Function Summary

Summary of functions contained in SolvePro version 8.

Although defined here using lower case, note that function names can be used using upper or lower case or a mixture of cases. For example Asin, ASIN and aSin are considered to be identical and all represent the function asin.

Trigonometric functions will operate in whatever mode is set in the ActiveEquation object. For example if set to degrees then asin(0.7071) will return 45 (degrees). If set to radians asin(0.7071) will return π/4 (radians).
.

average(vector), mean(vector)
Returns the arithmetic mean of a *vector* of numbers

abs(x)
Returns the absolute positive value of x

asin(x)
Returns the angle where x is the sine of that angle

acos(x)
Returns the angle where x is the cosine of that angle

atan(x)
Returns the angle where x is the tangent of that angle

atan2(x, y)
Returns the angle where y divided by x is the tangent of that angle

asinh(x)
Returns the angle where x is the hyperbolic sine of that angle

acosh(x)
Returns the angle where x is the hyperbolic cosine of that angle

atanh(x)
Returns the angle where x is the hyperbolic tangent of that angle

cos(x)
Returns the cosine of the angle x

cosh(x)
Returns the hyperbolic cosine of x

ceil(x)
Returns the ceiling of the specified number, which is the smallest integer that is greater than or equal to x

dnorm(x, mean, sd, max), dnorm(vector, mean, sd, max)
Generates a single value or a vector of numbers distributed according to the normal distribution where
x is a single value at which the normal distribution should be calculated
vector contains a series of values at which to calculate the normal distribution
mean is the centre of the distribution
sd is the standard deviation
max is the maximum value

dlognorm(x, mean, sd, max), dlognorm(vector, mean, sd, max)
Generates a single value or a vector of numbers distributed according to the lognormal distribution where
  x is the single value at which the lognormal distribution should be calculated
  vector contains a series of values at which to calculate the lognormal distribution
  mean is the centre of the distribution
  sd is the standard deviation
  max is the maximum value

dlognorm3(x, mean, sd, gamma, max), dlognorm3(vector, mean, sd, gamma, max)
Generates a single value or a vector of numbers distributed according to the three-parameter lognormal distribution where
  x is the single value at which the lognormal distribution should be calculated
  vector contains a series of values at which to calculate the lognormal distribution
  mean is the centre of the distribution
  sd is the standard deviation
  gamma is the threshold (offset) value
  max is the maximum value

exp(x)
Returns the value of *e* raised to the power x. i.e. value = e^x

e
Returns the mathematical constant *e* which is the base of natural logarithms, i.e. e = 2.7182818...

factorial(x), !(x)
Returns the factorial of specified number, i.e. 1*2*3* … (x-2)*(x-1)*x

floor(x)
Returns the floor of the specified number, which is the largest integer that is lower than or equal to x

g
Returns the value of the acceleration due to gravity i.e. g = 9.80665 ms^{-2}

hide()
This function no longer has any effect and has been removed.

histogram(vector, n), hist(vector, n)
Generates a histogram from the *vector* by dividing the range of values in *vector* into n bins or buckets. The result is a new table containing table[0] as the median bin values and table[1] containing the corresponding number of items in each bin.

if(condition, value if true, value if false)
If condition evaluates to true, then *value if true* is returned. If condition evaluates to false, then *value if false* is returned. This can be used to convey messages to the reader. For example,
  if(iseven(x), "Yes", "No") returns either Yes or No depending on whether x is an even number

imag(x), im(x)
Returns the imaginary part of the complex number x

int(x)
Returns the integer part of the real number x

iseven(x)
Returns 1 if the real number x is even, 0 if it is odd

isodd(x)
Returns 1 if the real number x is odd, 0 if it is even

isvar(str, val), var(str, val)
Returns the value of *str* if *str* already exists. If *str* does not exist, *str* is assigned the value of *val* and *val* is returned.

kurtosis(vector), kurt(vector)
Returns the kurtosis of the vectors of values in *vector*

ln(x)
Returns the natural logarithm of the number x

log(x), log10(x)
Returns the logarithm to base 10 of the number x

log2(x)
Returns the logarithm to base 2 of the number x

min(x, y)
Returns the minimum value of x and y

max(x, y)
Returns the maximum value of x or y

mean(vector), average(vector)
Returns the arithmetic mean of a *vector* of numbers

median(vector)
Returns the median value of a *vector*

mod(x, y), x%y
Modulus operator. Returns the remainder of x / y

norm(complex)
Returns the sum of the squares of a complex number such that *norm*(a + bi) = $a^2 + b^2$

pow(x, y), x^y
Returns the value of x raised to the power y.

pi(), π
Returns the mathematical constant pi. π = 3.14159...

polar(radius, angle), pol(radius, angle)
Returns a complex number x + yi corresponding to x and y Cartesian coordinates from the radius and angle polar coordinates supplied. For example if set to use degrees
  pol(1.0, 60) returns the complex number 0.5 + 0.866i

percentile(vector, y)
Returns the yth percentile value of the *vector*.

rangenum(start, end, n)
Generates a vector containing a series of numbers from *start* to *end* divided into *n* parts.

