

# STGraph - Funzioni di sistema

[Versione 23.2.16]

Legenda: A=array (o specificamente: S=scalare; V=vettore; M=matrice); E=espressione

- [Operatori](#)
- [Funzioni matematiche](#)
- [Funzioni statistiche](#)
- [Funzioni di controllo](#)
- [Funzioni per array](#)

Vedi anche:

- [Funzioni / operatori monadici polimorfi](#)
- [Funzioni / operatori diadici polimorfi](#)
- [Operatori booleani](#)
- [Funzioni polimorfe per la gestione di distribuzioni statistiche](#)
- [Funzioni di interpolazione](#)

## Operatori

[x+y](#) [x,y:A] x piu' y

[x&& y](#) [x,y:A] congiunzione logica: x e y

[x#y](#) [x,y:A] x concatenato con y

[x###y](#) [x:S, y:A; x:A, y:S] array ottenuto rimuovendo i primi x elementi da ogni vettore nell'ultima dimensione dell'array y o gli ultimi y elementi da ogni vettore nell'ultima dimensione dell'array x

[{x}](#) [x:E] sottoespressione assegnata alla variabile \$wn, dove n varia da 0 a 3 secondo l'ordine di valutazione

[x-y](#) [x,y:A] x meno y

[x==y](#) [x,y:A] confronto logico: x e' uguale a y?

[x>=y](#) [x,y:A] confronto logico: x e' maggiore o uguale di y?

[x>y](#) [x,y:A] confronto logico: x e' maggiore di y?

[x<=y](#) [x,y:A] confronto logico: x e' minore o uguale di y?

[\[x:y\]](#) [x,y:S] vettore dei valori da x a y

[\[x:y:z\]](#) [x,y,z:S] vettore dei valori da x a z separati per z

[x<y](#) [x,y:A] confronto logico: x e' minore di y?

[-x](#) [x:A] meno x

[x%y](#) [x,y:A] x modulo y

[x!=y](#) [x,y:A] confronto logico: x e' diverso da y?

[!x](#) [x:A] negazione logica: non x

[x||y](#) [x,y:A] disgiunzione logica: x o y

[f|x](#) [f:funzione; x:A] se x e' un vettore, vettore il cui primo elemento e' ottenuto applicando la funzione diadica f ai primi due elementi di x, il secondo elemento e' ottenuto applicando f al secondo e al terzo elemento, e cosi' via; se x e' un array di ordine superiore, array ottenuto nello stesso modo, applicando f in parallelo agli elementi di ogni vettore dell'ultima dimensione

[f|\[n\]x](#) [f:funzione; x:A; n:intero] come f|x, dove f e' applicato alla dimensione n di x

[x^y](#) [x,y:A] x elevato alla potenza y

[x\\*y](#) [x,y:A] x per y

[x/y](#) [x,y:A] x diviso y

[f/x](#) [f:funzione; x:A] se x e' un vettore, scalare ottenuto applicando la funzione diadica f ai primi due elementi di x, quindi al risultato e al terzo elemento, e cosi' via; se x e' un array di ordine n, array di ordine n-1 ottenuto nello stesso modo, applicando f in parallelo agli elementi di ogni vettore dell'ultima dimensione

[f/\[n\]x](#) [f:funzione; x:A; n:intero] come f/x, dove f e' applicato alla dimensione n di x

[f\ x](#) [f:funzione; x:A] se x e' un vettore, vettore il cui primo elemento e' ottenuto applicando la funzione diadica f ai primi due elementi di x, il secondo elemento e' ottenuto applicando f al risultato e al terzo elemento, e cosi' via; se x e' un array di ordine superiore, array ottenuto nello stesso modo, applicando f in parallelo agli elementi di ogni vettore dell'ultima dimensione

[f\[n\]x](#) [f:funzione; x:A; n:intero] come f\ x, dove f e' applicato alla dimensione n di x

[@x](#) [x:A] dimensione di x

## Funzioni matematiche

[acos\(x\)](#) [x:A] arcocoseno di x

[asin\(x\)](#) [x:A] arcoseno di x

[atan\(x\)](#) [x:A] arcotangente di  $x$

[bline\(vx,vy,x\)](#) [vx,vy:V, x:A] il valore  $y$  corrispondente a  $x$  sul segmento da  $(vx[0],vy[0])$  a  $(vx[1],vy[1])$ , e costante altrove

[cos\(x\)](#) [x:A] coseno di  $x$

[deg2rad\(x\)](#) [x:A] valore di  $x$  convertito da gradi a radianti

[exp\(x\)](#) [x:A] esponenziale di  $x$  ( $e$  elevato alla potenza  $x$ )

[FFT\(x,s\)](#) [x:V,M; s:S] se  $s==1$ , trasformata veloce di Fourier del vettore  $x$ ; se  $s==2$ , antitrasformata veloce di Fourier della matrice  $x$

[int\(x\)](#) [x:A] parte intera di  $x$

[integral\(x\)](#) [x:A] nelle transizioni di stato dei nodi di stato, somma iterativa di  $x$ , sulla base dell'algoritmo di integrazione scelto

[line\(vx,vy,x\)](#) [vx,vy:V, x:A] valore  $y$  corrispondente a  $x$  sulla linea retta per i punti  $vx[0],vy[0]$  e  $vx[1],vy[1]$

[log\(x\)](#) [x:A] logaritmo naturale di  $x$

[log\(x,y\)](#) [x,y:A] logaritmo di  $x$  in base  $y$

[max\(x,y\)](#) [x,y:A] massimo tra  $x$  e  $y$

[min\(x,y\)](#) [x,y:A] minimo tra  $x$  e  $y$

[mod\(x,y\)](#) [x,y:A]  $x$  modulo  $y$

[pline\(vx,vy,x\)](#) [vx,vy:V, x:A] valore  $y$  corrispondente a  $x$  sulla polinomiale i cui vertici sono nei vettori  $vx$  e  $vy$

[rad2deg\(x\)](#) [x:A] valore di  $x$  convertito da radianti a gradi

[round\(x,y\)](#) [x,y:A]  $x$  arrotondato a  $y$  decimali

[sigmoid\(vx,vy,x\)](#) [vx,vy:V, x:A] valore  $y$  corrispondente a  $x$  sulla sigmoide controllata dai punti  $vx[0],vy[0]$  e  $vx[1],vy[1]$

[sign\(x\)](#) [x:A] segno di  $x$ , cioè  $1$  se  $x > 0$ ,  $-1$  se  $x < 0$ ,  $0$  se  $x == 0$

[sin\(x\)](#) [x:A] seno di  $x$

[spline\(vx,vy,x\)](#) [vx,vy:V, x:A] valore  $y$  corrispondente a  $x$  sulla spline i cui nodi sono nei vettori  $vx$  e  $vy$

[sqrt\(x\)](#) [x:A] radice quadrata di  $x$

[tan\(x\)](#) [x:A] tangente di  $x$

[wrap\(x,y\)](#) [x,y:A]  $x$  modulo  $y$  con gestione dei valori negativi

## Funzioni statistiche

[beta\(v,x,s\)](#) [v:V; x:A; s:S] distribuzione di probabilita' beta, di parametri  $v=[p1,p2]$ , con  $p1=alpha$  (default: 1),  $p2=beta$  (default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s==0$ , valore della pdf in  $x$ ; se  $s==1$ , valore della cdf in  $x$

[binomial\(v,x,s\)](#) [v:V; x:A; s:S] distribuzione di probabilita' binomiale, di parametri  $v=[p1,p2]$ , con  $p1=mean$  (default: 0),  $p2=stddev$  (default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s==0$ , valore della pdf in  $x$ ; se  $s==1$ , valore della cdf in  $x$

[chiSquare\(v,x,s\)](#) [v:V; x:A; s:S] distribuzione di probabilita' chi quadro, con parametri  $v=[p1]$ , con  $p1=gradi$  di liberta' (default: 5): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s==0$ , valore della pdf in  $x$ ; se  $s==1$ , valore della cdf in  $x$ ; se  $s==2$ , valore della cdf inversa per la probabilita'  $x$

