DAQBench

32-bit ActiveX controls for
Measurement and Automation

DQAnalysis Controls Reference

Contents

DQAnalysis ActiveX control is an analysis tool that includes linear algebra, vector, matrix, complex number, and FFT operation.

ArrayElementCount Method

Syntax

Function DQAnalysis.ArrayElementCount (array as Variant) As long

Purpose

Returns the total number of elements in the input array.

Parameters

Return Value		The total number of elements in input array
Input	*array*	1D or Multidimensional Array.

Parameter Discussion

The input array is 0 based. This means that the first index is 0, not 1.

ArraySet Method

Syntax

Sub DQAnalysis.**ArraySet** (*x As Variant*, *val As Vatiant*)

Purpose

Sets the elements of the array *x* to a constant value. This function works with multidimensional arrays..

Parameters

Input	*x*	1D or Multidimensional array	Input array.
	val	Variant	The constant value.

ArrayClear Method

Syntax

Sub DQAnalysis.**ArrayClear** (*x As Variant*)

Purpose

Clear all elements of the x array to zero value. This function works with multidimensional arrays.

Parameters

Input	*x*	1D or Multidimensional array	Input array.

ArrayCopy Method

Syntax

Function DQAnalysis.**ArrayCopy** (*x As Variant)* As Variant

Purpose

Copies the elements of the x array to a new array. This function is useful to duplicate arrays for in-place operations. This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Duplicated array.
Input	*x*	1D or Multidimensional array	Input array.

ArrayClip Method

Syntax

Function DQAnalysis.**ArrayClip** (*x As Variant*, *upper As Variant*, *lower As Variant*) As Variant

Purpose

Clips the input array values. The range of the resulting output array is (lower : upper).

Parameters

Return Value		1D or Multidimensional array	Clipped array
Input	*x*	1D or Multidimensional array	Input data.
	upper	Variant	Upper limit
	lower	Variant	Lower limit

ArraySum Method

Syntax

Function DQAnalysis.**ArraySum** (*x As Variant*) As Variant

Purpose

Finds the sum of the elements in the input array. This function works with multidimensional arrays.

Parameters

Return Value		Variant	Sum of elements.
Input	*x*	1D or Multidimensional array	Input array.

ArrayProduct Method

Syntax

Function DQAnalysis.**ArrayProduct** (*x As Variant*) As Variant

Purpose

Finds the product of the n elements of the input array. This function works with multidimensional arrays.

Parameters

Return Value		Variant	Product of elements
Input	*x*	1D or Multidimensional array	Input array

ArrayAdd Method

Syntax

Function DQAnalysis.**ArrayAdd** (*x As Variant*, *y As Variant*) As Variant

Purpose

Adds arrays of any dimension. The ith element of the output array is obtained using the following formula.

$$zi = xi + yi$$

This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Result array
Input	*x*	1D or Multidimensional array	Input array
	y	1D or Multidimensional array	Input array

ArraySub Method

Syntax

Function DQAnalysis.**ArraySub** (*x As Variant*, *y As Variant*) As Variant

Purpose

Subtracts two arrays. The ith element of the output array z can be obtained using the following formula.

$z_i = x_i - y_i \qquad i = 0, 1, ..., n-1$

where n is the number of elements in the input arrays.

This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Result array.
Input	x	1D or Multidimensional array	Input array.
	y	1D or Multidimensional array	Input array.

ArrayMul Method

Syntax

Function DQAnalysis.**ArrayMul** (*x As Variant, y As Variant*) As Variant

Purpose

Multiplies two arrays. The ith element of the output array z is obtained using the following formula. This function works with multidimensional arrays.

$z_i = x_i * y_i \qquad i = 0, 1, ..., n-1$

where n is the number of elements in x or y.

Parameters

Return Value		1D or Multidimensional array	Result array.
Input	*x*	1D or Multidimensional array	Input array.
	y	1D or Multidimensional array	Input array.

Note

All input arrays should be the same size.

ArrayDiv Method

Syntax

Function DQAnalysis.**ArrayDiv** (*x As Variant, y As Variant*) As Variant

Purpose

Divides two arrays. The ith element of the output array is obtained using the following formula.

