How to Contact MathWorks

Latest news: www.mathworks.com
Sales and services: www.mathworks.com/sales_and_services
User community: www.mathworks.com/matlabcentral
Technical support: www.mathworks.com/support/contact_us
Phone: 508-647-7000

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

Fuzzy Logic Toolbox™ Release Notes

© COPYRIGHT 2000–2020 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government’s needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

Patents

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.
R2020a

K-Fold Cross Validation: Prevent overfitting of tuned fuzzy inference system parameters .......................................................... 1-2

R2019b

Interval Type-2 Fuzzy Inference Systems: Create, simulate, tune, and deploy fuzzy systems with additional membership function uncertainty .......................................................... 2-2

Functionality Being Removed or Changed .......................................................... 2-2
Support for representing fuzzy inference systems as structures will be removed .......................................................... 2-2

R2019a

Fuzzy Trees: Implement complex fuzzy inference systems as collections of interconnected FIS objects .......................................................... 3-2

Fuzzy System Tuning: Tune rules and parameters of fuzzy inference systems and fuzzy trees to satisfy cost functions .......................................................... 3-2

R2018b

Code Generation: Generate C and C++ code to load and evaluate fuzzy inference systems .......................................................... 4-2

Improved Fuzzy Inference System Architecture: Implement fuzzy systems using objects with improved error handling .......................................................... 4-2
evalfis Function: Obtain intermediate rule firing strengths when evaluating fuzzy systems .......................................................... 4-3
addRule Function: Specify rules using linguistic and symbolic expressions

Functionality Being Removed or Changed

Support for representing fuzzy inference systems as structures will be removed.

newfis will be removed.
parsrule will be removed.
addvar will be removed.
rmmf will be removed.
getfis will be removed.
setfis will be removed.
mf2mf will be removed.
mam2sug will be removed.
showfis will be removed.
evalfis will be removed.
evalmf now takes a fismf object as an input argument.
eaddmf is now addMF and its function syntax has changed.
addrule is now addRule.
writefis is now writeFIS.
Support for deploying fuzzy systems using standalone C-code fuzzy inference engine will be removed.

R2018a

Fuzzy Logic Controller Block: Troubleshoot FIS evaluation using new diagnostic messages

New evalfisOptions Option Set: Specify options for evaluating fuzzy systems

evalfis Function: New diagnostic message behavior

evalfis Function: Intermediate fuzzy inference outputs for Sugeno systems analogous to outputs for Mamdani systems

R2017b

Code Generation Improvements: Generate code for single and fixed-point data types, and custom membership and inference functions

PLC Deployment: Generate IEC 61131-3 Structured Text from fuzzy logic controllers
Fuzzy Logic Controller Block Improvements: Configure additional block parameters, and access intermediate fuzzy inference results ............................................ 6-2

evalfis Command: Evaluate FIS output variable ranges over a smaller number of sample points ................................................................. 6-2

R2017a

Unified genfis Command: Generate fuzzy inference system structures using a single command ................................................................. 7-2

anfisOptions Command: Specify options for training adaptive neuro-fuzzy inference systems ................................................................. 7-3

gensurfOptions Command: Specify options for generating fuzzy inference system output surfaces ................................................................. 7-4

newfis Command: Specify options using Name,Value pairs ................................................................. 7-4

parsrule Command: Specify options using Name,Value pairs ................................................................. 7-5

showrule Command: Specify options using Name,Value pairs ................................................................. 7-6

subclust Command: Specify options using Name,Value pairs ................................................................. 7-6

Obtain fuzzy inference system properties using improved getfis command ................................................................. 7-7

Functionality being removed or changed ................................................................. 7-7

R2016b

Standalone Applications for ANFIS Training: Deploy neuro-adaptive fuzzy inference code using MATLAB Compiler ................................................................. 8-2

R2016a

Bug Fixes
<table>
<thead>
<tr>
<th>R2015b</th>
<th>Bug Fixes</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2015a</td>
<td>Bug Fixes</td>
</tr>
<tr>
<td>R2014b</td>
<td>Commands to open Fuzzy Logic Designer and Neuro-Fuzzy Designer renamed .................................................. 12-2</td>
</tr>
<tr>
<td>R2014a</td>
<td>Example that shows how to use a fuzzy inference system to detect edges in an image ............................................. 13-2</td>
</tr>
<tr>
<td>R2013b</td>
<td>Bug Fixes</td>
</tr>
<tr>
<td>R2013a</td>
<td>No New Features or Changes</td>
</tr>
<tr>
<td>R2012b</td>
<td>No New Features or Changes</td>
</tr>
</tbody>
</table>
No New Features or Changes

No New Features or Changes

No New Features or Changes

No New Features or Changes

No New Features or Changes

No New Features or Changes
<table>
<thead>
<tr>
<th>Version</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2008b</td>
<td>No New Features or Changes</td>
</tr>
<tr>
<td>R2008a</td>
<td>No New Features or Changes</td>
</tr>
<tr>
<td>R2007b</td>
<td>New Demo ..................................  26-2</td>
</tr>
<tr>
<td>R2007a</td>
<td>No New Features or Changes</td>
</tr>
<tr>
<td>R2006b</td>
<td>No New Features or Changes</td>
</tr>
<tr>
<td>R2006a</td>
<td>No New Features or Changes</td>
</tr>
<tr>
<td>R14SP3</td>
<td>No New Features or Changes</td>
</tr>
</tbody>
</table>
No New Features or Changes
R2020a

Version: 2.7

New Features

Bug Fixes
K-Fold Cross Validation: Prevent overfitting of tuned fuzzy inference system parameters

You can now prevent overfitting of your fuzzy inference system (FIS) parameters to your training data by using k-fold cross validation. K-fold validation randomly partitions your training data into k subsets of approximately equal size. The function then performs k training-validation iterations. For each iteration, one data subset is used as validation data with the remaining k-1 subsets used as training data.

Each training-validation iteration ends when the ratio between the increase in validation cost and the current minimum validation cost exceeds a specified tolerance value.

For an example that tunes FIS parameters using k-fold cross validation, see “FIS Parameter Optimization with K-fold Cross Validation”. For more information on tuning fuzzy inference systems, see `tunefis` and `tunefisOptions`. 

R2019b

Version: 2.6

New Features

Bug Fixes

Compatibility Considerations
**Interval Type-2 Fuzzy Inference Systems: Create, simulate, tune, and deploy fuzzy systems with additional membership function uncertainty**

Fuzzy Logic Toolbox software now supports interval type-2 fuzzy inference systems (FISs). For any value in the universe of discourse, a traditional type-1 membership function has a single membership value. In type-2 membership functions, the degree of membership can have a range of values, which introduces additional uncertainty into the FIS.

