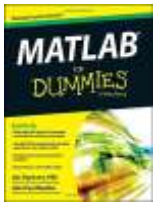
معرفی چند منبع در زمینه آموزش برنامه نویسی MATLAB یا متلب

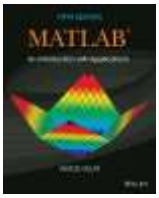
کتاب های به زبان انگلیسی

عنوان: Matlab, Third Edition: A Practical Introduction to Programming and Problem Solving
ترجمه عنوان: متلب: مقدمه ای عملی بر برنامه نویسی و حل مساله، چاپ سوم
مولفین: Stormy Attaway
سال چاپ: 2013
انتشارات: Butterworth-Heinemann
لینک دسترسی: لینک



عنوان: MATLAB For Dummies
ترجمه عنوان: تلب به زبان ساده
مولفین: Jim Sizemore, John Paul Mueller
سال چاپ: ۲۰۱۴
انتشارات: For Dummies
لینک دسترسی: لینک

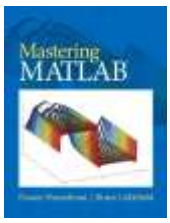
عنوان: Essential MATLAB for Engineers and Scientists
ترجمه عنوان: آنچه باید مهندسین و دانشمندان از متلب بدانند
مولفین: Brian Hahn, Daniel Valentine
سال چاپ: 2013
انتشارات: Academic Press
لینک دسترسی: لینک



عنوان: MATLAB: An Introduction with Applications
ترجمه عنوان: مقدمه ای بر متلب و کاربردهای آن
مولف: Amos Gilat
سال چاپ: 2014
انتشارات: Wiley
لینک دسترسی: لینک

عنوان: MATLAB For Beginners: A Gentle Approach
ترجمه عنوان: متلب برای افراد مبتدی با یک رویکرد تدریجی
مولف: Peter I. Kattan
سال چاپ: 2008
انتشارات: CreateSpace Independent Publishing Platform
لینک دسترسی: لینک

عنوان: MATLAB for Engineers
ترجمه عنوان: متلب برای مهندسین
مولف: Holly Moore
سال چاپ: 2011
انتشارات: Prentice Hall
لینک دسترسی: لینک

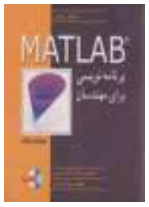


عنوان: Mastering MATLAB
ترجمه عنوان: تسلط بر متلب
مولفین: Duane C. Hanselman, Bruce L. Littlefield
سال چاپ: 2011
انتشارات: Prentice Hall
لینک دسترسی: لینک

کتاب های به زبان فارسی

عنوان: اصول و مبانی متلب برای علوم مهندسی
مولفین: برایان هان، دانیل تی، والنتین
مترجمین: رامین مولانا پور، سارا مولاناپور، نینا اسدی پور
انتشارات: سها دانش
لینک دسترسی: لینک

عنوان: کاربرد MATLAB در علوم مهندسی
مولفین: حیدرعلی شایانفر، حسین شایقی
انتشارات: یاوریان
لینک دسترسی: لینک



عنوان: برنامه نویسی MATLAB برای مهندسان
مولفین: محمود کشاورز مهر، بهزاد عبدی
انتشارات: نوپردازان
لینک دسترسی: لینک

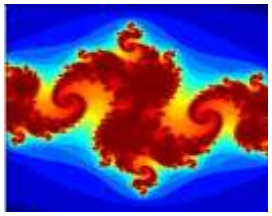
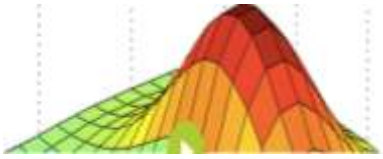
عنوان: آموزش کاربردی مباحث پیشرفته با MATLAB
مولفین: نیما جمشیدی، علی ابویی مهریزی، رسول مولایی
انتشارات: عابد
لینک دسترسی: لینک

عنوان: کاملترین مرجع آموزشی و کاربردی MATLAB
مولفین: علی اکبر علمداری، نسرین علمداری
انتشارات: نگارنده دانش
لینک دسترسی: لینک

عنوان: برنامه نویسی MATLAB برای مهندسین
مولف: استفن چاپمن
مترجم: سعدان زکائی
انتشارات: دانشگاه صنعتی خواجه نصیرالدین طوسی
لینک دسترسی: لینک

عنوان: آموزش گام به گام محاسبات عددی با متلب
مولف: کلیو مولر
مترجم: رسول نصیری
انتشارات: نشر گستر
لینک دسترسی: لینک

منابع آموزشی آنلاین

عنوان: مجموعه فرادرس‌های برنامه‌نویسی متلب **مدرس:** دکتر سید مصطفی کلامی هریس **مدت زمان:** ۹ ساعت و ۳ دقیقه **زبان:** فارسی **ارائه دهنده:** فرادرس **لینک دسترسی:** لینک	
عنوان: مجموعه فرادرس‌های متلب برای علوم و مهندسی **مدرس:** دکتر سید مصطفی کلامی هریس **مدت زمان:** ۱۴ ساعت و ۲۷ دقیقه **زبان:** فارسی **ارائه دهنده:** فرادرس **لینک دسترسی:** لینک	
عنوان: مجموعه فرادرس‌های برنامه نویسی متلب پیشرفته **مدرس:** دکتر سید مصطفی کلامی هریس **مدت زمان:** ۷ ساعت و ۱۲ دقیقه **زبان:** فارسی **ارائه دهنده:** فرادرس **لینک دسترسی:** لینک	
عنوان: Introduction to Programming with MATLAB **ترجمه عنوان:** آشنایی با برنامه‌نویسی متلب **مدرسین:** Akos Ledeczi, Michael Fitzpatrick, Robert Tairas **زبان:** انگلیسی **ارائه دهنده:** Vanderbilt University **لینک دسترسی:** لینک	
عنوان: Introduction to MATLAB **ترجمه عنوان:** مقدمه‌ای بر متلب **مدرس:** Danilo Šćepanović **زبان:** انگلیسی **ارائه دهنده:** MIT OCW **لینک دسترسی:** لینک	
عنوان: Up and Running with MATLAB **ترجمه عنوان:** شروع سریع کار با متلب **مدرس:** Patrick Royal **زبان:** انگلیسی **ارائه دهنده:** lynda.com **لینک دسترسی:** لینک	
عنوان: Modelling and Simulation using MATLAB **ترجمه عنوان:** مدل‌سازی و شبیه‌سازی با استفاده از متلب **مدرسین:** Prof. Dr.-Ing. Georg Fries و دیگران **زبان:** انگلیسی **ارائه دهنده:** iversity.org **لینک دسترسی:** لینک	

Fuzzy Logic Toolbox™

User's Guide

MATLAB®

R2015a

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

MathWorks®

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.
3 Apple Hill Drive
Natick, MA 01760-2098

Fuzzy Logic Toolbox™ User's Guide

© COPYRIGHT 1995–2015 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.

www.MatlabSite.com | متلب سایت

مرجع آموزش برنامه‌نویسی متلب در ایران

Revision History

January 1995	First printing	
April 1997	Second printing	
January 1998	Third printing	
September 2000	Fourth printing	Revised for Version 2 (Release 12)
April 2003	Fifth printing	
June 2004	Online only	Updated for Version 2.1.3 (Release 14)
March 2005	Online only	Updated for Version 2.2.1 (Release 14SP2)
September 2005	Online only	Updated for Version 2.2.2 (Release 14SP3)
March 2006	Online only	Updated for Version 2.2.3 (Release 2006a)
September 2006	Online only	Updated for Version 2.2.4 (Release 2006b)
March 2007	Online only	Updated for Version 2.2.5 (Release 2007a)
September 2007	Online only	Revised for Version 2.2.6 (Release 2007b)
March 2008	Online only	Revised for Version 2.2.7 (Release 2008a)
October 2008	Online only	Revised for Version 2.2.8 (Release 2008b)
March 2009	Online only	Revised for Version 2.2.9 (Release 2009a)
September 2009	Online only	Revised for Version 2.2.10 (Release 2009b)
March 2010	Online only	Revised for Version 2.2.11 (Release 2010a)
September 2010	Online only	Revised for Version 2.2.12 (Release 2010b)
April 2011	Online only	Revised for Version 2.2.13 (Release 2011a)
September 2011	Online only	Revised for Version 2.2.14 (Release 2011b)
March 2012	Online only	Revised for Version 2.2.15 (Release 2012a)
September 2012	Online only	Revised for Version 2.2.16 (Release 2012b)
March 2013	Online only	Revised for Version 2.2.17 (Release 2013a)
September 2013	Online only	Revised for Version 2.2.18 (Release 2013b)
March 2014	Online only	Revised for Version 2.2.19 (Release 2014a)
October 2014	Online only	Revised for Version 2.2.20 (Release 2014b)
March 2015	Online only	Revised for Version 2.2.21 (Release 2015a)