[exponential\(v,x,s\)](#) [v:V; x:A; s:S] distribuzione di probabilita' esponenziale, con parametri  $v=[p1]$ , dove  $p1=1/lambda$  ( $p1>0$ ; default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s==0$ , valore della pdf in  $x$ ; se  $s==1$ , valore della cdf in  $x$

[gamma\(v,x,s\)](#) [v:V; x:A; s:S] distribuzione di probabilita' gamma, di parametri  $v=[p1,p2]$ , con  $p1=alpha$  (default: 1),  $p2=beta$  (default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione;

se  $s=0$ , valore della pdf in  $x$ ; se  $s=1$ , valore della cdf in  $x$ ; se  $s=2$ , valore della cdf inversa per la probabilita'  $x$

[gaussian\(v,x,s\)](#) [ $v:V$ ;  $x:A$ ;  $s:S$ ] distribuzione di probabilita' gaussiana, di parametri  $v=[p1,p2]$ , con  $p1=mean$  (default: 0),  $p2=stddev$  (default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s=0$ , valore della pdf in  $x$ ; se  $s=1$ , valore della cdf in  $x$ ; se  $s=2$ , valore della cdf inversa per la probabilita'  $x$

[poisson\(v,x,s\)](#) [ $v:V$ ;  $x:A$ ;  $s:S$ ] distribuzione di probabilita' di Poisson, di parametri  $v=[p1]$ , con  $p1=mean$  ( $p1>0$ ; default: 5): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s=0$ , valore della pdf in  $x$ ; se  $s=1$ , valore della cdf in  $x$

[rand\(x,y\)](#) [ $x,y:S$ ] valore casuale estratto da una distribuzione uniforme tra 0 e 1, o tra 0 e  $x$  se solo  $x$  e' specificato, o tra  $x$  e  $y$  se entrambi sono specificati

[randInt\(x\)](#) [ $x:A$ ] valore casuale intero estratto da una distribuzione uniforme tra 0 e  $int(x-1)$

[tDistribution\(v,x,s\)](#) [ $v:V$ ;  $x:A$ ;  $s:S$ ] distribuzione di probabilita' t, di parametri  $v=[p1]$ , con  $p1=gradi$  di liberta' ( $p1>0$ ; default: 5): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s=0$ , valore della pdf in  $x$ ; se  $s=1$ , valore della cdf in  $x$

[uniform\(v,x,s\)](#) [ $v:V$ ;  $x:A$ ;  $s:S$ ] distribuzione di probabilita' uniforme (rettangolare), di parametri  $v=[p1,p2]$ , con  $p1=mean$  (default: 0),  $p2=stddev$  (default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s=0$ , valore della pdf in  $x$ ; se  $s=1$ , valore della cdf in  $x$ ; se  $s=2$ , valore della cdf inversa per la probabilita'  $x$

## Funzioni di controllo

[function\(x\)](#) [ $x:E$ ] nuova funzione, chiamata come il nodo in cui e' definita (il nome del nodo deve cominciare con un underscore) e definita dall'espressione  $x$

[getCProp\(n,x\)](#) [ $n$ :nome di nodo;  $x:stringa$ ] valore della proprieta' custom di nome  $x$ , o del nodo  $n$  o del nodo corrente se  $n$  non e' specificato

[getPhases\(\)](#) [ ] vettore di fasi specificato nella proprieta' custom "Phase"

[if\(c1,x1,c2,x2,...,cn,xn,xn+1\)](#) [ $c1,...,cn:A$ ;  $v1,...,vn+1:A$ ] struttura condizionale: restituisce  $x1$  se  $c1$  e' vero, altrimenti  $x2$  se  $c2$  e' vero, altrimenti ..., e  $xn+1$  altrimenti

[indexOrigin\(\)](#) [ ] origine degli indici degli array, 0 o 1, assegnata come propriet del modello

[isNumber\(x\)](#) [ $x:A$ ] vero se  $x$  e' un numero (cioe' non e' ne' NaN ne' Infinity), falso altrimenti

[iter\(x,e,z\)](#) [ $x:A$ ;  $e:E$ ;  $z:A$ ] valore ottenuto applicando ripetutamente l'espressione  $e$  (che puo' includere funzioni della forma  $fun(\$0,\$1)$  oppure  $\$0op\$1$ ), il cui elemento identita' e'  $z$ , agli elementi del vettore  $x$  o agli elementi di ogni vettore dell'ultima dimensione dell'array di ordine superiore  $x$

[readFromNet\(\)](#) [ ] valore letto da un server di rete Spacebrew

[readFromXLS\(x,y,r,c\)](#) [ $x:stringa$ ;  $y,r,c:S$ ] valore letto dalla riga  $r$  e colonna  $c$  del foglio di indice  $y$  del file xls il cui nome e'  $x$

[readFromXLS\(x,y,r1,c1,r2,c2\)](#) [ $x:stringa$ ;  $y,r1,c1,r2,c2:S$ ] vettore o matrice letti tra la cella di riga  $r1$  e colonna  $c1$  e la cella di riga  $r2$  e colonna  $c2$  del foglio di indice  $y$  del file xls il cui nome e'  $x$

[sysTime\(\)](#) [ ] numero di millisecondi dall'inizio della simulazione

[writeToNet\(x\)](#) [ $x:A$ ] invia  $x$  a un server di rete Spacebrew

## Funzioni per array

[array\(v,e\)](#) [ $v:V$ ;  $e:E$ ] array la cui dimensione e'  $v$  e i cui elementi sono specificati dall'espressione  $e$ , che puo' contenere le variabili di sistema  $\$i0,...,\$i5$ , ognuna di esse assegnata al valore della dimensione

corrispondente

[conc\(x,y,n\)](#) [x,y:A, n:intero]  $x$  concatenato con  $y$  lungo la dimensione  $n$  se specificata o l'ultima dimensione altrimenti

[dec\(x,y,n\)](#) [x:S, y:A; x:A, y:S, n:intero] array ottenuto rimuovendo i primi  $x$  elementi da ogni vettore lungo la dimensione  $n$  se specificata, o l'ultima dimensione altrimenti, dell'array  $y$  o gli ultimi  $y$  elementi da ogni vettore lungo la dimensione  $n$  se specificata, o l'ultima dimensione altrimenti, dell'array  $x$

[frequency\(x,d,c\)](#) [x:S; d,c:V] vettore che contiene la distribuzione dei dati in  $d$  categorizzati dai valori in  $c$ ; se  $x$  e' specificato, si suppone che  $d$  gia' contenga la distribuzione, a cui  $x$  e' aggiunto

[get\(x,v0,v1,...\)](#) [x:A; v0,v1,...:V] array ottenuto dall'array  $x$  estraendo il sottoarray di indici  $v0$  dalla prima dimensione, quindi il sottoarray di indici  $v1$  dalla seconda dimensione, e cosi' via (equivalente a  $x[v0, v1, \dots]$ )

[getData\(x,y\)](#) [x,y:A] vettore degli elementi dell'array  $y$  identificati dagli indici nel vettore  $x$  se  $y$  e' un vettore o dagli indici nella matrice  $x$  se  $y$  e' un array di ordine superiore

[getIndex\(s,x\)](#) [s:S, x:A] se  $x$  e' un vettore, vettore degli indici delle occorrenze di  $s$  in  $x$ ; se  $x$  e' un array di ordine superiore, matrice degli indici delle occorrenze di  $s$  in  $x$

[order\(x\)](#) [x:A] ordine, cioe' numero delle dimensioni, dell'array  $x$

[remove\(x,v\)](#) [x:A; v:S,V] array ottenuto eliminando l'elemento / gli elementi  $v$ -esimo/i dalla prima dimensione dell'array  $x$

[resize\(x,v\)](#) [x:A; v:V] array ottenuto ridimensionando l'array  $x$  alla dimensione  $v$ , ed eliminando gli elementi eccedenti in coda o aggiungendo zeri in coda, se richiesto

