Fuzzy Toolbox for MATLAB

Reference Guide
version 3.0

ROBERT BABUŠKA

May 18, 1995

Contents

1 Introduction

FUZZY TOOLBOX is a collection of MATLAB functions and algorithms, useful for implementing applications of fuzzy sets theory under MATLAB . Toolbox functions range from basic definitions in fuzzy sets theory, such as fuzzy set kernel, support, α-cuts, etc. through basic operations with fuzzy sets, relations, hedges, to special utilities supporting the design and implementation of fuzzy controllers and fuzzy modeling.

FUZZY TOOLBOX functions can be seen as templates for implementing many other concepts from fuzzy sets theory which are not included in the toolbox. Many of the FUZZY TOOLBOX functions can be embedded in SIMULINK blocks and used in simulations. In this way it is very easy for instance to design fuzzy controllers, to evaluate their performance in comparison to other types of controllers, etc.

We believe that the FUZZY TOOLBOX is not only a collection of routines implementing some concepts from fuzzy sets theory. We hope that this tool will be useful for research, education and perhaps also in industry.

2 New Features in Fuzzy Toolbox 3.0

As opposed to previous versions, in FUZZY TOOLBOX 3.0 all functions are coded as M-files. This ensures full portability over all MATLAB platforms. The most important changes were done in the routines supporting fuzzy controller design. The graphical rule-base editor from previous FUZZY TOOLBOX versions was replaced by a rule-base parser/compiler that converts the rule-base from a text form into a matrix form. Membership function editor is implemented with the aid of MATLAB Graphical User Interface (GUI) features. It has a similar structure to MATLAB figures, i.e. users can set, modify and get various properties. From the workspace, as many instances of the editor as needed can be created in a consistent way. Inference engine was optimized for speed and is now implemented in an M-file. Several new functions were added, including Sugeno–Takagi's fuzzy model and fuzzy C-means clustering algorithm. SIMULINK block library with full support for MATLAB 4.0 graphical features provides an effective environment for graphical fuzzy controller design and simulations.

3 Basic Concepts

In this section we will introduce the notation that will be used through the manual, we will discuss how fuzzy sets are represented and we will explain how the rule-base systems are implemented in the FUZZY TOOLBOX.

FUZZY TOOLBOX files are divided in three directories. The main directory contains basic fuzzy toolbox functions. Directory `FTSIMUL` contains SIMULINK block library and SIMULINK demon-

strations and directory `FTDEMOS` the basic FUZZY TOOLBOX demonstrations.

Fuzzy Set Representation

In FUZZY TOOLBOX a discrete-domain fuzzy set is usually stored in two vectors. The first vector, in the Reference Guide and Help texts usually denoted X, contains the elements of the universe of discourse. The second vector, usually denoted A or Ax, contains the membership grades of X elements into a fuzzy set A.

Continuous-domain fuzzy sets can be represented as vectors containing the parameters of analytically defined membership functions, i.e. the input parameters for the `mgrade` function. For specific operations, these sets must be usually discretized (using again the `mgrade` function). For more details run `ftdemo`.

Rule-Base Representation

Functions `txt2rls` `rls2txt`, `infer`, `dofmin`, etc. used for the design and implementation of fuzzy controllers, pass between each other the rule-base and membership functions matrices.

Rule-base matrix (rls)

This matrix contains the rules coded as integer numbers. These numbers can be regarded as row indices pointing to the membership functions matrix. Every row of the rule-base matrix defines one rule. The first column contains always the weights of the rules (real numbers from 0 to 1). The remaining columns are the codes of the premise and consequent fuzzy sets. Premise variables are connected with the AND operator. Example:

```
rls = [0.9 1 3 5;
       1.0 2 4 6]
```

can define two rules, with 2 premise variables and one consequent variable. However, it also can be two rules with one premise variable and two consequents. User on his/her own is responsible for the interpretation of the rule-base matrix. Note that inference functions do not require an extra parameter to specify the number of premise and consequent variables. For instance, in the `infer` function the number of premise and consequent variables is detected from the number of columns in the input data matrix. To continue the above example, the first column defines rule weights, 0.9 for the first rule and 1.0 for the second rule. The rest of the first row then means that the first, third and fifth row of the membership function matrix are present in the rule. If the membership function matrix defines membership functions for fuzzy sets SMALL, BIG for the first premise variable, POSITIVE, NEGATIVE for the second premise variable and SMALL and BIG for the consequent (in this order), that the first rule reads:

If inp1 is SMALL and inp2 is POSITIVE then output is SMALL.

Similarly, the second rule reads: *If inp1 is BIG and inp2 is NEGATIVE then output is BIG*. If the input or output is not specified in a given rule, then the rule-base matrix row contains zeros on the corresponding places. Complements of the fuzzy sets have the same codes as the sets but with negative sign. For example, a fuzzy expression **NOT BIG** in the premise of the above example would be coded as -2.

Membership Functions Matrix (mfs)

This matrix contains the definition of the analytical membership functions. Each membership function is defined by four break-points, see Fig. 1 for an example of a trapezoidal membership function. Each row of the matrix defines one membership function. First element in the row specifies the membership function type.

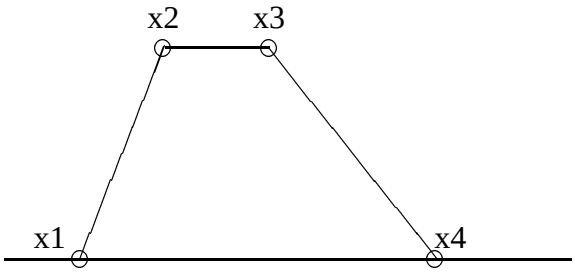


Figure 1: Membership function definition

Available membership function types are *trapezoidal*, *exponential* and *sigmoidal*. The remaining four elements are the four breakpoints. Example:

```
mfs = [1 -0.3 0.4 0.5 0.9;
       1  0.0 0.7 0.7 1.0]
```

defines two trapezoidal membership functions, first with left-bottom -0.3, left-shoulder 0.4, right-shoulder 0.5 and right-bottom 0.9, second is a triangular membership function with the base between 0.0 and 1.0 and the top in 0.7.

4 Quick Reference Tables

Fuzzy Sets – Basic Definitions	
`acut`	α-cut
`acuts`	Strong α-cut
`alevel`	α-level
`card`	Fuzzy set cardinality
`crossovr`	Crossover point
`hgt`	Fuzzy set height
`ker`	Fuzzy set kernel
`normal`	Fuzzy set normalization
`supp`	Fuzzy set support
`mgrade`	Membership grade
`mfplot`	Plots membership function(s)

Operations with Fuzzy Sets	
`and`	Conjunction of two fuzzy sets with type as a parameter
`and_lu`	Conjunction of two fuzzy sets - Lukasiewicz
`and_pr`	Conjunction of two fuzzy sets - probability
`and_re`	Conjunction of two fuzzy sets - restricted
`and_za`	Conjunction of two fuzzy sets - Zadeh
`or`	Disjunction of two fuzzy sets with type as a parameter
`or_lu`	Disjunction of two fuzzy sets - Lukasiewicz
`or_pr`	Disjunction of two fuzzy sets - probability
`or_re`	Disjunction of two fuzzy sets - restricted
`or_za`	Disjunction of two fuzzy sets - Zadeh
`sdif`	Symmetrical difference of two fuzzy sets
`csum`	Convex linear sum of min and max of two fuzzy sets
`not`	Complement with type as a parameter
`not_za`	Complement according to Zadeh
`not_su`	Complement according to Sugeno
`not_ya`	Complement according to Yager
`gendist`	Generalized distance function
`gencrit`	Generalized criterion function

Triangular Norms and Co-norms	
tnorm	Triangular norm with type as a parameter
t_do	Triangular norm - Dombi
t_du	Triangular norm - Dubois & Prade
t_fr	Triangular norm - Frank
t_ha	Triangular norm - Hamacher
t_la	Triangular norm - lambda
t_sk	Triangular norm - Schweizer & Sklar
t_ya	Triangular norm - Yager
snorm	Triangular co-norm with type as a parameter
s_do	Triangular co-norm - Dombi
s_du	Triangular co-norm - Dubois & Prade
s_fr	Triangular co-norm - Frank
s_ha	Triangular co-norm - Hamacher
s_la	Triangular co-norm - lambda
s_sk	Triangular co-norm - Schweizer & Sklar
s_ya	Triangular co-norm - Yager

Composition of Fuzzy Relations	
sup_min	Sup-min composition
sup_prod	Sup-product composition

Linguistic Hedges	
conc	Concentration operator
dil	Dilation operator
ext	Extensification
int	Intensification
normal	Fuzzy set normalization
power	General power
highly	Linguistic hedge highly
moreless	Linguistic hedge more or less
minus	Linguistic hedge minus
plus	Linguistic hedge plus
rather	Linguistic hedge rather
roughly	Linguistic hedge roughly
slightly	Linguistic hedge slightly
very	Linguistic hedge very

Fuzzy Implications	
`impl_ex`	Extended propositional calculus implication
`impl_ma`	Material implication
`impl_pr`	Propositional calculus implication
`impl_bg`	Brouwer-Gödel's implication.
`impl_go`	Goguen's implication.
`impl_kd`	Kleene-Dienes' implication.
`impl_la`	Larsen's implication
`impl_lu`	Lukasiewicz' implication
`impl_md`	Mamdani's implication
`impl_re`	Reichenbach's implication
`impl_rs`	Rescher's implication
`impl_wi`	Willmott's implication

Fuzzy Control and Modeling	
`dofmin`	Degree of fulfillment, conjunction calculated as minimum
`dofprod`	Degree of fulfillment, conjunction calculated as product
`doftn`	Degree of fulfillment, with user-supplied t-norm
`infer`	Mamdani's inference
`txt2rls`	Convert text form of the rules into matrix form
`rls2txt`	Convert matrix form of the rules into text form
`suglms`	LMS solution for rule consequents in Sugeno's model
`sugval`	Sugeno's inference
`rbfull`	Generate all possible combinations of premise variables
`mfequ`	Generate equidistant symmetrical membership functions

Defuzzification Methods	
`area`	The area of a fuzzy set
`coa`	Center of area (gravity) defuzzification
`coaw`	Weighted singleton center of area defuzzification
`mom`	Mean of maxima defuzzification

Fuzzy Control Blocks	
`Fuzzylib`	Fuzzy Toolbox block library
`Aggregation`	Aggregation of rules (consequent fuzzy sets)
`Defuzzifier`	Defuzzification
`DOF`	Calculates degrees of fulfillment of rules
`INFER`	Fuzzy inference block
`Fuzzifier`	Fuzzification of a crisp or fuzzy input
`Implication`	Fuzzy implication
`MAX-MIN`	MAX-MIN composition
`Sugeno`	Sugeno's inference

Utility Blocks	
`B-Scope`	Bar graph scope
`F-Scope`	Fuzzy set scope
`G-Scope`	Enhanced graphical scope
`Pause`	Pauses simulation for n seconds

Purpose:

α-cut of a fuzzy set.
Strong α-cut of a fuzzy set.

Synopsis:

```
[Xa,Aa,B] = acut(X,A,alpha)
[Xa,Aa,B] = acuts(X,A,alpha)
```

Description:

`acut(X,A,alpha)` calculates the α-cut of a fuzzy set `A` on a universe `X` for `alpha` being a real number from 0 to 1. α-cut of a fuzzy set A is an ordinary subset of `X`

$$A_\alpha = \{x; \alpha \leq Ax\}, \quad \alpha \in [0,1]$$

`acuts(X,A,alpha)` calculates the strong α-cut of a fuzzy set `A` on a universe `X` for `alpha` being a real number from 0 to 1. Strong α-cut of a fuzzy set A is an ordinary subset of `X`

$$A_\alpha = \{x; \alpha < Ax\}, \quad \alpha \in [0,1]$$

Output arguments are the following. `Xa` is a vector containing the α-cut elements, `Aa` contains the membership grades of `Xa` elements and `B` is a fuzzy set such that

$$B = A \cap A_\alpha$$

See also:

`alevel, crossovr, ker, supp, hgt, card, normal`

Purpose:

α-level of a fuzzy set.
Crossover points of a fuzzy set.

Synopsis:

```
[Xa,B] = alevel(X,A,alpha)
Xc = crossovr(X,A)
```

Description:

`alevel(X,A,alpha)` calculates the α-level of a fuzzy set `A` on a universe `X` for `alpha` being a real number from 0 to 1. α-level of a fuzzy set A is an ordinary subset of `X`

$$A_\alpha = \{x; \alpha = Ax\}, \quad \alpha \in [0, 1]$$

Output arguments are the following. `Xa` is a vector containing the α-level elements, and `B` is a fuzzy set which membership grades are zero for all x except the α-level points, where it equals to α.

