rank

Vector and Matrix

Syntax: rank(**A**)

Description: Returns the rank of a matrix **A**, i.e., the maximum number of linearly independent columns in **A**.

Arguments

A: real $m \times n$ matrix

Algorithm: Singular value computation (Wilkinson and Reinsch, 1971)

rbeta

Random Numbers

Syntax: rbeta($m, s1, s2$)

Description: Returns a vector of m random numbers having the beta distribution.

Arguments

m: integer, $m > 0$

$s1, s2$: real shape parameters, $s1 > 0$, $s2 > 0$

Algorithm: Best's XG algorithm, Johnk's generator (Devroye, 1986)

See also: rnd

rbinom

Random Numbers

Syntax: rbinom(m, n, p)

Description: Returns a vector of m random numbers having the binomial distribution.

Arguments

m, n: integers, $m > 0$, $n > 0$

p: real number, $0 \le p \le 1$

Algorithm: Waiting time and rejection algorithms (Devroye, 1986)

See also: rnd

rcauchy

Random Numbers

Syntax rcauchy(*m, l, s*)

Description Returns a vector of *m* random numbers having the Cauchy distribution.

Arguments

m integer, $m > 0$

l real location parameter

s real scale parameter, $s > 0$

Algorithm Inverse cumulative density method (Press *et al.*, 1992)

See also rnd

rchisq

Random Numbers

Syntax rchisq(*m, d*)

Description Returns the vector of *m* random numbers having the chi-squared distribution.

Arguments

m integer, $m > 0$

d integer degrees of freedom, $d > 0$

Algorithm Best's XG algorithm, Johnk's generator (Devroye, 1986)

See also rnd

Re

Complex Numbers

Syntax Re(*z*)

Description Returns the real part of *z*.

Arguments

z real or complex number

See also Im

READ_BLUE *(Professional)*

File Access

Syntax READ_BLUE(*file*)

Description Extracts only the blue component from *file* of a color image in BMP, GIF, JPG or TGA format. The result is a matrix with one-third as many columns as the matrix returned by READ_RGB.

Arguments

file string variable corresponding to color image filename or path

READBMP

File Access

Syntax READBMP(*file*)

Description Creates a matrix containing a grayscale representation of the bitmap image in *file*. Each element in the matrix corresponds to a pixel. The value of a matrix element determines the shade of gray associated with the corresponding pixel. Each element is an integer between 0 (black) and 255 (white).

Arguments

file string variable corresponding to grayscale image BMP filename or path

Comments After you have read an image file into Mathcad, you can use the *picture operator* to view it.

Mathcad Professional includes a function READ_IMAGE which reads not only BMP files but also GIF, JPG, and TGA files.

See also For color images, see READRGB.

READ_GREEN *(Professional)*

File Access

Syntax READ_GREEN(*file*)

Description Extracts only the green component from *file* of a color image in BMP, GIF, JPG or TGA format. The result is a matrix with one-third as many columns as the matrix returned by READ_RGB.

Arguments

file string variable corresponding to color image filename or path

READ_HLS *(Professional)*

File Access

Syntax READ_HLS(*file*)

Description Creates a matrix in which the color information in *file* is represented by the appropriate values of hue, lightness, and saturation. *file* is in BMP, GIF, JPG or TGA format.

Arguments

file string variable corresponding to color image filename or path

See also See READRGB for an overview.

READ_HLS_HUE *(Professional)*

File Access

Syntax READ_HLS_HUE(*file*)

Description Extracts only the hue component from *file* of a color image in BMP, GIF, JPG or TGA format. The result is a matrix with one-third as many columns as the matrix returned by READ_HLS.

Arguments

file string variable corresponding to color image filename or path

READ_HLS_LIGHT *(Professional)* File Access

Syntax READ_HLS_LIGHT(*file*)

Description Extracts only the lightness component from *file* of a color image in BMP, GIF, JPG or TGA format. The result is a matrix with one-third as many columns as the matrix returned by READ_HLS.

Arguments

file string variable corresponding to color image filename or path

READ_HLS_SAT *(Professional)* File Access

Syntax READ_HLS_SAT(*file*)

Description Extracts only the saturation component from *file* of a color image in BMP, GIF, JPG or TGA format. The result is a matrix with one-third as many columns as the matrix returned by READ_HLS.

Arguments

file string variable corresponding to color image filename or path

READ_HSV *(Professional)* File Access

Syntax READ_HSV(*file*)

Description Creates a matrix in which the color information in *file* is represented by the appropriate values of hue, saturation and value. *file* is in BMP, GIF, JPG or TGA format.

Arguments

file string variable corresponding to color image filename or path

See also See READRGB for an overview of reading color data files.

READ_HSV_HUE *(Professional)* File Access

Syntax READ_HSV_HUE(*file*)

Description Extracts only the hue component from *file* of a color image in BMP, GIF, JPG or TGA format. The result is a matrix with one-third as many columns as the matrix returned by READ_HSV.

Arguments

file string variable corresponding to color image filename or path

READ_HSV_SAT *(Professional)* File Access

Syntax READ_HSV_SAT(*file*)

Description Extracts only the saturation component from *file* of a color image in BMP, GIF, JPG or TGA format. The result is a matrix with one-third as many columns as the matrix returned by READ_HSV.

Arguments

file string variable corresponding to color image filename or path

READ_HSV_VALUE *(Professional)* File Access

Syntax READ_HSV_VALUE(*file*)

Description Extracts only the value component from *file* of a color image in BMP, GIF, JPG or TGA format. The result is a matrix with one-third as many columns as the matrix returned by READ_HSV.

Arguments

file string variable corresponding to color image filename or path

READ_IMAGE *(Professional)* File Access

Syntax READ_IMAGE(*file*)

Description Creates a matrix containing a grayscale representation of the image in *file*. Each element in the matrix corresponds to a pixel. The value of a matrix element determines the shade of gray associated with the corresponding pixel. Each element is an integer between 0 (black) and 255 (white). *file* is in BMP, GIF, JPG or TGA format.

Arguments

file string variable corresponding to grayscale image filename or path

See also For color images, see READRGB.

READPRN File Access

Syntax READPRN(*file*)

Description Reads a structured ASCII data file and returns a matrix. Each line in the data file becomes a row in the matrix. The number of elements in each row must be the same. Used as follows: **A** := READPRN(*file*).

Arguments

file string variable corresponding to structured ASCII data filename or path

Comments The READPRN function reads an entire data file, determines the number of rows and columns, and creates a matrix out of the data.

When Mathcad reads data with the READPRN function:

- Each instance of the READPRN function reads an entire data file.

- All lines in the data file must have the same number of values. (Mathcad ignores lines with no values.) If the lines in the file have differing numbers of values, Mathcad marks the READPRN equation with an error message. Use a text editor to replace the missing values with zeros before you use READPRN.
- The READPRN function ignores text in the data file.
- The result of reading the data file is an m-by-n matrix **A**, where m is the number of lines containing data in the file and n is the number of values per line.

WRITEPRN and READPRN allow you to write out and read in *nested arrays* created in Mathcad Professional.

READ_RED *(Professional)* — File Access

Syntax READ_RED(*file*)

Description Extracts only the red component from *file* of a color image in BMP, GIF, JPG or TGA format.The result is a matrix with one-third as many columns as the matrix returned by READ_RGB.

Arguments

file string variable corresponding to color image filename or path

READRGB — File Access

Syntax READRGB(*file*)

Description Creates a matrix in which the color information in the BMP file *file* is represented by the appropriate values of red, green, and blue. This matrix consists of three submatrices, each with the same number of columns and rows. Three matrix elements, rather than one, correspond to each pixel. Each element is an integer between 0 and 255. The three corresponding elements, when taken together, establish the color of the pixel.

Arguments

file string variable corresponding to color image filename or path

Example

```
color := "c:\images\monalisa.bmp"

gray := READBMP(color)
packed := READRGB(color)

r := rows(packed) - 1        c := cols(packed) / 3

red := submatrix(packed, 0, r, 0, c - 1)
green := submatrix(packed, 0, r, c, 2·c - 1)
blue := submatrix(packed, 0, r, 2·c, 3·c - 1)
```

Comments To partition the matrix for a color image into its red, green, and blue components, use the submatrix function formulas shown in the example above. In this example, the color bitmap file `monalisa.bmp` is read into a grayscale matrix **gray**, as well as the packed RGB matrix **packed**, and then converted into three submatrices called **red**, **green**, and **blue**.

After you have read an image file into Mathcad, you can use the *picture operator* to view it.

Mathcad Professional includes several specialized functions for reading color images or image components, including functions for reading images in GIF, JPG, and TGA formats.

Consult the following table to decide which function to use:

To separate a file into these components:	**Use these functions:**
red, green, and blue (RGB)	READ_RED, READ_GREEN, READ_BLUE
hue, lightness, and saturation (HLS)	READ_HLS, READ_HLS_HUE, READ_HLS_LIGHT, READ_HLS_SAT,
hue, saturation, and value (HSV)	READ_HSV, READ_HSV_HUE, READ_HSV_SAT, READ_HSV_VAL

Note READ_HLS and READ_HSV work in exactly the same way as READRGB. All the others work in exactly the same way as READBMP.

See also For grayscale images, see READBMP.

regress

Regression and Smoothing

One-dimensional Case

Syntax regress(**vx**, **vy**, *n*)

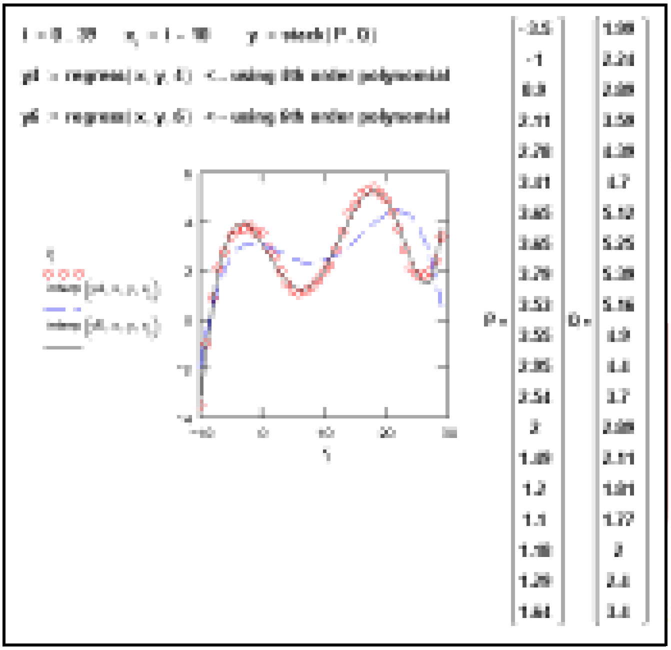
Description Returns the vector required by the interp function to find the *n*th order polynomial that best fits data arrays **vx** and **vy**.