[set\(x,v0,v1,...,y\)](#) [x:A; v0,v1,...:V; y:A] array ottenuto dall'array  $x$  sostituendo il suo sottoarray di indici  $v0$  nella prima dimensione, di indici  $v1$  nella seconda dimensione, ... con l'array  $y$

[shift\(x,y,s\)](#) [x,y:A; s:S] array ottenuto concatenando l'array  $y$  con l'array  $x$  a sinistra e rimuovendo lo stesso numero di elementi da  $x$  a destra se  $s==0$  o se non specificato, o in ordine inverso se  $s==1$

[shuffle\(x\)](#) [x:A] array ottenuto permutando in modo casuale gli elementi dell'array  $x$

[size\(x\)](#) [x:A] dimensione di  $x$

[sort\(x,s\)](#) [x:A; s:S] array ottenuto ordinando gli elementi dell'array  $x$ , in ordine diretto se  $s==0$  o se non specificato, o in ordine inverso se  $s==1$

[transpose\(x\)](#) [x:A] array ottenuto per trasposizione dell'array  $x$

## STGraph - Funzioni elencate per categorie funzionali

### STGraph - Funzioni / operatori monadici polimorfi

Ognuna delle funzioni / operatori seguenti ha un argomento, che è un array, e quindi in particolare scalare (array di ordine 0), vettore (array di ordine 1), o matrice (array di ordine 2), e si comporta in modo polimorfo.

Dato  $f(x)$ :

- se  $x$  is an empty  $n>0$ -order array, il risultato è  $x$  stesso
- se  $x$  è uno scalare, il risultato è uno scalare
- se  $x$  è un array di ordine  $n>0$ , il risultato è un array  $y$  di ordine  $n>0$  tale che  
 $y[i1, \dots, in] = [f(x[i1]), \dots, f(x[in])]$

[acos\(x\)](#)

[asin\(x\)](#)

[atan\(x\)](#)

[cos\(x\)](#)

[deg2rad\(x\)](#)

[exp\(x\)](#)

[int\(x\)](#)

[isNumber\(x\)](#)

[log\(x\)](#)

[-x](#)

[!x](#)

[rad2deg\(x\)](#)

[randInt\(x\)](#)

[sign\(x\)](#)

[sin\(x\)](#)

[sqrt\(x\)](#)

[tan\(x\)](#)

### STGraph - Funzioni / operatori diadici polimorfi

Ognuna delle funzioni / operatori seguenti ha due argomenti, che sono array, e quindi in particolare scalari (array di ordine 0), vettori (array di ordine 1), o matrici (array di ordine 2), e si comporta in modo polimorfo.

Dato  $f(x1, x2)$ :

- se  $x1$  o  $x2$  è un array di ordine  $n>0$  vuoto, il risultato è  $0.0$
- se  $x1$  e  $x2$  sono scalari, il risultato è uno scalare
- se  $x1$  è uno scalare e  $x2$  è un array di ordine  $n>0$ , il risultato è un array  $y$  di ordine  $n$  tale che  
 $y[i1, \dots, in] = f(x1, x2[i1, \dots, in])$  (o viceversa)
- se  $x1$  e  $x2$  sono array di ordine  $n>0$  della stessa dimensione, il risultato è un array  $y$  di ordine  $n$  tale che  
 $y[i1, \dots, in] = f(x1[i1, \dots, in], x2[i1, \dots, in])$
- se  $x1$  è un array di ordine  $n>1$  e  $x2$  è un array di ordine  $n-1$ , e le loro prime  $n-1$  dimensioni sono le stesse, il risultato è un array  $y$  di ordine  $n$  tale che  $y[i1, \dots, in] = f(x1[i1, \dots, in], x2[i1, \dots, in-1])$

Negli altri casi si genera un'eccezione.

[log\(x,y\)](#)  
[max\(x,y\)](#)  
[min\(x,y\)](#)  
[mod\(x,y\)](#)  
[x+y](#)  
[x&& y](#)  
[x-y](#)  
[x==y](#)  
[x>=y](#)  
[x>y](#)  
[x<=y](#)  
[x<y](#)  
[x%y](#)  
[x!=y](#)  
[x||y](#)  
[x^y](#)  
[x\\*y](#)  
[x/y](#)  
[round\(x,y\)](#)  
[wrap\(x,y\)](#)

## STGraph - Operatori booleani

Ognuno dei seguenti operatori tratta con i valori booleani vero e falso, che sono definiti in STGraph come segue:

- ogni valore maggiore di zero (effettivamente: maggiore o uguale di `EPSILON`) è valutato come vero;
- ogni valore minore o uguale a zero (effettivamente: minore di `EPSILON`) è valutato come falso.

L'unico tipo primitivo in STGraph è `double`: quindi, vero e falso sono implementati come `1.0` e `0.0` rispettivamente.

[x&& y](#)  
[x==y](#)  
[x>=y](#)  
[x>y](#)  
[x<=y](#)  
[x<y](#)  
[x!=y](#)  
[!x](#)  
[x||y](#)

## STGraph - Funzioni polimorfe per la gestione di distribuzioni statistiche

Ognuna delle funzioni seguenti può essere eseguita per generare un numero casuale estratto dalla distribuzione di probabilità data, o per restituire i valori `y` della funzione di densità di probabilità (PDF), o della funzione di densità di probabilità cumulata (CDF), o funzione di densità di probabilità cumulata inversa

per i valori  $x$  dati.

Assumendo che i parametri che caratterizzano la distribuzione  $f$  siano nel vettore  $v$  (p.es., nel caso della distribuzione gaussiana  $v=[mean, stddev]$ ), gli usi seguenti sono ammessi:

- $f()$  restituisce un valore casuale estratto dalla distribuzione con parametri di default
- $f(v)$  restituisce un valore casuale estratto dalla distribuzione con i parametri nel vettore  $v$
- $f(v, x, 0)$  restituisce i valori della PDF con parametri nel vettore  $v$  e per gli argomenti nell'array  $x$
- $f(v, x, 1)$  restituisce i valori della CDF con parametri nel vettore  $v$  e per gli argomenti nell'array  $x$
- $f(v, x, 2)$  restituisce i valori della CDF inversa con parametri nel vettore  $v$  e per gli argomenti nell'array  $x$ , nell'intervallo  $[0, 1]$  (o  $(0, 1)$ ).

Le relazioni algoritmiche tra PDF e CDF sono le seguenti:

- la CDF è approssimata da  $(+\backslash pdf) * deltax$ , dove  $pdf$  è il vettore dei valori della PDF e  $deltax$  è il passo dei argomenti nel dominio;
- la PDF è approssimata da  $(-|-cdf) / deltax$ , where  $cdf$  è il vettore dei valori della CDF e  $deltax$  è come sopra.

[beta\(v,x,s\)](#)

[binomial\(v,x,s\)](#)

[chiSquare\(v,x,s\)](#)

[exponential\(v,x,s\)](#)

[gamma\(v,x,s\)](#)

[gaussian\(v,x,s\)](#)

[poisson\(v,x,s\)](#)

[tDistribution\(v,x,s\)](#)

[uniform\(v,x,s\)](#)

## STGraph - Funzioni di interpolazione

Ognuna delle funzioni seguenti genera un valore di interpolazione secondo la stessa logica:

- due vettori  $v_x$  e  $v_y$  sono specificati, come punti di controllo che definiscono la curva di interpolazione
- un array di valori  $x$  è specificato, i cui corrispondenti valori  $y$  sulla curva sono calcolati dalla funzione.

[bline\(vx,vy,x\)](#)

[line\(vx,vy,x\)](#)

[pline\(vx,vy,x\)](#)

[sigmoid\(vx,vy,x\)](#)

[spline\(vx,vy,x\)](#)



## STGraph - Funzioni

### acos

**Formato:** `acos(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** arcocoseno di  $x$

$x$  e' espresso in radianti.