`crossovr(X,A)` finds the crossover points of a fuzzy set. Crossover points of a fuzzy set A are the elements of `X` such that $Ax = \frac{1}{2}$, i.e. the same as `alevel(X,A,0.5)`.

See also:

`acut, acuts, ker, supp, hgt, card, normal`

Purpose:

The intersection of two fuzzy sets. Alternative intersection operators can be specified.

Synopsis:

```
C = and(A,B)
C = and(A,B,itype)
C = and(A,B,itype,r)
```

Description:

`and(A,B)` calculates the intersection of two fuzzy sets `A` and `B` using the following formula proposed by Zadeh:

$$Cx = \min(Ax, Bx)$$

`and(A,B,itype)` calculates the intersection of two fuzzy sets `A` and `B` using the alternative intersection operators. The desired operator is selected using the optional parameter `itype`. Possible values of `itype` are following:

`'za'`	Zadeh (default), see `and_za`
`'lu'`	Lukasiewicz, see `and_lu`
`'pr'`	probability, see `and_pr`
`'re'`	restricted, see `and_re`

`and(A,B,itype,r)` is the synopsis for the restricted intersection, where `r` is a parameter, see `and_re` for details.

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`.

Instead of using function `and` with a particular parameter, you can call directly the corresponding `and_` function, e.g. instead of `and(A,B,'lu')` you can use `and_lu(A,B)`, which is slightly faster and therefore recommended for speed-critical applications.

See also:

`and_za, and_lu, and_re, and_pr`

Purpose:

The intersection of two fuzzy sets according to Lukasiewicz.
The probabilistic intersection of two fuzzy sets.
The intersection of two fuzzy sets according to Zadeh.

Synopsis:

```
C = and_lu(A,B)
C = and_pr(A,B)
C = and_za(A,B)
```

Description:

`and_lu(A,B)` calculates the intersection of two fuzzy sets `A` and `B` using the following formula proposed by Lukasiewicz:

$$Cx = \max(Ax + Bx - 1, 0)$$

`and_pr(A,B)` calculates the probabilistic intersection of two fuzzy sets A and B using the following formula from probability theory:

$$Cx = Ax.Bx$$

`and_za(A,B)` calculates the intersection of two fuzzy sets `A` and `B` using the following formula proposed by Zadeh:

$$Cx = \min(Ax, Bx)$$

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`. If only one input matrix `A` is supplied, the intersection of all rows in `A` is calculated.

See also:

`and, and_re`

Purpose:

The restricted intersection of two fuzzy sets.

Synopsis:

```
C = and_re(A,B,r)
```

Description:

`and_re(A,B,r)` calculates the restricted intersection of two fuzzy sets `A` and `B` using the following formulas:

$$\begin{aligned} &\text{if } r \geq 0 \ldots \quad Cx = Ax.Bx + r.(\min(Ax, Bx) - Ax.Bx) \\ &\text{if } r < 0 \ldots \quad Cx = Ax.Bx + r.(Ax.Bx - \max(0, Ax + Bx - 1)) \end{aligned}$$

Parameter `r` equals to 1 when `A` and `B` are strongly positively associated, to -1 when `A` and `B` are strongly negatively associated and to 0 when `A` and `B` are independent.

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`. If only one input matrix `A` is supplied, the intersection of all rows in `A` is calculated.

See also:

`and, and_za, and_lu, and_pr`

Purpose:

Absolute or relative scalar cardinality of a fuzzy set.

Synopsis:

```
C = card(A)
C = card(A,'rel')
```

Description:

`card(A)` calculates the scalar cardinality of a finite fuzzy set `A`. For finite fuzzy sets, scalar cardinality is defined as

$$|A| = \sum_{X} Ax$$

`card(A,'rel')` calculates the relative scalar cardinality of a finite fuzzy set `A` on a universe `X`. For finite fuzzy sets, relative scalar cardinality is defined as

$$||A|| = \frac{|A|}{|X|}$$

See also:

`acut, acuts, alevel, crossovr, supp, ker, hgt, normal`

Purpose:

Calculates centers of fuzzy clusters (cluster prototypes). Used by `fisodata`.
Calculates the distance of data points from cluster centers. Used by `fisodata`.

Synopsis:

```
v = center(x,f)
d = dist(x,v)
```

Description:

Function `center` calculates centers of fuzzy clusters (cluster prototypes). `x` is the data matrix, `f` is the fuzzy partition matrix. Matrix `v` with cluster center coordinates is returned. The centers are calculated as centers of gravity of the volume cluster.

Function `dist` calculates the distance of data points given in matrix `x` from cluster centers (prototypes) given in matrix `v`.

These functions are called at each iteration step of the fuzzy ISODATA clustering algorithm (function `fisodata`). By modifying these functions user can change the clusters from standard volume to e.g. elliptical, change the distance measure, etc.

See also:

`fisodata, fisofast`

Purpose:

Indexed center of area defuzzification.
Mean of maxima defuzzification.

Synopsis:

```
X0 = coa(X,A)
X0 = coa(X,A,i)
X0 = mom(X,A)
```

Description:

`coa(X,A)` calculates the center of area of a fuzzy set `A` on universe `X`, using the following formula:

$$X_0 = \sum_{x \in X} xAx / \sum_{x \in X} Ax$$

`coa(X,A,i)` calculates the center of area of a fuzzy set `A` on universe `X`. Elements of `A` less than threshold `i` are set to zero.

$$X_0 = \sum_{\{x; Ax > i\}} xAx / \sum_{Ax > i} Ax$$

`mom(X,A)` calculates the mean of maxima point for a fuzzy set `A` on universe `X`, using the following formula:

$$X_0 = \sum_{\{x; Ax = \max\}} xAx / \sum_{Ax = \max} Ax$$

See also:

`area, coaw`

Purpose:

Weighted center of area defuzzification.
Area of a fuzzy set.

Synopsis:

```
X0 = coaw(X,A,w)
S = area(X,A)
```

Description:

`coaw(X,A,w)` calculates the weighted center of area of a fuzzy set `A` on universe `X`, using the following formula:

$$X_0 = \sum_{x \in X} wxAx / \sum_{x \in X} wAx$$

This method is can be used in fuzzy controllers where the output terms are defined by fuzzy singletons. In this case the weights will be ones. If output fuzzy sets are not singletons, for speeding up the computations their centroids can be used instead of singletons. Then the weight can be for example the fuzzy set area. In this way also the shape of the membership function (not only the position of its maximum) is taken into account.

`area(X,A)` calculates the area a fuzzy set `A` on universe `X`, using numerical integration.

See also:

`coa, mom`

Purpose:

Concentration operator.
Dilution operator.
General power of a fuzzy set.

Synopsis:

```
B = conc(A)
B = dil(A)
B = dil(A,'za')
B = power(A,m)
```

Description:

`conc(A)` is the concentration operator used for defining linguistic hedges on fuzzy sets. It is defined as:

$$Bx = [Ax]^2$$

`dil(A)` is the dilution operator used for defining linguistic hedges on fuzzy sets. It is defined as:

$$Bx = 2Ax - [Ax]^2$$

`dil(A,'za')` is the dilution operator proposed by Zadeh as:

$$Bx = [Ax]^{\frac{1}{2}}$$

`power(A,m)` is the m-th power of a fuzzy set `A` defined by Zadeh as:

$$Bx = [Ax]^m$$

See also:

`ext, int, normal`

Purpose:

The convex linear sum of min and max. The symmetrical difference of two fuzzy sets.

Synopsis:

```
C = csum(A,B,lambda)
C = sdif(A,B)
```

Description:

`csum(A,B)` calculates the convex linear sum of min and max of two fuzzy sets `A` and `B` using the following formula:

$$Cx = \lambda.\min(Ax, Bx) + (1 - \lambda).max(Ax, Bx), \quad \lambda \in [0, 1]$$

It is a combination of fuzzy sets `A` and `B` that lies between union and intersection.

`sdif(A,B)` calculates the symmetrical difference of two fuzzy sets `A` and `B` using the following formula:

$$Cx = |Ax - Bx|$$

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`.

Purpose:

The degree of fulfillment of the rule premises, calculated as minimum of the antecedent membership grades.
The degree of fulfillment of the rule premises, calculated as product of the antecedent membership grades.
The degree of fulfillment of the rule premises, calculated with a user-supplied t-norm operator.

Synopsis:

```
dof=dofmin(rls,mfs,x,domain)
dof=dofprod(rls,mfs,x,domain)
dof=doftn(rls,mfs,x,domain,tnorm,par)
```

Description:

For given rule matrix `rls` (or only its premise part), membership function matrix `mfs` and matrix of crisp inputs `x`, these functions calculate the degrees of fulfillment of the rule premises. Matrix `rls` contains the rule weights and codes of the antecedent fuzzy sets (labels). The elements of `rls` are simply row indexes of the corresponding fuzzy sets stored in `mfs`. Value 0 is reserved for fuzzy label ANY, i.e. the situation when a particular input does not appear in the rule. Negative sign before the label number denotes the NOT operator. For details on membership function definition see `mgrade`. With the `doftn` function, `tnorm` is a string that contains the name of a function (M-file) that calculates the t-norm, `par` is an optional parameter passed to the specified t-norm function. Optionally, membership functions can be defined point-wise over a discretized domain. In this case, the domain is given in parameter `domain`.

Example: Consider the following two rules:

If error is LP and error change is LP then control is LP.
If error is LN and error change is not LN then control is LN.

Membership functions for the labels LP and LN are:

```
mfs = [ 1   -1   -1   -1  0.5; ...   % error LN
        1 -0.5    1    1    1; ...   % error LP
        1   -1   -1 -0.7  0.5; ...   % error change LN
        1 -0.5  0.7    1    1; ...   % error change LP
        1  -10  -10   -8    0; ...   % control LN
        1    0    8   10   10; ...   % control LP
      ]
```

The rule-base matrix for the above two rules reads:

```
rls = [ 1    2    4    6; ...        % 1st rule
        1    1   -3    5; ...        % 2nd rule
      ]
```

For calculating the degrees of fulfillment of the rules for five input error – error change pairs

```
input = [ -1 -1; -0.5 0; 0 0.2; 0.3 1; 1 1]
```

we use the following command:

```
dof = dofmin(rls,mfs,input)
```

See also:

`infer, txt2rls, rls2txt, sugval, mfset, mfget`

Purpose:

Fuzzy ISODATA clustering method (also called C-means).

Synopsis:

```
[f,v] = fisodata(x,f0)
[f,v] = fisodata(x,f0,m,e,s)
[f,v] = fisofast(x,f0)
[f,v] = fisofast(x,f0,m,e,s)
```

Description:

Function `fisodata` partitions a given data into a specified number of fuzzy clusters, using the fuzzy ISODATA (C-means) algorithm. Standard volume clusters and Euclidean distance are used. `x` is the data matrix, `f` is the initial fuzzy partition matrix or just a desired number of clusters. In the latter case, a random initial partition is generated automatically. An optional parameter `m` is a real number which determines the fuzziness of the clusters. For `m` close to 1, crisp clusters are obtained, with `m` increasing the fuzziness of the clusters increases. An optional parameter `e` defines the termination tolerance. The iterative algorithm stops when the difference between the partition matrices in two successive steps $|F(k-1) - F(k)| \leq e$. Default tolerance is 1e-3. An optional parameter `s` can be set to 1 in order to plot intermediate results (only for 1D and 2D data). Default value is 0, i.e. no plot. When specifying this parameter, arguments `m` and `e` can be empty matrices, in which case the default values are used. The function returns the partition matrix `f` and cluster prototype matrix `v`.

`fisofast` is a faster version of the same algorithm. The difference is that `fisofast` does not call the `center` and `dist` functions, but calculates the cluster centers and distances internally.

See also:

`center, dist`

Purpose:

The generalized criterion function.
The generalized distance function.

Synopsis:

```
C = gencrit(A,s)
C = gendist(A,B,s)
```

Description:

`gencrit(A,s)` is the generalized criterion function expressed by the following formula:

$$C = (\frac{1}{n}\sum_{i=1}^{N} Ax_i^s)^{\frac{1}{s}}$$

For different `s` we get the following functions:

$s \to -\infty$	C is the min operator
$s \to -1$	C is the harmonic mean operator
$s \to 0$	C is the geometric mean operator
$s \to \infty$	C is the max operator

`gendist(A,B,s)` is the generalized distance function used for comparing fuzzy sets in terms of their distance. It is defined by the following formula:

$$D = (\sum_{i=1}^{N} |Ax_i - Bx_i|^s)^{\frac{1}{s}}$$

For different `s` we get the following functions:

$s = 1$	D is the Hamming distance
$s = 2$	D is the Euclidean distance

Purpose:

The height of a fuzzy set.
Fuzzy set normalization.