Arguments

vx, vy real vectors of the same size

n integer, $n > 0$

Example



Comments

The regression functions regress and loess are useful when you have a set of measured y values corresponding to x values and you want to fit a polynomial of degree n through those y values. (For a simple linear fit, that is, n=1, you may as well use the slope and intercept functions.)

Use regress when you want to use a single polynomial to fit all your data values. The regress function lets you fit a polynomial of any order. However as a practical matter, you would rarely need to go beyond $n = 6$.

Since regress tries to accommodate all your data points using a single polynomial, it will not work well when your data does not behave like a single polynomial. For example, suppose you expect your y_i to be linear from x_1 to x_{10} and to behave like a cubic equation from x_{11} to x_{20}. If you use regress with $n = 3$ (a cubic), you may get a good fit for the second half but a poor fit for the first half.

The loess function, available in Mathcad Professional only, alleviates these kinds of problems by performing a more localized regression.

For regress, the first three components of the output vector $\mathbf{vr} := \mathrm{regress}(\mathbf{vx}, \mathbf{vy}, n)$ are $\mathbf{vr}_0$=3 (a code telling interp that **vr** is the output of regress as opposed to a spline function or loess), $\mathbf{vr}_1$=3 (the index within **vr** where the polynomial coefficients begin), and $\mathbf{vr}_2$=n (the order of the fit). The remaining $n + 1$ components are the coefficients of the fitting polynomial from the lowest degree term to the highest degree term.

Two-dimensional Case

Syntax regress(**Mxy**, **vz**, n)

Description Returns the vector required by the interp function to find the nth order polynomial that best fits data arrays **Mxy** and **vz**. **Mxy** is an $m \times 2$ matrix containing x-y coordinates. **vz** is an m-element vector containing the z coordinates corresponding to the m points specified in **Mxy**.

Arguments

Mxy real $m \times 2$ matrix containing x-y coordinates of the m data points

vz real m-element vector containing the z coordinates corresponding to the points specified in **Mxy**.

n integer, $n > 0$

Comments Assume, for example, that you have a set of measured z values corresponding to x and y values and you want to fit a polynomial surface through those z values. The meanings of the input arguments are more general than in the one-dimensional case:

- The argument **vx**, which was an m-element vector of x values, becomes an $m \times 2$ matrix, **Mxy**. Each row of **Mxy** contains an x in the first column and a corresponding y value in the second column.
- The argument x for the interp function becomes a 2-element vector **v** whose elements are the x and y values at which you want to evaluate the polynomial surface representing the best fit to the data points in **Mxy** and **vz**.

This discussion can be extended naturally to higher dimensional cases. You can add independent variables by simply adding columns to the **Mxy** array. You would then add a corresponding number of rows to the vector **v** that you pass to the interp function. The regress function can have as many independent variables as you want. However, regress will calculate more slowly and require more memory when the number of independent variables and the degree are greater than four. The loess function is restricted to at most four independent variables.

Keep in mind that for regress, the number of data values, m must satisfy $m > \binom{n+k-1}{n} \cdot \frac{n+k}{k}$, where k is the number of independent variables (hence the number of columns in **Mxy**), n is the degree of the desired polynomial, and m is the number of data values (hence the number of rows in **vz**). For example, if you have five explanatory variables and a fourth degree polynomial, you will need more than 126 observations.

The loess function, available in Mathcad Professional, works better than regress when your data does not behave like a single polynomial.

Algorithm Normal equation solution through Gauss-Jordan elimination (Press *et al.*, 1992)

relax

(Professional) Differential Equation Solving

Syntax relax(**A, B, C, D, E, F, U**, *rjac*)

Description Returns a matrix of solution values for a Poisson partial differential equation over a planar square region. More general than multigrid, which is faster.

Arguments

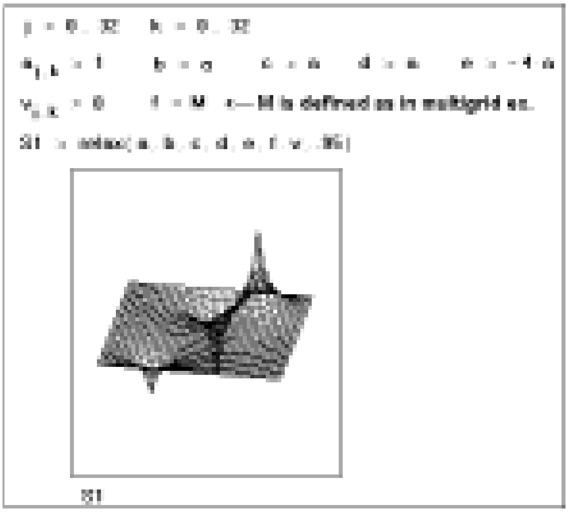
A, B, C, D, E real square matrices all of the same size containing coefficients of the discretized Laplacian (for example, the left-hand side of equations below).

F real square matrix containing the source term at each point in the region in which the solution is sought (for example, the right-hand side of equations below).

U real square matrix containing boundary values along the edges of the region and initial guesses for the solution inside the region.

rjac spectral radius of the Jacobi iteration, $0 < rjac < 1$, which controls the convergence of the relaxation algorithm. Its optimal value depends on the details of your problem.

Example



Comments Two partial differential equations that arise often in the analysis of physical systems are Poisson's equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \rho(x, y)$$ and its homogeneous form, Laplace's equation.

Mathcad has two functions for solving these equations over a square region, assuming the values taken by the unknown function $u(x, y)$ on all four sides of the boundary are known. The most general solver is the relax function. In the special case when $u(x, y)$ is known to be zero on all four sides of the boundary, you can use the multigrid function instead. This function will often solve the problem faster than relax. If the boundary condition is the same on all four sides, you can simply transform the equation to an equivalent one in which the value is zero on all four sides.

The relax function returns a square matrix in which:

- an element's location in the matrix corresponds to its location within the square region, and
- its value approximates the value of the solution at that point.

This function uses the relaxation method to converge to the solution. Poisson's equation on a square domain is represented by:

$$a_{j,k}u_{j+1,k} + b_{j,k}u_{j-1,k} + c_{j,k}u_{j,k+1} + d_{j,k}u_{j,k-1} + e_{j,k}u_{j,k} = f_{j,k}.$$

Algorithm: Gauss-Seidel with successive overrelaxation (Press *et al.*, 1992)

See also: multigrid

reverse — Sorting

One-dimensional Case

Syntax: reverse(**v**)

Description: Reverses the order of the elements of vector **v**.

Arguments
v vector

Two-dimensional Case

Syntax: reverse(**A**)

Description: Reverses the order of the rows of matrix **A**.

Arguments
A matrix

See also: See sort for sample application.

rexp — Random Numbers

Syntax: rexp(m, r)

Description: Returns a vector of m random numbers having the exponential distribution.

Arguments
m integer, $m > 0$
r real rate, $r > 0$

See also: rnd

Algorithm: Inverse cumulative density method (Press *et al.*, 1992)

rF

Random Numbers

Syntax rF(m, $d1$, $d2$)

Description Returns a vector of m random numbers having the F distribution.

Arguments

m integer, $m > 0$

d1, d2 integer degrees of freedom, $d1 > 0$, $d2 > 0$

Algorithm Best's XG algorithm, Johnk's generator (Devroye, 1986)

See also rnd

rgamma

Random Numbers

Syntax rgamma(m, s)

Description Returns a vector of m random numbers having the gamma distribution.

Arguments

m integer, $m > 0$

s real shape parameter, $s > 0$

Algorithm Best's XG algorithm, Johnk's generator (Devroye, 1986)

See also rnd

rgeom

Random Numbers

Syntax rgeom(m, p)

Description Returns a vector of m random numbers having the geometric distribution.

Arguments

m integer, $m > 0$

p real number, $0 < p \le 1$

Algorithm Inverse cumulative density method (Press *et al.*, 1992)

See also rnd

rhypergeom — Random Numbers

Syntax rhypergeom(*m, a, b, n*)

Description Returns a vector of *m* random numbers having the hypergeometric distribution.

Arguments

m — integer, $m > 0$

a, b, n — integers, $0 \le a$, $0 \le b$, $0 \le n \le a + b$

Algorithm Uniform sampling methods (Devroye, 1986)

See also rnd

rkadapt — *(Professional)* — Differential Equation Solving

Syntax rkadapt(**y**, *x1, x2, acc,* **D**, *kmax, save*)

Description Solves a differential equation using a slowly varying Runge-Kutta method. Provides DE solution estimate at *x2*.

Arguments *Several arguments for this function are the same as described for* rkfixed.

y — real vector of initial values

x1, x2 — real endpoints of the solution interval

D(*x*, **y**) — real vector-valued function containing the derivatives of the unknown functions

acc — real *acc* > 0 controls the accuracy of the solution; a small value of *acc* forces the algorithm to take smaller steps along the trajectory, thereby increasing the accuracy of the solution. Values of *acc* around 0.001 will generally yield accurate solutions.

kmax — integer *kmax* > 0 specifies the maximum number of intermediate points at which the solution will be approximated. The value of *kmax* places an upper bound on the number of rows of the matrix returned by these functions.

save — real *save* > 0 specifies the smallest allowable spacing between the values at which the solutions are to be approximated. *save* places a lower bound on the difference between any two numbers in the first column of the matrix returned by the function.

Comments The specialized DE solvers Bulstoer, Rkadapt, Stiffb, and Stiffr provide the solution *y(x)* over a number of uniformly spaced *x*-values in the integration interval bounded by *x1* and *x2*. When you want the value of the solution at only the endpoint, *y*(*x2*), use bulstoer, rkadapt, stiffb, and stiffr instead.

Algorithm Adaptive step 5th order Runge-Kutta method (Press *et al*., 1992)

See also rkfixed, a more general differential equation solver, for information on output and arguments; Rkadapt.

Rkadapt

(Professional) Differential Equation Solving

Syntax Rkadapt(**y**, *x1*, *x2*, *npts*, **D**)

Description Solves a differential equation using a slowly varying Runge-Kutta method; provides DE solution at equally spaced *x* values by repeated calls to rkadapt.