`acos` e' [una funzione matematica](#)

`acos` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

### array

**Formato:** `array(v,e)`

**Vincoli:**  $v$  e' uno scalare o un vettore;  $e$  e' un'espressione

**Descrizione:** `array` la cui dimensione e'  $v$  e i cui elementi sono specificati dall'espressione  $e$ , che puo' contenere le variabili di sistema  $\$i0, \dots, \$i5$ , ognuna di esse assegnata al valore della dimensione corrispondente

`array` e' [una funzione per array](#)

**Eccezioni:** no

### asin

**Formato:** `asin(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** arcoseno di  $x$

$x$  e' espresso in radianti.

`asin` e' [una funzione matematica](#)

`asin` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

### atan

**Formato:** `atan(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** arcotangente di  $x$

$x$  e' espresso in radianti.

`atan` e' [una funzione matematica](#)

`atan` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## beta

**Formato:** `beta(v,x,s)`

**Vincoli:**  $v$  (opzionale) e' un vettore;  $x$  (opzionale) e' un array;  $s$  (opzionale) e' uno scalare, o 0 o 1 o 2

**Descrizione:** distribuzione di probabilita' beta, di parametri  $v=[p1,p2]$ , con  $p1=\alpha$  (default: 1),  $p2=\beta$  (default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s==0$ , valore della pdf in  $x$ ; se  $s==1$ , valore della cdf in  $x$

`beta` e' [una funzione statistica](#)

`beta` e' [una funzione di distribuzione statistica](#)

**Eccezioni:**  $s$  deve essere 0 o 1 o 2.

## binomial

**Formato:** `binomial(v,x,s)`

**Vincoli:**  $v$  (opzionale) e' un vettore;  $x$  (opzionale) e' un array;  $s$  (opzionale) e' uno scalare, o 0 o 1 o 2

**Descrizione:** distribuzione di probabilita' binomiale, di parametri  $v=[p1,p2]$ , con  $p1=\text{mean}$  (default: 0),  $p2=\text{stddev}$  (default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s==0$ , valore della pdf in  $x$ ; se  $s==1$ , valore della cdf in  $x$

`binomial` e' [una funzione statistica](#)

`binomial` e' [una funzione di distribuzione statistica](#)

**Eccezioni:**  $s$  deve essere 0 o 1 o 2.

## bline

**Formato:** `bline(vx,vy,x)`

**Vincoli:**  $vx$  e  $vy$  sono vettori;  $x$  e' un array

**Descrizione:** il valore  $y$  corrispondente a  $x$  sul segmento da  $(vx[0],vy[0])$  a  $(vx[1],vy[1])$ , e costante altrove

`blinc` e' [una funzione matematica](#)

`blinc` e' [una funzione di interpolazione](#)

**Eccezioni:** Sia  $vx$  sia  $vy$  devono avere due elementi;  $vx[0]$  deve essere minore di  $vx[1]$ .

## chiSquare

**Formato:** `chiSquare(v,x,s)`

**Vincoli:**  $v$  (opzionale) e' un vettore;  $x$  (opzionale) e' un array;  $s$  (opzionale) e' uno scalare, o 0 o 1 o 2

**Descrizione:** distribuzione di probabilita' chi quadro, con parametri  $v=[p1]$ , con  $p1$ =gradi di liberta' (default: 5): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s==0$ , valore della pdf in  $x$ ; se  $s==1$ , valore della cdf in  $x$ ; se  $s==2$ , valore della cdf inversa per la probabilita'  $x$

`chiSquare` e' [una funzione statistica](#)

`chiSquare` e' [una funzione di distribuzione statistica](#)

**Eccezioni:**  $s$  deve essere 0 o 1 o 2; se  $s==2$ , gli elementi di  $x$  devono essere nell'intervallo  $[0,1]$ .

## conc

**Formato:** `conc(x,y,n)`

**Vincoli:**  $x$  e  $y$  sono array,  $n$  (opzionale) e' un intero

**Descrizione:**  $x$  concatenato con  $y$  lungo la dimensione  $n$  se specificata o l'ultima dimensione altrimenti. Se  $n$  e' omesso e' equivalente all'operatore  $x\#y$ .

`conc` e' [una funzione per array](#)

**Eccezioni:** Deve essere possibile concatenare  $x$  con  $y$  in termini delle loro dimensioni;  $n$  deve essere una dimensione corretta per  $x$ , attualmente 0 o 1.

## cos

**Formato:** `cos(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** coseno di  $x$

$x$  e' espresso in radianti.

`cos` e' [una funzione matematica](#)

`cos` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## **dec**

**Formato:** `dec(x,y,n)`

**Vincoli:** `x` e' uno scalare e `y` un array, o viceversa, `n` (opzionale) e' un intero

**Descrizione:** array ottenuto rimuovendo i primi `x` elementi da ogni vettore lungo la dimensione `n` se specificata, o l'ultima dimensione altrimenti, dell'array `y` o gli ultimi `y` elementi da ogni vettore lungo la dimensione `n` se specificata, o l'ultima dimensione altrimenti, dell'array `x`

Se `n` e' omesso e' equivalente all'operatore `x##y`.

`dec` e' [una funzione per array](#)

**Eccezioni:** `n` deve essere una dimensione corretta per `x`, attualmente 0 o 1.

## **deg2rad**

**Formato:** `deg2rad(x)`

**Vincoli:** `x` e' un array

**Descrizione:** valore di `x` convertito da gradi a radianti

`deg2rad` e' [una funzione matematica](#)

`deg2rad` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## **exp**

**Formato:** `exp(x)`

**Vincoli:** `x` e' un array

**Descrizione:** esponenziale di `x` (`e` elevato alla potenza `x`)

`exp` e' [una funzione matematica](#)

`exp` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## exponential

**Formato:** `exponential(v,x,s)`

**Vincoli:** `v` (opzionale) e' un vettore; `x` (opzionale) e' un array; `s` (opzionale) e' uno scalare, o 0 o 1

**Descrizione:** distribuzione di probabilita' esponenziale, con parametri `v=[p1]`, dove `p1=lambda` (`p1>0`; default: 1): se `x` e `s` non sono specificati, valore casuale estratto dalla distribuzione; se `s==0`, valore della pdf in `x`; se `s==1`, valore della cdf in `x`

`exponential` e' [una funzione statistica](#)

`exponential` e' [una funzione di distribuzione statistica](#)

**Eccezioni:** `s` deve essere 0 o 1; gli elementi di `x` devono essere strettamente positivi.

## FFT

**Formato:** `FFT(x,s)`

**Vincoli:** `s` e' uno scalare, 0 o 1 o 2; `x` e' un vettore se `x==1`, e una matrice se `x==2`

**Descrizione:** se `s==1`, trasformata veloce di Fourier del vettore `x`; se `s==2`, antitrasformata veloce di Fourier della matrice `x`

La FFT prende un vettore e restituisce una matrice `size(x) x 2`, da interpretare come un vettore di numeri complessi. La FFT inversa prende una matrice a 2 colonne, da interpretare come un vettore di numeri complessi, e produce un vettore.

`FFT` e' [una funzione matematica](#)

**Eccezioni:** Se si calcola la FFT, la dimensione di `x` deve essere una potenza di 2. Se si calcola la FFT inversa, il numero di righe di `x` deve essere una potenza di 2 e il numero di colonne deve essere 2.

## frequency

**Formato:** `frequency(x,d,c)`

**Vincoli:** `x` (opzionale) e' uno scalare; `d` e `c` sono vettori

**Descrizione:** vettore che contiene la distribuzione dei dati in `d` categorizzati dai valori in `c`; se `x` e' specificato, si suppone che `d` gia' contenga la distribuzione, a cui `x` e' aggiunto

Gli elementi del vettore `c` sono gli estremi destri degli intervalli che definiscono le categorie; l'ultima categoria e' aggiunta automaticamente per contenere i valori maggiori dell'ultimo elemento di `c`.

`frequency` e' [una funzione per array](#)

**Eccezioni:** `y` e `z` devono avere la stessa dimensione.

## function

**Formato:** `function(x)`

**Vincoli:** `x` e' un'espressione

**Descrizione:** nuova funzione, chiamata come il nodo in cui e' definita (il nome del nodo deve cominciare con un underscore) e definita dall'espressione `x`

Gli argomenti della funzione definita sono referenziati nell'espressione come `$a0`, `$a1`, ... (fino a `$a3`).