Synopsis:

```
H = hgt(A)
B = normal(A)
```

Description:

`hgt(A)` calculates the height of a fuzzy set `A`. The height of a finite fuzzy set A is a defined as

$$\text{Hgt } A = \max_{x \in X} Ax$$

`normal(A)` converts a fuzzy set `A` into a normal fuzzy set, using the following formula:

$$Bx = \frac{Ax}{\text{Hgt } A}$$

See also:

`acut, acuts, alevel, crossovr, supp, ker, card`

Purpose:

Linguistic hedge highly.
Linguistic hedge roughly.

Synopsis:

```
B = highly(A)
B = roughly(A)
```

Description:

`highly(A)` is the hedge *highly* used as a modifier of the meaning of A. It is defined as:

$$Bx = [Ax]^3$$

`roughly(A)` is the hedge *roughly* used as a modifier of the meaning of A. It is defined as:

$$B = \text{dil}(\text{dil}(A))$$

See also:

`very, moreless, plus, minus, rather, slightly`

Purpose:

Brouwer-Gödel's implication.
Goguen's implication.
Kleene-Dienes' implication.

Synopsis:

```
R = impl_bg(A,B)
R = impl_go(A,B)
R = impl_kd(A,B)
```

Description:

`impl_bg(A,B)` is the fuzzy implication proposed by Brouwer and Gödel as:

$$A \to B = \begin{cases} 1 & \text{if } Ax \leq Bx \\ Bx & \text{otherwise} \end{cases}$$

`impl_go(A,B)` is the fuzzy implication proposed by Goguen as:

$$A \to B = \begin{cases} \min(1, Bx/Ax) & \text{if } Ax \neq 0 \\ 1 & \text{otherwise} \end{cases}$$

`impl_kd(A,B)` is the fuzzy implication proposed by Kleene and Dienes as:

$$A \to B = \max(1 - Ax, Bx)$$

`A` can be a scalar (i.e. a single membership grade) or a column vector (i.e. a vector of membership grades). `B` can be a row vector (i.e. a fuzzy set defined point-wise) or a matrix (i.e. a vector of fuzzy sets). `R` is a vector (fuzzy set) or matrix (vector of fuzzy sets), depending on `B`.

See also:

`impl\_la, impl\_lu, impl\_md, impl\_re, impl\_rs, impl\_wi`

Purpose:

Larsen's implication.
Lukasiewicz' implication.
Mamdani's implication.

Synopsis:

```
R = impl_la(A,B)
R = impl_lu(A,B)
R = impl_md(A,B)
```

Description:

`impl_la(A,B)` is the fuzzy implication proposed by Larsen as:

$$A \rightarrow B = Ax.Bx$$

`impl_lu(A,B)` is the fuzzy implication proposed by Lukasiewicz as:

$$A \rightarrow B = \min(1, 1 - Ax + Bx)$$

`impl_md(A,B)` is the fuzzy implication proposed by Mamdani as:

$$A \rightarrow B = \min(Ax, Bx)$$

`A` can be a scalar (i.e. a single membership grade) or a column vector (i.e. a vector of membership grades). `B` can be a row vector (i.e. a fuzzy set defined point-wise) or a matrix (i.e. a vector of fuzzy sets). `R` is a vector (fuzzy set) or matrix (vector of fuzzy sets), depending on `B`.

See also:

`impl\_bg, impl\_go, impl\_kd, impl\_re, impl\_rs, impl\_wi`

Purpose:

Reichenbach's implication.
Rescher's implication.
Willmott's implication.

Synopsis:

```
R = impl_re(A,B)
R = impl_rs(A,B)
R = impl_wi(A,B)
```

Description:

`impl_re(A,B)` is the fuzzy implication proposed by Reichenbach as:

$$A \rightarrow B = 1 - Ax + Ax.Bx$$

`impl_rs(A,B)` is the fuzzy implication proposed by Rescher as:

$$A \rightarrow B = \begin{cases} 1 & \text{if } Ax \leq Bx \\ 0 & \text{otherwise} \end{cases}$$

`impl_wi(A,B)` is the fuzzy implication proposed by Willmott as:

$$A \rightarrow B = \max(1 - Ax, \min(Ax, Bx))$$

`A` can be a scalar (i.e. a single membership grade) or a column vector (i.e. a vector of membership grades). `B` can be a row vector (i.e. a fuzzy set defined point-wise) or a matrix (i.e. a vector of fuzzy sets). `R` is a vector (fuzzy set) or matrix (vector of fuzzy sets), depending on `B`.

See also:

`impl\_la, impl\_lu, impl\_md, impl\_bg, impl\_go, impl\_kd`

Purpose:

Material fuzzy implication.
Propositional calculus fuzzy implication.
Extended propositional calculus fuzzy implication.

Synopsis:

```
R = impl_ma(A,B)
R = impl_ma(A,B,snorm,par)
R = impl_pr(A,B)
R = impl_pr(A,B,snorm,tnorm,spar,tpar)
R = impl_ex(A,B,snorm,tnorm,spar,tpar)
```

Description:

`impl_ma(A,B)` is the arithmetic rule of fuzzy implication proposed by Zadeh as:

$$A \rightarrow B = \min(1, 1 - Ax + Bx)$$

`impl_ma(A,B,snorm,par)` is the general material fuzzy implication defined as:

$$A \rightarrow B = (\text{not } A) + B$$

where $+$ denotes a triangular co-norm. Parameters `snorm` and `par` are the triangular co-norm function and its parameter respectively. Parameter `par` should be supplied only for co-norms that require additional parameter (e.g. `s_du`, or `or_re`). Note that the Zadeh's arithmetic rule follows from the general material implication definition by using the bounded sum operator (i.e. $\min(1, Ax + Bx)$, toolbox function `or_lu`).

`impl_pr(A,B)` is the max-min rule of fuzzy implication proposed by Zadeh as:

$$A \rightarrow B = \max(\min(Ax, Bx), 1 - Ax)$$

`impl_pr(A,B,snorm,tnorm,spar,tpar)` is the general propositional calculus fuzzy implication defined as:

$$A \rightarrow B = (\text{not } A) + (A * B)$$

where $+$ denotes a triangular co-norm and $*$ denotes a triangular norm. Parameters `snorm` and `spar` are the triangular co-norm function and its parameter respectively. Parameters `tnorm` and `tpar` are the triangular norm function and its parameter respectively. Parameters `spar` and `tpar` should be supplied only for norms and co-norms that require additional parameter (e.g. `t_du, s_du`, or `and_re, or_re`). Note that the Zadeh's max-min rule follows from the general propositional calculus implication by using the intersection

and union operators (i.e. $\min(Ax, Bx)$ and $\max(Ax, Bx)$, toolbox functions `and` and `or` respectively).

`impl_ex(A,B,snorm,tnorm,spar,tpar)` is the general extended propositional calculus fuzzy implication defined as:

$$A \rightarrow B = (\text{not } A * \text{not } B) + B$$

where $+$ denotes a triangular co-norm and $*$ denotes a triangular norm. Parameters `snorm` and `spar` are the triangular co-norm function and its parameter respectively. Parameters `tnorm` and `tpar` are the triangular norm function and its parameter respectively. Parameters `spar` and `tpar` should be supplied only for norms and co-norms that require additional parameter (e.g. `t_du, s_du`, or `and_re, or_re`).

`A` and `B` must be vectors, `R` is always a matrix calculated on the Cartesian product universe `A` x `B`.

Purpose:

Performs fuzzy inference and defuzzification.

Synopsis:

```
[y,dof] = infer(rls,mfs,x)
[y,dof] = infer(rls,mfs,x,def)
```

Description:

For the rule-base `rls`, membership functions definition `mfs` and current input vector `x`, the output vector `y` is calculated. The degree-of-fulfillment matrix `dof` is returned as a second argument. Mamdani's inference with a fast center-of-area defuzzification is used. An optional input argument `def` specifies a vector of default values returned when no rules are applicable (optional, defaults to 0). By supplying a vector of inputs, a control surface can be evaluated and plotted. For example:

```
dom = -1:0.1:1;                 % common domain for both inputs
[x,y] = meshgrid(dom);          % create grid for the mesh plot
x = [x(:) y(:)];                % make a vector from each x and y
y(:) = infer(rls,mfs,x);        % fuzzy inference
mesh(dom,dom,y);                % plot the mesh
```

See also:

`dofmin, dofprod, doftn, txt2rls, mfset, mfget`

Purpose:

Contrast intensification.
Contrast extensification.

Synopsis:

```
B = int(A)
B = ext(A)
```

Description:

`int(A)` is the contrast intensification operator used for defining linguistic hedges on fuzzy sets. It is defined as:

$$Bx = \begin{cases} 2[Ax]^2 & \text{if } Ax \leq 0.5 \\ 1 - 2(1 - Ax)^2 & \text{otherwise} \end{cases}$$

`ext(A)` is the contrast extensification operator used for defining linguistic hedges on fuzzy sets. It is defined as:

$$Bx = \begin{cases} (\frac{1}{2}Ax)^{\frac{1}{2}} & \text{if } Ax \leq 0.5 \\ 1 - (\frac{1-Ax}{2})^{\frac{1}{2}} & \text{otherwise} \end{cases}$$

See also:

`conc, dil, normal, power`

Purpose:

Generates a set of equidistant symmetrical membership functions.
Generates all possible combinations of the input fuzzy sets in the premise part of a rule-base.

Synopsis:

```
mfs = mfequ(dom,mftype,n)
rls = rbfull(nlab)
```

Description:

Function `mfequ` generates a set of equidistant symmetrical membership functions. `dom` is the domain (universe of discourse) or just its minimum and maximum value (e.g. [min(dom) max(dom)]). `mftype` is the type of membership functions (see `mgrade`) and `n` is the number of membership functions or a vector of the peaks of membership functions (e.g. [c1 c2 c3 c4 c5 ...]).

Function `rbfull` generates all possible combinations of the input fuzzy sets in the premise part of a rule-base. Rule weights are equal to 1. This function is useful for generating the premise part automatically. The result can be converted in a text form using `rls2txt`. The `nlab` vector contains number of labels for each input, e.g. [2 3] for a premise part with two inputs, 2 labels in the first and 3 in the second input

See also:

`mfset, mfget, infer, txt2rls, rls2txt`

Purpose:

Sets the properties of the graphical membership function editor.
Gets the properties from the specified membership functions editor figure.

Synopsis:

```
mfset
mfset(H)
mfset(H,'PropertyName1',PropertyValue1,'PropertyName2',PropertyValue2
mfget(H,'PropertyName')
```

Description:

`mfset(H,'PropertyName',PropertyValue)` sets the value of the specified property for the editor in figure with handle `H`. `H` also can be a vector of figure handles, in which case `mfset` sets the editor properties for all specified figures. Multiple property values can be set with a single statement. `mfset` without parameters opens a new figure and initializes editor with defaults. `mfset(H)` initializes editor in figure with handle `H`.

Available properties: *Position* – defines figure position on the screen, *Input* – sets value of the crisp input, which fuzzy representation with respect to the defined fuzzy sets can be read with `mfget`, *MFs* – membership functions matrix (see `mgrade` for details) or a vector of MF types or just a number of membership functions, *Domain* – string defining the domain (universe of discourse) for the fuzzy sets, and *CMF* – sets (highlights) the current membership function. Property names are case insensitive.

Editor usage. With the four sliders in the corners of the figure, you can change the shape of the selected function or of all functions simultaneously, if no function is selected. Every membership function can be selected and deselected by clicking the curve of the function. The purpose of the wide slider in the upper part of the figure is two-fold. When no function is selected, it sets the crisp input value (red vertical line in the graph). If one of the membership functions is selected, the slider is used to move the function along the domain. The popup menu in the top center of the figure sets the membership function type (trapezoidal, exponential or sigmoidal). If no function is selected, the popup choice applies to all membership functions, otherwise only to the selected one. The editable text field in the bottom center of the figure defines the domain (universe of discourse) for the fuzzy sets. Note that the domain must be entered as a string, e.g. `'-10:0.1:10'`. The domain step is used as the resolution for membership function plots.