Arguments *All arguments for this function are the same as described for* rkfixed.

y real vector of initial values

x1, *x2* real endpoints of the solution interval

npts integer *npts* > 0 specifies the number of points beyond initial point at which the solution is to be approximated; controls the number of rows in the matrix output

D(*x*, **y**) real vector-valued function containing the derivatives of the unknown functions

Comments Given a fixed number of points, you can approximate a function more accurately if you evaluate it frequently wherever it's changing fast, and infrequently wherever it's changing more slowly. If you know that the solution has this property, you may be better off using Rkadapt. Unlike rkfixed which evaluates a solution at equally spaced intervals, Rkadapt examines how fast the solution is changing and adapts its step size accordingly. This “adaptive step size control” enables Rkadapt to focus on those parts of the integration domain where the function is rapidly changing rather than wasting time on the parts where change is minimal.

Although Rkadapt will use nonuniform step sizes internally when it solves the differential equation, it will nevertheless return the solution at equally spaced points.

Rkadapt takes the same arguments as rkfixed, and the matrix returned by Rkadapt is identical in form to that returned by rkfixed.

Algorithm Fixed step Runge-Kutta method with adaptive intermediate steps (5th order) (Press *et al.*, 1992)

See also rkfixed, a more general differential equation solver, for information on output and arguments.

rkfixed

Differential Equation Solving

Syntax rkfixed(**y**, *x1*, *x2*, *npts*, **D**)

Description Solves a differential equation using a standard Runge-Kutta method. Provides DE solution at equally spaced *x* values.

Arguments

y real vector of initial values (whose length depends on the order of the DE or the size of the system of DEs). For a first order DE like that in Example 1 or Example 2 below, the vector degenerates to one point, $y(0) = y(x1)$. For a second order DE like that in Example 3, the vector has two elements: the value of the function and its first derivative at *x1*. For higher order DEs like that in Example 4, the vector has *n* elements for specifying initial conditions of $y, y', y'', \ldots, y^{(n-1)}$. For a first order system like that in Example 5, the vector contains initial values for each unknown function. For higher order systems like that in Example 6, the vector contains initial values for the $n-1$ derivatives of each unknown function in addition to initial values for the functions themselves.

x1, *x2* real endpoints of the interval on which the solution to the DEs will be evaluated; initial values in **y** are the values at *x1*.

npts — integer *npts* > 0 specifies the number of points beyond the initial point at which the solution is to be approximated; controls the number of rows in the matrix output.

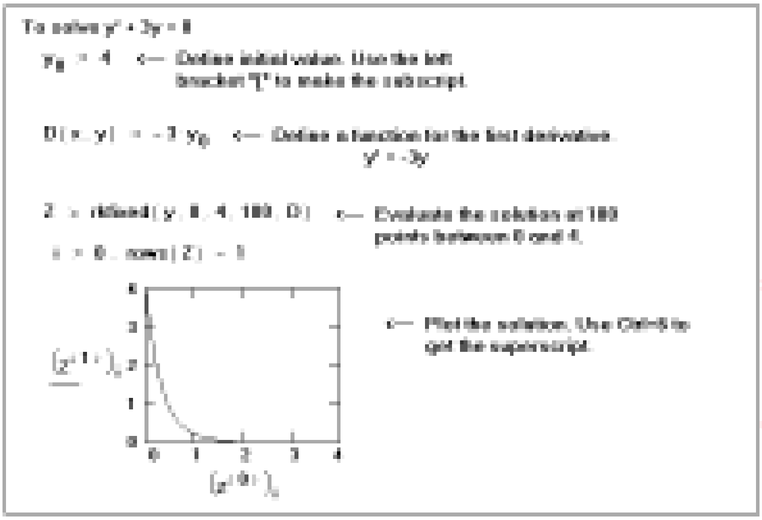
D$(x, \mathbf{y})$ — real vector-valued function containing derivatives of the unknown functions. For a first order DE like that in Example 1 or Example 2, the vector degenerates to a scalar function. For a second order DE like that in Example 3, the vector has two elements:

$$\mathbf{D}(t, \mathbf{y}) = \begin{bmatrix} y'(t) \\ y''(t) \end{bmatrix}$$

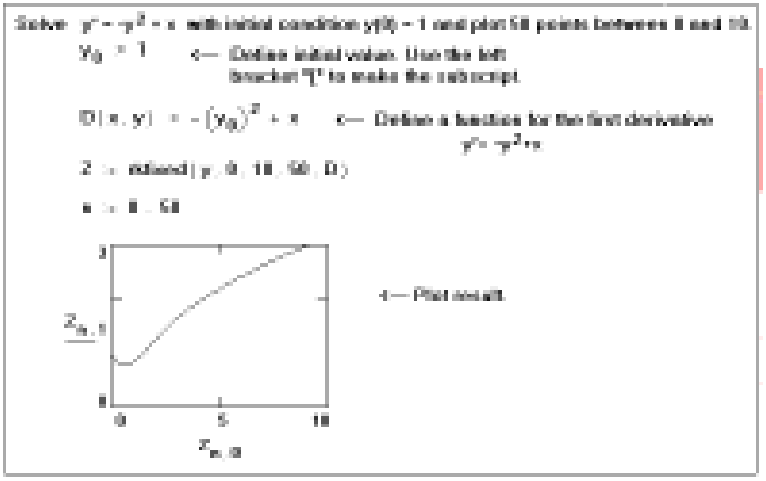
For higher order DEs like that in Example 4, the vector has n elements: $\mathbf{D}(t, \mathbf{y}) = \begin{bmatrix} y'(t) \\ y''(t) \\ \cdot \\ \cdot \\ \cdot \\ y^{(n)}(t) \end{bmatrix}$.

For a first order system like that in Example 5, the vector contains the first derivatives of each unknown function. For higher order systems like that in Example 6, the vector contains expressions for the $n-1$ derivatives of each unknown function in addition to nth derivatives.

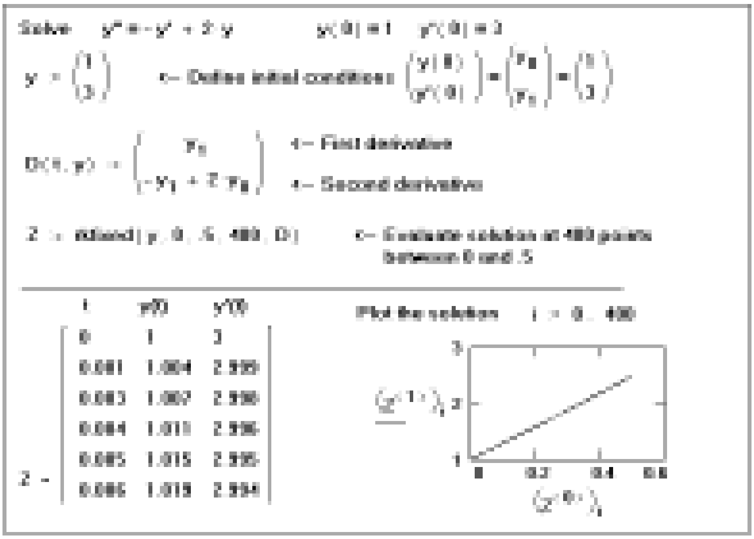
Examples



Example 1: Solving a first order differential equation.

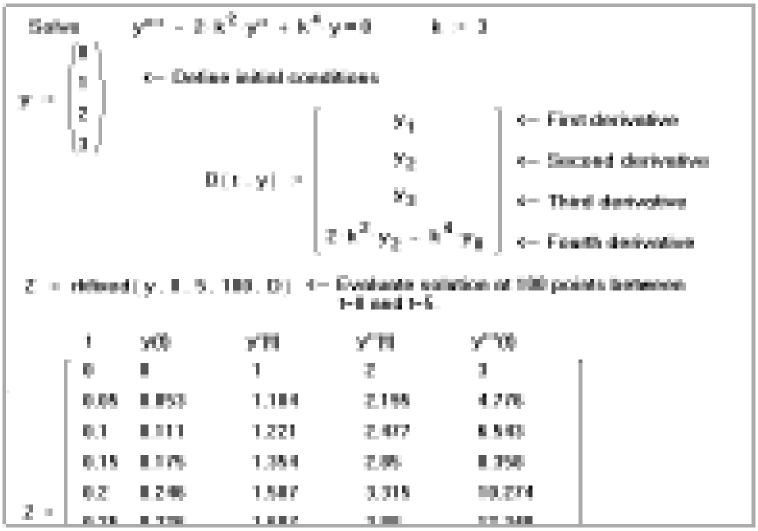


Example 2: Solving a nonlinear differential equation.

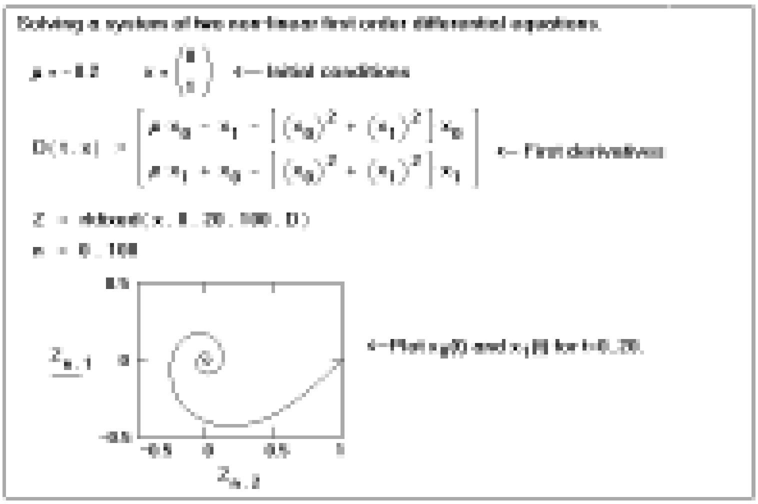


Example 3: Solving a second order differential equation.

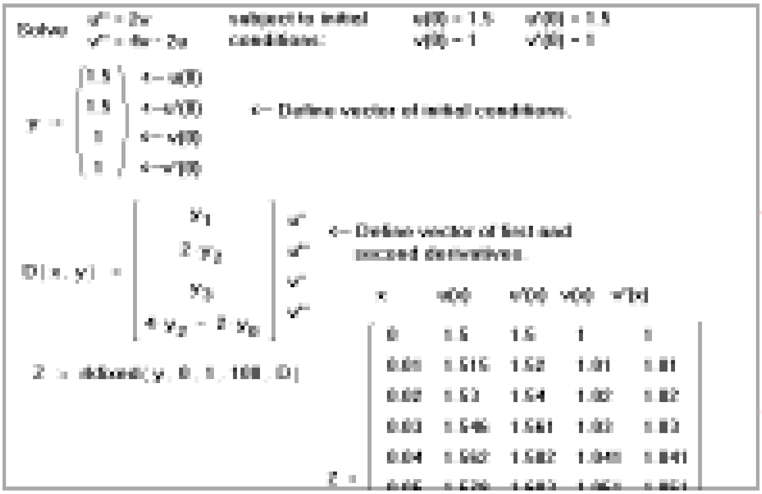
.