Per esempio, se l'espressione del nodo `_x` e' `function(($a0+$a1)/2)` allora `_x(2,3)` produce il risultato  $(2+3/2)=2.5$ .

Inoltre, l'espressione puo' contenere la variabile di sistema `$numArgs`, assegnata al numero degli argomenti della funzione.

Si noti che la (meta)funzione `function` consente definizioni ricorsive. Per esempio, la funzione prodotto fattoriale puo' essere definita in un nodo di nome `_fact` come: `function(if($a0==0,1,$a0*_fact($a0-1)))`.

`function` e' [una funzione di controllo](#)

**Eccezioni:** no

## gamma

**Formato:** `gamma(v,x,s)`

**Vincoli:** `v` (opzionale) e' un vettore; `x` (opzionale) e' un array; `s` (opzionale) e' uno scalare, o 0 o 1 o 2

**Descrizione:** distribuzione di probabilita' gamma, di parametri `v=[p1,p2]`, con `p1=alpha` (default: 1), `p2=beta` (default: 1): se `x` e `s` non sono specificati, valore casuale estratto dalla distribuzione; se `s==0`, valore della pdf in `x`; se `s==1`, valore della cdf in `x`; se `s==2`, valore della cdf inversa per la probabilita' `x`

`gamma` e' [una funzione statistica](#)

`gamma` e' [una funzione di distribuzione statistica](#)

**Eccezioni:** `s` deve essere 0 o 1 o 2; se `s==2`, gli elementi di `x` devono essere nell'intervallo `[0,1]`.

## gaussian

**Formato:** `gaussian(v,x,s)`

**Vincoli:**  $v$  (opzionale) e' un vettore;  $x$  (opzionale) e' un array;  $s$  (opzionale) e' uno scalare, o 0 o 1 o 2

**Descrizione:** distribuzione di probabilita' gaussiana, di parametri  $v=[p1, p2]$ , con  $p1=mean$  (default: 0),  $p2=stddev$  (default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s==0$ , valore della pdf in  $x$ ; se  $s==1$ , valore della cdf in  $x$ ; se  $s==2$ , valore della cdf inversa per la probabilita'  $x$

gaussian e' [una funzione statistica](#)

gaussian e' [una funzione di distribuzione statistica](#)

**Eccezioni:**  $s$  deve essere 0 o 1 o 2; se  $s==2$ , gli elementi di  $x$  devono essere nell'intervallo  $[0, 1]$ .

## get

**Formato:** `get(x, v0, v1, ...)`

**Vincoli:**  $x$  e' un array;  $v1, v2, \dots$  sono scalari o vettori

**Descrizione:** array ottenuto dall'array  $x$  estraendo il sottoarray di indici  $v0$  dalla prima dimensione, quindi il sottoarray di indici  $v1$  dalla seconda dimensione, e cosi' via (equivalente a `x[v0, v1, ...]`)

get e' [una funzione per array](#)

**Eccezioni:** no

## getCProp

**Formato:** `getCProp(n, x)`

**Vincoli:**  $n$  (opzionale) e' un nome di nodo;  $x$  e' una stringa

**Descrizione:** valore della proprieta' custom di nome  $x$ , o del nodo  $n$  o del nodo corrente se  $n$  non e' specificato

Sono gestiti solo valori numerici. Se specificato,  $n$  deve essere scritto senza virgolette.

getCProp e' [una funzione di controllo](#)

**Eccezioni:** Se specificato,  $n$  deve essere il nome di un nodo connesso.  $x$  deve essere il nome di una proprieta' custom esistente, il cui valore deve essere un numero.

## getData

**Formato:** `getData(x, y)`

**Vincoli:**  $x$  e  $y$  sono array

**Descrizione:** vettore degli elementi dell'array  $y$  identificati dagli indici nel vettore  $x$  se  $y$  e' un vettore o dagli indici nella matrice  $x$  se  $y$  e' un array di ordine superiore

Se non sono trovati elementi, restituisce lo scalare -1.

`getData` e' [una funzione per array](#)

**Eccezioni:** no

## **getIndex**

**Formato:** `getIndex(s,x)`

**Vincoli:** `s` e' uno scalare; `x` e' un array

**Descrizione:** se `x` e' un vettore, vettore degli indici delle occorrenze di `s` in `x`; se `x` e' un array di ordine superiore, matrice degli indici delle occorrenze di `s` in `x`  
Se `s` non e' trovato, restituisce il vettore `[-1]`.

`getIndex` e' [una funzione per array](#)

**Eccezioni:** no

## **getPhases**

**Formato:** `getPhases()`

**Vincoli:**

**Descrizione:** vettore di fasi specificato nella proprieta' custom "Phase"  
Il formato per la proprieta' custom e', per esempio, 1,5-8.

`getPhases` e' [una funzione di controllo](#)

**Eccezioni:** La proprieta' custom "Phase" deve essere definita e il suo valore deve essere assegnato come specificato.

## **if**

**Formato:** `if(c1,x1,c2,x2,...,cn,xn,xn+1)`

**Vincoli:** tutti gli argomenti sono array, o della stessa dimensione o di "dimensione compatibile" (vedi le Note)

**Descrizione:** struttura condizionale: restituisce `x1` se `c1` e' vero, altrimenti `x2` se `c2` e' vero, altrimenti ..., e `xn+1` altrimenti

La funzione `if` opera come una catena di `if ... then ... else if ... then ...`

La sua forma più semplice, `if(c,v1,v2)`, è equivalente alla struttura `if c then v1 else v2`.



Si comporta "più polimorficamente possibile": in particolare, se  $c$ ,  $v_1$  e  $v_2$  sono array della stessa dimensione, la funzione produce un array di quella dimensione, tale che `risultato[i1,...,in] == v1[i1,...,in]` se `c[i1,...,in]` è vero e `risultato[i1,...,in] == v2[i1,...,in]` altrimenti. Data la forma generale, `if(c1,v1,c2,v2,...,cn,vn,vn+1)`, le condizioni  $c_i$  devono avere la stessa dimensione. Se sono scalari, i valori  $v_j$  non sono vincolati. Altrimenti, i valori  $v_j$  devono avere la stessa dimensione delle condizioni  $c_i$  o devono essere scalari.

`if` e' [una funzione di controllo](#)

**Eccezioni:** Il numero degli argomenti deve essere dispari.

## indexOrigin

**Formato:** `indexOrigin()`

**Vincoli:**

**Descrizione:** origine degli indici degli array,, 0 o 1, assegnata come propriet del modello

`indexOrigin` e' [una funzione di controllo](#)

**Eccezioni:** no

## int

**Formato:** `int(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** parte intera di  $x$

`int` e' [una funzione matematica](#)

`int` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## integral

**Formato:** `integral(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** nelle transizioni di stato dei nodi di stato, somma iterativa di  $x$ , sulla base dell'algoritmo di integrazione scelto

Nel caso dell'algoritmo di Euler, il valore e' `this+x*timeD`.