For more information, see Type-2 Fuzzy Inference Systems.

**Functionality Being Removed or Changed**

**Support for representing fuzzy inference systems as structures will be removed**

**Warns**

Support for representing fuzzy inference systems as structures will be removed in a future release. Use `mamfis` and `sugfis` objects instead. There are differences between these representations that require updates to your code. These differences include:

- Object property names that differ from the corresponding structure fields
- Objects that store text data as strings rather than as character vectors

Also, all Fuzzy Logic Toolbox functions that accepted or returned fuzzy inference systems as structures now accept and return either `mamfis` or `sugfis` objects.

**Update Code**

To convert existing fuzzy inference system structures to objects, use the `convertfis` function.

This table shows some examples of how to update your code to use the new object property names.

<table>
<thead>
<tr>
<th>If your code has this form</th>
<th>Use this code instead</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fis.andMethod</code></td>
<td><code>fis.AndMethod</code></td>
</tr>
<tr>
<td><code>fis.aggMethod</code></td>
<td><code>fis.AggregationMethod</code></td>
</tr>
<tr>
<td><code>fis.input(1).name</code></td>
<td><code>fis.Inputs(1).Name</code></td>
</tr>
<tr>
<td><code>fis.output(1).mf(1).params</code></td>
<td><code>fis.Outputs(1).MembershipFunctions(1).Parameters</code></td>
</tr>
</tbody>
</table>
R2019a

Version: 2.5

New Features

Bug Fixes
**Fuzzy Trees: Implement complex fuzzy inference systems as collections of interconnected FIS objects**

You can now create a tree of interconnected fuzzy inference systems. To create a fuzzy tree, create a `fistree` object, specifying the component fuzzy inference systems and their interconnections. For more information, see Fuzzy Trees.

To evaluate fuzzy trees for specific input combinations, use the `evalfis` function.

**Fuzzy System Tuning: Tune rules and parameters of fuzzy inference systems and fuzzy trees to satisfy cost functions**

You can now tune the rules and parameters of a fuzzy system to satisfy input/output data or a cost function. You can tune the parameters of fuzzy inference systems (`mamfis` and `sugfis` objects) and fuzzy trees (`fistree` object).

Using the `tunefis` function, you can tune the parameters of a fuzzy system, learn new fuzzy rules, or both. You can tune your fuzzy system using any of the following methods. The first four methods require Global Optimization Toolbox software.

- Genetic algorithms
- Particle swarm optimization
- Simulated annealing
- Pattern search
- ANFIS

The ANFIS method supports only parameter tuning. The other tuning methods support both parameter tuning and rule learning.
R2018b

Version: 2.4
New Features
Bug Fixes
Compatibility Considerations
Code Generation: Generate C and C++ code to load and evaluate fuzzy inference systems

You can now generate code for loading and evaluating a fuzzy inference system (FIS) using MATLAB Coder™. The following existing functions now support code generation:

- evalfis
- evalfisOptions
- Built-in membership functions, such as gaussmf and trimf

For an example, see Generate Code for Fuzzy System Using MATLAB Coder.

Code generation using MATLAB Coder does not support the new FIS objects, mamfis and sugfis. To create a homogeneous FIS structure for code generation, use the new getFISCodeGenerationData.

Code generation using MATLAB Coder replaces the standalone C-code fuzzy inference engine. For more information, see “Support for deploying fuzzy systems using standalone C-code fuzzy inference engine will be removed” on page 4-10.

Improved Fuzzy Inference System Architecture: Implement fuzzy systems using objects with improved error handling

The Fuzzy Logic Toolbox now represents fuzzy inference systems using objects. These objects provide improved error handling over the previous structure-based representation.

The following table shows the new objects for representing a fuzzy inference system (FIS). The function for creating each object has the same name as the corresponding object.

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mamfis</td>
<td>Mamdani FIS</td>
</tr>
<tr>
<td>sugfis</td>
<td>Sugeno FIS</td>
</tr>
<tr>
<td>fisvar</td>
<td>Input or output variable</td>
</tr>
<tr>
<td>fismf</td>
<td>Membership function</td>
</tr>
<tr>
<td>fisrule</td>
<td>Fuzzy rule</td>
</tr>
</tbody>
</table>

In addition to new objects, the improved FIS architecture includes several new functions for configuring fuzzy systems.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addInput and addOutput</td>
<td>Add input and output variables to a FIS (replaces addvar)</td>
</tr>
<tr>
<td>removeInput and removeOutput</td>
<td>Remove input and output variables from a FIS (replaces rrmvar)</td>
</tr>
<tr>
<td>removeMF</td>
<td>Remove a membership function from an input or output variable in a FIS (replaces rrmf)</td>
</tr>
</tbody>
</table>
### Function Descriptions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>convertToSugeno</code></td>
<td>Convert a Mamdani FIS to a Sugeno FIS (replaces <code>mam2sug</code>)</td>
</tr>
<tr>
<td><code>convertToStruct</code></td>
<td>Convert a <code>mamfis</code> or <code>sugfis</code> object to a structure</td>
</tr>
<tr>
<td><code>update</code></td>
<td>Update the fields of a manually created <code>fisrule</code> object based on a given FIS</td>
</tr>
</tbody>
</table>

### Compatibility Considerations

The improved FIS architecture introduces several compatibility considerations that require updates to your code. For more information on how to update your code to use fuzzy inference system objects in general, see “Support for representing fuzzy inference systems as structures will be removed” on page 4-4.

Several functions have changes in behavior or will be removed in a future release, as indicated in the following table.

<table>
<thead>
<tr>
<th>Function</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>newfis</code></td>
<td>“newfis will be removed” on page 4-4</td>
</tr>
<tr>
<td><code>parsrule</code></td>
<td>“parsrule will be removed” on page 4-5</td>
</tr>
<tr>
<td><code>addvar</code></td>
<td>“addvar will be removed” on page 4-5</td>
</tr>
<tr>
<td><code>rmvar</code></td>
<td>“rmvar will be removed” on page 4-6</td>
</tr>
<tr>
<td><code>rmmf</code></td>
<td>“rmmf will be removed” on page 4-6</td>
</tr>
<tr>
<td><code>getfis</code></td>
<td>“getfis will be removed” on page 4-6</td>
</tr>
<tr>
<td><code>setfis</code></td>
<td>“setfis will be removed” on page 4-7</td>
</tr>
<tr>
<td><code>mf2mf</code></td>
<td>“mf2mf will be removed” on page 4-7</td>
</tr>
<tr>
<td><code>mam2sug</code></td>
<td>“mam2sug will be removed” on page 4-8</td>
</tr>
<tr>
<td><code>showfis</code></td>
<td>“showfis will be removed” on page 4-8</td>
</tr>
<tr>
<td><code>evalfis</code></td>
<td>“evalfis input argument order has changed” on page 4-8</td>
</tr>
<tr>
<td><code>evalmf</code></td>
<td>“evalmf now takes a <code>fismf</code> object as an input argument” on page 4-9</td>
</tr>
<tr>
<td><code>addmf</code></td>
<td>“addmf is now addMF and its function syntax has changed” on page 4-9</td>
</tr>
<tr>
<td><code>addrule</code></td>
<td>“addrule is now addRule” on page 4-10</td>
</tr>
<tr>
<td><code>writefis</code></td>
<td>“writefis is now writeFIS” on page 4-10</td>
</tr>
</tbody>
</table>