Example 4: Solving a higher order differential equation.



Example 5: Solving a system of first order linear equations.



Example 6: Solving a system of second order linear differential equations.

Comments

For a first order DE like that in Example 1 or Example 2, the output of rkfixed is a two-column matrix in which:

- The left-hand column contains the points at which the solution to the DE is evaluated.
- The right-hand column contains the corresponding values of the solution.

For a second order DE like that in Example 3, the output matrix contains three columns: the left-hand column contains the *t* values; the middle column contains $y(t)$; and the right-hand column contains $y'(t)$.
For higher order DEs like that in Example 4, the output matrix contains *n* columns: the left-hand one for the *t* values and the remaining columns for values of $y(t), y'(t), y''(t), \ldots, y^{(n-1)}(t)$.

For a first order system like that in Example 5, the first column of the output matrix contains the points at which the solutions are evaluated and the remaining columns contain corresponding values of the solutions. For higher order systems like that in Example 6:

- The first column contains the values at which the solutions and their derivatives are evaluated.
- The remaining columns contain corresponding values of the solutions and their derivatives. The order in which the solutions and their derivatives appear matches the order in which you put them into the vector of initial conditions.

The most difficult part of solving a DE is defining the function $\mathbf{D}(x, \mathbf{y})$. In Example 1 and Example 2, for example, it was easy to solve for $y'(x)$. In some more difficult cases, you can solve for $y'(x)$ symbolically and paste it into the definition for $\mathbf{D}(x, \mathbf{y})$. To do so, use the solve keyword or the **Solve for Variable** command from the **Symbolics** menu.

The function rkfixed uses a fourth order Runge-Kutta method, which is a good general-purpose DE solver. Although it is not always the fastest method, the Runge-Kutta method nearly always succeeds. There are certain cases in which you may want to use one of Mathcad's more specialized DE solvers. These cases fall into three broad categories:

- Your system of DEs may have certain properties which are best exploited by functions other than rkfixed. The system may be stiff (Stiffb, Stiffr); the functions could be smooth (Bulstoer) or slowly varying (Rkadapt).
- You may have a boundary value rather than an initial value problem (sbval and bvalfit).
- You may be interested in evaluating the solution only at one point (bulstoer, rkadapt, stiffb and stiffr).

You may also want to try several methods on the same DE to see which one works the best. Sometimes there are subtle differences between DEs that make one method better than another.

Algorithm Fixed step 4th order Runge-Kutta method (Press *et al.*, 1992)

See also Mathcad Resource Center QuickSheets and Differential Equations tutorial.

rlnorm

Random Numbers

Syntax rlnorm(m, μ, σ)

Description Returns a vector of m random numbers having the lognormal distribution.

Arguments

m integer, $m > 0$

μ real logmean

σ real logdeviation, $\sigma > 0$

Algorithm Ratio-of-uniforms method (Devroye, 1986)

See also rnd

rlogis

Random Numbers

Syntax rlogis(m, l, s)

Description Returns a vector of m random numbers having the logistic distribution.

Arguments

m integer, $m > 0$

l real location parameter

s real scale parameter, $s > 0$

Algorithm Inverse cumulative density method (Press *et al.*, 1992)

See also rnd

rnbinom

Random Numbers

Syntax rnbinom(m, n, p)

Description Returns a vector of m random numbers having the negative binomial distribution.

Arguments

m, n integers, $m > 0$, $n > 0$

p real number, $0 < p \le 1$

Algorithm Based on rpois and rgamma (Devroye, 1986)

See also rnd

rnd

Random Numbers

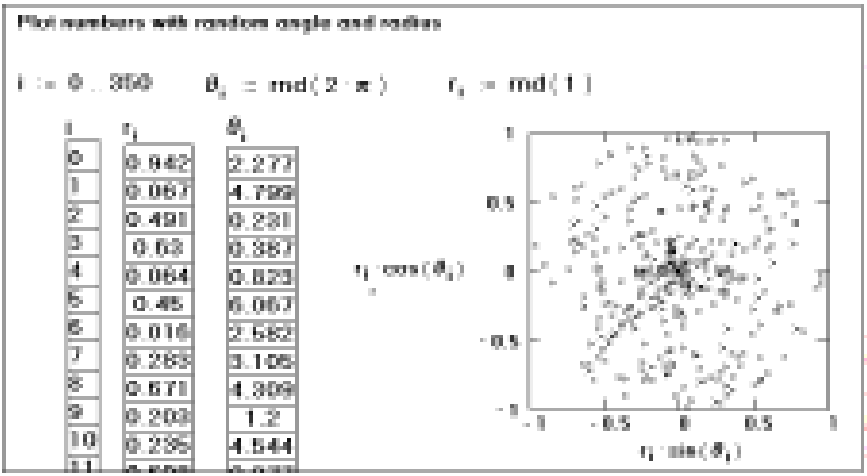
Syntax rnd(x)

Description Returns a random number between 0 and x. Identical to runif(1, 0, x) if $x > 0$.

Arguments

x real number

Example



Note: You won't be able to recreate this example exactly because the random number generator gives different numbers every time.

Comments Each time you recalculate an equation containing rnd or some other random variate built-in function, Mathcad generates new random numbers. Recalculation is performed by clicking on the equation and choosing **Calculate** from the **Math** menu.

These functions have a “seed value” associated with them. Each time you reset the seed, Mathcad generates new random numbers based on that seed. A given seed value will always generate the same sequence of random numbers. Choosing **Calculate** from the **Math** menu advances Mathcad along this random number sequence. Changing the seed value, however, advances Mathcad along an altogether different random number sequence.

To change the seed value, choose **Options** from the **Math** menu and change the value of “seed” on the Built-In Variables tab. Be sure to supply an integer.

To reset Mathcad's random number generator without changing the seed value, choose **Options** from the **Math** menu, click on the Built-In Variables tab, and click “OK” to accept the current seed. Then click on the equation containing the random number generating function and choose **Calculate** from the **Math** menu. Since the randomizer has been reset, Mathcad generates the same random numbers it would generate if you restarted Mathcad.

There are many other random variate generators in Mathcad.

Algorithm — Linear congruence method (Knuth, 1997)

---

rnorm — Random Numbers

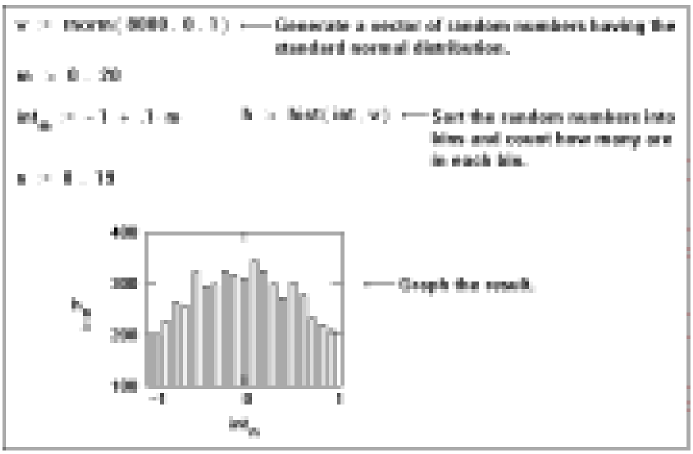
Syntax rnorm(m, μ, σ)

Description Returns a vector of m random numbers having the normal distribution.

Arguments

- m — integer, $m > 0$
- μ — real mean
- σ — real standard deviation, $\sigma > 0$

Example



Note: You won’t be able to recreate this example exactly because the random number generator gives different numbers every time.

Algorithm Ratio-of-uniforms method (Devroye, 1986)

See also rnd

root

Syntax root(*f(var), var*)

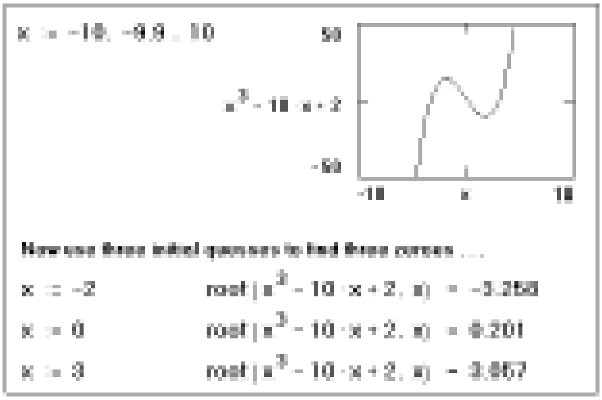
Description Returns a value of *var* at which the expression *f*(*var*) or function *f* is equal to 0.

Arguments

var real or complex scalar; *var* must be assigned a guess value before using this version of root.

f real or complex-valued function.

Example



Comments For expressions with several roots, your guess value determines which root Mathcad returns. The example shows a situation in which the root function returns several different values, each of which depends on the initial guess value.

You can't put numerical values in the list of unknowns; for example, root(*f(x)*, -2) or root(14, −2) is not permitted in the example above.

Mathcad solves for complex roots as well as real roots. To find a complex root, you must start with a complex value for the initial guess.

Solving an equation of the form $f(x) = g(x)$ is equivalent to using the root function as follows:

$$\mathrm{root}(f(x) - g(x), x)$$

The root function can solve only one equation in one unknown. To solve several equations simultaneously, use Find or Minerr. To solve an equation symbolically, that is, to find an exact numerical answer in terms of elementary functions, choose **Solve for Variable** from the **Symbolics** menu or use the solve keyword.

See also polyroot for an efficient means to compute all roots of a polynomial at once.

Mathcad evaluates the root function using the *secant method*. If that method fails to find a root, then the *Mueller method* is used. The guess value you supply for *x* becomes the starting point for successive approximations to the root value. When the magnitude of *f*(*x*) evaluated at the proposed root is less than the value of the predefined variable TOL, the root function returns a result.

If after many approximations Mathcad still cannot find an acceptable answer, it marks the root function with an error message indicating its inability to converge to a result. This error can be caused by any of the following:

- The expression has no roots.
- The roots of the expression are far from the initial guess.
- The expression has local maxima or minima between the initial guess and the roots.
- The expression has discontinuities between the initial guess and the roots.
- The expression has a complex root but the initial guess was real (or vice versa).

To find the cause of the error, try plotting the expression. This will help determine whether or not the expression crosses the *x*-axis and if so, approximately where. In general, the closer your initial guess is to where the expression crosses the *x*-axis, the more quickly the root function will converge on an acceptable result.