`integral` e' [una funzione matematica](#)

**Eccezioni:** no

## **isNumber**

**Formato:** `isNumber(x)`

**Vincoli:** `x` e' un array

**Descrizione:** vero se `x` e' un numero (cioe' non e' ne' NaN ne' Infinity), falso altrimenti

`isNumber` e' [una funzione di controllo](#)

`isNumber` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## **iter**

**Formato:** `iter(x,e,z)`

**Vincoli:** `x` e' un array; `e` e' un'espressione; `z` e' un array

**Descrizione:** valore ottenuto applicando ripetutamente l'espressione `e` (che puo' includere funzioni della forma `fun($0,$1)` oppure `$0op$1`), il cui elemento identita' e' `z`, agli elementi del vettore `x` o agli elementi di ogni vettore dell'ultima dimensione dell'array di ordine superiore `x`

`iter` e' [una funzione di controllo](#)

**Eccezioni:** `x` deve essere non nullo.

## **line**

**Formato:** `line(vx,vy,x)`

**Vincoli:** `vx` e `vy` sono vettori; `x` e' un array

**Descrizione:** valore `y` corrispondente a `x` sulla linea retta per i punti `vx[0],vy[0]` e `vx[1],vy[1]`

`line` e' [una funzione matematica](#)

`line` e' [una funzione di interpolazione](#)

**Eccezioni:** Sia `vx` sia `vy` devono avere due elementi; `vx[0]` deve essere minore di `vx[1]`.

## log-1

**Formato:**  $\log(x)$

**Vincoli:**  $x$  e' un array

**Descrizione:** logaritmo naturale di  $x$

$\log-1$  e' [una funzione matematica](#)

$\log-1$  e' [una funzione monadica polimorfa](#)

**Eccezioni:**  $x$  deve essere strettamente positivo.

## log-2

**Formato:**  $\log(x, y)$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** logaritmo di  $x$  in base  $y$

$\log-2$  e' [una funzione matematica](#)

$\log-2$  e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#). Inoltre,  $x$  e  $y$  devono essere strettamente positivi.

## max

**Formato:**  $\max(x, y)$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** massimo tra  $x$  e  $y$

$\max$  e' [una funzione matematica](#)

$\max$  e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## min

**Formato:**  $\min(x, y)$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** minimo tra  $x$  e  $y$

`min` e' [una funzione matematica](#)

`min` e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## **mod**

**Formato:** `mod(x,y)`

**Vincoli:** `x` e `y` sono array (i vincoli sono specificati [qui](#))

**Descrizione:** `x` modulo `y`

Equivalente all'operatore `x%y`.

`mod` e' [una funzione matematica](#)

`mod` e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## **Addition**

**Formato:** `x+y`

**Vincoli:** `x` e `y` sono array (i vincoli sono specificati [qui](#))

**Descrizione:** `x` piu' `y`

`Addition` e' [un operatore](#)

`Addition` e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## **And**

**Formato:** `x&& y`

**Vincoli:** `x` e `y` sono array (i vincoli sono specificati [qui](#))

**Descrizione:** congiunzione logica: `x` e `y`

`And` e' [un operatore](#)

`And` e' [una funzione diadica polimorfa](#), [un operatore booleano](#)

**Eccezioni:** Come specificato [qui](#).

## Conc

**Formato:**  $x \# y$

**Vincoli:**  $x$  e  $y$  sono array

**Descrizione:**  $x$  concatenato con  $y$

Equivalente alla funzione  $x \# y$ . Deve essere possibile concatenare  $x$  con  $y$  in termini delle loro dimensioni.

Conc e' [un operatore](#)

**Eccezioni:** no

## Dec

**Formato:**  $x \## y$

**Vincoli:**  $x$  e' uno scalare e  $y$  un array, o viceversa

**Descrizione:** array ottenuto rimuovendo i primi  $x$  elementi da ogni vettore nell'ultima dimensione dell'array  $y$  o gli ultimi  $y$  elementi da ogni vettore nell'ultima dimensione dell'array  $x$

Equivalente alla funzione  $\text{dec}(x, y)$ .

Dec e' [un operatore](#)

**Eccezioni:** no

## DefExpr

**Formato:**  $\{x\}$

**Vincoli:**  $x$  e' un'espressione

**Descrizione:** sottoespressione assegnata alla variabile  $\$w_n$ , dove  $n$  varia da 0 a 3 secondo l'ordine di valutazione

DefExpr e' [un operatore](#)

**Eccezioni:** no

## Difference

**Formato:**  $x-y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:**  $x$  meno  $y$

Difference e' [un operatore](#)

Difference e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## Eq

**Formato:**  $x==y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** confronto logico:  $x$  e' uguale a  $y$ ?

Eq e' [un operatore](#)

Eq e' [una funzione diadica polimorfa](#), [un operatore booleano](#)

**Eccezioni:** Come specificato [qui](#).

## GE

**Formato:**  $x>=y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** confronto logico:  $x$  e' maggiore o uguale di  $y$ ?

GE e' [un operatore](#)

GE e' [una funzione diadica polimorfa](#), [un operatore booleano](#)

**Eccezioni:** Come specificato [qui](#).

## GT

**Formato:**  $x>y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** confronto logico:  $x$  e' maggiore di  $y$ ?

GT e' [un operatore](#)

GT e' [una funzione diadica polimorfa](#), [un operatore booleano](#)

**Eccezioni:** Come specificato [qui](#).

## LE

**Formato:**  $x \leq y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** confronto logico:  $x$  e' minore o uguale di  $y$ ?

LE e' [un operatore](#)

LE e' [una funzione diadica polimorfa](#), [un operatore booleano](#)

**Eccezioni:** Come specificato [qui](#).

## GenVect1

**Formato:**  $[x:y]$

**Vincoli:**  $x$  e  $y$  sono scalari

**Descrizione:** vettore dei valori da  $x$  a  $y$

GenVect1 e' [un operatore](#)

**Eccezioni:** no

## GenVect2

**Formato:**  $[x:y:z]$

**Vincoli:**  $x$ ,  $y$  e  $z$  sono scalari

**Descrizione:** vettore dei valori da  $x$  a  $z$  separati per  $z$

GenVect2 e' [un operatore](#)

**Eccezioni:** no

## LT

**Formato:**  $x < y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** confronto logico:  $x$  e' minore di  $y$ ?

LT e' [un operatore](#)

LT e' [una funzione diadica polimorfa](#), [un operatore booleano](#)

**Eccezioni:** Come specificato [qui](#).

## Minus

**Formato:**  $-x$

**Vincoli:**  $x$  e' un array

**Descrizione:** meno  $x$

Minus e' [un operatore](#)

Minus e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## Mod

**Formato:**  $x \% y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:**  $x$  modulo  $y$

Equivalente alla funzione  $\text{mod}(x, y)$ .

Mod e' [un operatore](#)

Mod e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## NE

**Formato:**  $x \neq y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** confronto logico:  $x$  e' diverso da  $y$ ?

NE e' [un operatore](#)

NE e' [una funzione diadica polimorfa](#), [un operatore booleano](#)



**Eccezioni:** Come specificato [qui](#).

## Not

**Formato:**  $!x$

**Vincoli:**  $x$  e' un array

**Descrizione:** negazione logica: non  $x$

Not e' [un operatore](#)

Not e' [una funzione monadica polimorfa](#), [un operatore booleano](#)

**Eccezioni:** no

## Or

**Formato:**  $x \mid y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:** disgiunzione logica:  $x \text{ o } y$

Or e' [un operatore](#)

Or e' [una funzione diadica polimorfa](#), [un operatore booleano](#)

**Eccezioni:** Come specificato [qui](#).

## PairScan-1

**Formato:**  $f \mid x$

**Vincoli:**  $f$  e' una funzione o un operatore;  $x$  e' un array

**Descrizione:** se  $x$  e' un vettore, vettore il cui primo elemento e' ottenuto applicando la funzione diadica  $f$  ai primi due elementi di  $x$ , il secondo elemento e' ottenuto applicando  $f$  al secondo e al terzo elemento, e cosi' via; se  $x$  e' un array di ordine superiore, array ottenuto nello stesso modo, applicando  $f$  in parallelo agli elementi di ogni vettore dell'ultima dimensione

Se  $f$  e' una funzione,  $x$  deve essere delimitata da parentesi (p.es.,  $\max \mid (v)$ ).

PairScan-1 e' [un operatore](#)

**Eccezioni:**  $x$  deve essere un array non nullo.

## PairScan-2

**Formato:**  $f \mid [n]x$

**Vincoli:**  $f$  e' una funzione o un operatore;  $x$  e' un array;  $n$  e' un intero

**Descrizione:** come  $f|x$ , dove  $f$  e' applicato alla dimensione  $n$  di  $x$

Se  $f$  e' una funzione,  $x$  deve essere delimitata da parentesi (p.es.,  $\max \mid [0](v)$ ).