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

متلب سایت | www.MatlabSite.com
مرجع آموزش برنامه‌نویسی متلب در ایران

Contents

1 Getting Started

Fuzzy Logic Toolbox Product Description 1-2
Key Features 1-2

What Is Fuzzy Logic? 1-3
Description of Fuzzy Logic 1-3
Why Use Fuzzy Logic? 1-6
When Not to Use Fuzzy Logic 1-7
What Can Fuzzy Logic Toolbox Software Do? 1-8

Fuzzy vs. Nonfuzzy Logic 1-9

2 Tutorial

Foundations of Fuzzy Logic 2-2
Overview 2-2
Fuzzy Sets 2-3
Membership Functions 2-6
Logical Operations 2-11
If-Then Rules 2-15
References 2-18

Types of Fuzzy Inference Systems 2-20

Fuzzy Inference Process 2-22
Step 1. Fuzzify Inputs 2-23
Step 2. Apply Fuzzy Operator 2-24
Step 3. Apply Implication Method 2-25
Step 4. Aggregate All Outputs 2-25

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Step 5. Defuzzify 2-27
Fuzzy Inference Diagram 2-27

What Is Mamdani-Type Fuzzy Inference? 2-30

Build Mamdani Systems (GUI) 2-31
How to Build Mamdani Systems Using Fuzzy Logic Toolbox Graphical User Interface Tools 2-31
The Basic Tipping Problem 2-33
The Fuzzy Logic Designer 2-34
The Membership Function Editor 2-39
The Rule Editor 2-46
The Rule Viewer 2-49
The Surface Viewer 2-51
Importing and Exporting from the GUI Tools 2-53

Build Mamdani Systems Using Custom Functions 2-54
How to Build Fuzzy Inference Systems Using Custom Functions in the Designer 2-54
Specifying Custom Membership Functions 2-55
Specifying Custom Inference Functions 2-60

Build Mamdani Systems (Code) 2-65
Tipping Problem from the Command Line 2-65
System Display Functions 2-67
Building a System from Scratch 2-70
FIS Evaluation 2-73
The FIS Structure 2-73

Simulate Fuzzy Inference Systems in Simulink 2-78

Build Your Own Fuzzy Simulink Models 2-86
About the Fuzzy Logic Controller Block 2-86
About the Fuzzy Logic Controller with Ruleviewer Block . . . 2-87
Initializing Fuzzy Logic Controller Blocks 2-87
Example: Cart and Pole Simulation 2-88

What Is Sugeno-Type Fuzzy Inference? 2-91

Comparison of Sugeno and Mamdani Systems 2-97
Advantages of the Sugeno Method 2-97
Advantages of the Mamdani Method 2-97

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

About the Fuzzy Inference Engine 2-98

Simulate Systems on Windows Platforms 2-99
Including the `lcc` Command in the PATH Environment Variable 2-101
Including the `lcclnk` Command in the PATH Environment Variable 2-101

Simulate Systems on UNIX Platforms 2-103

3 Adaptive Neuro-Fuzzy Modeling

Neuro-Adaptive Learning and ANFIS 3-2
When to Use Neuro-Adaptive Learning 3-2
Model Learning and Inference Through ANFIS 3-3
References 3-5

Comparison of anfis and Neuro-Fuzzy Designer Functionality 3-7
Training Data 3-7
Input FIS Structure 3-7
Training Options 3-8
Display Options 3-9
Method 3-9
Output FIS Structure for Training Data 3-10
Training Error 3-10
Step-Size 3-10
Checking Data 3-11
Output FIS Structure for Checking Data 3-11
Checking Error 3-12

Train Adaptive Neuro-Fuzzy Inference Systems (GUI) 3-13
Loading, Plotting, and Clearing the Data 3-14
Generating or Loading the Initial FIS Structure 3-15
Training the FIS 3-15
Validating the Trained FIS 3-16

Test Data Against Trained System (GUI) 3-18
Checking Data Helps Model Validation 3-18

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Checking Data Does Not Validate Model 3-26

Save Training Error Data to MATLAB Workspace 3-31

Predict Chaotic Time-Series 3-39

Modeling Inverse Kinematics in a Robotic Arm 3-46

4 Data Clustering

Fuzzy Clustering 4-2
What Is Data Clustering? 4-2
Fuzzy C-Means Clustering 4-2
Subtractive Clustering 4-3
References 4-3

Cluster Quasi-Random Data Using Fuzzy C-Means Clustering 4-4

Model Suburban Commuting Using Subtractive Clustering 4-8

Data Clustering Using the Clustering Tool 4-20
Loading and Plotting the Data 4-21
Starting the Clustering 4-21
Saving the Cluster Center 4-22

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

5 Functions — Alphabetical List

6 Blocks — Alphabetical List

A Bibliography

Glossary

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

متلب سایت | www.MatlabSite.com
مرجع آموزش برنامه‌نویسی متلب در ایران

Getting Started

- "Fuzzy Logic Toolbox Product Description" on page 1-2
- "What Is Fuzzy Logic?" on page 1-3
- "Fuzzy vs. Nonfuzzy Logic" on page 1-9

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Fuzzy Logic Toolbox Product Description

Design and simulate fuzzy logic systems

Fuzzy Logic Toolbox™ provides MATLAB® functions, apps, and a Simulink® block for analyzing, designing, and simulating systems based on fuzzy logic. The product guides you through the steps of designing fuzzy inference systems. Functions are provided for many common methods, including fuzzy clustering and adaptive neurofuzzy learning.

The toolbox lets you model complex system behaviors using simple logic rules, and then implement these rules in a fuzzy inference system. You can use it as a stand-alone fuzzy inference engine. Alternatively, you can use fuzzy inference blocks in Simulink and simulate the fuzzy systems within a comprehensive model of the entire dynamic system.

Key Features

- Fuzzy Logic Design app for building fuzzy inference systems and viewing and analyzing results
- Membership functions for creating fuzzy inference systems
- Support for AND, OR, and NOT logic in user-defined rules
- Standard Mamdani and Sugeno-type fuzzy inference systems
- Automated membership function shaping through neuroadaptive and fuzzy clustering learning techniques
- Ability to embed a fuzzy inference system in a Simulink model
- Ability to generate embeddable C code or stand-alone executable fuzzy inference engines

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

What Is Fuzzy Logic?

In this section...

"Description of Fuzzy Logic" on page 1-3
"Why Use Fuzzy Logic?" on page 1-6
"When Not to Use Fuzzy Logic" on page 1-7
"What Can Fuzzy Logic Toolbox Software Do?" on page 1-8

Description of Fuzzy Logic

In recent years, the number and variety of applications of fuzzy logic have increased significantly. The applications range from consumer products such as cameras, camcorders, washing machines, and microwave ovens to industrial process control, medical instrumentation, decision-support systems, and portfolio selection.

To understand why use of fuzzy logic has grown, you must first understand what is meant by fuzzy logic.

Fuzzy logic has two different meanings. In a narrow sense, fuzzy logic is a logical system, which is an extension of multivalued logic. However, in a wider sense fuzzy logic (FL) is almost synonymous with the theory of fuzzy sets, a theory which relates to classes of objects with unsharp boundaries in which membership is a matter of degree. In this perspective, fuzzy logic in its narrow sense is a branch of FL. Even in its more narrow definition, fuzzy logic differs both in concept and substance from traditional multivalued logical systems.

In Fuzzy Logic Toolbox software, fuzzy logic should be interpreted as FL, that is, fuzzy logic in its wide sense. The basic ideas underlying FL are explained very clearly and insightfully in "Foundations of Fuzzy Logic" on page 2-2. What might be added is that the basic concept underlying FL is that of a linguistic variable, that is, a variable whose values are words rather than numbers. In effect, much of FL may be viewed as a methodology for computing with words rather than numbers. Although words are inherently less precise than numbers, their use is closer to human intuition. Furthermore, computing with words exploits the tolerance for imprecision and thereby lowers the cost of solution.

Another basic concept in FL, which plays a central role in most of its applications, is that of a fuzzy if-then rule or, simply, fuzzy rule. Although rule-based systems have a

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

long history of use in Artificial Intelligence (AI), what is missing in such systems is a mechanism for dealing with fuzzy consequents and fuzzy antecedents. In fuzzy logic, this mechanism is provided by the calculus of fuzzy rules. The calculus of fuzzy rules serves as a basis for what might be called the Fuzzy Dependency and Command Language (FDCL). Although FDCL is not used explicitly in the toolbox, it is effectively one of its principal constituents. In most of the applications of fuzzy logic, a fuzzy logic solution is, in reality, a translation of a human solution into FDCL.