$z_i = x_i / y_i \qquad i = 0, 1, ..., n-1$

where n is the number of elements in each input array.

This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Result array.
Input	x	1D or Multidimensional array	Input array.
	y	1D or Multidimensional array	Input array.

Note

All input and output arrays should be the same size in each dimension, have the same number of elements, and the output arrays will be the size of the input arrays.

ArrayAbs Method

Syntax

Function DQAnalysis.**ArrayAbs** (*x As Variant*) As Variant

Purpose

Returns the absolute value of the x input array. This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Absolute value of input array.
Input	x	1D or Multidimensional array	Input array.

ArrayNeg Method

Syntax

Function DQAnalysis.**ArrayNeg** (*x As Variant*) As Variant

Purpose

Negates the elements of the input array. This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Negated values of Input array
Input	x	1D or Multidimensional array	Input array

ArrayLinearEval Method

Syntax

Function DQAnalysis.**ArrayLinearEval** (*x As Variant, a As Variant, b As Variant*) As Variant

Purpose

Performs a linear evaluation of a 1D or multidimensional array. The ith element of the output array y is obtained using the formula:

$$y_i = a*x_i + b \qquad i = 0, 1, ..., n-1$$

where n is the number of elements in array x.

Parameters

Return Value		1D or Multidimensional array	Linearly evaluated array.
Input	*x*	1D or Multidimensional array	Input array.
	a	Variant	Multiplicative constant.
	b	Variant	Additive constant.

ArrayPolyEval Method

Syntax

Function DQAnalysis.**ArrayPolyEval** (*src As Variant, coef As Variant*) As Variant

Purpose

Performs a polynomial evaluation on the input array. The ith element of the output array is obtained using the following formula.

$$dst_i \;?\; \overset{k?1}{\underset{j?0}{?}} coef_i * src_i^{\,j} \qquad \text{i=0,1,2.., n-1}$$

where n is the number of elements in input array src, and
j is the number of elements in the coefficient array.
This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Polynomially evaluated array.
Input	*src*	1D or Multidimensional array	Input array.
	coef	1D array	Coefficients array.

Parameter Discussion

The order of the polynomial is equal to the number of elements in the coefficients array minus one. If there are j elements in the coef array, order = j - 1.

ArrayScale Method

Syntax

Function DQAnalysis.**ArrayScale** (*x As Variant*) As Variant

Purpose

Scales the input array. The scaled output array is in the range (-1 : 1). The ith element of the scaled array can be obtained using the following formulas:

scaleconst = (max - min) / 2
offset = min + scaleconst
yi = (xi- offset) / scaleconst   i = 0, 1, ..., n-1

where   max is the maximum value in the input array,
min is the minimum value in the input array, and
n is the number of elements in array x.

The function determines the values of the constants scaleconst and offset. x and y can be the same array. This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Scaled array
Input	*x*	1D or Multidimensional array	Input array

ArrayQuickScale Method

Syntax

Function DQAnalysis.**ArrayQuickScale** (*x As Variant, factor As Variant*) As Variant

Purpose

Scales the input array by its maximum absolute value. The ith element of the scaled array y can be obtained using the following formula.

Yi = Xi / factor   for i = 0, 1, ..., n-1

where factor is the maximum absolute value in the input array, and
n is the number of elements in x.

The constant factor is determined by the function. This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Scaled array.
Input	*x*	1D or Multidimensional array	Input array.
	factor	Variant	factor constant.

ArrayNormalize Method

Syntax

Function DQAnalysis.**ArrayNormalize** (*x As Variant*) As Variant

Purpose

Normalizes an input array. The output array y is of the following form.

yi = (xi- ave) / sDev   i=0, 1, ..., n-1

where n is the number of elements in input array x,
ave is the mean of the input array, and
sDev is the standard deviation of the input vector.

Refer to the StdDev function for the formulas used to find the mean and the standard deviation. This function works with multidimensional arrays.

Parameters

Return Value		1D or Multidimensional array	Normalized vector
Input	*x*	1D or Multidimensional array	Input vector.