PairScan-2 e' [un operatore](#)

**Eccezioni:**  $x$  deve essere un array non nullo;  $n$  deve essere una dimensione corretta per  $x$ , attualmente tra 0 e 2.

## Power

**Formato:**  $x^y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:**  $x$  elevato alla potenza  $y$

Power e' [un operatore](#)

Power e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## Product

**Formato:**  $x*y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:**  $x$  per  $y$

Product e' [un operatore](#)

Product e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## Quotient

**Formato:**  $x/y$

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:**  $x$  diviso  $y$

Quotient e' [un operatore](#)

Quotient e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## Reduction-1

**Formato:**  $f/x$

**Vincoli:**  $f$  e' una funzione o un operatore;  $x$  e' un array

**Descrizione:** se  $x$  e' un vettore, scalare ottenuto applicando la funzione diadica  $f$  ai primi due elementi di  $x$ , quindi al risultato e al terzo elemento, e cosi' via; se  $x$  e' un array di ordine  $n$ , array di ordine  $n-1$  ottenuto nello stesso modo, applicando  $f$  in parallelo agli elementi di ogni vettore dell'ultima dimensione

Se  $f$  e' una funzione,  $x$  deve essere delimitata da parentesi (p.es.,  $\max/(v)$ ).

Reduction-1 e' [un operatore](#)

**Eccezioni:**  $x$  deve essere un array non nullo.

## Reduction-2

**Formato:**  $f/[n]x$

**Vincoli:**  $f$  e' una funzione o un operatore;  $x$  e' un array;  $n$  e' un intero

**Descrizione:** come  $f/x$ , dove  $f$  e' applicato alla dimensione  $n$  di  $x$

Se  $f$  e' una funzione,  $x$  deve essere delimitata da parentesi (p.es.,  $\max/[0](v)$ ).

Reduction-2 e' [un operatore](#)

**Eccezioni:**  $x$  deve essere un array non nullo;  $n$  deve essere una dimensione corretta per  $x$ , attualmente tra 0 e 2.

## Scan-1

**Formato:**  $f \backslash x$

**Vincoli:**  $f$  e' una funzione o un operatore;  $x$  e' un array

**Descrizione:** se  $x$  e' un vettore, vettore il cui primo elemento e' ottenuto applicando la funzione diadica  $f$  ai primi due elementi di  $x$ , il secondo elemento e' ottenuto applicando  $f$  al risultato e al terzo elemento, e cosi' via; se  $x$  e' un array di ordine superiore, array ottenuto nello stesso modo, applicando  $f$  in parallelo agli elementi di ogni vettore dell'ultima dimensione

Se  $f$  e' una funzione,  $x$  deve essere delimitata da parentesi (p.es.,  $\max\{v\}$ ).

Scan-1 e' [un operatore](#)

**Eccezioni:**  $x$  deve essere un array non nullo.

## Scan-2

**Formato:**  $f[n]x$

**Vincoli:**  $f$  e' una funzione o un operatore;  $x$  e' un array;  $n$  e' un intero

**Descrizione:** come  $f[x]$ , dove  $f$  e' applicato alla dimensione  $n$  di  $x$

Se  $f$  e' una funzione,  $x$  deve essere delimitata da parentesi (p.es.,  $\max\{0\}(v)$ ).

Scan-2 e' [un operatore](#)

**Eccezioni:**  $x$  deve essere un array non nullo;  $n$  deve essere una dimensione corretta per  $x$ , attualmente tra 0 e 2.

## Size

**Formato:**  $@x$

**Vincoli:**  $x$  e' un array

**Descrizione:** dimensione di  $x$

Equivalente alla funzione `size(x)`. Il valore e' zero se  $x$  e' uno scalare, e un vettore altrimenti.

Size e' [un operatore](#)

**Eccezioni:** no

## order

**Formato:** `order(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** ordine, cioe' numero delle dimensioni, dell'array  $x$

order e' [una funzione per array](#)

**Eccezioni:** no

## pline

**Formato:** `pline(vx,vy,x)`

**Vincoli:** `vx` e `vy` sono vettori; `x` e' un array

**Descrizione:** valore `y` corrispondente a `x` sulla polinomiale i cui vertici sono nei vettori `vx` e `vy`

`pline` e' [una funzione matematica](#)

`pline` e' [una funzione di interpolazione](#)

**Eccezioni:** `vx` e `vy` devono avere lo stesso numero di elementi, almeno due; per ogni `i`, `vx[i]` deve essere minore di `vx[i+1]`.

## poisson

**Formato:** `poisson(v,x,s)`

**Vincoli:** `v` (opzionale) e' un vettore; `x` (opzionale) e' un array; `s` (opzionale) e' uno scalare, o 0 o 1

**Descrizione:** distribuzione di probabilita' di Poisson, di parametri `v=[p1]`, con `p1=mean` (`p1>0`; default: 5): se `x` e `s` non sono specificati, valore casuale estratto dalla distribuzione; se `s==0`, valore della pdf in `x`; se `s==1`, valore della cdf in `x`

`poisson` e' [una funzione statistica](#)

`poisson` e' [una funzione di distribuzione statistica](#)

**Eccezioni:** `s` deve essere 0 o 1; gli elementi di `x` devono essere strettamente positivi.

## rad2deg

**Formato:** `rad2deg(x)`

**Vincoli:** `x` e' un array

**Descrizione:** valore di `x` convertito da radianti a gradi

`rad2deg` e' [una funzione matematica](#)

`rad2deg` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## rand

**Formato:** `rand(x,y)`

**Vincoli:** `x` (opzionale) e' uno scalare; `y` (opzionale) e' uno scalare

**Descrizione:** valore casuale estratto da una distribuzione uniforme tra 0 e 1, o tra 0 e `x` se solo `x` e' specificato, o tra `x` e `y` se entrambi sono specificati

`rand` e' [una funzione statistica](#)

**Eccezioni:** Se solo `x` e' specificato, deve essere strettamente positivo; se sia `x` sia `y` sono specificati, il primo deve essere minore del secondo.

## **randInt**

**Formato:** `randInt(x)`

**Vincoli:** `x` e' un array

**Descrizione:** valore casuale intero estratto da una distribuzione uniforme tra 0 e `int(x-1)`

`randInt` e' [una funzione statistica](#)

`randInt` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## **readFromNet**

**Formato:** `readFromNet()`

**Vincoli:**

**Descrizione:** valore letto da un server di rete Spacebrew

`readFromNet` e' [una funzione di controllo](#)

**Eccezioni:** Il server di rete Spacebrew deve essere raggiungibile.

## **ReadFromXLS-1**

**Formato:** `readFromXLS(x,y,r,c)`

**Vincoli:** `x` e' una stringa; `y`, `r`, e `c` sono scalari

**Descrizione:** valore letto dalla riga `r` e colonna `c` del foglio di indice `y` del file xls il cui nome e' `x`  
Il file xls deve essere nella stessa cartella del modello, e il nome del file `x` deve essere indicato senza percorso. Tutti gli indici partono da 1.