A trend that is growing in visibility relates to the use of fuzzy logic in combination with neurocomputing and genetic algorithms. More generally, fuzzy logic, neurocomputing, and genetic algorithms may be viewed as the principal constituents of what might be called soft computing. Unlike the traditional, hard computing, *soft computing* accommodates the imprecision of the real world. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, and low solution cost. In the future, soft computing could play an increasingly important role in the conception and design of systems whose MIQ (Machine IQ) is much higher than that of systems designed by conventional methods.

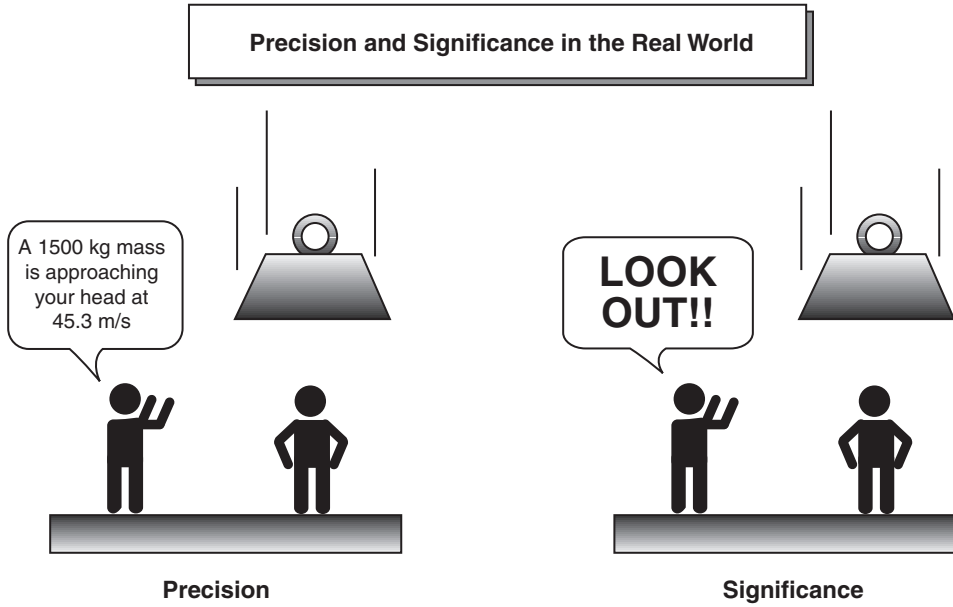
Among various combinations of methodologies in soft computing, the one that has highest visibility at this juncture is that of fuzzy logic and neurocomputing, leading to neuro-fuzzy systems. Within fuzzy logic, such systems play a particularly important role in the induction of rules from observations. An effective method developed by Dr. Roger Jang for this purpose is called ANFIS (Adaptive Neuro-Fuzzy Inference System). This method is an important component of the toolbox.

Fuzzy logic is all about the relative importance of precision: How important is it to be exactly right when a rough answer will do?

You can use Fuzzy Logic Toolbox software with MATLAB technical computing software as a tool for solving problems with fuzzy logic. Fuzzy logic is a fascinating area of research because it does a good job of trading off between significance and precision—something that humans have been managing for a very long time.

In this sense, fuzzy logic is both old and new because, although the modern and methodical science of fuzzy logic is still young, the concepts of fuzzy logic relies on age-old skills of human reasoning.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

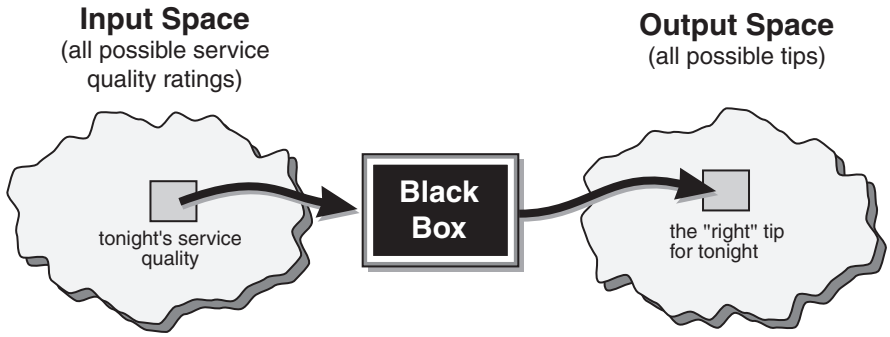


Fuzzy logic is a convenient way to map an input space to an output space. Mapping input to output is the starting point for everything. Consider the following examples:

- With information about how good your service was at a restaurant, a fuzzy logic system can tell you what the tip should be.
- With your specification of how hot you want the water, a fuzzy logic system can adjust the faucet valve to the right setting.
- With information about how far away the subject of your photograph is, a fuzzy logic system can focus the lens for you.
- With information about how fast the car is going and how hard the motor is working, a fuzzy logic system can shift gears for you.

A graphical example of an input-output map is shown in the following figure.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



An input-output map for the tipping problem:
"Given the quality of service, how much should I tip?"

To determine the appropriate amount of tip requires mapping inputs to the appropriate outputs. Between the input and the output, the preceding figure shows a black box that can contain any number of things: fuzzy systems, linear systems, expert systems, neural networks, differential equations, interpolated multidimensional lookup tables, or even a spiritual advisor, just to name a few of the possible options. Clearly the list could go on and on.

Of the dozens of ways to make the black box work, it turns out that fuzzy is often the very best way. Why should that be? As Lotfi Zadeh, who is considered to be the father of fuzzy logic, once remarked: "In almost every case you can build the same product without fuzzy logic, but fuzzy is faster and cheaper."

Why Use Fuzzy Logic?

Here is a list of general observations about fuzzy logic:

- Fuzzy logic is conceptually easy to understand.

  The mathematical concepts behind fuzzy reasoning are very simple. Fuzzy logic is a more intuitive approach without the far-reaching complexity.
- Fuzzy logic is flexible.

  With any given system, it is easy to layer on more functionality without starting again from scratch.
- Fuzzy logic is tolerant of imprecise data.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Everything is imprecise if you look closely enough, but more than that, most things are imprecise even on careful inspection. Fuzzy reasoning builds this understanding into the process rather than tacking it onto the end.

- Fuzzy logic can model nonlinear functions of arbitrary complexity.

  You can create a fuzzy system to match any set of input-output data. This process is made particularly easy by adaptive techniques like Adaptive Neuro-Fuzzy Inference Systems (ANFIS), which are available in Fuzzy Logic Toolbox software.

- Fuzzy logic can be built on top of the experience of experts.

  In direct contrast to neural networks, which take training data and generate opaque, impenetrable models, fuzzy logic lets you rely on the experience of people who already understand your system.

- Fuzzy logic can be blended with conventional control techniques.

  Fuzzy systems don't necessarily replace conventional control methods. In many cases fuzzy systems augment them and simplify their implementation.

- Fuzzy logic is based on natural language.

  The basis for fuzzy logic is the basis for human communication. This observation underpins many of the other statements about fuzzy logic. Because fuzzy logic is built on the structures of qualitative description used in everyday language, fuzzy logic is easy to use.

The last statement is perhaps the most important one and deserves more discussion. Natural language, which is used by ordinary people on a daily basis, has been shaped by thousands of years of human history to be convenient and efficient. Sentences written in ordinary language represent a triumph of efficient communication.

When Not to Use Fuzzy Logic

Fuzzy logic is not a cure-all. When should you not use fuzzy logic? The safest statement is the first one made in this introduction: fuzzy logic is a convenient way to map an input space to an output space. If you find it's not convenient, try something else. If a simpler solution already exists, use it. Fuzzy logic is the codification of common sense — use common sense when you implement it and you will probably make the right decision. Many controllers, for example, do a fine job without using fuzzy logic. However, if you take the time to become familiar with fuzzy logic, you'll see it can be a very powerful tool for dealing quickly and efficiently with imprecision and nonlinearity.

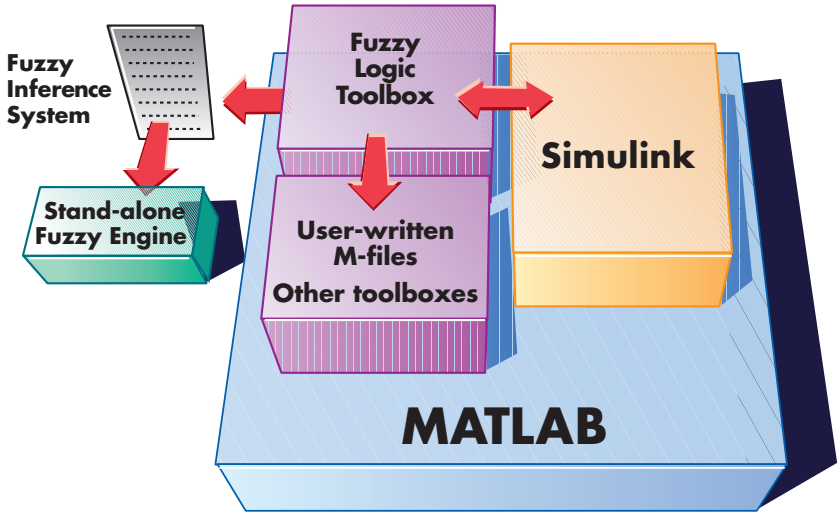
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

What Can Fuzzy Logic Toolbox Software Do?