ArrayMaxMin1D Method

Syntax

Sub DQAnalysis.**ArrayMaxMin1D** (*x As Variant, max As Variant, imax As Long, min As Variant, imin As Long*)

Purpose

Finds the maximum and minimum values in the input array, as well as the respective indices of the first occurrence of the maximum and minimum values.

Parameters

Input	*x*	1D array	Input array.
Output	*max*	Variant	Maximum value.
	imax	Long	Index of max in x array.
	min	Variant	Minimum value.
	imin	Long	Index of min in x array.

ArrayMaxMin2D Method

Syntax

Sub DQAnalysis.**ArrayMaxMin2D** (*x As Variant, max As Variant, imax As Long, jmax As Long, min As Variant, imin As Long, jmin As Long*)

Purpose

Finds the maximum and minimum values in the 2D input array, as well as the respective indices of the first occurrence of the maximum and minimum values. The x array is scanned by rows.

Parameters

Input	*x*	1D array	Input array.
Output	*max*	Variant	Maximum value.

imax	Long	Index of max in x array (first dimension).
imax	Long	Index of max in x array (second dimension).
min	Variant	Minimum value.
imin	Long	Index of min in x array (first dimension).
jmin	Long	Index of min in x array (second dimension).

ArraySubset1D Method

Syntax

Function DQAnalysis.**ArraySubset1D** (*x As Variant, start As Long, length As Long*) As Variant

Purpose

Extracts a subset of the input array containing the number of elements specified by the length and starting at the index element.

Parameters

Return Value		1D array	Subset array
Input	*x*	1D array	Input array.
	start	Long	Initial index for the subset.
	length	Long	Number of elements copied to the subset.

ArrayReverse1D Method

Syntax

Function DQAnalysis.**ArrayReverse1D** (*x As Variant*) As Variant

Purpose

Reverses the order of the elements of the input array using the following formula:

xi = xn-1-I for i=0,1,…,n-1

where n is the number of elements in array x.

Parameters

Return Value		1D array	Reversed array.
Input	*x*	1D array	Input array.

ArrayShift1D Method

Syntax

Function DQAnalysis.**ArrayShift1D** (*x As Variant, count As Long*) As Variant

Purpose

Shifts the elements of the input array using the following formula.

xi=xi-count i = 0, 1, ..., n-1

where n is the number of elements in input array x.

The number of count specified can be in the positive (right) or negative (left) direction.

Parameters

Return Value		1D array	Shifted array.
Input	*x*	1D array	Input array.
	count	Long	Number of shifts.

Parameter Discussion

This is not a circular shift. Shifted values are not retained, and the trailing portion of the shift is replaced with zero. The operation cannot be done in place, thus the input and output arrays cannot be the same. The input and output arrays are the same size.

ArraySort1D Method

Syntax

Function DQAnalysis.**ArraySort1D** (*x As Variant, direction As Integer*) As Variant

Purpose

Sorts the input array in ascending or descending order.

Parameters

Return Value		1D array	Sorted array.
Input	*x*	1D array	Input array.
	direction	Integer	Zero: ascending; Non-zero: descending.

CxAdd Method

Syntax

Sub DQAnalysis.**CxAdd** (*xr As Variant, xi As Variant, yr As Variant, yi As Variant, zr As Variant, zi As Variant*)

Purpose

Adds two complex numbers. The resulting complex number is obtained using the following formulas.

zr = xr + yr
zi = xi + yi

Parameters

Input	*xr*	Variant	Real part of x
	xi	Variant	Imaginary part of x
	yr	Variant	Real part of y
	yi	Variant	Imaginary part of y
Output	*zr*	Variant	Real part of z
	zi	Variant	Imaginary part of z

CxSub Method

Syntax

Sub DQAnalysis.**CxSub** (*xr As Variant, xi As Variant, yr As Variant, yi As Variant, zr As Variant, zi As Variant*)

Purpose

Subtracts two complex numbers. The resulting complex number is obtained using the following formulas.

zr = xr - yr
zi = xi - yi

Parameters

Input	*xr*	Variant	Real part of x.
	xi	Variant	Imaginary part of x.
	yr	Variant	Real part of y.
	yi	Variant	Imaginary part of y.
Output	*zr*	Variant	Real part of z.
	zi	Variant	Imaginary part of z.

