum(8,9,12,9,7,10) = 55
Mean(a,b,...)	Arithmetic mean	mean(8,9,12,9,7,10) = 9.16666666666667
Meanq(a,b,...)	Quadratic mean	meanq(8,9,12,9,7,10) = 9.30053761886914
Meang(a,b,...)	Arithmetic mean	meang(8,9,12,9,7,10) = 9.03598945281812
Var(a,b,...)	Variance	var(1,2,3,4,5,6,7) = 4.66666666666667
Varp(a,b,...)	Variance pop.	varp(1,2,3,4,5,6,7) = 4
Stdev(a,b,...)	Standard deviation	Stdev(1,2,3,4,5,6,7) = 2.16024689946929
Stdevp(a,b,...)	Standard deviation pop.	Stdevp(1,2,3,4,5,6,7) = 2
Step(x,a)	Haveside's step function	Returns 1 if x ◄ a , 0 otherwise

Typical mixed math-physical expressions

1+(2-5)*3+8/(5+3)^2	sqr(2)
(a+b)*(a-b)	x^2+3*x+1
300 km + 123000 m	(3000000 km/s)/144 Mhz
256.33*Exp(-t/12 us)	(1+(2-5)*3+8/(5+3)^2)/sqr(5^2+3^2)
2+3x+2x^2	0.25x + 3.5y + 1
0.1uF*6.8kohm	sqr(4^2+3^2)
(12.3 mm)/(856.6 us)	(-1)^(2n+1)*x^n/n!
And((x<2);(x<=5))	sin(2*pi*x)+cos(2*pi*x)

Variables can be any alphanumeric string and must start with a letter

```
x  y  a1  a2  time  alpha   beta
```

Also the symbol "_" is accepted for writing variables in "programming style"..

```
time_1  alpha_b1   rise_time
```

Implicit multiplication is not supported because of its intrinsic ambiguity. So "xy" stands for a variable named "xy" and not for x*y. The multiplication symbol "*" cannot generally be omitted.

It can be omitted only for <u>coefficients</u> of the three classic math variables ***x, y, z***. It means that strings like “2x” and “2*x” are equivalent

```
2x  3.141y  338z^2   ↑   2*x  3.141*y  338*z^2
```

On the contrary, the following expressions are illegal.

```
2a  3(x+1)  334omega  2pi
```

Constant numbers can be integers, decimal, or exponential

```
2  -3234  1.3333  -0.00025   1.2345E-12
```

From version 4.2, MathParser accepts both decimal symbols "." or ",". See international setting

Physical numbers are numbers followed by a unit of measure

```
"1s"  for 1 second     "200m"  for 200 meters    "25kg"  for 25 kilograms
```

For better reading they may contain a blank

```
"1 s"   "200 m"   "25 kg"   "150 MHz"   "0.15 uF"   "3600 kohm"
```

They may also contain the following **multiplying factors**:

```
T=10^12  G=10^9  M=10^6  k=10^3  m=10^-3  u=10^-6  n=10^-9  p=10^-12
```

Functions are called by their function-name followed by parentheses. Arguments can be: numbers, variables, expressions, or even other functions

```
sin(x)   log(x)   cos(2*pi*t+phi)   atan(4*sin(x))
```

For functions which have more than one argument, the successive arguments are separated by commas (default)

```
max(a,b)     root(x,y)    BesselJ(x,n)    HypGeom(x,a,b,c)
```

Note. From version 4.2 , the argument separator depends on the MathParser decimal separator setting. If decimal symbol is point "." (i.e. 3.14) , the argument separator is ",". If it is comma "," (i.e. 3,14) , the argument separator is ";".

Logical expressions are now supported

```
x<1   x+2y >= 4    x^2+5x-1>0   t<>0   (0<x<1)
```

Logical expressions return always 1 (True) or 0 (False). Compact expressions, like “0<x<1” , are now supported; you can enter: `(0<x<1)` as well `(0<x)*(x<1)`

Numerical range can be inserted using logical symbols and Boolean functions. For example:

```
For   2<x<5             insert     (2<x)*(x<5)       or also (2<x<5)

For  x<2 , x>=10        insert     OR(x<2, x>=10)    or also   (x<2)+(x>=10)

For  -1<x<1             insert     (x>-1)*(x<1)      or  (-1<x<1)    or also
|x|<1
```

Percentage. (changed) Now it simply returns the argument divided by 100

```
3%     =>  returns the number  3/100 = 0.03
```

Math Constants supported are: Pi Greek (▲), Euler-Mascheroni (→) , Euler-Napier's (e), Goldean mean (↖). Constant numbers must be suffixed with # symbol (except pi-greek that can written also without a suffix for compatibility with previous versions)

```
pi = 3.14159265358979    or  pi# = 3.14159265358979

pi2# = 1.5707963267949  (▲ /2),  pi4# = 0.785398163397448  (▲ /4))

eu# = 0.577215664901533

e# = 2.71828182845905

phi# = 1.61803398874989
```

Note: pi-greek constant can be indicated with "pi" or "PI" as well. All other constants are case sensitive.

Angle expressions

This version supports angles in radians, sexagesimal degrees, or centesimal degrees. The right angle unit can be set by the property AngleUnit ("RAD" is the default unit). This affects all angle computation of the parser.

For example if you set the unit "DEG", all angles will be read and converted in degree

```
sin(120)  =>  0.86602540378444

asin(0.86602540378444)  =>  60

rad(pi/2)  =>  90         grad(400)  =>  360       deg(360)  =>  360
```

Angles can also be written in **ddmmss** format like for example 45° 12' 13"

```
sin(29°59'60")  =>  0.5            29°59'60"    =>  30
```

```
sin(29d 59m 60s)  =>  0.5            29d 59s 60m  =>  30
```

Physical Constants supported are:

Planck constant	**h#**	6.6260755e-34 J s
Boltzmann constant	**K#**	1.380658e-23 J/K
Elementary charge	**q#**	1.60217733e-19 C
Avogadro number	**A#**	6.0221367e23 particles/mol
Speed of light	**c#**	2.99792458e8 m/s
Permeability of vacuum (↘)	**mu#**	12.566370614e-7 T^2m^3/J
Permittivity of vacuum (⇨)	**eps#**	8.854187817e-12 C^2/Jm
Electron rest mass	**me#**	9.1093897e-31 kg
Proton rest mass	**mp#**	1.6726231e-27 kg
Neutron rest mass	**mn#**	n 1.6749286e-27 kg
Gas constant	**R#**	8.31451 m^2kg/s^2k mol
Gravitational constant	**G#**	6.672e-11 $m^3/kg\ s^2$
Acceleration due to gravity	**g#**	9.80665 m/s^2

Physical constants can be used like any other symbolic math constant.

Just remember that they have their own dimension units listed in the above table.

Example of physical formulas are:

```
  m*c# ^2          1/(4*pi*eps#)*q#/r^2       eps# * S/d

 sqr(m*h*g#)       s0+v*t+0.5*g#*t^2
```