### evalfis Function: Obtain intermediate rule firing strengths when evaluating fuzzy systems

You can now obtain intermediate rule firing strengths when evaluating a fuzzy inference system using `evalfis`. The rule firing strengths are obtained by applying the fuzzy operator to the values of the fuzzified inputs.
addRule Function: Specify rules using linguistic and symbolic expressions

The addRule function now supports adding rules to a fuzzy system using linguistic and symbolic expressions. This addRule functionality replaces the equivalent parsrule functionality. For more information, see “parsrule will be removed” on page 4-5.

Functionality Being Removed or Changed

Support for representing fuzzy inference systems as structures will be removed

Still runs

Support for representing fuzzy inference systems as structures will be removed in a future release. Use mamfis and sugfis objects instead. There are differences between these representations that require updates to your code. These differences include:

- Object property names that differ from the corresponding structure fields
- Objects that store text data as strings rather than as character vectors

Also, all Fuzzy Logic Toolbox functions that accepted or returned fuzzy inference systems as structures now accept and return either mamfis or sugfis objects.

Update Code

To convert existing fuzzy inference system structures to objects, use the convertfis function.

This table shows some examples of how to update your code to use the new object property names.

<table>
<thead>
<tr>
<th>If your code has this form:</th>
<th>Use this code instead:</th>
</tr>
</thead>
<tbody>
<tr>
<td>fis.andMethod</td>
<td>fis.AndMethod</td>
</tr>
<tr>
<td>fis.aggMethod</td>
<td>fis.AggregationMethod</td>
</tr>
<tr>
<td>fis.input(1).name</td>
<td>fis.Inputs(1).Name</td>
</tr>
<tr>
<td>fis.output(1).mf(1).params</td>
<td>fis.Outputs(1).MembershipFunctions(1).Parameters</td>
</tr>
</tbody>
</table>

newfis will be removed

Still runs

newfis will be removed in a future release. Use mamfis or sugfis instead. There are differences between these functions that require updates to your code.

To create a Mamdani or Sugeno FIS, use mamfis or sugfis, respectively.

Update Code

This table shows some typical usages of newfis for creating fuzzy systems and how to update your code to use mamfis or sugfis instead.

<table>
<thead>
<tr>
<th>If your code has this form:</th>
<th>Use this code instead:</th>
</tr>
</thead>
<tbody>
<tr>
<td>fis = newfis(name)</td>
<td>fis = mamfis('Name',name)</td>
</tr>
<tr>
<td>fis = newfis(name,'FISType','mamdani')</td>
<td>fis = mamfis('Name',name)</td>
</tr>
<tr>
<td>fis = newfis(name,'FISType','sugeno')</td>
<td>fis = sugfis('Name',name)</td>
</tr>
</tbody>
</table>
If your code has this form: | Use this code instead:
---|---
fis = newfis(name,...  | fis = mamfis('Name',name,...
  'FISType','mamdani',...  |  'AndMethod','prod')
  'AndMethod','prod')
fis = newfis(name,...  | fis = sugfis('Name',name,...
  'FISType','sugeno',....  |  'OrMethod','probor')
  'OrMethod','probor')

parsrule will be removed

*Still runs*

parsrule will be removed in a future release. Use addRule instead.

**Update Code**

If you previously added rules using linguistic or symbolic expressions with parsrule, you can specify rules using the same expressions with addRule. addRule automatically detects the format of the strings or character vectors in your rule list. Therefore, it is no longer necessary to specify the rule format. To add a rule list using addRule, use the following command:

```matlab
fis = addRule(fis,rules);
```

Previously, you could add rules using indexed expressions with parsrule.

```matlab
rule1 = "1 2, 1 4 (1) : 1";
rule2 = "-1 1, 3 2 (1) : 1";
rules = [rule1 rule2];
fis = parsrule(fis,rules,'Format','indexed');
```

Now, specify these rules using arrays of indices.

```matlab
rule1 = [1 2 1 4 1 1];
rule2 = [-1 1 3 2 1 1];
rules = [rule1; rule2];
fis = addRule(fis,rules);
```

If you previously specified rules using the 'Lanuage' name-value pair argument with parsrule, this functionality has been removed and there is no replacement. Specify your rules using addRule with a different rule format.

Previously, parsrule replaced the entire rule list in your fuzzy system. addRule appends your specified rules to the rule list.

**addvar will be removed**

*Still runs*

addvar will be removed in a future release. Use addInput or addOutput instead. There are differences between these functions that require updates to your code.

To add input or output variables to a fuzzy system, use addInput or addOutput, respectively.

**Update Code**

This table shows some typical usages of addvar and how to update your code to use addInput or addOutput instead.
If your code has this form: | Use this code instead:
---|---
`fis = addvar(fis,'input',... 'service',[0 10])` | `fis = addInput(fis,[0 10],... 'Name','service')`  
`fis = addvar(fis,'output',... 'tip',[0 30])` | `fis = addOutput(fis,[0 30],... 'Name','tip')`  

**rmvar will be removed**

*Still runs*

`rmvar` will be removed in a future release. Use `removeInput` or `removeOutput` instead. There are differences between these functions that require updates to your code.

To remove input or output variables from a fuzzy system, use `removeInput` or `removeOutput`, respectively.

**Update Code**

This table shows some typical usages of `rmvar` and how to update your code to use `removeInput` or `removeOutput` instead. Previously, you specified the index of the variable that you wanted to remove. Now, to remove a variable, specify the variable name.

| If your code has this form: | Use this code instead: |
---|---|
`fis = rmvar(fis,'input',1)` | `fis = removeInput(fis,"service")`  
`fis = rmvar(fis,'output',1)` | `fis = removeOutput(fis,"tip")`  

Previously, you had to delete any rules from your fuzzy system that contained the variable you wanted to remove. `removeInput` and `removeOutput` automatically remove these variables from the rule set of your fuzzy system.