`mfget` reads the parameters from the specified membership functions editor. Available properties are: *Input* – crisp input value, *MFs* – membership functions matrix (see `mgrade`

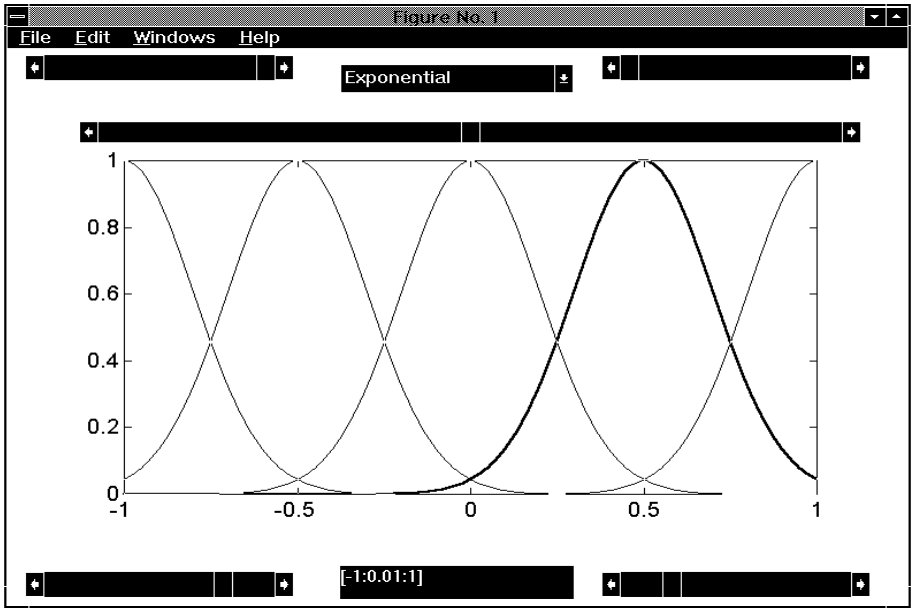


Figure 2: *Membership function editor window.*

for details), *Domain* – string defining the domain (universe of discourse) for the fuzzy sets, *CMF* – current membership function, and *Grades* – membership grades of the current input value. Property names are case insensitive. Only one property can be read at a time.

Example: The first of the two following command opens a new figure and initializes membership functions editor with 7 equidistant trapezoidal (default) functions defined on the domain `-1:0.1:1`. The second command reads the parameters of the membership functions.

```
mfset(1,'MFs',7,'Domain','-1:0.1:1')
mfget(1,'mfs')
```

See also:

`mgrade, mfequ`

Purpose:

Calculates membership grade(s) of the data into fuzzy set(s) defined either analytically or point-wise on a discrete domain.
Plots analytically defined membership functions.

Synopsis:

```
Ax = mgrade(x,mf)
Ax = mgrade(x,mf,dom)
mfplot(x,mf)
```

Description:

Function `mgrade(mf,x)` returns the membership grade (degree of belonginess) of value `x` into the fuzzy set with membership function defined in matrix `mf`. The size of this matrix is m x 5, where m is the number of membership functions. The first column determines the membership function type according to the following table:

0	One everywhere
1	Trapezoidal membership function
2	Exponential membership function
3	Sigmoidal membership function

The remaining four columns define the four breakpoints of the membership function ([left bottom, left shoulder, right shoulder, right bottom] = [p1, p2, p3, p4]). The degree of values less than left bottom and greater than right bottom is always zero. Membership degree of values between left and right shoulder is always one. The shape of the curve between the left bottom and left shoulder and right shoulder and right bottom is determined by the membership function type.

Trapezoidal membership function

$$f_t(x,a,b,c,d) = \begin{cases} 1 & \text{if } b \le x \le c \\ (x-a)/(b-a) & \text{if } a < x < b \\ 1-(x-c)/(d-c) & \text{if } c < x < d \\ 0 & \text{otherwise} \end{cases}$$

Exponential membership function

$$f_e(x,c_1,c_2,w_1,w_2) = \begin{cases} e^{-(\frac{x-c_1}{2w_1})^2} & \text{if } x < c_1 \\ e^{-(\frac{x-c_2}{2w_2})^2} & \text{if } x > c_2 \\ 1 & \text{otherwise} \end{cases}$$

Left shoulder c_1 and right shoulder c_2 are supplied directly, w_1, w_2 are calculated.

Sigmoidal membership function

$$f_s(x, c_1, c_2, w_1, w_2) = \begin{cases} 1 & \text{if } c_1 \le x \le c_2 \\ \frac{1}{2}\left(\frac{x-c_1+2w_1}{w_1}\right)^2 & \text{if } c_1 - 2w_1 < x < c_1 - w_1 \\ 1 - \frac{1}{2}\left(\frac{x-c_1}{w_1}\right)^2 & \text{if } c_1 - w_1 < x < c_1 \\ \frac{1}{2}\left(\frac{x-c_2+2w_2}{w_2}\right)^2 & \text{if } c_2 < x < c_2 + w_2 \\ 1 - \frac{1}{2}\left(\frac{x-c_2}{w_2}\right)^2 & \text{if } c_2 + w_2 < x < c_2 + 2w_2 \\ 0 & \text{otherwise} \end{cases}$$

Left shoulder c_1 and right shoulder c_2 are supplied directly, w_1, w_2 are calculated.

The membership degree is in the range from 0 to 1. This function is typically used for fuzzification, i.e. it converts crisp data into its fuzzy representation. If `mf` is a matrix with more than one row and `x` is a scalar, the result is a column vector, which elements represent the grade of membership of `x` into the fuzzy sets with membership functions defined in rows of `mf`. If `x` is a row vector, a membership grade matrix is returned.

When the third input argument `dom` is supplied, the function calculates the membership grade of value `x` into the fuzzy set `mf` defined point-wise on domain `dom`. Values which membership grades are not defined in `mf` are interpolated linearly.

Function `mfplot` plots the membership functions defined in matrix `mf` over the domain `dom`.

Purpose:

The complement of a fuzzy set. Alternative complement operators can be specified.

Synopsis:

```
C = not(A)
C = not(A,ctype)
C = not(A,ctype,p)
```

Description:

`not(A)` calculates the complement of a fuzzy set `A` using the following formula proposed by Zadeh:

$$Cx = 1 - Ax$$

`not(A,ctype)` calculates the complement of a fuzzy set `A` using the alternative complement operators. The desired operator is selected using the optional parameter `ctype`. Possible values of `ctype` are following:

`'za'`	Zadeh (default), see `not_za`
`'su'`	Sugeno, see `not_su`
`'ya'`	Yager, see `not_ya`

`not(A,ctype,p)` is the synopsis for the complements according to Sugeno and Yager, where `p` is a parameter, see `not_su` and `not_ya` for details.

Instead of using function `not` with a particular parameter, you can call directly the corresponding `not_` function, e.g. instead of `not(A,'su',p)` you can use `not_su(A,p)`, which is slightly faster and therefore recommended for speed-critical applications.

See also:

`not_za, not_su, not_ya`

Purpose:

The λ-complement of a fuzzy set according to Sugeno.
The complement of a fuzzy set according to Yager.
The complement of a fuzzy set according to Zadeh.

Synopsis:

```
C = not_su(A,lambda)
C = not_ya(A,w)
```

Description:

`not_su(A,lambda)` calculates the λ-complement of a fuzzy set `A` using the following formula proposed by Sugeno:

$$Cx = (1 - Ax)/(1 + lambda.Ax), \quad lambda \in [-1, \infty)$$

`not_ya(A,w)` calculates the complement of a fuzzy set `A` using the following formula proposed by Yager:

$$Cx = (1 - Ax^w)^{\frac{1}{w}}, \quad w \in (0, \infty)$$

`not_za(A)` calculates the complement of a fuzzy set `A` using the following formula proposed by Zadeh:

$$Cx = 1 - Ax$$

See also:

`not`

Purpose:

The union of two fuzzy sets. Alternative union operators can be specified.

Synopsis:

```
C = or(A,B)
C = or(A,B,utype)
C = or(A,B,utype,r)
```

Description:

`or(A,B)` calculates the union of two fuzzy sets `A` and `B` using the following formula proposed by Zadeh:

$$Cx = \max(Ax, Bx)$$

`or(A,B,utype)` calculates the union of two fuzzy sets `A` and `B` using the alternative union operators. The desired operator is selected using the optional parameter `utype`. Possible values of `utype` are following:

`'za'`	Zadeh (default), see `or_za`
`'lu'`	Lukasiewicz, see `or_lu`
`'pr'`	probability, see `or_pr`
`'re'`	restricted, see `or_re`

`or(A,B,utype,r)` is the synopsis for the restricted union, where `r` is a parameter, see `or_re` for details.

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`.

Instead of using function `or` with a particular parameter, you can call directly the corresponding `or_` function, e.g. instead of `or(A,B,'lu')` you can use `or_lu(A,B)`, which is slightly faster and therefore recommended for speed-critical applications.

See also:

`or_za, or_lu, or_re, or_pr`

Purpose:

The union of two fuzzy sets according to Lukasiewicz.
The probabilistic union of two fuzzy sets.
The union of two fuzzy sets according to Zadeh.

Synopsis:

```
C = or_lu(A,B)
C = or_pr(A,B)
C = or_za(A,B)
```

Description:

`or_lu(A,B)` calculates the union of two fuzzy sets `A` and `B` using the following formula proposed by Lukasiewicz:

$$Cx = \min(Ax + Bx, 1)$$

`or_pr(A,B)` calculates the probabilistic union of two fuzzy sets A and B using the following formula from the probability theory:

$$Cx = Ax + Bx - Ax.Bx$$

`or_za(A,B)` calculates the union of two fuzzy sets `A` and `B` using the following formula proposed by Zadeh:

$$Cx = \max(Ax, Bx)$$

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`. If only one input matrix `A` is supplied, the union of all rows in `A` is calculated.

See also:

`or, or_re`

Purpose:

The restricted union of two fuzzy sets.

Synopsis:

```
C = or_re(A,B,r)
```

Description:

`or_re(A,B,r)` calculates the restricted union of two fuzzy sets A and B using the following formulas:

$$\text{if } r \geq 0 \ldots \quad Cx = Ax + Bx - Ax.Bx + r.(\max(Ax, Bx) - (Ax + Bx - Ax.Bx))$$
$$\text{if } r < 0 \ldots \quad Cx = Ax + Bx - Ax.Bx + r.(Ax + Bx - Ax.Bx - \min(1, Ax + Bx))$$

Parameter r equals to 1 when A and B are strongly positively associated, to -1 when A and B are strongly negatively associated and to 0 when A and B are independent.

If A and B are both row or column vectors, then C is also a row or column vector. If A is a column vector and B a row vector, then the matrix C is calculated on the Cartesian product universe A x B. If only one input matrix A is supplied, the union of all rows in A is calculated.

See also:

`or, or_za, or_lu, or_pr`

Purpose:

Linguistic hedge plus.
Linguistic hedge minus.

Synopsis:

```
B = plus(A)
B = minus(A)
```

Description:

`plus(A)` is the hedge *plus* used as a modifier of the meaning of `A`. It is defined as:

$$Bx = [Ax]^{1.25}$$

`minus(A)` is the hedge `minus` used as a modifier of the meaning of `A`. It is defined as:

$$Bx = [Ax]^{0.75}$$

See also:

`very, moreless, highly, rather, slightly, roughly`

Purpose:

Linguistic hedge rather.
Linguistic hedge slightly.

Synopsis:

```
B = rather(A)
B = slightly(A)
```

Description:

`rather(A)` is the hedge *rather* used as a modifier of the meaning of A. It is defined as:

$$B = \text{int}(\text{conc}(A))$$

`slightly(A)` is the hedge *slightly* used as a modifier of the meaning of A. It is defined as:

$$B = \text{int}(\text{normal}(\text{min}(\text{plus}(A),\text{not}(\text{very}(A)))))$$

See also:

`very, moreless, plus, minus, highly, roughly`

Purpose:

Converts the rules from the matrix form to the text form.

Synopsis:

```
txt = rls2txt(rls,mfs,inputs,outputs,nlab,labels,filename)
```

Description:

Function `rls2txt` converts the rules from the matrix form to the text form. Input arguments `rls` to `labels` can be obtained as the output of `txt2rls`. For their description, see `txt2rls`. Optional input argument `filename` is a text string that specifies a name of a file where the rule-base will be written. Output argument `txt` is a text matrix containing the rule-base. In certain sense, `rls2txt` is an inverse function to `txt2rls`, i.e. after

```
[rls,mfs,inputs,outputs,nlab,labels] = txt2rls('rbase1.rb');
rls2txt(rls,mfs,inputs,outputs,nlab,labels,'rbase2.rb');
```

the rule-base file *rbase2.rb* contains the same rule-base as *rbase1.rb*, except comments and formatting.

See also:

`txt2rls, mfset, mfget`

Purpose:

The general triangular co-norm function. The type (family) can be specified by a parameter.