You can create and edit fuzzy inference systems with Fuzzy Logic Toolbox software. You can create these systems using graphical tools or command-line functions, or you can generate them automatically using either clustering or adaptive neuro-fuzzy techniques.

If you have access to Simulink software, you can easily test your fuzzy system in a block diagram simulation environment.

The toolbox also lets you run your own stand-alone C programs directly. This is made possible by a stand-alone Fuzzy Inference Engine that reads the fuzzy systems saved from a MATLAB session. You can customize the stand-alone engine to build fuzzy inference into your own code. All provided code is ANSI® compliant.



Because of the integrated nature of the MATLAB environment, you can create your own tools to customize the toolbox or harness it with another toolbox, such as the Control System Toolbox™, Neural Network Toolbox™, or Optimization Toolbox™ software.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Fuzzy vs. Nonfuzzy Logic

The Basic Tipping Problem

To illustrate the value of fuzzy logic, examine both linear and fuzzy approaches to the following problem:

What is the right amount to tip your waitperson?

First, work through this problem the conventional (nonfuzzy) way, writing MATLAB® commands that spell out linear and piecewise-linear relations. Then, look at the same system using fuzzy logic.

The Basic Tipping Problem. Given a number between 0 and 10 that represents the quality of service at a restaurant (where 10 is excellent), what should the tip be?

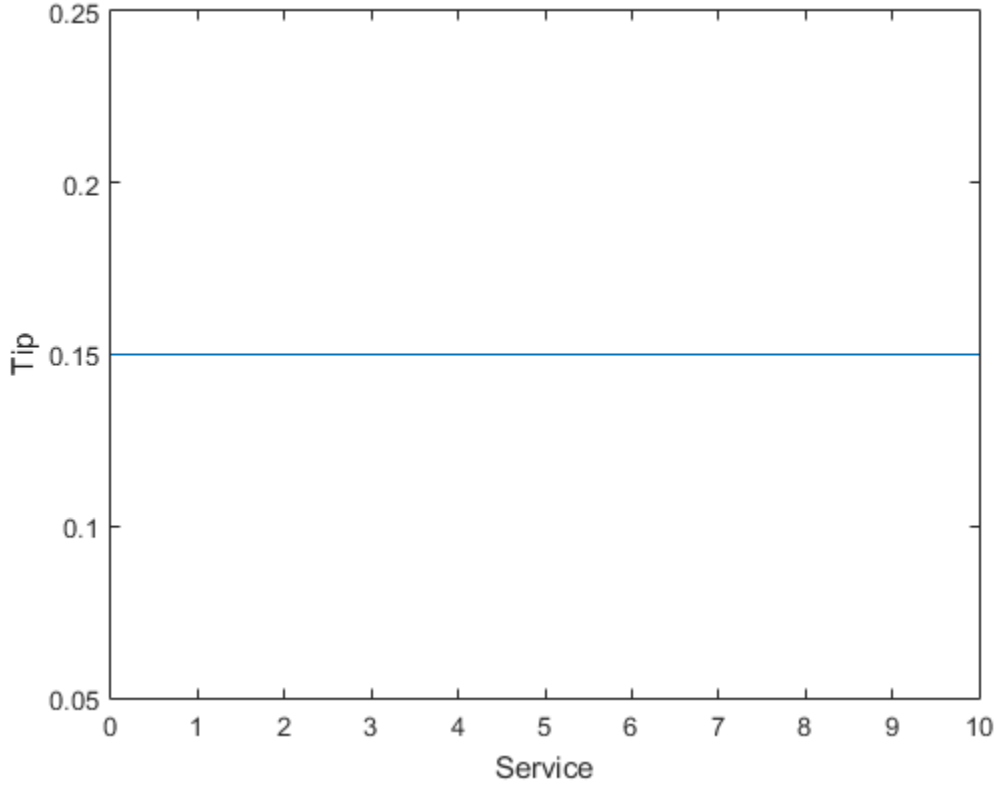
(This problem is based on tipping as it is typically practiced in the United States. An average tip for a meal in the U.S. is 15%, though the actual amount may vary depending on the quality of the service provided.)

The Nonfuzzy Approach

Begin with the simplest possible relationship. Suppose that the tip always equals 15% of the total bill.

```
service = 0:.5:10;
tip = 0.15*ones(size(service));
plot(service,tip);
xlabel('Service'); ylabel('Tip');
ylim([0.05 0.25])
```