**rmmf will be removed**

*Still runs*

`rmmf` will be removed in a future release. Use `removeMF` instead. There are differences between these functions that require updates to your code.

**Update Code**

The following table shows some typical usages of `rmmf` and how to update your code to use `removeMF` instead. Previously, you specified the index of the variable from which you wanted to remove the membership function and the index of the membership function that you wanted to remove. Now, to remove a membership function, specify the variable name and the membership function name.

| If your code has this form: | Use this code instead: |
---|---|
`fis = rmmf(fis,'input',1,'mf',1)` | `fis = removeMF(fis,"service","poor")`  
`fis = rmmf(fis,'output',1,'mf',1)` | `fis = removeMF(fis,"tip","cheap")`  

Previously, you had to delete any references to a membership function you wanted to remove from the rule set. `removeMF` automatically removes these references from the rule set of your fuzzy system.

**getfis will be removed**

*Still runs*
getfis will be removed in a future release. Access fuzzy inference system properties using dot notation instead. There are differences between these approaches that require updates to your code.

**Update Code**

This table shows some typical usages of `getfis` for accessing fuzzy inference system properties and how to update your code to use dot notation instead.

<table>
<thead>
<tr>
<th>If your code has this form:</th>
<th>Use this code instead:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>get(fis,'andmethod')</code></td>
<td><code>fis.AndMethod</code></td>
</tr>
<tr>
<td><code>getfis(fis,'input',1)</code></td>
<td><code>fis.Inputs(1)</code></td>
</tr>
<tr>
<td><code>getfis(fis,'input',1,'name')</code></td>
<td><code>fis.Inputs(1).Name</code></td>
</tr>
<tr>
<td><code>getfis(fis,'input',2,'mf',1)</code></td>
<td><code>fis.Inputs(2).MembershipFunctions(1)</code></td>
</tr>
<tr>
<td><code>getfis(fis,'input',2,'mf',1,...</code></td>
<td><code>fis.Inputs(2).MembershipFunctions(1).Parameters</code></td>
</tr>
<tr>
<td><code>params)</code></td>
<td></td>
</tr>
</tbody>
</table>

Previously, fuzzy inference systems were represented as structures. Now, fuzzy inference systems are represented as objects. Fuzzy inference system object properties have different names than the corresponding structure fields. For more information on fuzzy inference system objects, see `mamfis` and `sugfis`.

**setfis will be removed**

*Still runs*

`setfis` will be removed in a future release. Set fuzzy inference system properties using dot notation instead. There are differences between these approaches that require updates to your code.

**Update Code**

This table shows some typical usages of `setfis` for setting fuzzy inference system properties and how to update your code to use dot notation instead.

<table>
<thead>
<tr>
<th>If your code has this form:</th>
<th>Use this code instead:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fis = setfis(fis,'andmethod','prod')</code></td>
<td><code>fis.AndMethod = 'prod'</code></td>
</tr>
<tr>
<td><code>fis = setfis(fis,'input',1,...</code></td>
<td><code>fis.Inputs(1).Name = &quot;service&quot;</code></td>
</tr>
<tr>
<td><code>name','service')</code></td>
<td></td>
</tr>
<tr>
<td><code>fis = setfis(fis,'input',2,...</code></td>
<td><code>fis.Inputs(2).MembershipFunctions(1).Parameters = ...</code></td>
</tr>
<tr>
<td><code>mf',1,...</code></td>
<td><code>[5 10 15]</code></td>
</tr>
<tr>
<td><code>params,[5 10 15])</code></td>
<td></td>
</tr>
</tbody>
</table>

Previously, fuzzy inference systems were represented as structures. Now, fuzzy inference systems are represented as objects. Fuzzy inference system object properties have different names than the corresponding structure fields. For more information on fuzzy inference system objects, see `mamfis` and `sugfis`.

**mf2mf will be removed**

*Still runs*

`mf2mf` will be removed in a future release. Convert membership functions using dot notation on `fismf` objects instead. There are differences between these approaches that require updates to your code.
**Update Code**

Previously, to change the type of a membership function in a fuzzy inference system, you converted the parameters using `mf2mf`.

```matlab
fis = readfis('tipper');
oldType = fis.input(1).mf(1).type;
oldParams = fis.input(1).mf(1).params;
fis.input(1).mf(1).type = newType;
fis.input(1).mf(1).params = mf2mf(oldParams,oldType,newType);
```

Now, when you change the type of membership function, the parameters are converted automatically.

```matlab
fis = readfis('tipper');
fis.Inputs(1).MembershipFunctions(1).Type = newType;
```

Previously, membership functions were represented as structures within a fuzzy inference system structure. Now, membership functions are represented as `fismf` objects within `mamfis` and `sugfis` objects. For more information on fuzzy inference system objects, see `mamfis` and `sugfis`.

**mam2sug will be removed**

*Still runs*

`mam2sug` will be removed in a future release. Use `convertToSugeno` instead. To update your code, change the function name from `mam2sug` to `convertToSugeno`. The syntaxes are equivalent.

**showfis will be removed**

*Still runs*

`showfis` will be removed in a future release. View the properties of your FIS directly instead.

Previously, you could view the properties of your fuzzy system, `myFIS`, using the `showfis` function.

```matlab
showfis(myFIS)
```

Now, you can view the properties directly instead.

```matlab
myFIS
```

To view additional FIS properties, use dot notation. For example, view information about the membership functions of the first input variable.

```matlab
myFIS.Inputs(1).MembershipFunctions
```

For more information on fuzzy inference systems and their properties, see `mamfis` and `sugfis`.

**evalfis input argument order has changed**

*Behavior change*

The order of input arguments for `evalfis` has changed, which requires updates to your code.

**Update Code**

Previously, to evaluate a fuzzy inference system, `fis`, you specified the input variable values, `input`, as the first input argument. For example:

```matlab
output = evalfis(input,fis);
output = evalfis(input,fis,options);
```
Update your code to specify the fuzzy inference system as the first input argument. For example:

```plaintext
output = evalfis(fis,input);
output = evalfis(fis,input,options);
```

**evalmf now takes a fismf object as an input argument**

Behavior change

`evalmf` now takes a `fismf` object as an input argument rather than the type and parameters of the membership function. Also, you can now evaluate multiple membership functions by passing an array of `fismf` objects to `evalmf`. There are differences between these approaches that require updates to your code.