Synopsis:

```
S = snorm(A,B,p,family)
```

Description:

`snorm(A,B,p)` calculates the disjunction of two fuzzy sets `A` and `B` using triangular co-norm proposed by Dubois and Prade:

$$S(x,y) = \frac{x + y - xy - \min(x, y, 1-\alpha)}{\max(1-x, 1-y, \alpha)} \quad p \in (0,1)$$

`snorm(A,B,p,family)` calculates the disjunction of two fuzzy sets `A` and `B` using the triangular co-norm specified in parameter `family`. Possible values of `family` are the following:

`'du'`	Dubois and Prade (default), see `s_du`
`'do'`	Dombi, see `s_do`
`'la'`	λ, see `s_la`
`'ya'`	Yager, see `s_ya`
`'fr'`	Frank, see `s_fr`
`'ha'`	Hamacher, see `s_ha`
`'sk'`	Schweizer and Sklar, see `s_sk`

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`.

Instead of using function `snorm` with a particular parameter, you can call directly the corresponding `s_`function, e.g. instead of `snorm(A,B,0.5,'ha')` you can use `s_ha(A,B,0.5)`, which is slightly faster and therefore recommended for speed-critical applications.

See also:

`s_du, s_do, s_la, s_ya, s_fr, s_ha, s_sk`

Purpose:

Least squares optimization of the linear consequents in Sugeno–Takagi's model. Sugeno–Takagi's inference.

Synopsis:

```
[p,ym,yl,ylm] = suglms(x,y,f,def)
[ym,yl,ylm] = sugval(p,x,f,def)
```

Description:

Function `suglms` calculates a least-mean-squares solution for the consequent parameters in the Sugeno-Takagi's model. `x` is the input data matrix, `y` the output data vector, and `f` the matrix of degrees of fulfillment of all data points (rows in `x`). `f` can be obtained e.g. as an output of `dofmin`. An optional argument `def` is a scalar defining a default value returned when the sum of grades equals to one (defaults to 0). The function returns the consequents parameters `p` for each rule (i.e. column of `f`), model output `ym` for the given input data `x`, local output of each rule `yl` and local output of each rule `ylm` with data corresponding to degrees less than 0.2 masked with NaN's (useful for plots).

Function `sugval` calculates the output of the Sugeno–Takagi's model. `p` is the consequent parameter matrix, obtained with e.g. `suglms`, `x` is the input data matrix, `f` the matrix of degrees of fulfillment of all data points (rows in `x`). `f` can be obtained e.g. as an output of `dofmin`. An optional argument `def` is a scalar defining a default value returned when the sum of grades equals to one (defaults to 0). The function returns the model output `ym` for the given input data `x`, local output of each rule `yl` and local output of each rule `ylm` with data corresponding to degrees less than 0.2 masked with NaN's (useful for plots).

Example:

```
x = (0:0.02:1)'; y = sin(7*x);
f = mgrade(x',mfequ(x,2,3))';
[p,ym,yl,ylm] = suglms([x ones(size(x))],y,f);
subplot(211); plot(x,ylm,'.',x,[y ym]); title('Fitting y = sin(7*x)
subplot(212); plot(x,f); title('Membership functions')
```

See also:

`dofmin, dofprod, infer`

Purpose:

The support of a fuzzy set.
The kernel of a fuzzy set.

Synopsis:

```
[Xs,As] = supp(X,A)
[Xk,B] = ker(X,A)
```

Description:

`supp(X,A)` calculates the support of a fuzzy set `A` on a universe `X`. The support of a fuzzy set A is an ordinary subset of `X`

$$\text{Supp } A = \{x; Ax > 0\}$$

Output arguments are the following. `Xs` is a vector containing the support elements and `As` contains the membership grades of `X` elements.

`ker(X,A)` calculates the kernel of a fuzzy set `A` on a universe `X`. The kernel of a fuzzy set A is an ordinary subset of `X`

$$\text{Ker } A = \{x; Ax = 1\}$$

Output arguments are the following. `Xk` is a vector containing the kernel elements and `B` is a fuzzy set defined on `X`, with membership grades equal to one for the kernel elements and zero otherwise. Note that `ker(X,A)` is the same as `acut(X,A,1)`. A fuzzy set A is *normal* when Ker $A \neq \emptyset$ otherwise it is *subnormal*.

See also:

`acut, acuts, alevel, crossovr, hgt, normal, card`

Purpose:

Sup-min composition of fuzzy relations.
Sup-product composition of fuzzy relations.

Synopsis:

```
R = sup_min(S,T)
R = sup_prd(S,T)
```

Description:

`sup_min(S,T)` is the sup-min composition of fuzzy relations `S` and `T`. The sup-min composition of finite fuzzy relations can be viewed as a matrix product, where $\sum$ is in fact the max operation and product is the min operation.

`sup_prod(S,T)` is the sup-product composition of fuzzy relations `S` and `T`. The sup-product composition of finite fuzzy relations can be viewed as a matrix product, where $\sum$ is in fact the max operation and product remains product.

See also:

`sup_mean, inf_max`

Purpose:

The general triangular norm function. The type (family) can be specified by a parameter.

Synopsis:

```
T = tnorm(A,B,p,family)
```

Description:

`tnorm(A,B,p)` calculates the conjunction of two fuzzy sets `A` and `B` using triangular norm proposed by Dubois and Prade:

$$T(x,y) = \frac{xy}{\max(x,y,p)} \qquad p \in (0,1)$$

`tnorm(A,B,p,family)` calculates the conjunction of two fuzzy sets `A` and `B` using the triangular norm specified in parameter `family`. Possible values of `family` are the following:

`'du'`	Dubois and Prade (default), see `t_du`
`'do'`	Dombi, see `t_do`
`'la'`	λ, see `t_la`
`'ya'`	Yager, see `t_ya`
`'fr'`	Frank, see `t_fr`
`'ha'`	Hamacher, see `t_ha`
`'sk'`	Schweizer and Sklar, see `t_sk`

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`.

Instead of using function `tnorm` with a particular parameter, you can call directly the corresponding `t_`function, e.g. instead of `tnorm(A,B,0.5,'ha')` you can use `t_ha(A,B,0.5)`, which is slightly faster and therefore recommended for speed-critical applications.

See also:

`t_du, t_do, t_la, t_ya, t_fr, t_ha, t_sk`

Purpose:

Converts the rules from the text form into the coded matrix form.

Synopsis:

```
rls = txt2rls(filename)
[rls, mfs, inputs, outputs, nlab, labels] = txt2rls(filename)
```

Description:

Function `txt2rls` converts the rules from the text form to the matrix form. Input argument `filename` is a string that specifies a name of the text file containing the rule-base. Output argument `rls` contains the rule-base matrix, `mfs` the membership functions matrix, `inputs` is a text matrix containing the input labels, `outputs` is a text matrix containing the output labels, vector `nlab` specifies the number of labels for all inputs and outputs, `labels` contains the input and output labels concatenated in one text matrix.

In the rule-base, input and output identifiers and fuzzy set labels for each input and output must be defined. If-Then rules can contain logical operators AND and NOT (NOT is applicable only in rule premises). For a sample rule-base see the SAMPLE.RB file.

Rule-base items are:

directives	–	#input, #output, #rules, #mf (optional)
keywords	–	If, Then, Is, And, Not, comma (,)
rule terminator	–	;
input and output names	–	user-defined identifiers
fuzzy set labels	–	user-defined identifiers
rule weights	–	real numbers
membership functions definitions	–	five-tuples of real numbers
singletons	–	alternatively, rule consequents can be defined as singletons.

Identifiers (input and output names and fuzzy set labels) can consist of multiple words (e.g. error change) but cannot contain keywords, directives and a semicolon. Identifiers are case-sensitive, keywords and directives case-insensitive. Input and output names follow immediately after their respective directives. Each fuzzy set label must be written on a separate line, i.e.:

```
#input error change
       Big Positive
       Medium Positive
       ...
```

A rule is a text fragment beginning with "If" and ending with a semicolon (;). A rule must begin on a new line and can span multiple lines. Every rule can have a weight which is a real number preceding the rule on the same or separate line. Example:

```
0.85 IF error is POSITIVE BIG and output is POSITIVE
    THEN control is NEGATIVE BIG;
```

Membership functions definitions can be placed anywhere in the text. Every membership function is defined by a five-tuple [type a b c d]. For details on types and parameters see `mgrade`. Example:

```
#input error change
       Big Positive      #mf 1 0.0 0.8 1.0 1.0
       Medium Positive  #mf 1 0.0 0.4 0.5 0.9
```

Any text following the ’of membership functions, input, output definitions and rules is arbitrary.

See also:

`rls2txt, infer, mfset, mfget`

Purpose:

The conjunction of two fuzzy sets with triangular norm according to Dombi.
The disjunction of two fuzzy sets with triangular co-norm according to Dombi.

Synopsis:

```
T = t_do(A,B,lambda)
S = s_do(A,B,lambda)
```

Description:

`t_do(A,B,lambda)` calculates the conjunction of two fuzzy sets `A` and `B` using the triangular norm proposed by Dombi:

$$T(x,y) = \frac{1}{1+[(\frac{1}{x}-1)^{-\lambda}+(\frac{1}{y}-1)^{-\lambda}]^{\frac{1}{\lambda}}} \qquad p \in [0,\infty)$$

`s_do(A,B,lambda)` calculates the disjunction of two fuzzy sets `A` and `B` using the triangular co-norm proposed by Dombi:

$$S(x,y) = \frac{1}{1+[(\frac{1}{x}-1)^{\lambda}+(\frac{1}{y}-1)^{\lambda}]^{\frac{1}{\lambda}}} \qquad p \in [0,\infty)$$

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`. If only one input matrix `A` is supplied, the t-(co)norm of all rows in `A` is calculated.

See also:

`tnorm, snorm, t_du, s_du, t_la, s_la, t_ya, s_ya, t_fr, s_fr, t_ha, s_ha, t_sk, s_sk`

Purpose:

The conjunction of two fuzzy sets with triangular norm according to Dubois and Prade.
The disjunction of two fuzzy sets with triangular co-norm according to Dubois and Prade.

Synopsis:

```
T = t_du(A,B,alpha)
S = s_du(A,B,alpha)
```

Description:

`t_du(A,B,alpha)` calculates the conjunction of two fuzzy sets A and B using the triangular norm proposed by Dubois and Prade:

$$T(x,y) = \frac{xy}{\max(x,y,\alpha)} \qquad p \in (0,1)$$

`s_du(A,B,alpha)` calculates the disjunction of two fuzzy sets A and B using the triangular co-norm proposed by Dubois and Prade:

$$S(x,y) = \frac{x+y-xy-\min(x,y,1-\alpha)}{\max(1-x,1-y,\alpha)} \qquad p \in (0,1)$$

If A and B are both row or column vectors, then C is also a row or column vector. If A is a column vector and B a row vector, then the matrix C is calculated on the Cartesian product universe A x B. If only one input matrix A is supplied, the t-(co)norm of all rows in A is calculated.

See also:

`tnorm, snorm, t_do, s_do, t_la, s_la, t_ya, s_ya, t_fr, s_fr, t_ha, s_ha, t_sk, s_sk`

Purpose:

The conjunction of two fuzzy sets with triangular norm according to Frank.
The disjunction of two fuzzy sets with triangular co-norm according to Frank.

Synopsis:

```
T = t_fr(A,B,w)
S = s_fr(A,B,w)
```

Description:

`t_fr(A,B,w)` calculates the conjunction of two fuzzy sets `A` and `B` using the triangular norm proposed by Frank:

$$T(x,y) = \log_w[1 + \frac{(w^x - 1)(w^y - 1)}{w - 1}] \quad w \in (0,1),\ w \neq 1$$

`s_fr(A,B,w)` calculates the disjunction of two fuzzy sets `A` and `B` using the triangular co-norm proposed by Frank:

$$S(x,y) = 1 - \log_w[1 + \frac{(w^{1-x} - 1)(w^{1-y} - 1)}{w - 1}] \quad w \in (0,1),\ w \neq 1$$

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`. If only one input matrix `A` is supplied, the t-(co)norm of all rows in `A` is calculated.

See also:

`tnorm, snorm, t_do, s_do, t_la, s_la, t_ya, s_ya, t_du, s_du, t_ha, s_ha, t_sk, s_sk`

Purpose:

The conjunction of two fuzzy sets with triangular norm according to Hamacher.
The disjunction of two fuzzy sets with triangular co-norm according to Hamacher.

Synopsis:

```
T = t_ha(A,B,gamma)
S = s_ha(A,B,gamma)
```

Description:

`t_ha(A,B,gamma)` calculates the conjunction of two fuzzy sets A and B using the triangular norm proposed by Hamacher:

$$T(x,y) = \frac{x + y - (2-\gamma)xy}{1-(1-\gamma)xy} \qquad \gamma \in (0,\infty)$$

`s_ha(A,B,gamma)` calculates the disjunction of two fuzzy sets A and B using the triangular co-norm proposed by Hamacher:

$$S(x,y) = \frac{xy}{\gamma + (1-\gamma)(x+y-xy)} \qquad \gamma \in (0,\infty)$$

If A and B are both row or column vectors, then C is also a row or column vector. If A is a column vector and B a row vector, then the matrix C is calculated on the Cartesian product universe A x B. If only one input matrix A is supplied, the t-(co)norm of all rows in A is calculated.