CxMul Method

Syntax

Sub DQAnalysis.**CxMul** (*xr As Variant*, *xi As Variant*, *yr As Variant*, *yi As Variant*, *zr As Variant*, *zi As Variant*)

Purpose

Multiplies two complex numbers. The resulting complex number is obtained using the following formulas.

zr = xr*yr - xi*yi
zi = xr*yi + xi*yr

Parameters

Input	*xr*	Variant	Real part of x.
	xi	Variant	Imaginary part of x.
	yr	Variant	Real part of y.
	yi	Variant	Imaginary part of y.
Output	*zr*	Variant	Real part of z.
	zi	Variant	Imaginary part of z.

CxDiv Method

Syntax

Sub DQAnalysis.**CxDiv** (*xr As Variant*, *xi As Variant*, *yr As Variant*, *yi As Variant*, *zr As Variant*, *zi As Variant*)

Purpose

Divides two complex numbers. The resulting number is obtained using the following formulas.

zr = (xr*yr + xi*yi) / (yr2 + yi2)
zi = (xi*yr - xr*yi) / (yr2 + yi2)

Parameters

Input	*xr*	Variant	Real part of x.
	xi	Variant	Imaginary part of x.
	yr	Variant	Real part of y.
	yi	Variant	Imaginary part of y.
Output	*zr*	Variant	Real part of z.
	zi	Variant	Imaginary part of z.

CxRecip Method

Syntax

Sub DQAnalysis.**CxRecip** (*xr As Variant*, *xi As Variant*, *yr As Variant*, *yi As Variant*)

Purpose

Computes the reciprocal of a complex number. The resulting complex number is obtained using the following formulas.

yr = xr / (xr2 + xi2)
yi = -xi / (xr2 + xi2)

Parameters

Input	*xr*	Variant	Real part of x
	xi	Variant	Imaginary part of x

Output	*yr*	Variant	Real part of y
	yi	Variant	Imaginary part of y

CxToPolar Method

Syntax

Sub DQAnalysis.**CxToPolar** (*x As Variant, y As Variant, mag As Variant, phase As Variant*)

Purpose

Converts the rectangular coordinates (x, y) to polar coordinates (mag, phase). The formulas used to obtain the polar coordinates are as follows.

mag = | x |
phase = arctan (y / x)

The phase value is in the range of (-p to p).

Parameters

Input	x	Variant	Coordinate.
	y	Variant	Coordinate.
Output	mag	Variant	Magnitude.
	phase	Variant	Phase (in radians).

CxToRect Method

Syntax

Sub DQAnalysis.**CxToRect** (*mag As Variant, phase As Variant, x As Variant, y As Variant*)

Purpose

Converts the polar coordinates (mag, phase) to rectangular coordinates (x, y). The formulas used to obtain the rectangular coordinates are as follows.

x = mag * cos(phase)
y = mag * sin(phase)

Parameters

Input	*mag*	Variant	Magnitude.
	phase	Variant	Phase (in radians).
Output	*x*	Variant	x coordinate.
	y	Variant	y coordinate.

CxPow Method

Syntax

Sub DQAnalysis.**CxPow** (*xr As Variant, xi As Variant, exp As Variant, yr As Variant, yi As Variant*)

Purpose

Computes the power of a complex number. The resulting complex number is obtained using the following formula.

(yr, yi) = (xr, xi)exp

Parameters

Input	*xr*	Variant	Real part of x.
	xi	Variant	Imaginary part of x.
	exp	Variant	Exponent.

Output	*yr*	Variant	Real part of y.
	yi	Variant	Imaginary part of y.

CxSqrt Method

Syntax

Sub DQAnalysis.**CxSqrt** (*xr As Variant, xi As Variant, yr As Variant, yi As Variant*)

Purpose

Computes the square root of a complex number. The resulting complex number is obtained using the following formula.