**Update Code**

Previously, you evaluated a membership function for given input values, `x`, by specifying the type of membership function, `type`, and the membership functions parameters, `params`.

```plaintext
y = evalmf(x,params,type);
```

Update your code to first create a `fismf` object, `mf`. Then, pass this object to `evalmf`.

```plaintext
mf = fismf(type,params);
y = evalmf(mf,x);
```

Also, previously, to evaluate multiple membership functions you called `evalmf` once for each membership function.

```plaintext
y1 = evalmf(x,params1,type1);
y2 = evalmf(x,params2,type2);
y3 = evalmf(x,params3,type3);
```

Now, you can evaluate multiple membership functions by passing an array of `fismf` objects to `evalmf`.

```plaintext
mf1 = fismf(type1,params1);
mf2 = fismf(type2,params2);
mf3 = fismf(type3,params3);
y = evalmf([mf1 mf2 mf3],x);
```

Here, `y = [y1 y2 y3]'`;

**addmf is now addMF and its function syntax has changed**

Behavior change

The name and behavior of the `addmf` function has changed. Now:

- `addmf` is `addMF`
- You specify the variable to which you want to add the membership function by name rather than by index.
- You specify the name of the membership function using a name-value pair argument.

These changes require updates to your code. For more information, see `addMF`.
Update Code

The following table shows some typical usages of `addmf` for adding membership functions to fuzzy variables and how to update your code. In this table, `fis` is a fuzzy inference system with two inputs, `service` and `food`, and one output, `tip`.

<table>
<thead>
<tr>
<th>If your code has this form:</th>
<th>Use this code instead:</th>
</tr>
</thead>
<tbody>
<tr>
<td>`fis = addmf(fis'input',1,...</td>
<td>`fis = addMF(fis,&quot;service&quot;,...</td>
</tr>
<tr>
<td>'poor','gaussmf',[1.5 0])</td>
<td>&quot;gaussmf&quot;,[1.5 0],</td>
</tr>
<tr>
<td></td>
<td>'Name',&quot;poor&quot;)</td>
</tr>
<tr>
<td>`fis = addmf(fis,'input',2,...</td>
<td>`fis = addMF(fis,&quot;food&quot;,...</td>
</tr>
<tr>
<td>'rancid','trapmf',[-2 0 1 3])</td>
<td>&quot;trapmf&quot;,[-2 0 1 3],</td>
</tr>
<tr>
<td></td>
<td>'Name','rancid&quot;)</td>
</tr>
<tr>
<td>`fis = addmf(fis,'output',1,...</td>
<td>`fis = addMF(fis,&quot;tip&quot;,...</td>
</tr>
<tr>
<td>'cheap','trimf',[0 5 10])</td>
<td>&quot;trimf&quot;,[0 5 10],</td>
</tr>
<tr>
<td></td>
<td>'Name','cheap&quot;)</td>
</tr>
</tbody>
</table>

`addrule` is now `addRule`  
*Behavior change*

`addrule` is now `addRule`. To update your code, change the function name from `addrule` to `addRule`. The syntaxes are equivalent. For more information see `addRule`.

`writefis` is now `writeFIS`  
*Behavior change*

`writefis` is now `writeFIS`. To update your code, change the function name from `writefis` to `writeFIS`. The syntaxes are equivalent. For more information, see `writefis`.

Support for deploying fuzzy systems using standalone C-code fuzzy inference engine will be removed  
*Still runs*

Support for deploying fuzzy systems using a standalone C-code fuzzy inference engine will be removed in a future release. Generate code using MATLAB Coder or Simulink® Coder instead.

For more information on:
- Code generation in MATLAB and Simulink, see Deploy Fuzzy Inference Systems.
- The standalone inference engine, see Fuzzy Inference Engine in the R2018a documentation.
Version: 2.3.1

New Features

Bug Fixes

Compatibility Considerations
Fuzzy Logic Controller Block: Troubleshoot FIS evaluation using new diagnostic messages

The Fuzzy Logic Controller block now reports potential problems that can produce unexpected output values during a simulation. The block generates diagnostic messages for the following conditions:

- Input values outside of their specified variable ranges
- No rules fired for a given output at the current input values
- Empty output fuzzy sets

You can specify whether these diagnostic messages are reported as warnings, reported as errors, or ignored.

For more information, see Fuzzy Logic Controller.

New evalfisOptions Option Set: Specify options for evaluating fuzzy systems

You can now specify options for evaluating fuzzy systems at the command line using the new evalfisOptions option set. Using this option set, you can specify:

- The number of sample points to use when evaluating output fuzzy sets.
- Whether diagnostic messages for potential problems are reported as warnings, reported as errors, or ignored.

For more information, see evalfisOptions and evalfis.

Compatibility Considerations

Previously, when evaluating a FIS using evalfis, you specified the number of sample points in output fuzzy sets using the optional input argument numPts.

\[ \text{output} = \text{evalfis}(\text{fis}, \text{input}, \text{numPts}); \]

Starting in R2018a, modify your code to use an evalfisOptions option set.

\[ \text{opt} = \text{evalfisOptions}(\text{'NumSamplePoints'}, \text{numPts}); \]
\[ \text{output} = \text{evalfis}(\text{fis}, \text{input}, \text{opt}); \]

evalfis Function: New diagnostic message behavior

You can now control how the evalfis function reports potential problems that can produce unexpected output values during a simulation. The function generates diagnostic messages for the following conditions:

- Input values outside of their specified variable ranges
- No rules fired for a given output at the current input values
- Empty output fuzzy sets

You can specify whether these diagnostic messages are reported as warnings, reported as errors, or ignored. To do so, specify the corresponding options in an evalfisOptions option set.
For more information, see `evalfisOptions` and `evalfis`.

**Compatibility Considerations**

Previously, the `evalfis` function had the following behaviors for diagnostic conditions.

<table>
<thead>
<tr>
<th>Diagnostic Condition</th>
<th>Previous Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input values outside of their specified variable ranges</td>
<td>MATLAB warning</td>
</tr>
<tr>
<td>No rules fired for a given output at the current input values</td>
<td>MATLAB Command Window message</td>
</tr>
<tr>
<td>Empty output fuzzy sets</td>
<td>MATLAB Command Window message</td>
</tr>
</tbody>
</table>

Starting in R2018a, these diagnostic conditions are reported as MATLAB warnings by default. You can change this behavior by specifying the corresponding options in an `evalfisOptions` option set.

**evalfis Function: Intermediate fuzzy inference outputs for Sugeno systems analogous to outputs for Mamdani systems**

When evaluating a Sugeno system using the following syntax, the intermediate fuzzy inference results are now analogous to the intermediate results for Mamdani systems.

```matlab
[output,fuzzifiedInputs,ruleOutputs,aggregatedOutput] = evalfis(input,fis);
```

For a Sugeno system:

- `ruleOutputs` now returns an array that contains the scalar output value for each rule; that is, the product of the rule firing strength and the rule output level.
- `aggregatedOutput` now returns the sum of all the rule output values for each output variable.