See also:

`tnorm, snorm, t_do, s_do, t_la, s_la, t_ya, s_ya, t_du, s_du, t_fr, s_fr, t_sk, s_sk`

Purpose:

The conjunction of two fuzzy sets with λ triangular norm.
The disjunction of two fuzzy sets with λ triangular co-norm.

Synopsis:

```
T = t_la(A,B,lambda)
S = s_la(A,B,lambda)
```

Description:

`t_la(A,B,lambda)` calculates the conjunction of two fuzzy sets `A` and `B` using the λ triangular norm.

$$T(x,y) = \max(0,(\lambda+1)(x+y-1)-\lambda xy) \quad \lambda \in [-1,\infty)$$

`s_la(A,B,lambda)` calculates the disjunction of two fuzzy sets `A` and `B` using the λ triangular co-norm.

$$S(x,y) = \min(1,x+y+xy) \quad \lambda \in [-1,\infty)$$

Note: For $\lambda = -1$ we get the triangular norm and co-norm according to Frank. For $\lambda = 0$ we get the triangular norm and co-norm according to Yager.

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`. If only one input matrix `A` is supplied, the t-(co)norm of all rows in `A` is calculated.

See also:

`tnorm, snorm, t_do, s_do, t_fr, s_fr, t_ya, s_ya, t_du, s_du, t_ha, s_ha, t_sk, s_sk`

Purpose:

The conjunction of two fuzzy sets with triangular norm according to Schweizer and Sklar.
The disjunction of two fuzzy sets with triangular co-norm according to Schweizer and Sklar.

Synopsis:

```
T = t_sk(A,B,p)
S = s_sk(A,B,p)
```

Description:

`t_sk(A,B,p)` calculates the conjunction of two fuzzy sets `A` and `B` using the triangular norm proposed by Schweizer and Sklar:

$$T(x,y) = 1 - [(1-x)^p + (1-y)^p - (1-x)^p(1-y)^p]^{\frac{1}{p}} \quad p \in (0,\infty)$$

`s_sk(A,B,p)` calculates the disjunction of two fuzzy sets `A` and `B` using the triangular co-norm proposed by Schweizer and Sklar:

$$S(x,y) = [x^p + y^p - x^p y^p]^{\frac{1}{p}} \quad p \in (0,\infty)$$

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`. If only one input matrix `A` is supplied, the t-(co)norm of all rows in `A` is calculated.

See also:

`tnorm, snorm, t_do, s_do, t_la, s_la, t_ya, s_ya, t_du, s_du, t_fr, s_fr, t_ha, s_ha`

Purpose:

The conjunction of two fuzzy sets with triangular norm according to Yager.
The disjunction of two fuzzy sets with triangular co-norm according to Yager.

Synopsis:

```
T = t_ya(A,B,p)
S = s_ya(A,B,p)
```

Description:

`t_ya(A,B,p)` calculates the conjunction of two fuzzy sets `A` and `B` using the triangular norm proposed by Yager:

$$T(x,y) = \begin{cases} 1 - \min(1, [(1-x)^p + (1+y)^p]^{\frac{1}{p}}) & \text{if } (1-x)^p + (1-y)^p \geq 1, \quad p \geq 1 \\ 1 - [(1-x)^p + (1+y)^p]^{\frac{1}{p}} & \text{otherwise}, \quad p \geq 1 \end{cases}$$

`s_ya(A,B,p)` calculates the disjunction of two fuzzy sets `A` and `B` using the triangular co-norm proposed by Yager:

$$S(x,y) = \begin{cases} \min(1, [x^p + y^p]^{\frac{1}{p}}) & \text{if } x^p + y^p \geq 1, \quad p \geq 1 \\ (x^p + y^p)^{\frac{1}{p}} & \text{otherwise}, \quad p \geq 1 \end{cases}$$

For $p = 1$ we get bold intersection and bold union respectively.

If `A` and `B` are both row or column vectors, then `C` is also a row or column vector. If `A` is a column vector and `B` a row vector, then the matrix `C` is calculated on the Cartesian product universe `A` x `B`. If only one input matrix `A` is supplied, the t-(co)norm of all rows in `A` is calculated.

See also:

`tnorm, snorm, t_do, s_do, t_la, s_la, t_ha, s_ha, t_du, s_du, t_fr, s_fr, t_sk, s_sk`

Purpose:

Linguistic hedge very.
Linguistic hedge more or less.

Synopsis:

```
B = very(A)
B = moreless(A)
```

Description:

`very(A)` is the hedge *very* used as a modifier of the meaning of A. It is defined as:

$$B = \text{conc}(A)$$

`moreless(A)` is the hedge *more or less* used as a modifier of the meaning of A. It is defined as:

$$B = \text{dil}(A)$$

See also:

`plus, minus, highly, rather, slightly, roughly`

5 Fuzzy Control Library

The Fuzzy Control library for SIMULINK consists of the building blocks for simulations of dynamical systems with fuzzy controllers. It includes three groups of blocks:

1. Elementary blocks for FLC design (fuzzifier, premise evaluation block, implication, aggregation and defuzzification), with the possibility of combining various operators for logical connectives, implication, aggregation and defuzzification. Every block allows for on-line plots of intermediate results (e.g. degrees of fulfillment of the rules, fuzzy sets, etc.).
2. Fuzzy controller blocks integrating the above functions in one, useful for more complicated and speed-demanding applications.
3. Utility blocks – bar graph scope, fuzzy set scope, enhanced graph scope and a pause block that allows for single-stepping the simulation.

Each block is based on an S-function, implemented in an M-file. The functions used for particular operations (i.e. implication, aggregation, defuzzification, etc.) are defined by the user in the block dialogue box. Because S-functions are implemented in MATLAB language in M-files users can easily modify them if needed, or use them as templates for their own functions. The library relies on Handle Graphics of MATLAB 4.0 and therefore it is not compatible with previous MATLAB versions. On the other hand, no machine-dependent code is used, which ensures full portability over all platforms where MATLAB 4.0 runs.

Building of fuzzy controllers with the library block is very easy and straightforward. SIMULINK capabilities allow users to evaluate the performance of FLC's for various dynamical systems in presence of different kinds of disturbances, etc. The demonstration examples illustrate the basic concepts of fuzzy control and the usage of Fuzzy control library blocks.

The following table lists all the library blocks with a short description of their function. In the text, each block is described in more detail with all the parameter options, etc. On-line help on the block functions can be obtained by pressing the Help button in the dialogue box, for the information on the S-function parameters invoke the MATLAB help command with the particular function name from the MATLAB workspace. For the very details look at the function itself.

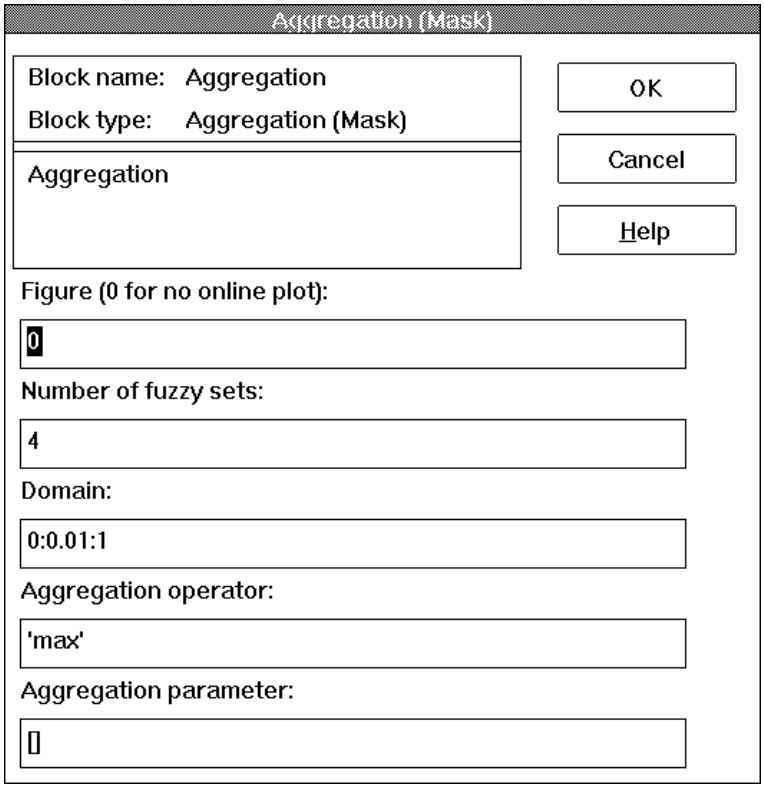
Fuzzy Control Blocks	
`Aggregation`	Aggregation of rules (consequent fuzzy sets)
`Defuzzifier`	Defuzzification
`DOF`	Calculates degrees of fulfillment of rules
`Infer`	Fuzzy inference block
`Fuzzifier`	Fuzzification of a crisp or fuzzy input
`Implication`	Fuzzy implication
`Sugeno`	Sugeno's fuzzy inference

Utility Blocks	
`B-Scope`	Bar graph scope
`F-Scope`	Fuzzy set scope
`G-Scope`	Enhanced graphical scope
`Pause`	Pauses simulation for n seconds

Purpose:

Aggregates the fuzzy sets in rule consequents (obtained after applying the implication operator) in one fuzzy set.

Description:



Parameter **Figure** specifies a figure (graph window) where the on-line visualization of the aggregation process is directed. **Figure** must be a valid figure handle (i.e. an integer number). Zero (0) switches the visualization off. On the block initialization, **Figure** is automatically opened (if it has not been already opened) and cleared (`clf`). All figures with numbers less then **Figure** are opened automatically as well.

The **Number of fuzzy sets** is the number of fuzzy sets entering the block. The total length of the input vector is **Number of fuzzy sets** times the length of **Domain**.

Domain is the common domain (universe of discourse) of the fuzzy sets.

Aggregation operator is a text matrix that specifies the name of a function implementing the aggregation e.g. `max`). The synopsis of the aggregation function must be the same as

synopsis of MATLAB function `max`. Note that different types of implications use different aggregation functions. For instance Mamdani's MIN implication operator requires MAX aggregation and for example Lukasiewicz implication uses MIN as the aggregation operator.

Parameter is an additional parameter passed to the aggregation function. If an empty matrix ([]) is specified, no additional parameter is passed to the function.

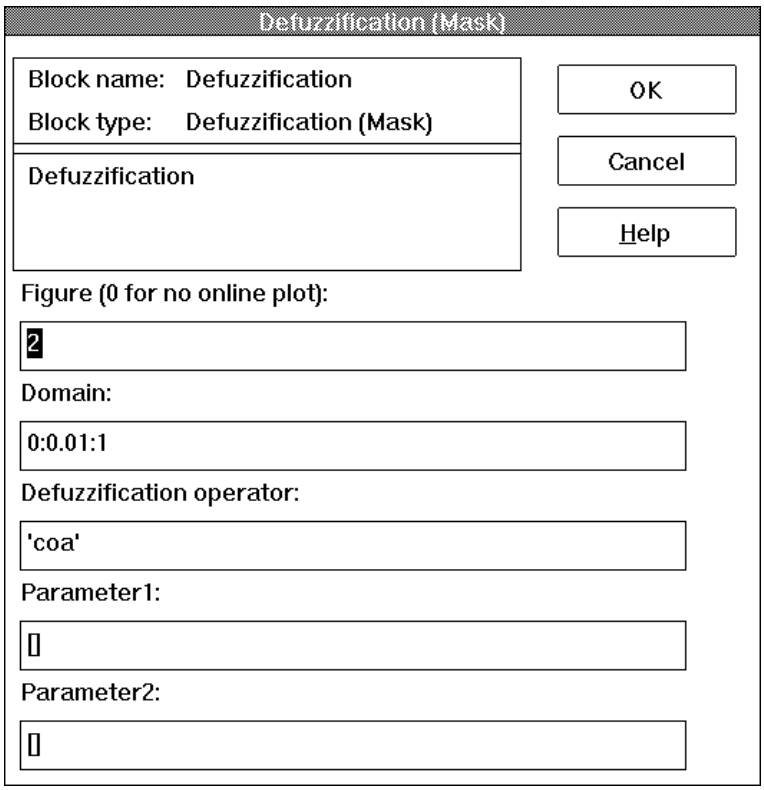
Notes

library:	Fuzzylib
direct feedthrough:	yes
based on S-function:	`sfunagr`

Purpose:

Performs the defuzzification of a fuzzy set. The particular defuzzification method is defined in the dialogue box. Allows for on-line plotting of the defuzzification process.