(yr, yi) = (xr, xi)^1/2

Parameters

Input	*xr*	Variant	Real part of x.
	xi	Variant	Imaginary part of x.
Output	*yr*	Variant	Real part of y.
	yi	Variant	Imaginary part of y.

CxLn Method

Syntax

Sub DQAnalysis.**CxLn** (*xr As Variant, xi As Variant, yr As Variant, yi As Variant*)

Purpose

Computes the natural logarithm of a complex number. The resulting complex number is obtained using the following formula.

(yr, yi) = Loge(xr, xi)

where e = 2.718---.

Parameters

Input	*xr*	Variant	Real part of x.
	xi	Variant	Imaginary part of x.
Output	*yr*	Variant	Real part of y.
	yi	Variant	Imaginary part of y.

CxLog Method

Syntax

Sub DQAnalysis.**CxLog** (*xr As Variant, xi As Variant, yr As Variant, yi As Variant*)

Purpose

Computes the logarithm (base 10) of a complex number. The resulting complex number is obtained using the following formula.

(yr, yi) = Log10(xr, xi)

Parameters

Input	*xr*	Variant	Real part of x.
	xi	Variant	Imaginary part of x.
Output	*yr*	Variant	Real part of y
	yi	Variant	Imaginary part of y

CxArrayAdd Method

Syntax

Sub DQAnalysis.**CxArrayAdd** (*xr As Variant*, *xi As Variant*, *yr As Variant*, *yi As Variant*, *zr As Variant*, *zi As Variant*)

Purpose

Adds two complex arrays. The ith element of the resulting complex array is obtained using the following formulas.

zri = xri + yri
zii = xii + yii   i = 0, 1, ..., n-1

where n is the number of elements in the input array.
The function works with multidimensional arrays.

Parameters

Input	*xr*	1D or Multidimensional array	Real part of x.
	xi	1D or Multidimensional array	Imaginary part of x.
	yr	1D or Multidimensional array	Real part of y.
	yi	1D or Multidimensional array	Imaginary part of y.
Output	*zr*	1D or Multidimensional array	Real part of z.
	zi	1D or Multi dimensional array	Imaginary part of z.

Note

All input arrays should be the same size in each dimension, and the output arrays are the size of the input arrays.

CxArraySub Method

Syntax

Sub DQAnalysis.**CxArraySub** (*xr As Variant*, *xi As Variant*, *yr As Variant*, *yi As Variant*, *zr As Variant*, *zi As Variant*)

Purpose

Subtracts two complex arrays. The ith element of the resulting complex array is obtained using the following formulas.

zri = xri- yri
zii= xii- yii   i = 0, 1, .., n-1

where n is the number of elements in the input arrays.
The function works with multidimensional arrays.

Parameters

Input	*xr*	1D or Multidimensional array	Real part of x.
	xi	1D or Multidimensional array	Imaginary part of x.
	yr	1D or Multidimensional array	Real part of y.
	yi	1D or Multidimensional array	Imaginary part of y.
Output	*zr*	1D or Multidimensional array	Real part of z.
	zi	1D or Multidimensional array	Imaginary part of z.

Note

All input arrays should be the same size in each dimension, and the output arrays are the size of the input arrays.

CxArrayMul Method

Syntax

Sub DQAnalysis.**CxArrayMul** (*xr As Variant, xi As Variant, yr As Variant, yi As Variant, zr As Variant, zi As Variant*)

Purpose

Multiplies two complex arrays. The ith element of the resulting complex array is obtained using the following formulas.

zri = xri*yri - xii*yii
zii = xri*yii + xii*yri   i = 0, 1, .., n-1

where n is the number of elements in the input arrays.

The function works with multidimensional arrays.

Parameters

Input	xr	1D or Multidimensional array	Real part of x.
	xi	1D or Multidimensional array	Imaginary part of x.
	yr	1D or Multidimensional array	Real part of y.
	yi	1D or Multidimensional array	Imaginary part of y.
Output	zr	1D or Multidimensional array	Real part of z.
	zi	1D or Multidimensional array	Imaginary part of z.

Note

All input arrays should be the same size in each dimension, and the output arrays are the size of the input arrays.