For more information, see `evalfis`.

**Compatibility Considerations**

Previously, for a Sugeno fuzzy system:

- `ruleOutputs` returned an array that contained the output level for each rule.
- `aggregatedOutput` returned an array that contained the firing strength for each rule.

Starting in R2018a, if your code returns intermediate fuzzy inference results when evaluating a Sugeno system using `evalfis`, modify your code to use the new `ruleOutputs` and `aggregatedOutput` results.
## Functionality being removed or changed

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Result</th>
<th>Use This Instead</th>
<th>Compatibility Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>output = evalfis(input,fis,numPts);</td>
<td>Still works</td>
<td>opt = evalfisOptions('NumSamplePoints',numPts);</td>
<td>To specify the number of sample points for output fuzzy sets, use an evalfisOptions option set. For more information, see “New evalfisOptions Option Set: Specify options for evaluating fuzzy systems” on page 5-2.</td>
</tr>
<tr>
<td>diagStrTofis3.m</td>
<td>Not applicable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnostic messages when evaluating fuzzy systems using evalfis</td>
<td>Still works</td>
<td></td>
<td>By default, diagnostic messages are now reported as warnings. To change this behavior, specify the corresponding options in an evalfisOptions option set. For more information see, “evalfis Function: New diagnostic message behavior” on page 5-2.</td>
</tr>
<tr>
<td>[output,fuzzifiedInputs,ruleOutputs,aggregatedOutput] = evalfis(input,fis)</td>
<td>Still works</td>
<td></td>
<td>The behaviors of the ruleOutputs and aggregatedOutput argument of evalfis have changed for evaluating Sugeno systems. For more information, see “evalfis Function: Intermediate fuzzy inference outputs for Sugeno systems analogous to outputs for Mamdani systems” on page 5-3.</td>
</tr>
<tr>
<td>when evaluating Sugeno systems</td>
<td>Not applicable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Version: 2.3

New Features

Bug Fixes
**Code Generation Improvements: Generate code for single and fixed-point data types, and custom membership and inference functions**

The Fuzzy Logic Controller block now supports code generation for fuzzy systems using:

- Single-precision data.
- Fixed-point data. To generate code for fixed-point data, you need Fixed-Point Designer™ software.
- Custom membership functions and custom inference functions. For more information on specifying custom functions for a fuzzy system, see Build Fuzzy Systems Using Custom Functions.

You can generate code using either Simulink Coder or Simulink PLC Coder™ software.

**PLC Deployment: Generate IEC 61131-3 Structured Text from fuzzy logic controllers**

The Fuzzy Logic Controller block now supports generation of IEC 61131-3 Structured Text for PLC deployment using Simulink PLC Coder software.

**Fuzzy Logic Controller Block Improvements: Configure additional block parameters, and access intermediate fuzzy inference results**

For the Fuzzy Logic Controller block, you can now:

- Use a double-precision, single-precision, or fixed-point data type.
- Specify the number of sample points for evaluating the output range of a Mamdani system.
- Access intermediate fuzzy inference results using new optional output ports.

For more information, see Fuzzy Logic Controller.

**evalfis Command: Evaluate FIS output variable ranges over a smaller number of sample points**

You can now specify the number of sample points for evaluating the output range of a Mamdani fuzzy inference system at the command line as any value greater than 1. Previously, the minimum value was 101.

This change applies to the:

- numPts input argument of evalfis.
- NumSamplePoints property of gensurfOptions.
R2017a

Version: 2.2.25

New Features

Bug Fixes

Compatibility Considerations
Unified genfis Command: Generate fuzzy inference system structures using a single command

The commands for generating the structure of a fuzzy inference system have been unified into a single genfis command, which you configure using a new genfisOptions option set.

Starting in R2017a, to generate a FIS structure, first create a default genfisOptions option set, specifying one of the following structure generation algorithms:

- Grid partitioning
  ```
  opt = genfisOptions('GridPartition');
  ```
- Subtractive clustering
  ```
  opt = genfisOptions('SubtractiveClustering');
  ```
- Fuzzy c-means clustering
  ```
  opt = genfisOptions('FCMClustering');
  ```

You can then modify the options using dot notation. Any options you do not modify remain at their default values.

Compatibility Considerations

Previously, to generate FIS structures, you used the genfis1, genfis2, or genfis3 commands with optional input arguments.

These commands may be removed in a future release and, starting in R2017a, using these commands generates a warning. If your code uses genfis1, genfis2, or genfis3, modify the code to use the genfis command, specifying options using a genfisOptions option set.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Old Syntax</th>
<th>New Syntax</th>
</tr>
</thead>
</table>
| Grid partitioning    | ```
  fis = genfis1(data,...
  numMFs,...
  inmftype,...
  outmftype)
  ``` | ```
  opt = genfisOptions('GridPartition');
  opt.NumMembershipFunctions = numMFs;
  opt.InputMembershipFunctionType = inmftype;
  opt.OutputMembershipFunctionType = outmftype;
  inputData = data(:,end-1);
  outputData = data(:,end);
  fis = genfis(inputData,outputData,opt);
  ``` |
| Subtractive clustering| ```
  fis = genfis2(inputData,...
  outputData,...
  radii,...
  xBounds,...
  options,...
  userCenters)
  ``` | ```
  opt = genfisOptions('SubtractiveClustering');
  opt.ClusterInfluenceRange = radii;
  opt.DataScale = xBounds;
  opt.SquashFactor = options(1);
  opt.AcceptRatio = options(2);
  opt.RejectRatio = options(3);
  opt.Verbose = options(4);
  opt.CustomClusterCenters = userCenters;
  fis = genfis(inputData,outputData,opt);
  ``` |
The syntaxes in this table assume that you are specifying all the options for each algorithm. Since the initial `genfisOptions` option set contains default algorithm options, you have to specify only nondefault options. For example, create an FIS using FCM clustering with three clusters, leaving all other options at their default values.

```matlab
opt = genfisOptions('FCMClustering');
opt.NumClusters = 3;
fis = genfis(Xin,Xout,opt);
```

**anfisOptions Command: Specify options for training adaptive neuro-fuzzy inference systems**

To specify options for training adaptive neuro-fuzzy inference systems using the `anfis` command, you now create an `anfisOptions` option set. You can then modify the options using dot notation. Any options you do not modify remain at their default values.