Description:



Parameter **Figure** specifies a figure (graph window) where the on-line visualization of the defuzzification process is directed. **Figure** must be a valid figure handle (i.e. an integer number). Zero (0) switches the visualization off. On the block initialization, **Figure** is automatically opened (if it has not been already opened) and cleared (`clf`). All figures with numbers less then **Figure** are opened automatically as well.

Domain vector defines the discretization of the universe. This vector is used in the defuzzification procedure (for instance for the calculation of the center of area of the resulting fuzzy set). If for example the "Fuzzy Mean" defuzzification technique (i.e. center of area on singletons in output fuzzy sets) is used, the **Domain** vector elements define the positions of these singletons.

Defuzzification operator is a text matrix that contains name of a function implementing the defuzzification. The syntax of this function must be the same as the syntax of for instance

functions `coa` or `mom` that implement the center of area and mean of maxima defuzzification methods respectively. If an empty matrix ([]) is specified, default center of area method is used. The possibility of specifying the defuzzification method in the dialogue box allows you to compare different defuzzification techniques and experiment with your own algorithms.

Parameter1 and `Parameter2` are additional parameters passed to the defuzzification routine. It can be for instance the weight vector for weighted fuzzy mean technique or an index for indexed center of area defuzzification. If an empty matrix ([]) is specified, no additional parameters are passed to the defuzzification function. `Parameter2` can have a special value 'singleton' which is used to get better plots when output singletons are involved.

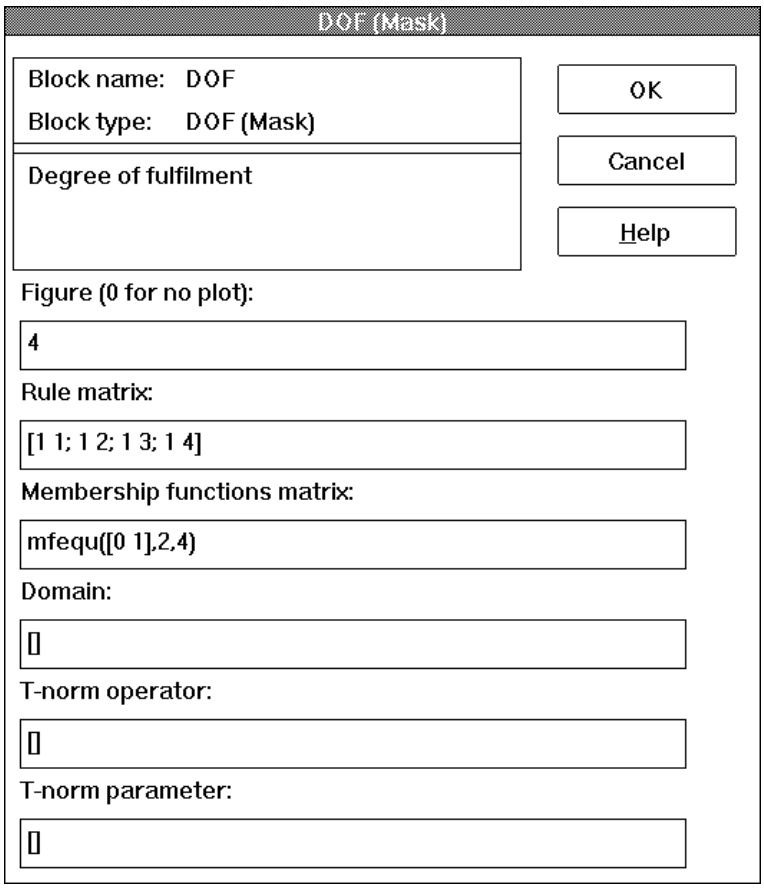
Notes

library:	Fuzzylib
direct feedthrough:	yes
based on S-function:	`sfundef`

Purpose:

For the input vector, calculates the degrees of fulfillment (DOF's) of the rules specified in the block parameter. `DOF` is the `doftn` function encapsulated in a SIMULINK S-function block. See `doftn` for details.

Description:



Parameter **Figure** specifies a figure (graph window) where the graphical visualization of the degrees of fulfillment is directed in the form of a bar graph. **Figure** must be a valid figure handle (i.e. an integer number). Zero (0) switches the visualization off. On the block initialization, **Figure** is automatically opened (if it has not been already opened) and cleared (`clf`). All figures with numbers less then **Figure** are opened automatically as well.

Rule matrix defines the antecedent part of the rules as codes of membership functions given in the **Membership functions matrix**. See `infer` and `txt2rls` for details.

Membership functions matrix defines the membership functions parameters. See `mgrade` for details.

Domain specifies the domain for point-wise defined membership functions. See `mgrade` for details.

T-norm operator is a text matrix that specifies the T-norm function used as the logical AND connective. The synopsis of this function must be the same as the synopsis of the MATLAB function `min`.

T-norm parameter is an additional parameter passed to the T-norm function. If an empty matrix ([]) is specified, no additional parameter is passed to the function.

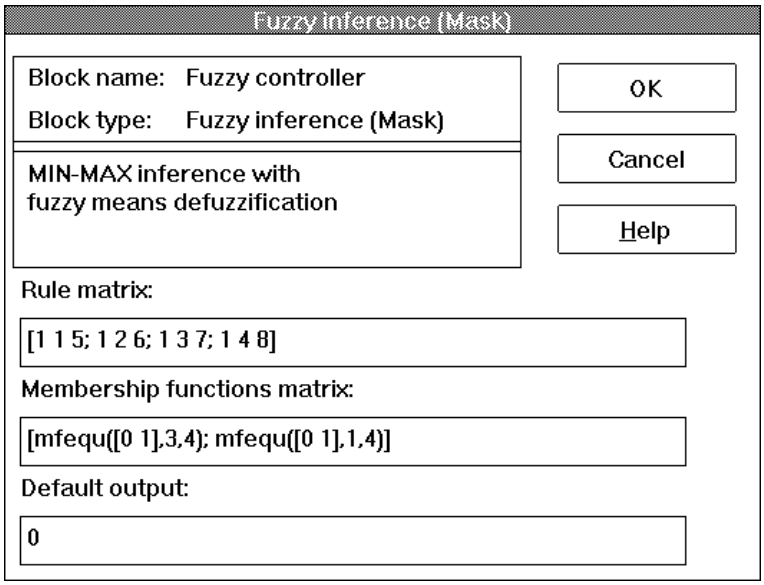
Notes

library:	Fuzzylib
direct feedthrough:	yes
based on S-function:	`sfundof`

Purpose:

Fuzzy logic controller block encapsulates the `infer` function. For a crisp input vector, it calculates a crisp output, using the specified rule-base and membership functions definition. MIN-MAX inference and fuzzy mean defuzzification are used. See `infer` for details.

Description:



Rule matrix defines the rules as codes of membership functions given in the **Membership functions matrix**. See `infer` and `txt2rls` for details.

Membership functions matrix defines the membership functions parameters. See `mgrade` for details.

Default output is a vector of the same dimension of the output vector, containing the default values assigned to each output when no rule is applicable. As opposed to the `infer` function, in this SIMULINK block the **Default output** specification is compulsory, e.g. is cannot be skipped or defined as an empty matrix. The reason is that on the block initialization, the number of inputs must be known. From the rule matrix, only the total number of inputs and outputs is known, thus the number of outputs must be defined through the **Default output vector**.

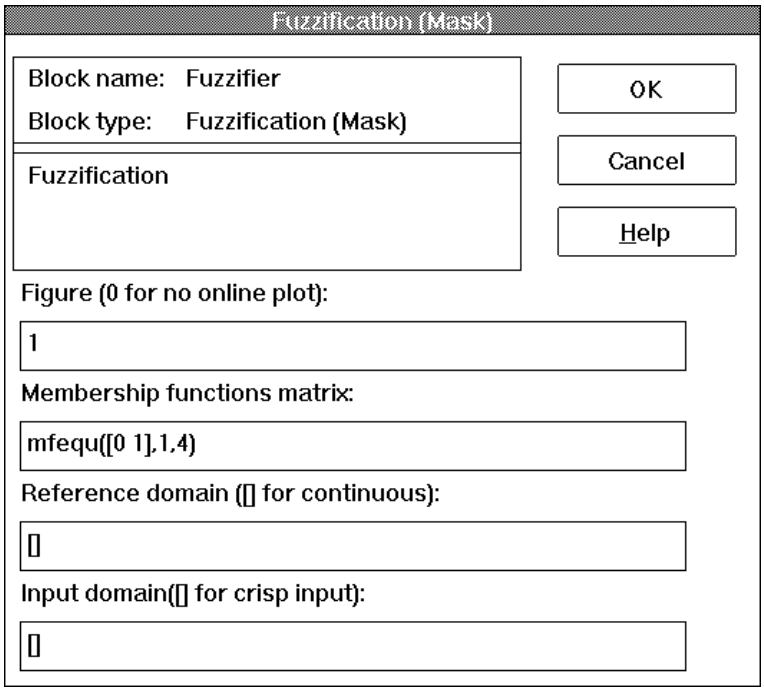
Notes

library:	Fuzzylib
direct feedthrough:	yes
based on S-function:	`sfuninf`

Purpose:

Converts crisp (real) values in corresponding fuzzy representation. In addition, fuzzy sets can be converted into the representation in terms of membership grades into the reference fuzzy sets (so called fuzzy discretization). Reference fuzzy sets are defined on a continuous domain using an analytical description of their membership functions.

Description:



The Fuzzifier block can be used for two purposes. First, it converts an input value (scalar) into its fuzzy representation, i.e. a vector of membership grades of this value into the defined reference fuzzy sets. Second, instead of a crisp value, a fuzzy set may enter this block. As a result, this fuzzy set is expressed in terms of the membership grades into the reference fuzzy sets, using the following formula.

$$\mu_{A_{r_i}}(A) = \sup \min(\mu_{A_{r_i}}(x), \mu_A(x))$$

where A_{r_i} is the i-th reference fuzzy set and A is the input fuzzy set. For this purpose the fuzzy sets are discretized over the **Domain** vector x (see below).

Parameter **Figure** specifies a figure (graph window) where the on-line visualization of the fuzzification process is directed. **Figure** must be a valid figure handle (i.e. an integer number). Zero (0) switches the visualization off. On the block initialization, **Figure** is

automatically opened (if it has not been already opened) and cleared (`clf`). All figures with numbers less then **Figure** are opened automatically as well.

Membership functions matrix defines the membership functions parameters. See `mgrade` for details.

Reference domain vector defines the discretization of the universe of discourse for reference fuzzy sets defined point-wise. When analytical membership functions are used, set to an empty matrix.

Input domain vector defines the discretization of the universe of discourse for fuzzy input. If the input domain is different from an empty matrix, fuzzy input to the block is expected (of the same width as is the width of the domain). When crisp input is used, set to an empty matrix.

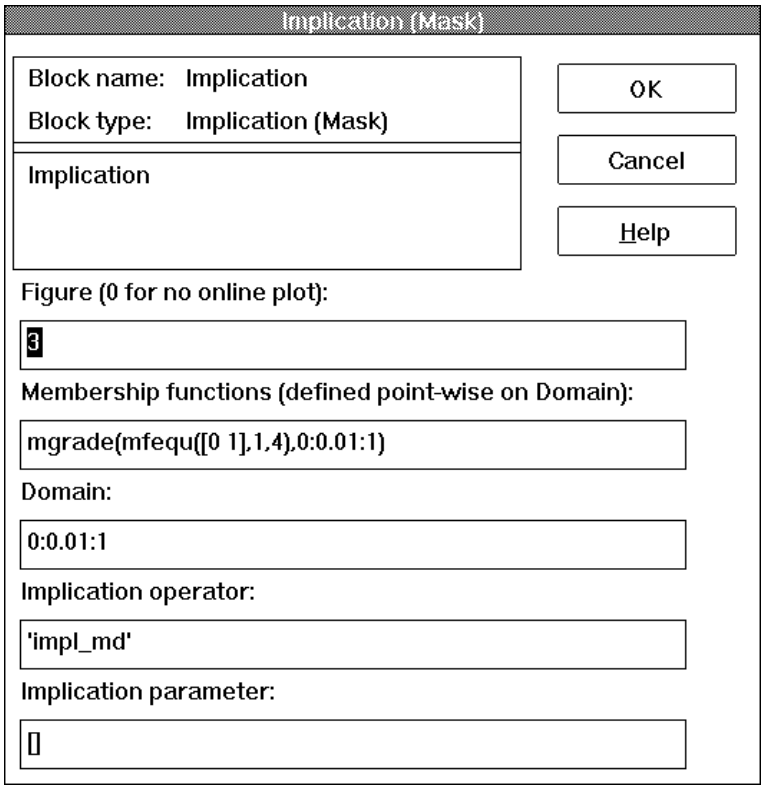
Notes

library:	Fuzzylib
direct feedthrough:	yes
based on S-function:	`sfunfuzz`

Purpose:

Performs fuzzy implication. Implication function may be specified in the dialogue box.