ReadFromXLS-1 e' [una funzione di controllo](#)

**Eccezioni:** no

## readFromXLS-2

**Formato:** `readFromXLS(x,y,r1,c1,r2,c2)`

**Vincoli:** `x` e' una stringa; `y`, `r1`, `c1`, `r2`, e `c2` sono scalari

**Descrizione:** vettore o matrice letti tra la cella di riga `r1` e colonna `c1` e la cella di riga `r2` e colonna `c2` del foglio di indice `y` del file xls il cui nome e' `x`

Il file xls deve essere nella stessa cartella del modello, e il nome del file `x` deve essere indicato senza percorso. Tutti gli indici partono da 1.

readFromXLS-2 e' [una funzione di controllo](#)

**Eccezioni:** no

## remove

**Formato:** `remove(x,v)`

**Vincoli:** `x` e' un array; `v` e' o uno scalare o un vettore

**Descrizione:** array ottenuto eliminando l'elemento / gli elementi `v`-esimo/i dalla prima dimensione dell'array `x`

remove e' [una funzione per array](#)

**Eccezioni:** no

## resize

**Formato:** `resize(x,v)`

**Vincoli:** `x` e' un array; `v` e' un vettore

**Descrizione:** array ottenuto ridimensionando l'array `x` alla dimensione `v`, ed eliminando gli elementi eccedenti in coda o aggiungendo zeri in coda, se richiesto

resize e' [una funzione per array](#)

**Eccezioni:** no

## round

**Formato:** `round(x,y)`

**Vincoli:** `x` e `y` sono array (i vincoli sono specificati [qui](#))

**Descrizione:** `x` arrotondato a `y` decimali

`round` e' [una funzione matematica](#)

`round` e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## set

**Formato:** `set(x,v0,v1,...,y)`

**Vincoli:** `x` e' un array; `v1`, `v2`, ... sono scalari o vettori; `y` e' un array

**Descrizione:** array ottenuto dall'array `x` sostituendo il suo sottoarray di indici `v0` nella prima dimensione, di indici `v1` nella seconda dimensione, ... con l'array `y`

`set` e' [una funzione per array](#)

**Eccezioni:** no

## shift

**Formato:** `shift(x,y,s)`

**Vincoli:** `x` e `y` sono array; `s` (opzionale) e' uno scalare

**Descrizione:** array ottenuto concatenando l'array `y` con l'array `x` a sinistra e rimuovendo lo stesso numero di elementi da `x` a destra se `s==0` o se non specificato, o in ordine inverso se `s==1`

`shift` e' [una funzione per array](#)

**Eccezioni:** Deve essere possibile concatenare `x` con `y` in termini delle loro dimensioni; se specificato, `s` deve essere 0 o 1.

## shuffle

**Formato:** `shuffle(x)`

**Vincoli:** `x` e' un array



**Descrizione:** array ottenuto permutando in modo casuale gli elementi dell'array  $x$

`shuffle` e' [una funzione per array](#)

**Eccezioni:** no

## sigmoid

**Formato:** `sigmoid(vx,vy,x)`

**Vincoli:**  $vx$  e  $vy$  sono vettori;  $x$  e' un array

**Descrizione:** valore  $y$  corrispondente a  $x$  sulla sigmoide controllata dai punti  $vx[0],vy[0]$  e  $vx[1],vy[1]$

`sigmoid` e' [una funzione matematica](#)

`sigmoid` e' [una funzione di interpolazione](#)

**Eccezioni:** Sia  $vx$  sia  $vy$  devono avere due elementi;  $vx[0]$  deve essere minore di  $vx[1]$ .

## sign

**Formato:** `sign(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** segno di  $x$ , cioe' 1 se  $x > 0$ , -1 se  $x < 0$ , 0 se  $x == 0$

`sign` e' [una funzione matematica](#)

`sign` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## sin

**Formato:** `sin(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** seno di  $x$

$x$  e' espresso in radianti.

`sin` e' [una funzione matematica](#)

`sin` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## size

**Formato:** `size(x)`

**Vincoli:** `x` e' un array

**Descrizione:** dimensione di `x`

Equivalente all'operatore `@x`. Il valore e' zero se `x` e' uno scalare, e un vettore altrimenti.

`size` e' [una funzione per array](#)

**Eccezioni:** no

## sort

**Formato:** `sort(x,s)`

**Vincoli:** `x` e' un array; `s` (opzionale) e' uno scalare

**Descrizione:** array ottenuto ordinando gli elementi dell'array `x`, in ordine diretto se `s==0` o se non specificato, o in ordine inverso se `s==1`

`sort` e' [una funzione per array](#)

**Eccezioni:** Se specificato, `s` deve essere o 0 o 1.

## spline

**Formato:** `spline(vx,vy,x)`

**Vincoli:** `vx` e `vy` sono vettori; `x` e' un array

**Descrizione:** valore `y` corrispondente a `x` sulla spline i cui nodi sono nei vettori `vx` e `vy`

`spline` e' [una funzione matematica](#)

`spline` e' [una funzione di interpolazione](#)

**Eccezioni:** `vx` e `vy` devono avere lo stesso numero di elementi, almeno due; per ogni `i`, `vx[i]` deve essere minore di `vx[i+1]`.

## sqrt

**Formato:** `sqrt(x)`

**Vincoli:** `x` e' un array

**Descrizione:** radice quadrata di `x`

`sqrt` e' [una funzione matematica](#)

`sqrt` e' [una funzione monadica polimorfa](#)

**Eccezioni:** `x` deve essere positivo.

## sysTime

**Formato:** `sysTime()`

**Vincoli:**

**Descrizione:** numero di millisecondi dall'inizio della simulazione

`sysTime` e' [una funzione di controllo](#)

**Eccezioni:** no

## tan

**Formato:** `tan(x)`

**Vincoli:** `x` e' un array

**Descrizione:** tangente di `x`

`x` e' espresso in radianti.

`tan` e' [una funzione matematica](#)

`tan` e' [una funzione monadica polimorfa](#)

**Eccezioni:** no

## tDistribution

**Formato:** `tDistribution(v,x,s)`

**Vincoli:** `v` (opzionale) e' un vettore; `x` (opzionale) e' un array; `s` (opzionale) e' uno scalare, o 0 o 1

**Descrizione:** distribuzione di probabilita' `t`, di parametri `v=[p1]`, con `p1`=gradi di liberta' (`p1`>0; default: 5):  
se `x` e `s` non sono specificati, valore casuale estratto dalla distribuzione; se `s`==0, valore della pdf in `x`; se `s`==1, valore della cdf in `x`

tDistribution e' [una funzione statistica](#)

tDistribution e' [una funzione di distribuzione statistica](#)

**Eccezioni:**  $s$  deve essere 0 o 1; gli elementi di  $x$  devono essere strettamente positivi.

## transpose

**Formato:** `transpose(x)`

**Vincoli:**  $x$  e' un array

**Descrizione:** array ottenuto per trasposizione dell'array  $x$

Questa funzione implementa un concetto generalizzato di trasposizione. Se l'argomento e' uno scalare, e' restituito senza cambiamenti; altrimenti e' restituito l'array ottenuto sostituendo la  $i$ -esima dimensione dell'argomento con la  $(i+1)$ -esima dimensione.

transpose e' [una funzione per array](#)

**Eccezioni:** no

## uniform

**Formato:** `uniform(v,x,s)`

**Vincoli:**  $v$  (opzionale) e' un vettore;  $x$  (opzionale) e' un array;  $s$  (opzionale) e' uno scalare, 0 o 1 o 2

**Descrizione:** distribuzione di probabilita' uniforme (rettangolare), di parametri  $v=[p1,p2]$ , con  $p1=mean$  (default: 0),  $p2=stddev$  (default: 1): se  $x$  e  $s$  non sono specificati, valore casuale estratto dalla distribuzione; se  $s==0$ , valore della pdf in  $x$ ; se  $s==1$ , valore della cdf in  $x$ ; se  $s==2$ , valore della cdf inversa per la probabilita'  $x$

uniform e' [una funzione statistica](#)

uniform e' [una funzione di distribuzione statistica](#)

**Eccezioni:**  $s$  deve essere 0 o 1 o 2; se  $s==2$ , gli elementi di  $x$  devono essere nell'intervallo  $[0,1]$ .

## wrap

**Formato:** `wrap(x,y)`

**Vincoli:**  $x$  e  $y$  sono array (i vincoli sono specificati [qui](#))

**Descrizione:**  $x$  modulo  $y$  con gestione dei valori negativi

Mentre `mod(-2,5) == -2`, `wrap(-2,5) == 3`.

`wrap` e' [una funzione matematica](#)

`wrap` e' [una funzione diadica polimorfa](#)

**Eccezioni:** Come specificato [qui](#).

## **writeToNet**

**Formato:** `writeToNet(x)`

**Vincoli:** `x` e' un array

**Descrizione:** invia `x` a un server di rete Spacebrew

`writeToNet` e' [una funzione di controllo](#)

**Eccezioni:** Il server di rete Spacebrew deve essere raggiungibile.