Here are some hints for getting the most out of the root function:

- To change the accuracy of the root function, change the value of the built-in variable TOL. If you increase TOL, the root function will converge more quickly, but the answer will be less accurate. If you decrease TOL, the root function will converge more slowly, but the answer will be more accurate. To change TOL at a specified point in the worksheet, include a definition like $TOL := 0.01$. To change TOL for the whole worksheet, choose **Options** from the **Math** menu, click on the Built-In Variables tab, and replace the number in the text box beside "TOL." After you click "OK," choose **Calculate Worksheet** from the **Math** menu to update the entire worksheet using the new value of TOL.
- If an expression has multiple roots, try different guess values to find them. Plotting the function is a good way to determine how many roots there are, where they are, and what initial guesses are likely to find them. Refer to the previous example. If two roots are close together, you may have to reduce TOL to distinguish between them.
- If $f(x)$ has a small slope near its root, then $\mathrm{root}(f(x), x)$ may converge to a value r that is relatively far from the actual root. In such cases, even though $|f(r)| < TOL$, r may be far from the point where $f(r) = 0$. To find a more accurate root, decrease the value of TOL. Or, try finding root($g(x)$,x), where $g(x) = \dfrac{f(x)}{\frac{d}{dx}f(x)}$.
- For an expression $f(x)$ with a known root r, solving for additional roots of $f(x)$ is equivalent to solving for roots of $h(x) = (f(x))/(x-r)$. Dividing out known roots like this is useful for resolving two roots that may be close together. It's often easier to solve for roots of $h(x)$ as defined here than it is to try to find other roots for $f(x)$ with different guesses.

Algorithm Secant and Mueller methods (Press *et al.*, 1992; Lorczak)

round

Truncation and Round-off

One-argument Version

Syntax round(x)

Description Rounds the real number x to the nearest integer. Same as round(x, 0).

Arguments

x real number

Two-argument Version

Syntax round(x, n)

Description Rounds the real number x to n decimal places. If $n < 0$, x is rounded to the left of the decimal point.

Arguments

x real number

n integer

See also ceil, floor, trunc

rows

Vector and Matrix

Syntax rows(**A**)

Description Returns the number of rows in array **A**.

Arguments

A matrix or vector

See also cols for example

rpois

Random Numbers

Syntax rpois(m, λ)

Description Returns a vector of m random numbers having the Poisson distribution.

Arguments

m integer, $m > 0$

λ real mean, $\lambda > 0$

Algorithm Devroye, 1986

See also rnd

rref

Vector and Matrix

Syntax rref(**A**)

Description Returns a matrix representing the row-reduced echelon form of **A**.

Arguments

A real $m \times n$ matrix

Algorithm Elementary row reduction (Anton)

rsort

Sorting

Syntax rsort(**A**, *i*)

Description Sorts the columns of the matrix **A** by placing the elements in row *i* in ascending order. The result is the same size as **A**.

Arguments

A $m \times n$ matrix or vector

i integer, $0 \le i \le m - 1$

Algorithm Heap sort (Press *et al.*, 1992)

See also sort for more details, csort

rt

Random Numbers

Syntax rt(*m*, *d*)

Description Returns a vector of *m* random numbers having Student's *t* distribution.

Arguments

m integer, $m > 0$

d integer degrees of freedom, $d > 0$

Algorithm Best's XG algorithm, Johnk's generator (Devroye, 1986)

See also rnd

runif

Random Numbers

Syntax runif(*m*, *a*, *b*)

Description Returns a vector of *m* random numbers having the uniform distribution

Arguments

m integer, $m > 0$

a, *b* real numbers, $a < b$

Algorithm Linear congruence method (Knuth, 1997)

See also rnd

rweibull

Random Numbers

Syntax rweibull(*m, s*)

Description Returns a vector of *m* random numbers having the Weibull distribution.

Arguments

m integer, $m > 0$

s real shape parameter, $s > 0$

Algorithm Inverse cumulative density method (Press *et al*., 1992)

See also rnd

SaveColormap

File Access

Syntax SaveColormap(*file*, **M**)

Description Creates a colormap *file* containing the values in the matrix **M**. Returns the number of rows written to *file*.

Arguments

file string variable corresponding to CMP filename

M integer matrix with three columns and whose elements $\mathbf{M}_{i,j}$ all satisfy $0 \le \mathbf{M}_{i,j} \le 255$.

Comments The file *file* is the name of a colormap located in the CMAPS subdirectory of your Mathcad directory. After you use SaveColormap, the colormap is available on the Advanced tab in the 3D Plot Format dialog box. See on-line Help for more information.

See also LoadColormap

sbval

(Professional)

Differential Equation Solving

Syntax sbval(**v**, *x1*, *x2*, **D, load, score**)

Description Converts a boundary value differential equation to an initial value problem. Useful when derivatives are continuous throughout.

Arguments

v real vector containing guesses for missing initial values

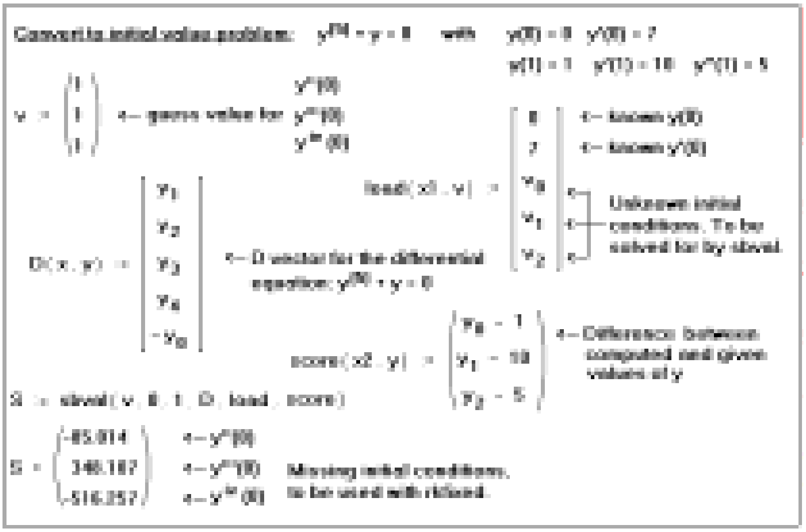
x1, *x2* real endpoints of the interval on which the solution to the DEs will be evaluated

D(*x*, **y**) real *n*-element vector-valued function containing the derivatives of the unknown functions

load(*x1*, **v**) real vector-valued function whose *n* elements correspond to the values of the *n* unknown functions at *x1*. Some of these values will be constants specified by your initial conditions. If a value is unknown, you should use the corresponding guess value from **v**

score(*x2*, **y**) real *n*-element vector-valued function which measures solution discrepancy at *x2*

Example



Comments

Initial value DE solvers like rkfixed assume that you know the value of the solution and its first $n-1$ derivatives at the beginning of the interval of integration. Two-point boundary value DE solvers, like sbval and bvalfit, may be used if you lack this information about the solution at the beginning of the interval of integration, but you do know something about the solution elsewhere in the interval. In particular:

- You have an *n*th order differential equation.
- You know some but not all of the values of the solution and its first $n-1$ derivatives at the beginning of the interval of integration, *x1*.
- You know some but not all of the values of the solution and its first $n-1$ derivatives at the end of the interval of integration, *x2*.
- Between what you know about the solution at *x1* and what you know about it at *x2*, you have *n* known values.

If there is a discontinuity at a point intermediate to *x1* and *x2*, you should use bvalfit. If continuity holds throughout, then use sbval to evaluate those initial values left unspecified at *x1*. sbval does not actually return a solution to a differential equation; it merely computes the initial values the solution must have in order for the solution to match the final values you specify. You must then take the initial values returned by sbval and solve the resulting initial value problem using rkfixed or any of the other more specialized DE solvers.

Algorithm

Shooting method with 4th order Runge-Kutta method (Press *et al.*, 1992)

See also

rkfixed for more details

search

(Professional) String

Syntax search($S, SubS, m$)

Description Returns the starting position of the substring *SubS* in *S* beginning from position m. Returns -1 if the substring is not found.

Arguments

S string expression; Mathcad assumes that the first character in *S* is at position 0

SubS substring expression

m integer, $m \geq 0$

sec

Trigonometric

Syntax sec(z), for z in radians;
sec($z \cdot \text{deg}$), for z in degrees

Description Returns the secant of z.

Arguments

z real or complex number

sech

Hyperbolic

Syntax sech(z)

Description Returns the hyperbolic secant of z.

Arguments

z real or complex number

sign

Piecewise Continuous

Syntax sign(x)

Description Returns 0 if x=0, 1 if $x > 0$, and -1 otherwise.

Arguments

x real number

See also csgn, signum

signum

Complex Numbers

Syntax signum(z)

Description Returns 1 if z=0 and $z/|z|$ otherwise.

Arguments

z real or complex number

See also csgn, sign

sin

Trigonometric

Syntax sin(z), for z in radians;
sin($z\cdot$deg), for z in degrees

Description Returns the sine of z.

Arguments

z real or complex number

sinh

Hyperbolic

Syntax sinh(z)

Description Returns the hyperbolic sine of z.

Arguments

z real or complex number

skew

Statistics

Syntax skew(**A**)

Description Returns the skewness of the elements of **A**:

$$\mathrm{skew}(\mathbf{A}) = \frac{mn}{(mn-1)(mn-2)}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left(\frac{\mathbf{A}_{i,j}-\mathrm{mean}(\mathbf{A})}{\mathrm{Stdev}(\mathbf{A})}\right)^3$$

Arguments

A real or complex $m \times n$ matrix or vector, $m \cdot n \geq 3$

slope

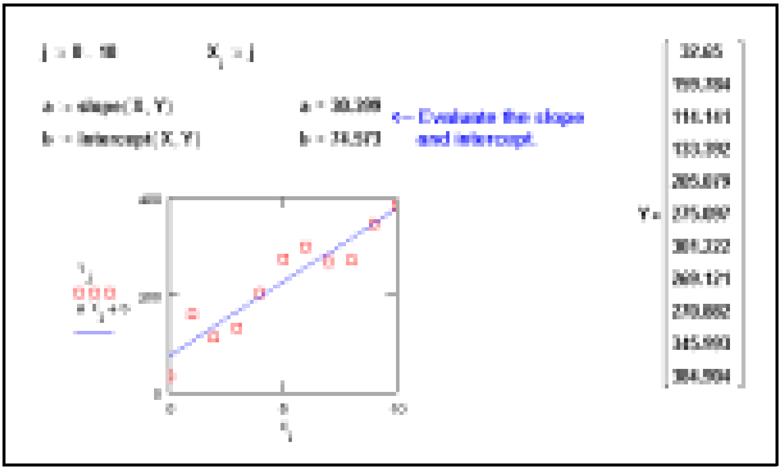
Syntax slope(**vx**, **vy**)

Description Returns the slope of the least-squares regression line.