Description:



Parameter **Figure** specifies a figure (graph window) where the on-line visualization of the implication results is directed. **Figure** must be a valid figure handle (i.e. an integer number). Zero (0) switches the visualization off. On the block initialization, **Figure** is automatically opened (if it has not been already opened) and cleared (`clf`). All figures with numbers less then **Figure** are opened automatically as well.

Matrix **Output fuzzy sets** contains in rows point-wise defined membership functions of the consequent fuzzy sets.

Domain vector defines the discretization of the consequent universe. Fuzzy implication is calculated point-wise over this domain.

Implication operator is a text matrix which specifies the name of a function implementing the fuzzy implication (e.g. `min` or `impl_lu`). The synopsis of the implication function must

be the same as synopsis of MATLAB function `min`.

Parameter is an additional parameter passed to the implication function. If an empty matrix ([]) is specified, no additional parameter is passed to the function.

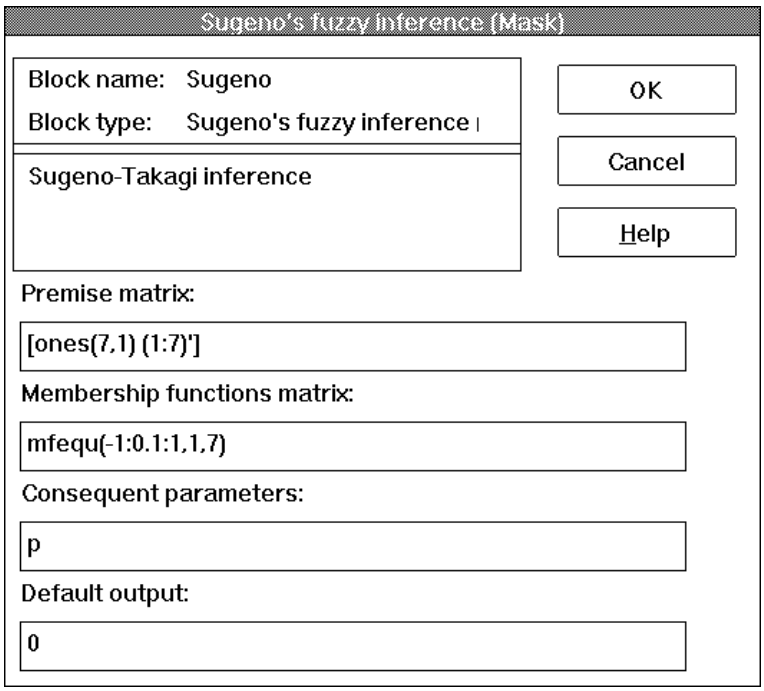
Notes

library:	Fuzzylib
direct feedthrough:	yes
based on S-function:	`sfunimpl`

Purpose:

Sugeno inference block, encapsulates the `dofmin` and `sugval` functions. See those functions for details. The block input must contain both premise and consequent inputs, even in case that they are identical. First premise inputs are supplied, then the consequent inputs.

Description:



Premise matrix defines the rule premises, as codes of membership functions given in the **Membership functions matrix**. See `dofmin` and `txt2rls` for details.

Membership functions matrix defines the membership functions parameters. See `mgrade` for details.

Consequent parameters contains the parameters of linear rule consequents. See `sugval` for details.

Default output is a vector containing the default value assigned to the output when no rule is applicable.

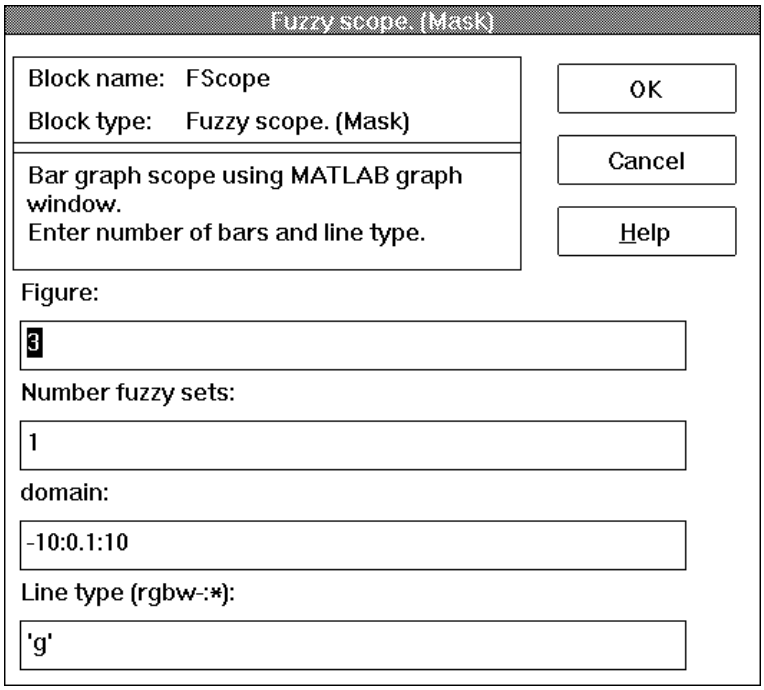
Notes

library:	Fuzzylib
direct feedthrough:	yes
based on S-function:	`sfunsug`

Purpose:

Fuzzy set scope, displays on-line fuzzy set(s).

Description:



Parameter **Figure** specifies a figure (graph window) where the plot is directed. It must be a valid figure handle (i.e. an integer number). On the block initialization, **Figure** is automatically opened (if it has not been already opened) and cleared (`clf`). All figures with numbers less then **Figure** are opened automatically as well.

Number of fuzzy sets specifies the number of fuzzy sets to be displayed.

Domain vector defines the fuzzy set domain (universe of discourse). The length of this vector must be equal to the length of the input vector (i.e. vector of membership degrees) times **Number of fuzzy sets**.

Line type specifies the color and line type. See `plot` fro details.

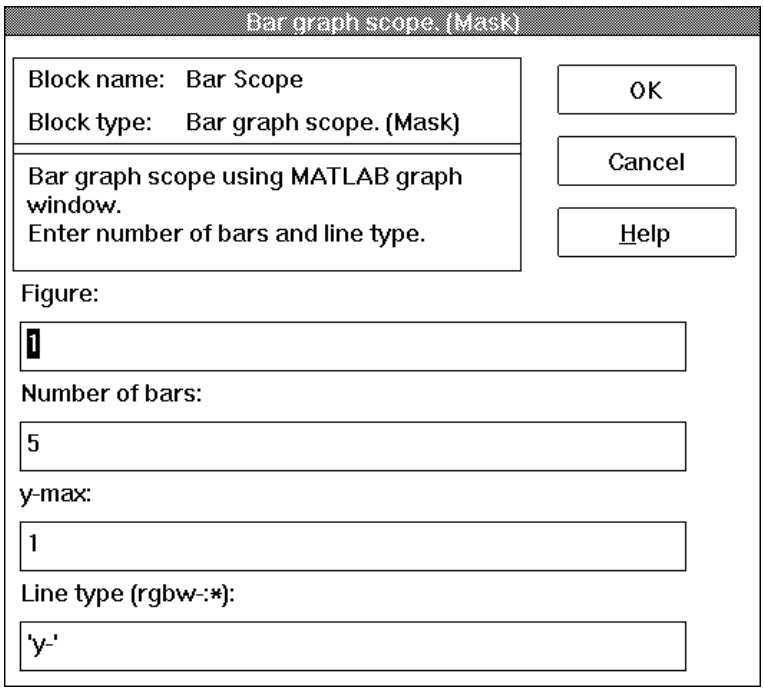
Notes

library:	Fuzzylib
based on S-function:	`sfunfset`

Purpose:

Bar graph scope, displays on-line a bar graph of the inputs. Can be used for investigating membership degrees, or truth values of rules.

Description:



Parameter **Figure** specifies a figure (graph window) where the plot is directed. It must be a valid figure handle (i.e. an integer number). On the block initialization, **Figure** is automatically opened (if it has not been already opened) and cleared (`clf`). All figures with numbers less then **Figure** are opened automatically as well.

Number of bars specifies the number of bars to be displayed. Must be equal to the length of the input vector.

y-max defines the maximum value to be displayed on the vertical axis. When working with membership degrees, y-max is usually set to 1.

Line type specifies the color and line type. See `plot` for details.

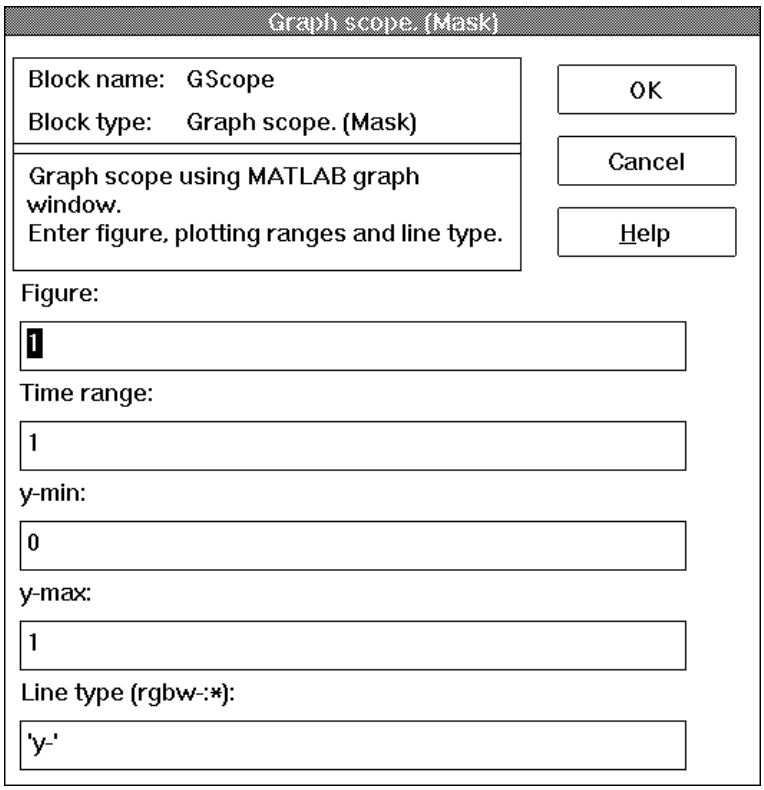
Notes

library:	Fuzzylib
based on S-function:	`sfunbar`

Purpose:

Graph scope, enhanced version of the standard SIMULINK scope. A target figure can be specified.

Description:



All the parameters are exactly the same as in SIMULINK Graph Scope, except for **Figure** which allows for specifying a figure (graph window) where the plot is directed. It must be a valid figure handle (i.e. an integer number). On the block initialization, **Figure** is automatically opened (if it has not been already opened) and cleared (`clf`). All figures with numbers less then **Figure** are opened automatically as well.

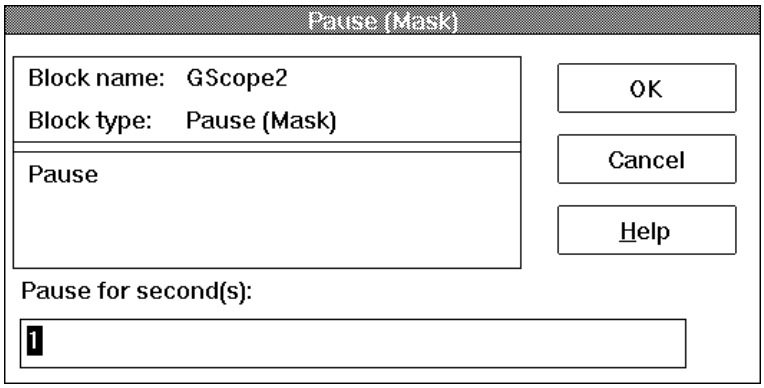
Notes

library:	Fuzzylib
based on S-function:	`sfunbar`

Purpose:

Pauses the simulation for a specified number of seconds or waits for a key (the same as MATLAB `pause` command).

Description:



Pause for seconds specifies the number of seconds for which the simulation pauses at every step. Special values are 0 – does not pause, and [] (empty matrix) – wait for a key.

Notes

library:	Fuzzylib
based on S-function:	`sfunpaus`