CxArrayDiv Method

Syntax

Sub DQAnalysis.**CxArrayDiv** (*xr As Variant, xi As Variant, yr As Variant, yi As Variant, zr As Variant, zi As Variant*)

Purpose

Divides two complex arrays. The ith element of the resulting complex array is obtained using the following formula.

zri = (xri*yri+ xii*yii) / (yri2 + yii2)
zii = (xii*yri- xri*yii) / (yri2 + yii2)   i = 0, 1, ..., n-1

where n is the size of the input arrays.

The function works with multidimensional arrays.

Parameters

Input	*xr*	1D or Multidimensional array	Real part of x.
	xi	1D or Multidimensional array	Imaginary part of x.
	yr	1D or Multidimensional array	Real part of y.
	yi	1D or Multidimensional array	Imaginary part of y.
Output	*zr*	1D or Multidimensional array	Real part of z.
	zi	1D or Multidimensional array	Imaginary part of z.

Note

All input arrays should be the same size in each dimension, and the output arrays are the size of the input arrays.

CxArrayToPolar Method

Syntax

Sub DQAnalysis.**CxArrayToPolar** (*x As Variant, y As Variant, mag As Variant, phase As Variant*)

Purpose

Converts the set of rectangular coordinate points (x, y) to a set of polar coordinate points (mag, phase). The ith element of the polar coordinate set is obtained using the following formulas.

magi = | xi + yi |
phasei = arctan(yi / xi) i = 0, 1, ..., n-1

where n is the number of elements in input array x.
The phase value is in the range of (-p to p).

Parameters

Input	x	1D or Multidimensional array	x coordinate.
	y	1D or Multidimensional array	y coordinate.
Output	mag	1D or Multidimensional array	Magnitude.
	phase	1D or Multidimensional array	Phase (in radians).

Note

All input arrays and explicitly bounded output arrays should be the same size.

CxArrayToRect Method

Syntax

Sub DQAnalysis.**CxArrayToRect** (*mag As Variant, phase As Variant, x As Variant, y As Variant*)

Purpose

Converts the set of polar coordinate points (mag, phase) to a set of rectangular coordinate points (x, y). The ith element of the rectangular set is obtained using the following formulas.

xi = magi* cos(phasei)
yi= magi* sin(phasei)   i = 0, 1, ..., n-1

where n is the number of elements in the input arrays.

Parameters

Input	*mag*	1D or multidimensional array.	Magnitude.
	phase	1D or multidimensional array.	Phase (in radians).
Output	*x*	1D or multidimensional array.	x coordinate
	y	1D or multidimensional array.	y coordinate

Note

All input and output arrays should be the same size.

CxArrayLinearEval Method

Syntax

Sub DQAnalysis.**CxArrayLinearEval** (*xr As Variant, xi As Variant, ar As Variant, ai As Variant, br As Variant, bi As Variant, yr As Variant, yi As Variant*)

Purpose

Performs a complex linear evaluation of a 1D complex array. The ith element of the resulting complex array is obtained using the following formulas.

yri= ar*xri- ai*xii+ br
yii= ar*xii+ ai*xri+ bi   i = 0, 1, .., n-1

where n is the number of elements in the input arrays.

Parameters

Input	*xr*	1D or Multidimensional array	Real part of x.
	xi	1D or Multidimensional array	Imaginary part of x.
	ar	Variant	Real part of a.
	ai	Variant	Imaginary part of a.
	br	Variant	Real part of b.
	bi	Variant	Imaginary part of b.
Output	*yr*	1D or Multidimensional array	Real part of y.
	yi	1D or Multidimensional array	Imaginary part of y.

Mean Method

Syntax

Function DQAnalysis.**Mean** (*x As Variant*) As Variant

Purpose

Computes the mean (average) value of the input array

Parameters

Return Value		Variant	Mean value.
Input	*x*	1D array	Input array.

Variance Method

Syntax

Function DQAnalysis.**Variance** (*x As Variant*) As Variant

Purpose

Computes the variance values of the input array.

Parameters

Return Value		Variant	Variance.
Input	*x*	1D array	Input array.