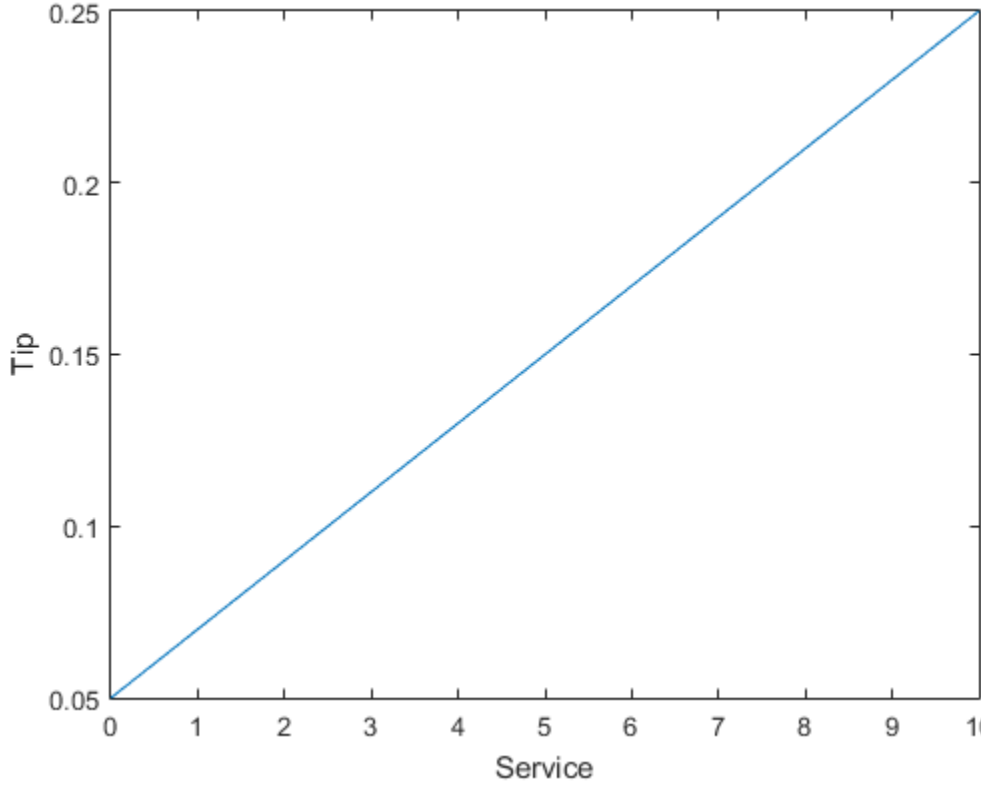

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



This relationship does not take into account the quality of the service, so you need to add a new term to the equation. Because service is rated on a scale of 0 to 10, you might have the tip go linearly from 5% if the service is bad to 25% if the service is excellent. Now the relation looks like the following plot:

```
tip = (.20/10)*service+0.05;
plot(service,tip);
xlabel('Service'); ylabel('Tip');
ylim([0.05 0.25])
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



The formula does what you want it to do, and is straight forward. However, you may want the tip to reflect the quality of the food as well. This extension of the problem is defined as follows.

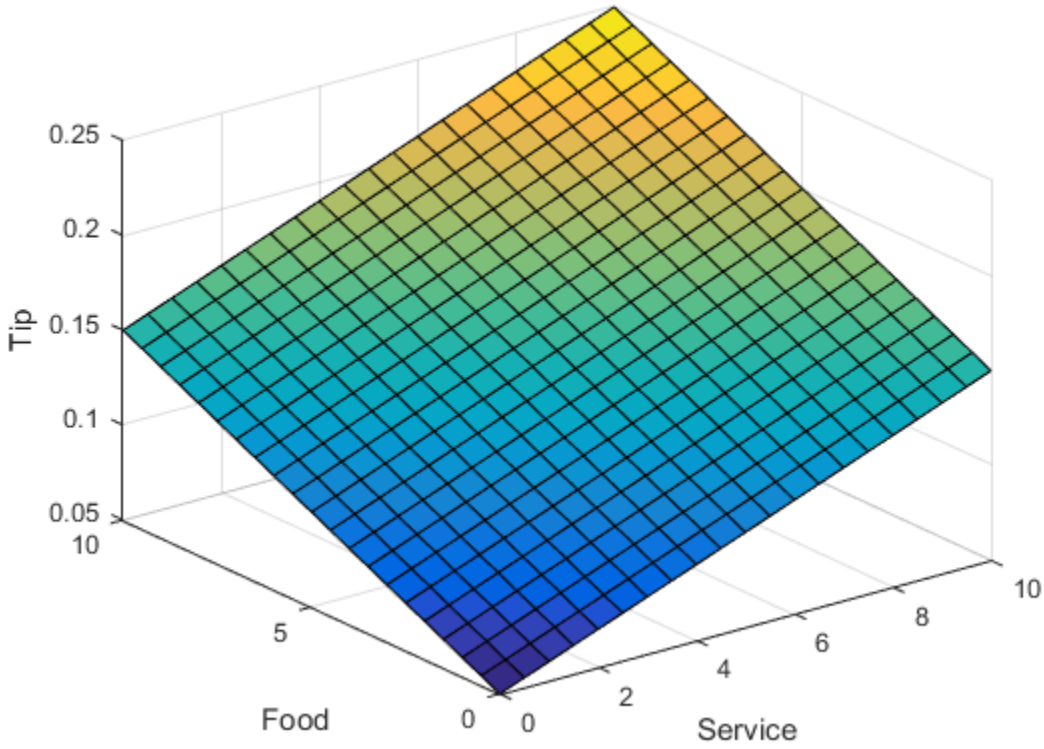
The Extended Tipping Problem. Given two sets of numbers between 0 and 10 (where 10 is excellent) that respectively represent the quality of the service and the quality of the food at a restaurant, what should the tip be?

See how the formula is affected now that you have added another variable. Try the following equation:

```
food = 0:.5:10;
[F,S]=meshgrid(food,service);
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
tip = (0.20/20).*(S+F)+0.05;
surf(S,F,tip);
xlabel('Service'); ylabel('Food');zlabel('Tip');
```

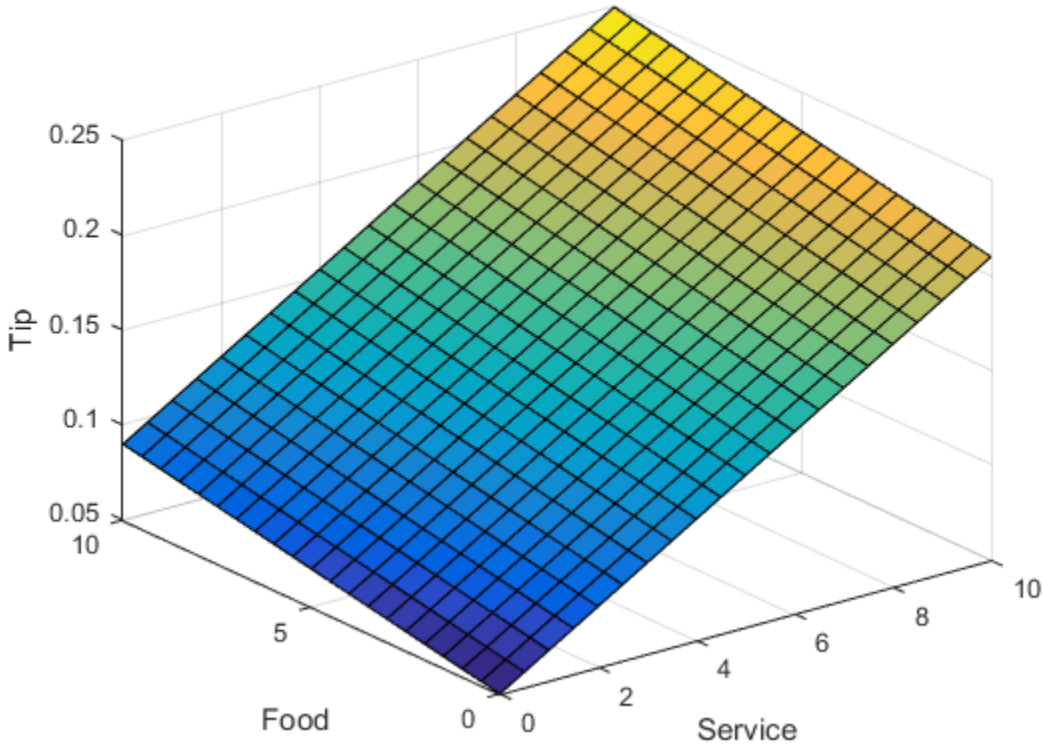


In this case, the results look satisfactory, but when you look at them closely, they do not seem quite right. Suppose you want the service to be a more important factor than the food quality. Specify that service accounts for 80% of the overall tipping grade and the food makes up the other 20%. Try this equation:

```
servRatio = 0.8;
tip = servRatio*(0.20/10*S+0.05) + ...
   (1-servRatio)*(0.20/10*F+0.05);
surf(S,F,tip);
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
xlabel('Service'); ylabel('Food'); zlabel('Tip');
```

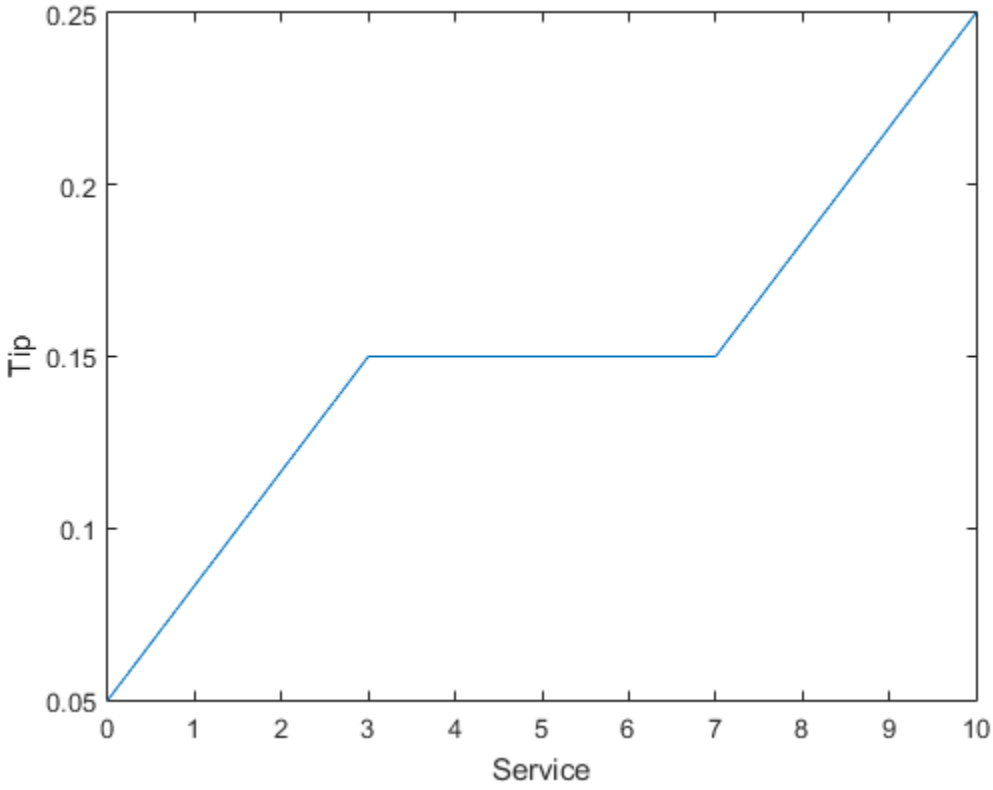


The response is still some how too uniformly linear. Suppose you want more of a flat response in the middle, i.e., you want to give a 15% tip in general, but want to also specify a variation if the service is exceptionally good or bad. This factor, in turn, means that the previous linear mappings no longer apply. You can still use the linear calculation with a piecewise linear construction. Now, return to the one-dimensional problem of just considering the service. You can string together a simple conditional tip assignment, using logical indexing.