**Compatibility Considerations**

Previously, to train an adaptive neuro-fuzzy inference system using `anfis`, you specified the training options using optional input arguments.

```matlab
fis = anfis(trnData,initFIS,trnOpt,dispOpt,chkData,optMethod);
```

Starting in R2017a, if your code uses `anfis`, modify the code to use an `anfisOptions` option set.

```matlab
opt = anfisOptions;
opt.InitialFIS = 3;
fis = anfis(trnData,opt);
```

The following table shows the mapping of the old `anfis` input arguments to the new `anfisOptions` option set.

<table>
<thead>
<tr>
<th>Old anfis Input Argument</th>
<th>New anfisOptions Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>initFIS</td>
<td>InitialFIS</td>
</tr>
<tr>
<td>trnOpt(1)</td>
<td>EpochNumber</td>
</tr>
<tr>
<td>trnOpt(2)</td>
<td>ErrorGoal</td>
</tr>
<tr>
<td>trnOpt(3)</td>
<td>InitialStepSize</td>
</tr>
<tr>
<td>trnOpt(4)</td>
<td>StepSizeDecreaseRate</td>
</tr>
<tr>
<td>trnOpt(5)</td>
<td>StepSizeIncreaseRate</td>
</tr>
</tbody>
</table>
Old anfis Input Argument | New anfisOptions Option
--- | ---
dispOpt(1) | DisplayANFISInformation
dispOpt(2) | DisplayErrorValues
dispOpt(3) | DisplayStepSize
dispOpt(4) | DisplayFinalResults
chkData | ValidationData
optMethod | OptimizationMethod

**gensurfOptions Command: Specify options for generating fuzzy inference system output surfaces**

To specify options for generating fuzzy inference system output surfaces using the `gensurf` command, you now create a `gensurfOptions` option set. You can then modify the options using dot notation. Any options you do not modify remain at their default values.

**Compatibility Considerations**

Previously, to generate an output surface for a fuzzy inference system using `gensurf`, you specified the generation options using optional input arguments.

```
gensurf(fis, inputs, output, grids, refInput, points);
```

Starting in R2017a, if your code uses `gensurf`, modify the code to use a `gensurfOptions` option set.

```
opt = gensurfOptions;
opt.InputIndex = [1 3];
fis = gensurf(fis, opt);
```

The following table shows the mapping of the old `gensurf` input arguments to the new `gensurfOptions` option set.

<table>
<thead>
<tr>
<th>Old gensurf Input Argument</th>
<th>New gensurfOptions Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputs</td>
<td>InputIndex</td>
</tr>
<tr>
<td>output</td>
<td>OutputIndex</td>
</tr>
<tr>
<td>grids</td>
<td>NumGridPoints</td>
</tr>
<tr>
<td>refinput</td>
<td>ReferenceInputs</td>
</tr>
<tr>
<td>points</td>
<td>NumSamplePoints</td>
</tr>
</tbody>
</table>

**newfis Command: Specify options using Name,Value pairs**

To specify options for creating new fuzzy inference systems using the `newfis` command, you now use Name,Value pair arguments. Any Name,Value pair arguments that you do not specify remain at their default values.
Compatibility Considerations

Previously, you specified options for the `newfis` command using optional input arguments.

```matlab
fis = newfis('My FIS',fisType,andMethod,orMethod,impMethod,aggMethod,defuzzMethod);
```

Starting in R2017a, if your code uses `newfis`, modify the code to use one or more Name,Value pair arguments. For example, create a Mamdani FIS with default options.

```matlab
fis = newfis('My FIS','FISType','mamdani');
```

The following table shows the mapping of the old input arguments to the new Name,Value pair arguments.

<table>
<thead>
<tr>
<th>Old <code>newfis</code> Input Argument</th>
<th>New Name,Value Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fisType</code></td>
<td>'FISType'</td>
</tr>
<tr>
<td><code>andMethod</code></td>
<td>'AndMethod'</td>
</tr>
<tr>
<td><code>orMethod</code></td>
<td>'OrMethod'</td>
</tr>
<tr>
<td><code>impMethod</code></td>
<td>'ImplicationMethod'</td>
</tr>
<tr>
<td><code>aggMethod</code></td>
<td>'AggregationMethod'</td>
</tr>
<tr>
<td><code>defuzzMethod</code></td>
<td>'DefuzzificationMethod'</td>
</tr>
</tbody>
</table>

parsrule Command: Specify options using Name,Value pairs

To specify options for creating new fuzzy inference systems using the `parsrule` command, you now use Name,Value pair arguments. Any Name,Value pair arguments that you do not specify remain at their default values.

Compatibility Considerations

Previously, you specified options for the `parsrule` command using optional input arguments `ruleFormat` and `lang`.

```matlab
outFIS = parsrule(inFIS,ruleList,ruleFormat,lang);
```

Starting in R2017a, if your code uses `newfis`, modify the code to use one or more Name,Value pair arguments. For example, add a list of rules in 'symbolic' format.

```matlab
fis = parsrule(inFIS,ruleList,'Format','symbolic');
```

The following table shows the mapping of the old input arguments to the new Name,Value pair arguments.

<table>
<thead>
<tr>
<th>Old <code>parsrule</code> Input Argument</th>
<th>New Name,Value Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ruleFormat</code></td>
<td>'Format'</td>
</tr>
<tr>
<td><code>lang</code></td>
<td>'Language'</td>
</tr>
</tbody>
</table>
showrule Command: Specify options using Name,Value pairs

To specify options for viewing the rules of a fuzzy inference system using the `showrule` command, you now use `Name,Value` pair arguments. Any `Name,Value` pair arguments that you do not specify remain at their default values.

Compatibility Considerations

Previously, you specified options for the `showrule` command using optional input arguments `indexList`, `format`, and `lang`.

```matlab
text = showrule(fis,indexList,format,lang);
```

Starting in R2017a, if your code uses `newfis`, modify the code to use one or more `Name,Value` pair arguments. For example, view the first fuzzy rule in `fis`.

```matlab
text = showrule(fis,'RuleIndex',1);
```

The following table shows the mapping of the old input arguments to the new `Name,Value` pair arguments.

<table>
<thead>
<tr>
<th>Old showrule Input Argument</th>
<th>New Name,Value Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>indexList</td>
<td>'RuleIndex'</td>
</tr>
<tr>
<td>format</td>
<td>'Format'</td>
</tr>
<tr>
<td>lang</td>
<td>'Language'</td>
</tr>
</tbody>
</table>

subclust Command: Specify options using Name,Value pairs

To specify options for subtractive clustering using the `subclust` command, you now use `Name,Value` pair arguments. Any `Name,Value` pair arguments that you do not specify remain at their default values.