Arguments

vx, vy real vector arguments of the same size

Example



Comments The functions slope and intercept return the slope and intercept of the line which best fits the data in a least-squares sense: $y = \mathrm{slope}(\mathbf{vx}, \mathbf{vy}) \cdot x + \mathrm{intercept}(\mathbf{vx}, \mathbf{vy})$.

Be sure that every element in the **vx** and **vy** arrays contains a data value. Since every element in an array must have a value, Mathcad assigns 0 to any elements not explicitly assigned.

These functions are useful not only when the data is inherently linear, but also when it is exponential. If x and y are related by $y = Ae^{kx}$, you can apply these functions to the logarithm of the data values and make use of the fact that $\ln(y) = \ln(A) + kx$, hence $A = \exp(\mathrm{intercept}(\mathbf{vx}, \ln(\mathbf{vy})))$ and $k = \mathrm{slope}(\mathbf{vx}, \ln(\mathbf{vy}))$.

The resulting fit weighs the errors differently from a least-squares exponential fit (which the function genfit provides) but is usually a good approximation.

See also intercept, stderr

sort

Sorting

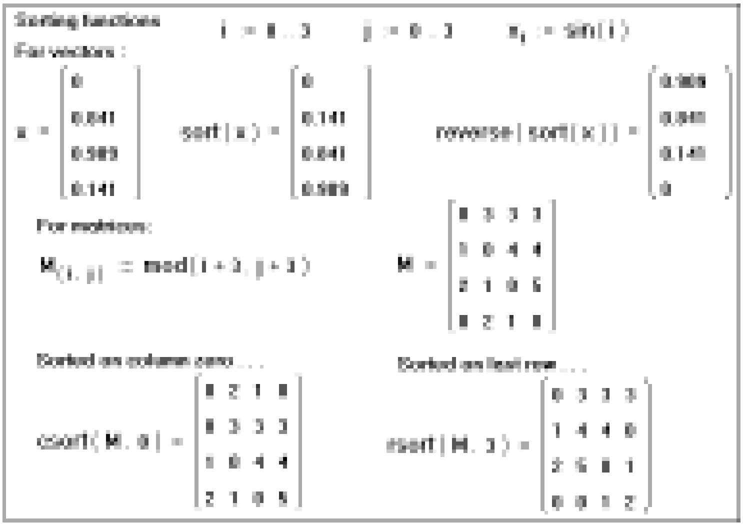
Syntax sort(**v**)

Description Returns the elements of vector **v** sorted in ascending order.

Arguments

v vector

Example



Comments All of Mathcad's sorting functions accept matrices and vectors with complex elements. However in sorting them, Mathcad ignores the imaginary part.

To sort a vector or matrix in descending order, first sort in ascending order, then use reverse. For example, reverse(sort(**v**)) returns the elements of **v** sorted in descending order.

Unless you change the value of ORIGIN, matrices are numbered starting with row zero and column zero. If you forget this, it's easy to make the error of sorting a matrix on the wrong row or column by specifying an incorrect *n* argument for rsort and csort. To sort on the first column of a matrix, for example, you must use csort(**A**, 0).

Algorithm Heap sort (Press *et al.*, 1992)

stack

Vector and Matrix

Syntax stack(**A**, **B**)

Description Returns a matrix formed by placing the matrix **A** above **B**.

Arguments

A, **B** two matrices or vectors; **A** and **B** must have the same number of columns

See also augment for example

stderr

Syntax
stderr(**vx, vy**)

Description
Returns the standard error associated with simple linear regression, measuring how closely data points are spread about the regression line.

$$\mathrm{stderr}(\mathbf{vx},\mathbf{vy}) = \sqrt{\frac{1}{n-2}\sum_{i=0}^{n-1}\left(\mathbf{vy}_i - (\mathrm{intercept}(\mathbf{vx},\mathbf{vy}) + \mathrm{slope}(\mathbf{vx},\mathbf{vy})\cdot\mathbf{vx}_i)\right)^2}\ .$$

Arguments
vx, vy real vector arguments of the same size

See also
slope, intercept

stdev

Syntax
stdev(**A**)

Description
Returns the standard deviation of the elements of **A**, where *mn* (the sample size) is used in the denominator: $\mathrm{stdev}(\mathbf{A}) = \sqrt{\mathrm{var}(\mathbf{A})}$.

Arguments
A real or complex $m \times n$ matrix or vector

See also
Stdev, var, Var

Stdev

Syntax
Stdev(**A**)

Description
Returns the standard deviation of the elements of **A**, where *mn* – 1 (the sample size less one) is used in the denominator: $\mathrm{Stdev}(\mathbf{A}) = \sqrt{\mathrm{Var}(\mathbf{A})}$.

Arguments
A real or complex $m \times n$ matrix or vector

See also
stdev, var, Var

stiffb

Syntax stiffb(**y**, *x1*, *x2*, *acc*, **D**, **J**, *kmax*, *save*)

Description Solves a differential equation using the stiff Bulirsch-Stoer method. Provides DE solution estimate at *x2*.

Arguments *Several arguments for this function are the same as described for* rkfixed.

y real vector of initial values.

x1, *x2* real endpoints of the solution interval.

D(*x*, **y**) real vector-valued function containing the derivatives of the unknown functions.

acc real $acc > 0$ controls the accuracy of the solution; a small value of *acc* forces the algorithm to take smaller steps along the trajectory, thereby increasing the accuracy of the solution. Values of *acc* around 0.001 will generally yield accurate solutions.

J(*x*, **y**) real vector-valued function which returns the $n \times (n + 1)$ matrix whose first column contains the derivatives $\partial \mathbf{D} / \partial x$ and whose remaining columns form the Jacobian matrix ($\partial \mathbf{D} / \partial y_k$) for the system of DEs.

kmax integer $kmax > 0$ specifies maximum number of intermediate points at which the solution will be approximated; places an upper bound on the number of rows of the matrix returned by these functions.

save real $save > 0$ specifies the smallest allowable spacing between values at which the solutions are to be approximated; places a lower bound on the difference between any two numbers in the first column of the matrix returned by the function.

Comments The specialized DE solvers Bulstoer, Rkadapt, Stiffb, and Stiffr provide the solution *y(x)* over a number of uniformly spaced *x*-values in the integration interval bounded by *x1* and *x2*. When you want the value of the solution at only the endpoint, *y*(*x2*), use bulstoer, rkadapt, stiffb, and stiffr instead.

Algorithm Bulirsch-Stoer method with adaptive step size for stiff systems (Press *et al*., 1992)

See also rkfixed, a more general differential equation solver, for information on output and arguments; Stiffb.

Stiffb (Professional) Differential Equation Solving

Syntax Stiffb(**y**, *x1*, *x2*, *npts*, **D**, **J**)

Description Solves a differential equation using the stiff Bulirsch-Stoer method. Provides DE solution at equally spaced x values by repeated calls to stiffb.

Arguments *Several arguments for this function are the same as described for* rkfixed.

y real vector of initial values.

x1, *x2* real endpoints of the solution interval.

D(*x*, **y**) real vector-valued function containing the derivatives of the unknown functions.

npts integer $npts > 0$ specifies the number of points beyond initial point at which the solution is to be approximated; controls the number of rows in the matrix output.

J(*x*, **y**) real vector-valued function which returns the $n \times (n+1)$ matrix whose first column contains the derivatives $\partial \mathbf{D} / \partial x$ and whose remaining columns form the Jacobian matrix ($\partial \mathbf{D} / \partial y_k$) for the system of DEs. For example, if:

$$\mathbf{D}(x, \mathbf{y}) = \begin{bmatrix} x \cdot y_1 \\ -2 \cdot y_1 \cdot y_0 \end{bmatrix} \quad \text{then} \quad \mathbf{J}(x, \mathbf{y}) = \begin{bmatrix} y_1 & 0 & x \\ 0 & -2 \cdot y_1 & -2 \cdot y_0 \end{bmatrix}$$

Comments A system of DEs expressed in the form $\mathbf{y} = \mathbf{A} \cdot \mathbf{x}$ is a stiff system if the matrix **A** is nearly singular. Under these conditions, the solution returned by rkfixed may oscillate or be unstable. When solving a stiff system, you should use one of the two DE solvers specifically designed for stiff systems: Stiffb and Stiffr. These use the Bulirsch-Stoer method and the Rosenbrock method, respectively, for stiff systems.

The form of the matrix returned by these functions is identical to that returned by rkfixed. However, Stiffb and Stiffr require an extra argument **J**(*x*, **y**).

Algorithm Fixed-step Bulirsch-Stoer method with adaptive intermediate step size for stiff systems (Press *et al.*, 1992)

See also rkfixed, a more general differential equation solver, for information on output and arguments.

stiffr (Professional) Differential Equation Solving

Syntax stiffr(**y**, *x1*, *x2*, *acc*, **D**, **J**, *kmax*, *save*)

Description Solves a differential equation using the stiff Rosenbrock method. Provides DE solution estimate at *x2*.

Arguments *Several arguments for this function the same as described for rkfixed.*

y real vector of initial values.

x1, *x2* real endpoints of the solution interval.

D(*x*, **y**) real vector-valued function containing the derivatives of the unknown functions.

acc real $acc > 0$ controls the accuracy of the solution; a small value of *acc* forces the algorithm to take smaller steps along the trajectory, thereby increasing the accuracy of the solution. Values of *acc* around 0.001 will generally yield accurate solutions.

J$(x, \mathbf{y})$ real vector-valued function that returns the $n \times (n+1)$ matrix whose first column contains the derivatives $\partial \mathbf{D} / \partial x$ and whose remaining columns form the Jacobian matrix ($\partial \mathbf{D} / \partial y_k$) for the system of DEs.

kmax integer $kmax > 0$ specifies maximum number of intermediate points at which the solution will be approximated; places an upper bound on the number of rows of the matrix returned by these functions.

save real $save > 0$ specifies the smallest allowable spacing between values at which the solutions are to be approximated; places a lower bound on the difference between any two numbers in the first column of the matrix returned by the function.

Comments

The specialized DE solvers Bulstoer, Rkadapt, Stiffb, and Stiffr provide the solution *y(x)* over a number of uniformly spaced *x*-values in the integration interval bounded by *x1* and *x2*. When you want the value of the solution at only the endpoint, $y(x2)$, use bulstoer, rkadapt, stiffb, and stiffr instead.

Algorithm

4th order Rosenbrock method with adaptive intermediate step size for stiff systems (Press *et al.*, 1992)

See also

rkfixed, a more general differential equation solver for information on output and arguments, and Stiffr

Stiffr *(Professional)* Differential Equation Solving

Syntax

Stiffb(**y**, *x1, x2, npts,* **D**, **J**)

Description

Solves a differential equation using the stiff Rosenbrock method. Provides DE solution at equally spaced *x* values by repeated calls to stiffr.