```
tip = zeros(size(service));
tip(service<3) = (0.10/3)*service(service<3)+0.05;
tip(service>=3 & service<7) = 0.15;
tip(service>=7 & service<=10) = ...
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
   (0.10/3)*(service(service>=7 & service<=10)-7)+0.15;
plot(service,tip);
xlabel('Service'); ylabel('Tip');
ylim([0.05 0.25]);
```

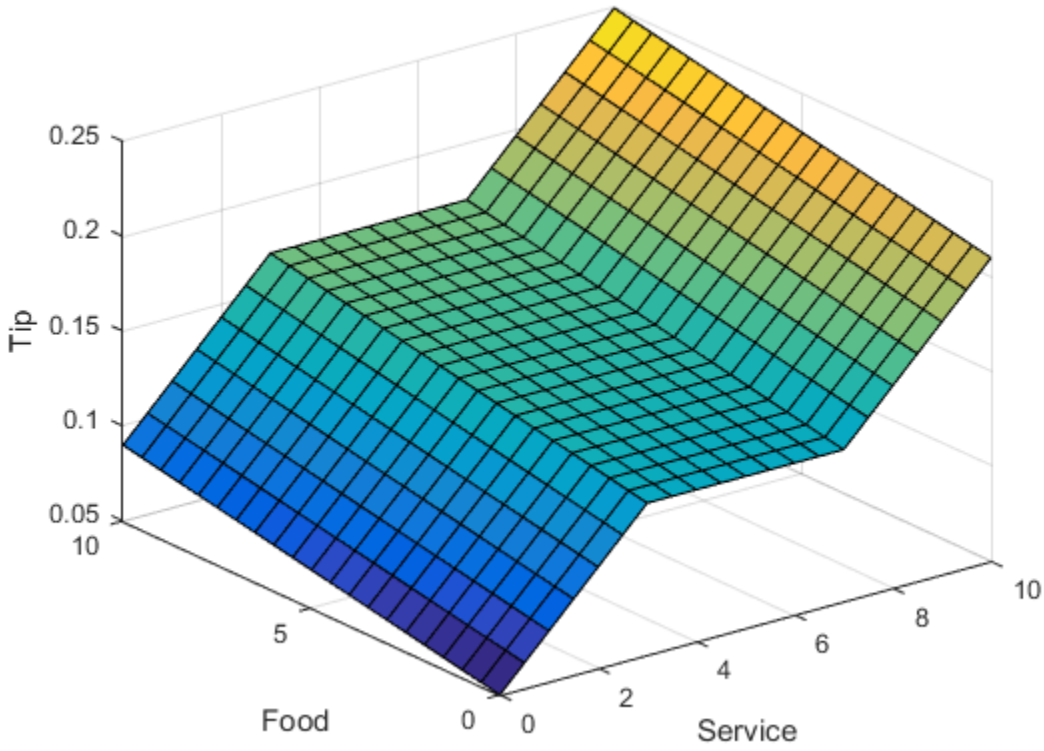


Suppose you extend this to two dimensions, where you take food into account again.

```
servRatio=0.8;
tip = zeros(size(S));
tip(S<3) = ((0.10/3)*S(S<3)+0.05)*servRatio + ...
   (1-servRatio)*(0.20/10*F(S<3)+0.05);
tip(S>=3 & S<7) = (0.15)*servRatio + ...
   (1-servRatio)*(0.20/10*F(S>=3 & S<7)+0.05);
tip(S>=7 & S<=10) = ((0.10/3)*(S(S>=7 & S<=10)-7)+0.15)*servRatio + ...
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
        (1-servRatio)*(0.20/10*F(S>=7 & S<=10)+0.05);
surf(S,F,tip);
xlabel('Service'); ylabel('Food');zlabel('Tip');
```



The plot looks good, but the function is surprisingly complicated. It was a little difficult to code this correctly, and it is definitely not easy to modify this code in the future. Moreover, it is even less apparent how the algorithm works to someone who did not see the original design process.

The Fuzzy Logic Approach

You need to capture the essentials of this problem, leaving aside all the factors that could be arbitrary. If you make a list of what really matters in this problem, you might end up with the following rule descriptions.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Tipping Problem Rules - Service Factor

- If service is poor, then tip is cheap
- If service is good, then tip is average
- If service is excellent, then tip is generous

The order in which the rules are presented here is arbitrary. It does not matter which rules come first. If you want to include the food's effect on the tip, add the following two rules.

Tipping Problem Rules - Food Factor

- If food is rancid, then tip is cheap
- If food is delicious, then tip is generous

You can combine the two different lists of rules into one tight list of three rules like so.

Tipping Problem Rules - Both Service and Food Factors

- If service is poor or the food is rancid, then tip is cheap
- If service is good, then tip is average
- If service is excellent or food is delicious, then tip is generous

These three rules are the core of your solution. Coincidentally, you have just defined the rules for a fuzzy logic system. When you give mathematical meaning to the linguistic variables (what is an average tip, for example) you have a complete fuzzy inference system. The methodology of fuzzy logic must also consider:

- How are the rules all combined?
- How do I define mathematically what an average tipis?

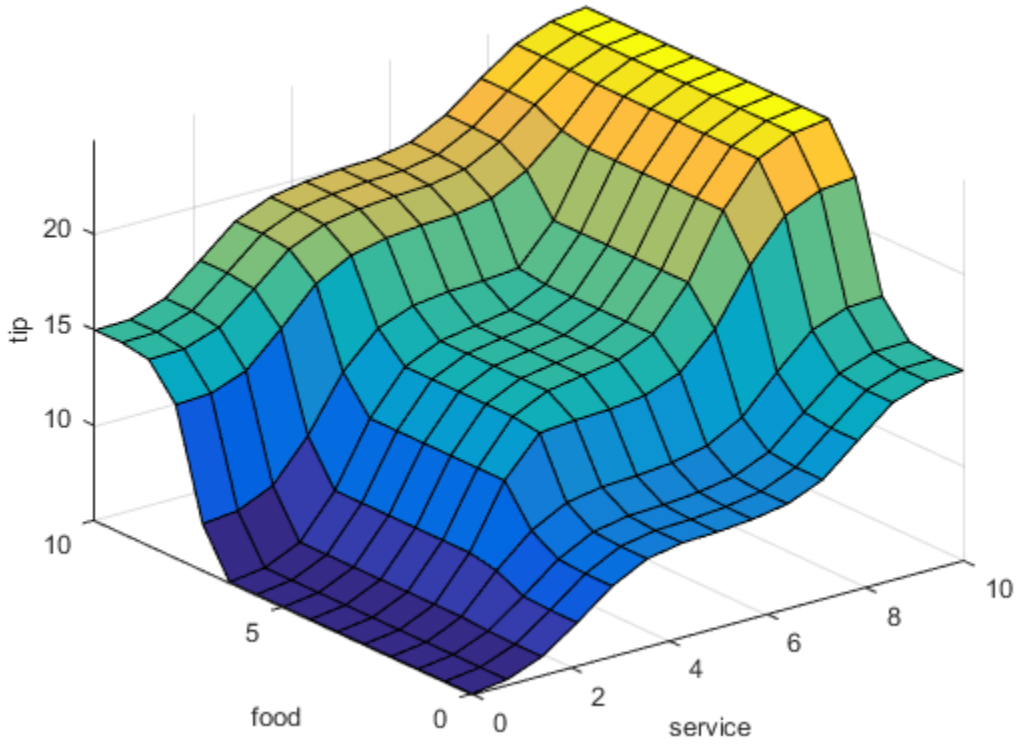
See other sections of the documentation for detailed answers to these questions. The details of the method don't really change much from problem to problem - the mechanics of fuzzy logic aren't terribly complex. What matters is that you understand that fuzzy logic is adaptable, simple, and easily applied.

Problem Solution

The following plot represents the fuzzy logic system that solves the tipping problem.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
gensurf(readfis('tipper'));
```



This plot was generated by the three rules that accounted for both service and food factors. The mechanics of how fuzzy inference works is explained in the Overview section of Foundations of Fuzzy Logic topic. In the topic, Build Mamdani Systems (GUI), the entire tipping problem is worked through using the Fuzzy Logic Toolbox (TM) apps.

Observations Consider some observations about the example so far. You found a piecewise linear relation that solved the problem. It worked, but it was problematic to derive, and when you wrote it down as code, it was not very easy to interpret. Conversely, the fuzzy logic system is based on some common sense statements. Also, you were able to add two more rules to the bottom of the list that influenced the shape of the overall

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

output without needing to undo what had already been done. Making the subsequent modification was relatively easy.

Moreover, by using fuzzy logic rules, the maintenance of the structure of the algorithm decouples along fairly clean lines. The notion of an average tip might change from day to day, city to city, country to country, but the underlying logic is the same: if the service is good, the tip should be average.

Recalibrating the Method You can recalibrate the method quickly by simply shifting the fuzzy set that defines average without rewriting the fuzzy logic rules.

You can shift lists of piecewise linear functions, but there is a greater likelihood that recalibration will not be so quick and simple.

In the following example, the piecewise linear tipping problem slightly rewritten to make it more generic. It performs the same function as before, only now the constants can be easily changed.

```
% Establish constants
lowTip = 0.05; averTip = 0.15; highTip = 0.25;
tipRange = highTip-lowTip;
badService = 0; okayService = 3;
goodService = 7; greatService = 10;
serviceRange = greatService-badService;
badFood = 0; greatFood = 10;
foodRange = greatFood-badFood;

% If service is poor or food is rancid, tip is cheap
if service<okayService,
  tip = (((averTip-lowTip)/(okayService-badService)) ...
      *service+lowTip)*servRatio + ...
      (1-servRatio)*(tipRange/foodRange*food+lowTip);

% If service is good, tip is average
elseif service<goodService,
  tip = averTip*servRatio + (1-servRatio)* ...
        (tipRange/foodRange*food+lowTip);

% If service is excellent or food is delicious, tip is generous
else,
  tip = (((highTip-averTip)/ ...
        (greatService-goodService))* ...
        (service-goodService)+averTip)*servRatio + ...
        (1-servRatio)*(tipRange/foodRange*food+lowTip);
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
end
```

As with all code, the more generality that is introduced, the less precise the algorithm becomes. You can improve clarity by adding more comments, or perhaps rewriting the algorithm in slightly more self-evident ways. But, the piecewise linear methodology is not the optimal way to resolve this issue.