rangeint(start, end, interval)
Generates a vector containing a series of numbers from *start* to *end* with the specified *interval*. Note that if the interval from the last value and the *end* is not an integer multiple of the *interval*, then the *end* value is not included

real(x), re(x)
Returns the real part of the complex number x

root(x, y)
Returns the yth root of the complex number x

rnorm(number, mean, sd), rnorm(n, μ, σ)
Generates a vector of random numbers distributed according to the normal distribution where
  number (n) is the number of values to be generated
  mean (μ) is the centre of the distribution
  sd (σ) is the standard deviation

runif(number, min, max)
Generates a vector of random numbers distributed uniformly across the specified range where
number is the number of values to be generated
min is the minimum value required
max is the maximum number required

rtri(number, min, max, mode)
Generates a vector of random numbers distributed according to the triangular distribution where
number is the number of values to be generated
min is the minimum value required
max is the maximum number required
mode is the most likely or 'best-guess' value

rbeta(number, alpha, beta)
Generates a vector of random numbers distributed according to the beta distribution where
number is the number of values to be generated
alpha is a shape parameter
beta is a shape parameter

rpert(number, min, max, mode)
Generates a vector of random numbers distributed according to the PERT distribution where
number is the number of values to be generated
min is the minimum value required
max is the maximum number required
mode is the most likely or 'best-guess' value

rmpert(number, min, max, mode, weight)
Generates a vector of random numbers distributed according to the modified PERT distribution where
number is the number of values to be generated
min is the minimum value required
max is the maximum number required
mode is the most likely or 'best-guess' value
weight is the weighting applied to the mode

randomreal(min, max)
Returns a random number between min and max

randomint(min, max)
Returns a random integer between min and max

randomsample(vector, nSamples, nMeans), rsample(vector, nSamples, nMeans)
Randomly samples nSamples elements from *vector*, calculates the mean value from each sample and returns a vector containing nMean values.

round(x)
Returns the nearest integer to the real number x. Note that 0.5 rounds up to 1

sum(vector), vsum(vector)
Returns the sum of the values contained in *vector*

sumsqr(vector), vsumsqr(vector)
Returns the sum of the squares of the values contained in *vector*

show()
This function no longer has any effect and has been removed.

sqrt(x), sqr(x), √(x)
Returns the square of the complex number x

sin(x)
Returns the sine of the angle x

stdev(vector)
Returns the sample standard deviation of the values contained in *vector*

stdevs(vector)
Returns the sample standard deviation of the values contained in vector

stdevp(vector)
Returns the population standard deviation of the values contained in vector

skew(vector)
Returns the skew of the values contained in *vector*

sort(vector), vsort(vector)
Generates a new vector containing the elements of *vector* sorted from lowest to highest.

tan(x)
Returns the tangent to the angle x

tanh(x)
Returns the hyperbolic tangent of the angle x

vadd(vector1, vector2), vjoin(vector1, vector2)
Returns a new vector with vector 2 appended at the end of vector1

vremove(vector, pos, n)
Removes *n* items starting at position *pos* from *vector* and returns a new vector

vreplace(vector, pos, x)
Replaces the item at position *pos* with the value *x*

vtable(vector1, vector2)
Returns a table containing the two vectors vector1 and vector2

var(str, val), isvar(str, val, y)
Returns the value of *str* if *str* already exists. If *str* does not exist, *str* is assigned the value of *val* and *val* is returned.

vat(vector, pos), vector[pos], table[n][pos]
Returns the value contained at position *pos* in *vector*. To return the value at position *pos* in vector *n* in a table use *table[n][pos]*

vmax(vector)
Returns the maximum value of the items in *vector*

vmin(vector)
Returns the minimum value of the items in *vector*

vsum(vector), sum(vector)
Returns the sum of the items in *vector*

vsumsqr(vector), sumsqr(vector)
Returns the sum of the squares of the items in *vector*

vview(vector, n, mode)
Displays the first *n* items in *vector*. If mode is 0, the values are displayed as a list. If mode is 1 the values are displayed as a simple table. Note that the displayed list or table is regenerated when the document is recalculated so any formatting applied to a table will be lost.

vsort(vector), sort(vector)
Returns a new vector with the contents of *vector* sorted in order from lowest value to highest value

vreverse(vector)
Returns a new vector with the elements of *vector* in arranged in reverse order

Mathematical operators

Operator	Description
+	add
-	subtract
* or x	multiply
/ or ÷	divide
%	modulus
^	power

Logical operators

Operator	Description
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
= or ==	equal to

Vector processing

ActiveEquation functions work within SolvePro by performing whatever calculation is defined and then returning the result. Processing vectors of numbers requires a different approach since there may be huge numbers of similar calculations to be performed using different variables within the one ActiveEquation object. This type of processing is defined by checking the 'Use vectors' box in an ActiveEquation object. SolvePro deals with this issue by compiling each ActiveEquation calculation into a small section of code and then supplying this code with each of the variables one at a time to perform a calculation. Ultimately this produces a vector of numbers containing all the results of the individual calculations.

Many of the full range of functions in SolvePro are not directly applicable to mathematical processing. As a consequence vector processing operates on a subset of the all the functions which have been defined in SolvePro. The full list of functions and operators which can be used with vectors of numbers are listed below.

Operators and functions

Operator / Function	Description
+, -, *, /	Add, subtract, multiply, divide
sqrt(x) or √(x)	Square root
pow(x, y) or x^y	Power
root(x, y)	Root
x%y or mod(x, y)	Modulus
abs(x)	Absolute value
sig(x)	Sign
min(x, y), max(x, y)	Minima, maxima
ceil(x), floor(x)	Ceiling, floor
round(x)	Rounding
cos(x), sin(x), tan(x)	Trigonometrical functions
acos(x), asin(x), atan(x), atan2(x)	Inverse trigonometrical functions
cosh(x), sinh(x), tanh(x)	Hyperbolic functions
acosh(x), asinh(x), atanh(x)	Inverse hyperbolic functions
exp(x), In(x)	Natural logarithms (Base *e*)
log(x) or log10(x)	Logarithm to base 10
log2(x)	Logarithm to base 2
e, pi, g	Built-in constants