Arguments

Several arguments for this function are the same as described for rkfixed*.*

y real vector of initial values.

x1, x2 real endpoints of the solution interval.

D$(x, \mathbf{y})$ real vector-valued function containing the derivatives of the unknown functions.

npts integer $npts > 0$ specifies the number of points beyond initial point at which the solution is to be approximated; controls the number of rows in the matrix output.

J$(x, \mathbf{y})$ real vector-valued function which returns the $n \times (n+1)$ matrix whose first column contains the derivatives $\partial \mathbf{D} / \partial x$ and whose remaining columns form the Jacobian matrix ($\partial \mathbf{D} / \partial y_k$) for the system of DEs. For example, if:

$$\mathbf{D}(x, \mathbf{y}) = \begin{bmatrix} x \cdot y_1 \\ -2 \cdot y_1 \cdot y_0 \end{bmatrix}, \quad \text{then} \quad \mathbf{J}(x, \mathbf{y}) = \begin{bmatrix} y_1 & 0 & x \\ 0 & -2 \cdot y_1 & -2 \cdot y_0 \end{bmatrix}$$

Comments

A system of DEs expressed in the form $\mathbf{y} = \mathbf{A} \cdot \mathbf{x}$ is a stiff system if the matrix **A** is nearly singular. Under these conditions, the solution returned by rkfixed may oscillate or be unstable. When solving a stiff system, you should use one of the two DE solvers specifically designed for stiff systems: Stiffb and Stiffr. These use the Bulirsch-Stoer method and the Rosenbrock method, respectively, for stiff systems.

The form of the matrix returned by these functions is identical to that returned by rkfixed. However, Stiffb and Stiffr require an extra argument **J**$(x, \mathbf{y})$.

Algorithm Fixed-step 4th order Rosenbrock method with adaptive intermediate step size for stiff systems (Press *et al.*, 1992)

See also rkfixed, a more general differential equation solver, for information on output and arguments

str2num *(Professional)* String

Syntax str2num(*S*)

Description Returns the constant formed by converting the characters in *S* into a number. Characters in *S* must constitute an integer such as 17, a real floating-point number such as -16.5, a complex floating-point number such as 2.1+6i or 3.241 - 9.234j, or an e-format number such as 4.51e-3 (for $4.51 \cdot 10^{-3}$). Mathcad ignores any spaces in the string.

Arguments
S string expression

See also num2str

str2vec *(Professional)* String

Syntax str2vec(*S*)

Description Returns the vector of ASCII codes corresponding to the characters in string *S*. For a list of ASCII codes, see Appendix to the *Mathcad User's Guide*. For example, the ASCII code for letter "a" is 97, that for letter "b" is 98, and that for letter "c" is 99.

Arguments
S string expression

See also vec2str

strlen *(Professional)* String

Syntax strlen(*S*)

Description Returns the number of characters in *S*.

Arguments
S string expression

submatrix

Syntax submatrix(**A**, *ir, jr, ic, jc*)

Description Returns a submatrix of **A** consisting of all elements common to rows *ir* through *jr* and columns *ic* through *jc*. Make certain that $ir \le jr$ and $ic \le jc$, otherwise the order of rows and/or columns will be reversed.

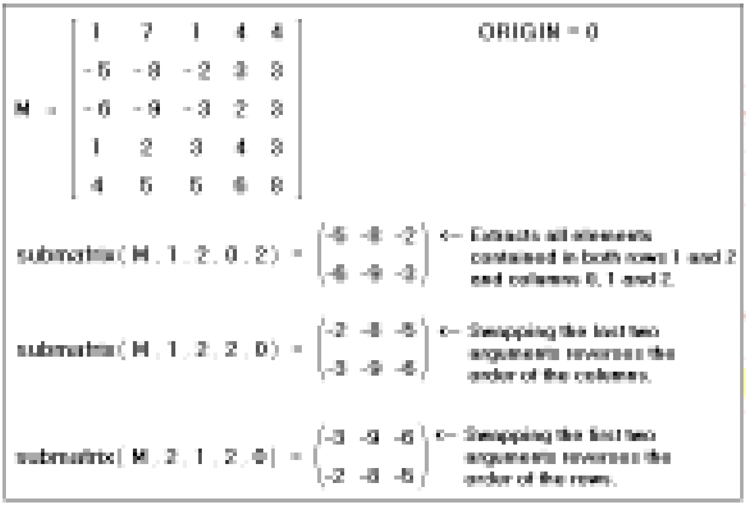
Arguments

A $m \times n$ matrix or vector

ir, jr integers, $0 \le ir \le jr \le m$

ic, jc integers, $0 \le ic \le jc \le n$

Example



substr

(Professional) String

Syntax substr(*S, m, n*)

Description Returns a substring of *S* beginning with the character in the *m*th position and having at most *n* characters.

Arguments

S string expression. Mathcad assumes that the first character in *S* is at position 0.

m, n integers, $m \ge 0$, $n \ge 0$

supsmooth *(Professional)* Regression and Smoothing

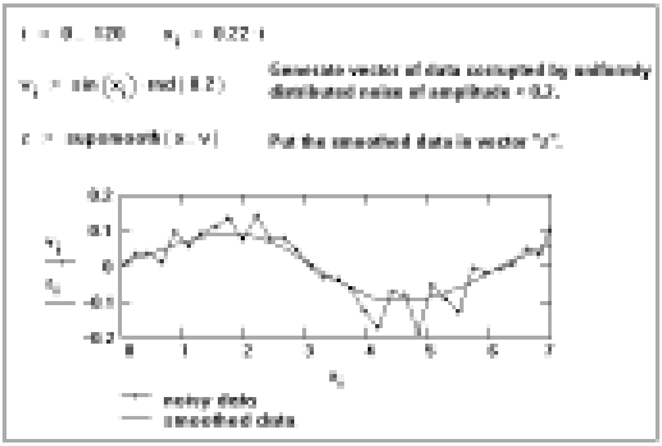
Syntax supsmooth(**vx**, **vy**)

Description Creates a new vector, of the same size as **vy**, by piecewise use of a symmetric k-nearest neighbor linear least square fitting procedure in which k is adaptively chosen.

Arguments

vx, vy real vectors of the same size; elements of **vx** must be in ascending order

Example



Comments The supsmooth function uses a symmetric k nearest neighbor linear least-squares fitting procedure to make a series of line segments through the data. Unlike ksmooth which uses a fixed bandwidth for all the data, supsmooth will adaptively choose different bandwidths for different portions of the data.

Algorithm Variable span super-smoothing method (Friedman)

See also medsmooth and ksmooth

svd *(Professional)* Vector and Matrix

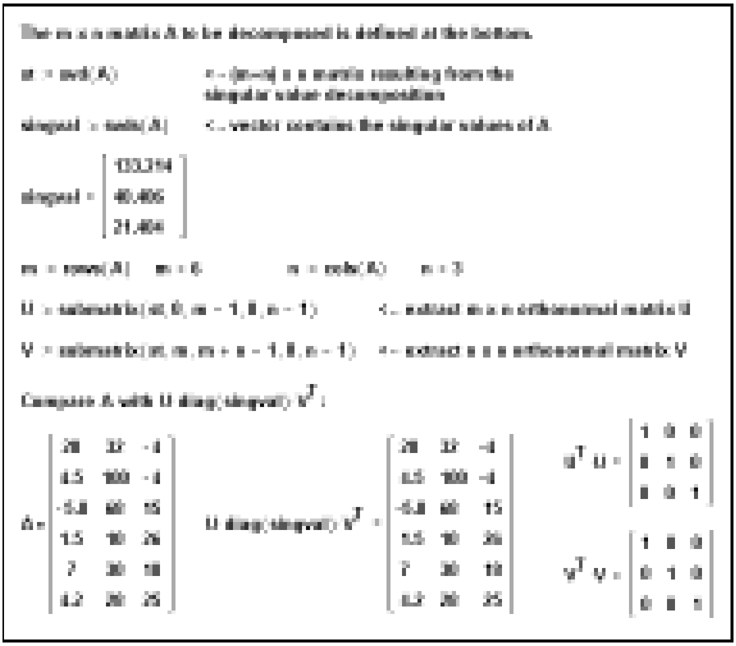
Syntax svd(**A**)

Description Returns an $(m + n) \times n$ matrix whose first m rows contain the $m \times n$ orthonormal matrix **U**, and whose remaining n rows contain the $n \times n$ orthonormal matrix **V**. Matrices **U** and **V** satisfy the equation $\mathbf{A} = \mathbf{U} \cdot \mathrm{diag}(\mathbf{s}) \cdot \mathbf{V}^{\mathbf{T}}$, where **s** is the vector returned by svds(**A**).

Arguments

A $m \times n$ real matrix, where $m \geq n$

Example



Algorithm Householder reduction with QR transformation (Wilkinson and Reinsch, 1971)

See also svds

svds *(Professional)* Vector and Matrix

Syntax svds(**A**)

Description Returns a vector containing the singular values of **A**.

Arguments

A $m \times n$ real matrix, where $m \geq n$

Algorithm Householder reduction with QR transformation (Wilkinson and Reinsch, 1971)

See also svd

tan Trigonometric

Syntax tan(z) for z in radians;
tan(z·deg), for z in degrees

Description Returns the tangent of z.

Arguments

z real or complex number

tanh

Hyperbolic

Syntax tanh(z)

Description Returns the hyperbolic tangent of z.

Arguments

z real or complex number

Tcheb

(Professional)

Special

Syntax Tcheb(n, x)

Description Returns the value of the Chebyshev polynomial of degree n of the first kind.

Arguments

n integer, $n \ge 0$

x real number

Comments Solution of the differential equation $(1 - x^2) \cdot \frac{d^2}{dx^2}y - x \cdot \frac{d}{dx}y + n^2 \cdot y = 0$.

Algorithm Recurrence relation (Abramowitz and Stegun, 1972)

See also Ucheb

tr

Vector and Matrix

Syntax tr($\mathbf{M}$)

Description Returns the trace of $\mathbf{M}$, the sum of diagonal elements.

Arguments

$\mathbf{M}$ real or complex square matrix

trunc

Truncation and Round-off

Syntax trunc(x)

Description Returns the integer part of x. Same as floor(x) for $x > 0$ and ceil(x) for $x < 0$.

Arguments

x real number

See also ceil, floor, round

Ucheb *(Professional)* Special

Syntax Ucheb(n, x)

Description Returns the value of the Chebyshev polynomial of degree n of the second kind.