StdDev Method

Syntax

Function DQAnalysis.**StdDev** (*x As Variant*) As Variant

Purpose

Computes the standard deviation values of the input array.

Parameters

Return Value		Variant	Standard deviation.
Input	*x*	1D array	Input array.

RMS Method

Syntax

Function DQAnalysis.**RMS** (*x As Variant*) As Variant

Purpose

Computes the root mean squared (rms) value of the input array.

Parameters

Return Value		Variant	Root mean squared value.
Input	*x*	1D array	Input array.

Moment Method

Syntax

Function DQAnalysis.**Moment** (*x As Variant, order As Variant*) As Variant

Purpose

Computes the moment about the mean of the input array with the specified order.

Parameters

Return Value		Variant	Moment about the mean.
Input	*x*	1D array	Input array.
	order	Variant	Moment order.

Note

order must be greater than zero.

Median Method

Syntax

Function DQAnalysis.**Median** (*x As Variant*) As Variant

Purpose

Finds the median value of the input array. To find the median value, the input array first is sorted in ascending order.

Parameters

Return Value		Variant	Median value.
Input	*x*	1D array	Input array.

Note

The x input array is not changed.

Histogram Method

Syntax

Sub DQAnalysis.**Histogram** (*x As Variant, xBase As Variant, xTop As Variant, intervals As Long, hist As Variant As, axis As Variant*)

Purpose

Computes the histogram of the x input array. The histogram is obtained by counting the number of times that the elements in the input array fall in the ith interval.

Parameters

Input	*x*	1D array	Input data.
	xBase	Variant	Lower range.
	xTop	Variant	Upper range.
	intervals	Long	Number of intervals.
Output	*hist*	1D array	Histogram of x. The size of this array is intervals elements.
	axis	1D array	Histogram axis array. The size of this array is intervals elements.

Mode Method

Syntax

Function DQAnalysis.**Mode** (x As Variant, xBase As Variant, xTop As Variant, intervals As Long) As Variant

Purpose

Finds the mode of the input array. The mode is defined as the value that most often occurs in a given set of samples. This function determines the mode in terms of the histogram of the input array.

Parameters

Return Value		Variant	Mode value.

Input	*x*	1D array	Input array.
	xBase	Variant	Lower range.
	xTop	Variant	Upper range.
	intervals	Long	Number of intervals.

DotProduct Method

Syntax

Function DQAnalysis.**DotProduct** (*x As Variant, y As Variant*) As Variant

Purpose

Computes the dot product of the x and y input arrays.

Parameters

Return Value		Variant	Dot product.
Input	*x*	1D array	Input vector.
	y	1D array	Input vector.

MatrixMul Method

Syntax

Function DQAnalysis.**MatrixMul** (*x As Variant, y As Variant*) As Variant

Purpose

Multiplies two 2D input matrices or a 2D matrix with a vector.

Parameters

Return Value		2D or 1D array	Resultant Matrix.
Input	*x*	2D or 1D array	Input matrix.
	y	2D or 1D array	Input matrix.

Parameter Discussion

The following array sizes must be met:

1. x must be (m by k).
2. y must be (k by n).
3. z will be (m by n).
4. If y is a vector, z is also a vector.

MatrixTranspose Method

Syntax

Function DQAnalysis.**MatrixTranspose** (x As Variant) As Variant

Purpose

Finds the transpose of a 2D input matrix. The (ith, jth) element of the resulting matrix is given by the following formula.

$$y_{ij}=x_{ji}$$

Parameters

Input	x	2D array	Input matrix.
Return Value		2D array	Transpose matrix.

Note:

If the input matrix is dimensioned (n by m), then the output matrix will be dimensioned (m by n).

MatrixDet Method

Syntax

Function DQAnalysis.**MatrixDet** (x As Variant) As Variant

Purpose

Finds the determinant of an n-by-n 2D input matrix, where n is the number of rows and columns of x.

Parameters

Return Value		Variant	Determinant.
Input	*x*	2D array	Input matrix.

Note:

The input matrix must be an n-by-n square matrix.