Compatibility Considerations

Previously, you specified options for the `subclust` command using optional input arguments `xBounds` and `options`.

```matlab
fisOut = subclust(fisIn,radii,xBounds,options);
```

Starting in R2017a, if your code uses `newfis`, modify the code to use one or more `Name,Value` pair arguments. For example, specify clustering options.

```matlab
fisOut = subclust(fisIn,radii,'Options',options);
```

The following table shows the mapping of the old input arguments to the new `Name,Value` pair arguments.

<table>
<thead>
<tr>
<th>Old subclust Input Argument</th>
<th>New Name,Value Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>xBounds</td>
<td>'DataScale'</td>
</tr>
<tr>
<td>options</td>
<td>'Options'</td>
</tr>
</tbody>
</table>
Obtain fuzzy inference system properties using improved `getfis` command

Several `getfis` syntaxes that previously printed formatted properties to the Command Window and also returned properties now perform a single action.

- `getfis(fis)` now just prints FIS properties to the Command Window.
- `getfis(fis,vartype,varindex)` now just returns variable properties in a structure.
- `getfis(fis,vartype,varindex,'mf',mfIndex)` now just returns membership function properties in a structure.

Compatibility Considerations

Starting in R2017a, the following `getfis` syntaxes have a new behavior:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Previous Behavior</th>
<th>New Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getfis(fis)</code></td>
<td>Print formatted list of FIS properties to Command Window, and return FIS name.</td>
<td>Print formatted list of FIS properties to Command Window.</td>
</tr>
<tr>
<td><code>getfis(fis,vartype,varIndex)</code></td>
<td>Print formatted list variable properties to Command Window, and return structure that contains variable properties.</td>
<td>Return structure that contains variable properties.</td>
</tr>
<tr>
<td><code>getfis(fis,vartype,varIndex,'mf',mfIndex)</code></td>
<td>Print formatted list membership function properties to Command Window, and return structure that contains membership function properties.</td>
<td>Return structure that contains membership function properties.</td>
</tr>
</tbody>
</table>

Functionality being removed or changed

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Result</th>
<th>Use This Instead</th>
<th>Compatibility Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>genfis1</code>, <code>genfis2</code>, and <code>genfis3</code> commands</td>
<td>Warns</td>
<td><code>genfis</code> command</td>
<td>See “Unified <code>genfis</code> Command: Generate fuzzy inference system structures using a single command” on page 7-2.</td>
</tr>
<tr>
<td>Specify optional input arguments for the <code>anfis</code> command</td>
<td>Still works</td>
<td>Specify options using <code>anfisOptions</code> command.</td>
<td>See “<code>anfisOptions</code> Command: Specify options for training adaptive neuro-fuzzy inference systems” on page 7-3.</td>
</tr>
<tr>
<td>Specify optional input arguments for the <code>gensurf</code> command</td>
<td>Still works</td>
<td>Specify options using <code>gensurfOptions</code> command.</td>
<td>See “<code>gensurfOptions</code> Command: Specify options for generating fuzzy inference system output surfaces” on page 7-4.</td>
</tr>
<tr>
<td>Specify optional input arguments for <code>newfis</code> command</td>
<td>Still works</td>
<td>Specify options using Name,Value pair arguments.</td>
<td>See “<code>newfis</code> Command: Specify options using Name,Value pairs” on page 7-4.</td>
</tr>
<tr>
<td>Functionality</td>
<td>Result</td>
<td>Use This Instead</td>
<td>Compatibility Considerations</td>
</tr>
<tr>
<td>------------------------------------------------------------------------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Specify optional input arguments for parsrule command</td>
<td>Still works</td>
<td>Specify options using Name,Value pair arguments.</td>
<td>See “parsrule Command: Specify options using Name,Value pairs” on page 7-5.</td>
</tr>
<tr>
<td>Specify optional input arguments for showrule command</td>
<td>Still works</td>
<td>Specify options using Name,Value pair arguments.</td>
<td>See “showrule Command: Specify options using Name,Value pairs” on page 7-6.</td>
</tr>
<tr>
<td>Specify optional input arguments for subclust command</td>
<td>Still works</td>
<td>Specify options using Name,Value pair arguments.</td>
<td>See “subclust Command: Specify options using Name,Value pairs” on page 7-6.</td>
</tr>
<tr>
<td>getfis command syntaxes that both print and return properties</td>
<td>Still works</td>
<td>Syntaxes now either print or return properties, not both.</td>
<td>See “Obtain fuzzy inference system properties using improved getfis command” on page 7-7.</td>
</tr>
</tbody>
</table>
R2016b

Version: 2.2.24

New Features

Bug Fixes
Standalone Applications for ANFIS Training: Deploy neuro-adaptive fuzzy inference code using MATLAB Compiler

The `anfis` command now supports application deployment using MATLAB Compiler™. For more information on building and deploying standalone applications from MATLAB programs, see MATLAB Compiler.
R2016a

Version: 2.2.23

Bug Fixes
R2015b

Version: 2.2.22

Bug Fixes
R2015a

Version: 2.2.21

Bug Fixes
R2014b

Version: 2.2.20

New Features

Bug Fixes
Commands to open Fuzzy Logic Designer and Neuro-Fuzzy Designer renamed

fuzzy is renamed to fuzzyLogicDesigner. Use this command to open the Fuzzy Logic Designer app.

anfisedit is renamed to neuroFuzzyDesigner. Use this command to open the Neuro-Fuzzy Designer app.
R2014a

Version: 2.2.19

New Features

Bug Fixes
Example that shows how to use a fuzzy inference system to detect edges in an image

The Fuzzy Logic Image Processing example shows how to use a fuzzy inference system to detect edges in an image.
Version: 2.2.18

Bug Fixes
R2013a

Version: 2.2.17

No New Features or Changes
R2012b

Version: 2.2.16

No New Features or Changes
R2012a

Version: 2.2.15

No New Features or Changes
Version: 2.2.14

No New Features or Changes
R2011a

Version: 2.2.13
No New Features or Changes
R2010b

Version: 2.2.12

No New Features or Changes
Version: 2.2.11
No New Features or Changes
R2009b

Version: 2.2.10

No New Features or Changes
R2009a

Version: 2.2.9

No New Features or Changes
R2008b

Version: 2.2.8

No New Features or Changes
R2008a

Version: 2.2.7

No New Features or Changes
R2007b

Version: 2.2.6

New Features
New Demo

Fuzzy Logic Toolbox software has a new demo Fuzzy C-Means Clustering for Iris Data, which illustrates the use of Fuzzy C-Means clustering for Iris dataset.
R2007a

Version: 2.2.5

No New Features or Changes
Version: 2.2.4

No New Features or Changes
R14SP3

Version: 2.2.2

No New Features or Changes
Version: 2.2.1

No New Features or Changes