If you remove everything from the algorithm except for three comments, what remain are exactly the fuzzy logic rules you previously wrote down.

```
% If service is poor or food is rancid, tip is cheap
% If service is good, tip is average
% If service is excellent or food is delicious, tip is generous
```

If, as with a fuzzy system, the comment is identical with the code, think how much more likely your code is to have comments. Fuzzy logic uses the language that is clearest to you, high level comments, also have meaning to the machine, which is why it is a very successful technique for bridging the gap between people and machines.

By making the equations as simple as possible (linear) you make things simpler for the machine, but more complicated for you. However, the limitation is really no longer the computer - it is your mental model of what the computer is doing. Computers have the ability to make things hopelessly complex; fuzzy logic reclaims the middleground and lets the machine work with your preferences rather than the other way around.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

متلب سایت | www.MatlabSite.com
مرجع آموزش برنامه‌نویسی متلب در ایران

Tutorial

- "Foundations of Fuzzy Logic" on page 2-2
- "Types of Fuzzy Inference Systems" on page 2-20
- "Fuzzy Inference Process" on page 2-22
- "What Is Mamdani-Type Fuzzy Inference?" on page 2-30
- "Build Mamdani Systems (GUI)" on page 2-31
- "Build Mamdani Systems Using Custom Functions" on page 2-54
- "Build Mamdani Systems (Code)" on page 2-65
- "Simulate Fuzzy Inference Systems in Simulink" on page 2-78
- "Build Your Own Fuzzy Simulink Models" on page 2-86
- "What Is Sugeno-Type Fuzzy Inference?" on page 2-91
- "Comparison of Sugeno and Mamdani Systems" on page 2-97
- "About the Fuzzy Inference Engine" on page 2-98
- "Simulate Systems on Windows Platforms" on page 2-99
- "Simulate Systems on UNIX Platforms" on page 2-103

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

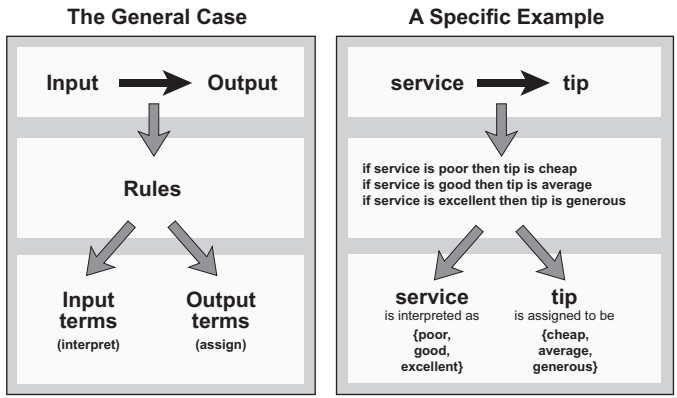
Foundations of Fuzzy Logic

In this section...

"Overview" on page 2-2
"Fuzzy Sets" on page 2-3
"Membership Functions" on page 2-6
"Logical Operations" on page 2-11
"If-Then Rules" on page 2-15
"References" on page 2-18

Overview

The point of fuzzy logic is to map an input space to an output space, and the primary mechanism for doing this is a list of if-then statements called rules. All rules are evaluated in parallel, and the order of the rules is unimportant. The rules themselves are useful because they refer to variables and the adjectives that describe those variables. Before you can build a system that interprets rules, you must define all the terms you plan on using and the adjectives that describe them. To say that the water is hot, you need to define the range that the water's temperature can be expected to vary as well as what we mean by the word *hot*. The following diagram provides a roadmap for the fuzzy inference process. It shows the general description of a fuzzy system on the left and a specific fuzzy system on the right.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

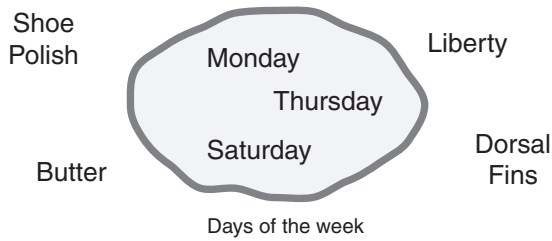
To summarize the concept of fuzzy inference depicted in this figure, *fuzzy inference is a method that interprets the values in the input vector and, based on some set of rules, assigns values to the output vector*.

This topic guides you through the fuzzy logic process step by step by providing an introduction to the theory and practice of fuzzy logic.

Fuzzy Sets

Fuzzy logic starts with the concept of a fuzzy set. A *fuzzy set* is a set without a crisp, clearly defined boundary. It can contain elements with only a partial degree of membership.

To understand what a fuzzy set is, first consider the definition of a *classical set*. A classical set is a container that wholly includes or wholly excludes any given element. For example, the set of days of the week unquestionably includes Monday, Thursday, and Saturday. It just as unquestionably excludes butter, liberty, and dorsal fins, and so on.



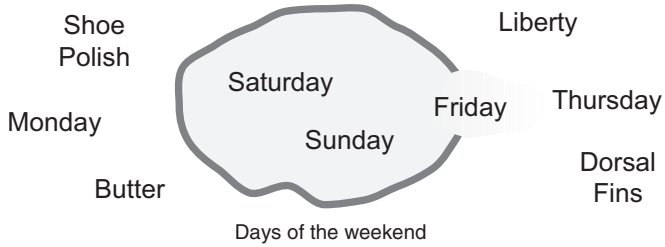
This type of set is called a classical set because it has been around for a long time. It was Aristotle who first formulated the Law of the Excluded Middle, which says X must either be in set A or in set not-A. Another version of this law is:

Of any subject, one thing must be either asserted or denied.

To restate this law with annotations: “Of any subject (say Monday), one thing (a day of the week) must be either asserted or denied (I assert that Monday is a day of the week).” This law demands that opposites, the two categories A and not-A, should between them contain the entire universe. Everything falls into either one group or the other. There is no thing that is both a day of the week and not a day of the week.

Now, consider the set of days comprising a weekend. The following diagram attempts to classify the weekend days.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Most would agree that Saturday and Sunday belong, but what about Friday? It feels like a part of the weekend, but somehow it seems like it should be technically excluded. Thus, in the preceding diagram, Friday tries its best to "straddle on the fence." Classical or normal sets would not tolerate this kind of classification. Either something is in or it is out. Human experience suggests something different, however, straddling the fence is part of life.

Of course individual perceptions and cultural background must be taken into account when you define what constitutes the weekend. Even the dictionary is imprecise, defining the weekend as the period from Friday night or Saturday to Monday morning. You are entering the realm where sharp-edged, yes-no logic stops being helpful. Fuzzy reasoning becomes valuable exactly when you work with how people really perceive the concept *weekend* as opposed to a simple-minded classification useful for accounting purposes only. More than anything else, the following statement lays the foundations for fuzzy logic.

In fuzzy logic, the truth of any statement becomes a matter of degree.

Any statement can be fuzzy. The major advantage that fuzzy reasoning offers is the ability to reply to a yes-no question with a not-quite-yes-or-no answer. Humans do this kind of thing all the time (think how rarely you get a straight answer to a seemingly simple question), but it is a rather new trick for computers.