MatrixTrace Method

Syntax

Function DQAnalysis.**MatrixTrace** (x As Variant) As Variant

Purpose

Finds the trace of the 2D input matrix. The trace is the sum the main diagonal elements of the matrix. The trace is obtained using the following formula.

Trace = x[0][0] + x[1][1] + ... + x[n-1][n-1]

The input matrix must be an n-by-n square matrix.

Parameters

Input	x	2D array	Input matrix.
Return Value		Variant	Trace.

MatrixInverse Method

Syntax

Function DQAnalysis. **MatrixInverse** (*x As Variant*) As Variant

Purpose

Finds the inverse matrix of an input matrix.

Parameters

Return Value		2D array	Inverse matrix.
Input	x	2D array	Input matrix.

Note:

The input matrix must be an n by n square matrix, and input and output matrices are the same size.

MatrixLU Method

Syntax

Sub DQAnalysis.**MatrixLU** (*a As Variant*, *b As Variant*, *p As Variant*)

Purpose

Performs an LU matrix decomposition,

Pa = L * U

where L is an n-by-n lower triangular matrix whose diagonal elements are ones,

U is an upper triangular matrix, and

P is an identity matrix with some rows exchanged based on the information in array p.

Parameters

Input	*a*	2D array	Input matrix.
Output	*b*	2D array	LU decomposition.

	p	2D array	Permutation matrix..

Parameter Discussion

After the function executes, the output matrix b consists of two triangular matrices. L occupies the lower triangular part and U occupies the upper triangular part of b. The permutation vector p records possible row exchange information in the LU decomposition.

LU is most useful when used in conjunction with BackSub and ForwSub to solve a set of linear equations with the same matrix a. For more information, refer to Numerical Recipes by Press, et al., Cambridge University Press.

MatrixForwSub Method

Syntax

Sub DQAnalysis.**MatrixForwSub** (*a As Variant, y As Variant, p As Variant, x As Variant*)

Purpose

Solves the linear equations a*x = y by forward substitution. a is assumed to be an n-by-n lower triangular matrix whose diagonal elements are all ones.

x and y can be the same array.

Parameters

Input	*a*	2D array	Input matrix.
	y	1D array	Input vector.
	p	2D array	Permutation matrix.
Output	*x*	1D array	Solution vector.

Note

ForwSub is used in conjunction with LU and BackSub to solve linear equations. The parameter p is obtained from LU. If you are not using the LU function, set p(i) = i.

MatrixBackSub Method

Syntax

Function DQAnalysis.**MatrixBackSub** (*a As Variant, y As Variant*) As Variant

Purpose

Solves the linear equations a*x = y by backward substitution. a is assumed to be an n-by-n lower triangular matrix whose diagonal elements are allones.This function is used in conjunction with LU and ForwSub to solvelinear equations. Refer to the LU function description for more information.

Parameters

Return value		1D array	Solution Vector
Input	a	2D array	Input matrix.
	y	1D array	Input vector.

Note

The size of y and the return value variable must be the same as n, and a must be an n-by-n square matrix.

MatrixSolveLinearEqs Method

Syntax

Function DQAnalysis.**MatrixSolveLinearEqs** (*A As Variant, y As Variant*) As Variant

Purpose

Solves the linear system of equations with the following formula.

Ax = y

Parameters

Return Value		1D array	Solution vector
Input	A	2D array	Input matrix
	y	1D array	Known vector

Note

The A input matrix must be an n by n square matrix, and x and y must have n.

FFT Method

Syntax

Sub DQAnalysis.**FFT** (*xr As Variant*, *xi As Variant*, *yr as Variant*, *yi as Variant*)

Purpose

Computes the Fast Fourier Transform of the complex data. Let x=xr+jxi be the complex array, then:

cy = FFT {cx}

About the Fast Fourier Transform (FFT)

Parameters

Input	xr	1D array	Real part of complex array.
	xi	1D array	Imaginary part of complex array.
Output	yr	1D array	Real part of FFT.
	yi	1D array	Imaginary part of FFT.

Remark

Using this function all input arrays should be the same size. The size of the arrays should be a power of two.