Arguments

n integer, $n \geq 0$

x real number

Comments Solution of the differential equation $(1-x^2) \cdot \frac{d^2}{dx^2}y - 3 \cdot x \cdot \frac{d}{dx}y + n \cdot (n+2) \cdot y = 0$.

Algorithm Recurrence relation (Abramowitz and Stegun, 1972)

See also Tcheb

var Statistics

Syntax var(**A**)

Description Returns the variance of the elements of **A**: $\mathrm{var}(\mathbf{A}) = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left|A_{i,j} - \mathrm{mean}(\mathbf{A})\right|^2$.

This expression is normalized by the sample size mn.

Arguments

A real or complex $m \times n$ matrix or array

See also stdev, Stdev, Var

Var Statistics

Syntax Var(**A**)

Description Returns the variance of the elements of **A**: $\mathrm{var}(\mathbf{A}) = \frac{1}{mn-1}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left|A_{i,j} - \mathrm{mean}(\mathbf{A})\right|^2$.

This expression is normalized by the sample size less one, $mn - 1$.

Arguments

A real or complex $m \times n$ matrix or array

See also stdev, Stdev, var

vec2str

(Professional) String

Syntax vec2str(**v**)

Description Returns the string formed by converting a vector **v** of ASCII codes to characters. The elements of **v** must be integers between 0 and 255.

Arguments

v vector of ASCII codes

See also str2vec

wave

(Professional) Wavelet Transform

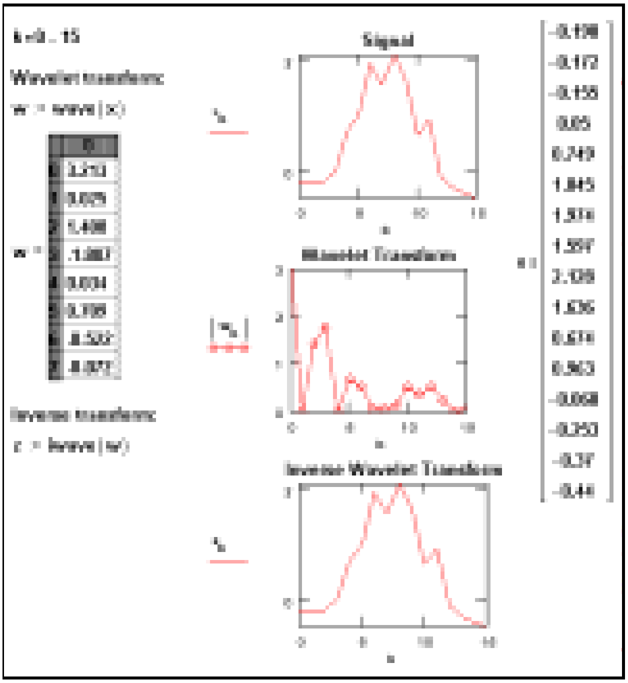
Syntax wave(**v**)

Description Returns the discrete wavelet transform of real data using Daubechies four-coefficient wavelet filter.

Arguments

v real vector of 2^n elements, where $n > 0$ is an integer

Example



Comments — When you define a vector **v** for use with Fourier or wavelet transforms, be sure to start with v_0 (or change the value of ORIGIN). If you do not define v_0, Mathcad automatically sets it to zero. This can distort the results of the transform functions.

Algorithm — Pyramidal Daubechies 4-coefficient wavelet filter (Press *et al.*, 1992)

See also — iwave

WRITEBMP — File Access

Syntax — WRITEBMP(*file*)

Description — Creates a grayscale BMP image file *file* out of a matrix.
Used as follows: WRITEBMP(*file*) := **M** . The function must appear alone on the left side of a definition.

Arguments

file — string variable corresponding to BMP filename or path

M — integer matrix, each element satisfying $0 \le \mathbf{M}_{i,j} \le 255$

WRITE_HLS *(Professional)* — File Access

Syntax — WRITE_HLS(*file*)

Description — Creates a color BMP image file *file* out of a matrix formed by juxtaposing the three matrices giving the hue, lightness, and saturation components of an image.

Arguments

file — string variable corresponding to BMP filename or path

M — integer matrix, each element satisfying $0 \le \mathbf{M}_{i,j} \le 255$

See also — See WRITERGB for an overview of creating color data files.

WRITE_HSV *(Professional)* — File Access

Syntax — WRITE_HSV(*file*)

Description — Creates a color BMP image file *file* out of a matrix formed by juxtaposing the three matrices giving the hue, saturation, and value components of an image.

Arguments

file — string variable corresponding to BMP filename or path

M — integer matrix, each element satisfying $0 \le \mathbf{M}_{i,j} \le 255$

See also — See WRITERGB for overview.

WRITEPRN

Syntax WRITEPRN(*file*) := **A**

Description Writes a matrix **A** into a structured ASCII data file *file*. Each row becomes a line in the file. The function must appear alone on the left side of a definition.

Arguments

file string variable corresponding to structured ASCII data filename or path

A matrix or vector

Comments The WRITEPRN and APPENDPRN functions write out data values neatly lined up in rows and columns. When you use these functions:

- Equations using WRITEPRN or APPENDPRN must be in a specified form. On the left should be WRITEPRN(*file*) or APPENDPRN(*file*). This is followed by a definition symbol (:=) and a matrix expression. Do not use range variables or subscripts on the matrix expression.
- Each new equation involving WRITEPRN writes a new file; if two equations write to the same file, the data written by the second equation will overwrite the data written by the first. Use APPENDPRN if you want to append values to a file rather than overwrite the file.
- The built-in variables *PRNCOLWIDTH* and *PRNPRECISION* determine the format of the data file that Mathcad creates. The value of *PRNCOLWIDTH* specifies the width of the columns (in characters). The value of *PRNPRECISION* specifies the number of significant digits used. By default, *PRNCOLWIDTH*=8 and *PRNPRECISION*=4. To change these values, choose **Options** from the **Math** menu and edit the numbers on the Built-In Variables tab, or enter definitions for these variables in your Mathcad document above the WRITEPRN function.

WRITEPRN and READPRN allow you to write out and read in *nested arrays* created in Mathcad Professional.

If the array you are writing is either a nested array (an array whose elements are themselves arrays) or a complex array (an array whose elements are complex), then WRITEPRN will *not* create a simple ASCII file. Instead, WRITEPRN creates a file using a special format unlikely to be readable by other applications. This file can, however, be read by Mathcad's READPRN function.

By using the augment function, you can concatenate several variables and write them all using WRITEPRN to a data file.

See also APPENDPRN

WRITERGB

File Access

Syntax WRITERGB(*file*)

Description Creates a color BMP image file *file* out of a single matrix formed by juxtaposing the three matrices giving the red, green, and blue values of an image. Used as follows: WRITERGB(*file*) := **M** . The function must appear alone on the left side of a definition.

Arguments

file string variable corresponding to BMP filename or path

M integer matrix, each element satisfying $0 \le \mathbf{M}_{i,j} \le 255$

Comments The function augment is helpful for combining submatrices prior to using WRITERGB.

Mathcad Professional has functions for creating color BMP files out of matrices in which the image is stored in HLS or HSV format. These work in exactly the same way as WRITERGB.

See also WRITE_HLS and WRITE_HSV

Y0

Bessel

Syntax Y0(x)

Description Returns the value of the Bessel function $Y_0(x)$ of the second kind. Same as Yn(0, x).

Arguments

x real number, $x > 0$

Algorithm Steed's method (Press *et al*., 1992)

Y1

Bessel

Syntax Y1(x)

Description Returns the value of the Bessel function $Y_1(x)$ of the second kind. Same as Yn(1, x).

Arguments

x real number, $x > 0$

Algorithm Steed's method (Press *et al*., 1992)

Yn

Bessel

Syntax Yn(m, x)

Description Returns the value of the Bessel function $Y_m(x)$ of the second kind.

Arguments

- m integer, $0 \le m \le 100$
- x real number, $x > 0$

Comments Solution of the differential equation $x^2 \cdot \frac{d^2}{dx^2}y + x \cdot \frac{d}{dx}y + (x^2 - n^2) \cdot y = 0$.

Algorithm Steed's method (Press *et al.*, 1992)

See also Jn

ys

(Professional)

Bessel

Syntax ys(n, x)

Description Returns the value of the spherical Bessel function of the second kind, of order n, at x.

Arguments

- x real number, $x > 0$
- n integer

Comments Solution of the differential equation: $x^2 \cdot \frac{d^2}{dx^2}y + 2 \cdot x \cdot \frac{d}{dx}y + (x^2 - n \cdot (n+1))y = 0$.

Algorithm Recurrence relation (Abramowitz and Stegun, 1972)

See also js

δ

Piecewise Continuous

Syntax δ(m, n)

Description Returns the value of the Kronecker delta function. Output is 1 if $m = n$ and 0 otherwise. (To type δ, press **d**[**Ctrl**]**G**).

Arguments

- m, n integers

Algorithm Continued fraction expansion (Abramowitz and Stegun, 1972; Lorczak)

ε

Syntax $\varepsilon(i, j, k)$

Description Returns the value of a completely antisymmetric tensor of rank three. Output is 0 if any two arguments are the same, 1 if the three arguments are an even permutation of (0 1 2), and –1 if the arguments are an odd permutation of (0 1 2). (To type ε, press **e**[**Ctrl**]**g**).

Arguments

i, j, k integers between 0 and 2 inclusive (or between ORIGIN and ORIGIN+2 inclusive if ORIGIN≠0)

Γ

Classical Definition

Syntax $\Gamma(z)$

Description Returns the value of the classical Euler gamma function. (To type Γ, press **G**[**Ctrl**]**g**).

Arguments

z real or complex number; undefined for $z = 0, -1, -2, \ldots$

Description For Re(z) > 0, $\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$.

For Re(z) < 0, function values analytically continue the above formula. Because $\Gamma(z+1) = z!$, the gamma function extends the factorial function (traditionally defined only for positive integers).

Extended Definition

Syntax $\Gamma(x, y)$

Description Returns the value of the extended Euler gamma function. (To type Γ, press **G**[**Ctrl**]**g**).

Arguments

x, y real numbers, $x > 0, y \geq 0$

Description Although restricted to real arguments, the function $\Gamma(x, y) = \int_y^\infty t^{x-1} e^{-t} dt$

extends the classical gamma function in the sense that the lower limit of integration y is free to vary. In the special case when y=0, the classical formulation applies and the first argument may assume complex values.

Syntax

Φ(*x*)

Description

Returns the value of the Heaviside step function. Output is 1 if $x \geq 0$ and 0 otherwise. (To type Φ, press **F**[**Ctrl**]**g**).

Arguments

x real number

Example