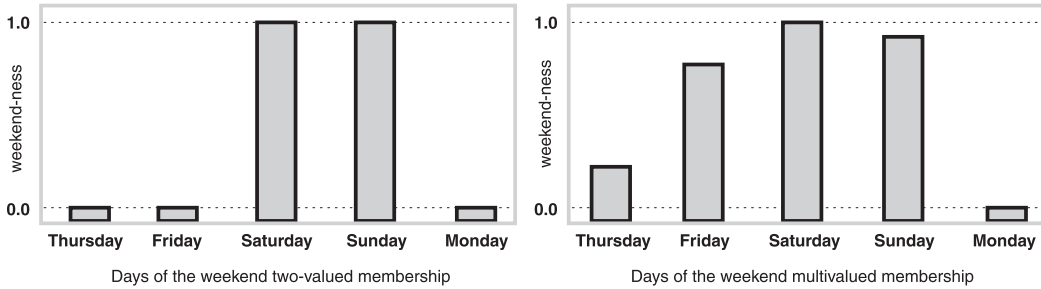
How does it work? Reasoning in fuzzy logic is just a matter of generalizing the familiar yes-no (Boolean) logic. If you give true the numerical value of 1 and false the numerical value of 0, this value indicates that fuzzy logic also permits in-between values like 0.2 and 0.7453. For instance:

Q: Is Saturday a weekend day?
A: 1 (yes, or true)
Q: Is Tuesday a weekend day?
A: 0 (no, or false)

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

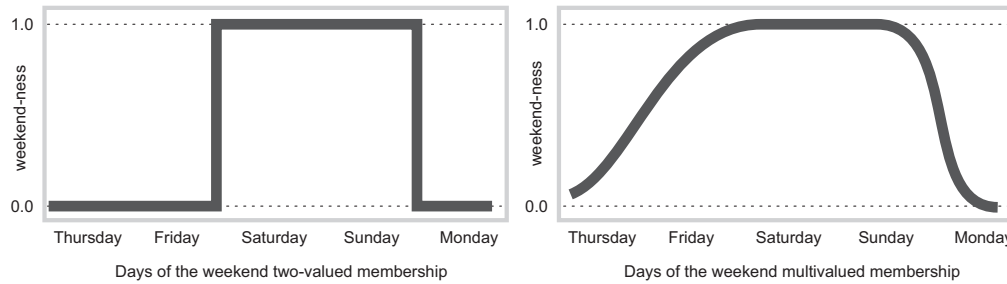
Q: Is Friday a weekend day?
A: 0.8 (for the most part yes, but not completely)
Q: Is Sunday a weekend day?
A: 0.95 (yes, but not quite as much as Saturday).

The following plot on the left shows the truth values for weekend-ness if you are forced to respond with an absolute yes or no response. On the right, is a plot that shows the truth value for weekend-ness if you are allowed to respond with fuzzy in-between values.



Technically, the representation on the right is from the domain of *multivalued logic* (or multivalent logic). If you ask the question "Is X a member of set A?" the answer might be yes, no, or any one of a thousand intermediate values in between. Thus, X might have partial membership in A. Multivalued logic stands in direct contrast to the more familiar concept of two-valued (or bivalent yes-no) logic.

To return to the example, now consider a continuous scale time plot of weekend-ness shown in the following plots.



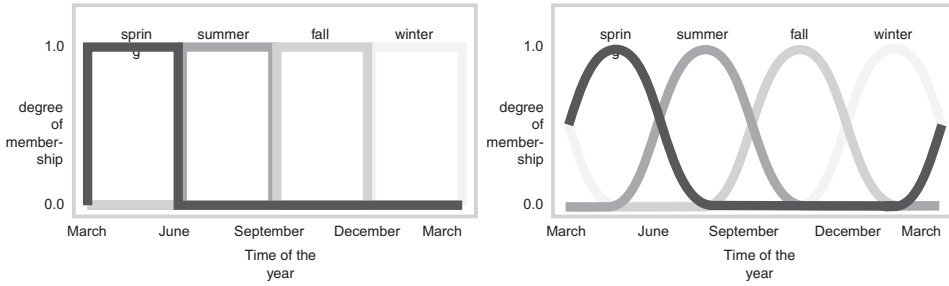
By making the plot continuous, you are defining the degree to which any given instant belongs in the weekend rather than an entire day. In the plot on the left, notice that at

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

midnight on Friday, just as the second hand sweeps past 12, the weekend-ness truth value jumps discontinuously from 0 to 1. This is one way to define the weekend, and while it may be useful to an accountant, it may not really connect with your own real-world experience of weekend-ness.

The plot on the right shows a smoothly varying curve that accounts for the fact that all of Friday, and, to a small degree, parts of Thursday, partake of the quality of weekend-ness and thus deserve partial membership in the fuzzy set of weekend moments. The curve that defines the weekend-ness of any instant in time is a function that maps the input space (time of the week) to the output space (weekend-ness). Specifically it is known as a *membership function*. See "Membership Functions" on page 2-6 for a more detailed discussion.

As another example of fuzzy sets, consider the question of seasons. What season is it right now? In the northern hemisphere, summer officially begins at the exact moment in the earth's orbit when the North Pole is pointed most directly toward the sun. It occurs exactly once a year, in late June. Using the astronomical definitions for the season, you get sharp boundaries as shown on the left in the figure that follows. But what you experience as the seasons vary more or less continuously as shown on the right in the following figure (in temperate northern hemisphere climates).



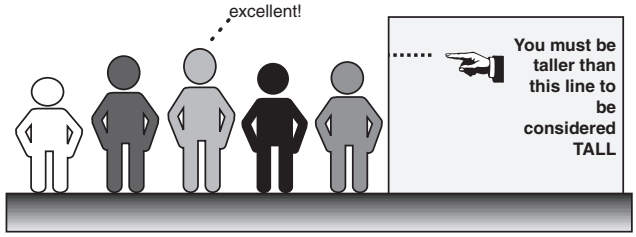
Membership Functions

A *membership function* (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the *universe of discourse*, a fancy name for a simple concept.

One of the most commonly used examples of a fuzzy set is the set of tall people. In this case, the universe of discourse is all potential heights, say from 3 feet to 9 feet, and the

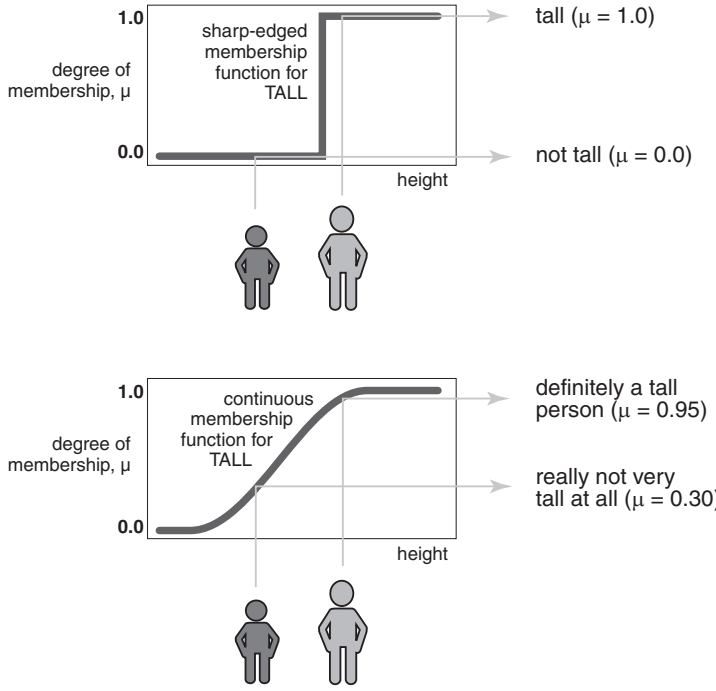
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

word tall would correspond to a curve that defines the degree to which any person is tall. If the set of tall people is given the well-defined (crisp) boundary of a classical set, you might say all people taller than 6 feet are officially considered tall. However, such a distinction is clearly absurd. It may make sense to consider the set of all real numbers greater than 6 because numbers belong on an abstract plane, but when we want to talk about real people, it is unreasonable to call one person short and another one tall when they differ in height by the width of a hair.



If the kind of distinction shown previously is unworkable, then what is the right way to define the set of tall people? Much as with the plot of weekend days, the figure following shows a smoothly varying curve that passes from not-tall to tall. The output-axis is a number known as the membership value between 0 and 1. The curve is known as a *membership function* and is often given the designation of μ. This curve defines the transition from not tall to tall. Both people are tall to some degree, but one is significantly less tall than the other.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Subjective interpretations and appropriate units are built right into fuzzy sets. If you say "She's tall," the membership function tall should already take into account whether you are referring to a six-year-old or a grown woman. Similarly, the units are included in the curve. Certainly it makes no sense to say "Is she tall in inches or in meters?"

Membership Functions in Fuzzy Logic Toolbox Software

The only condition a membership function must really satisfy is that it must vary between 0 and 1. The function itself can be an arbitrary curve whose shape we can define as a function that suits us from the point of view of simplicity, convenience, speed, and efficiency.

A classical set might be expressed as

$A = \{x \mid x > 6\}$

A fuzzy set is an extension of a classical set. If X is the universe of discourse and its elements are denoted by x, then a fuzzy set A in X is defined as a set of ordered pairs.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

$A = \{x, \mu_A(x) \mid x \in X\}$

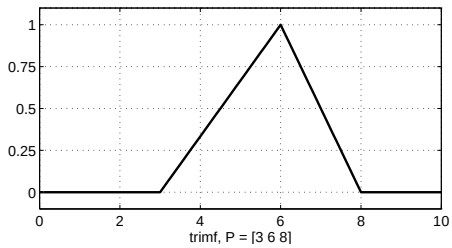
$\mu_A(x)$ is called the membership function (or MF) of x in A. The membership function maps each element of X to a membership value between 0 and 1.

The toolbox includes 11 built-in membership function types. These 11 functions are, in turn, built from several basic functions:

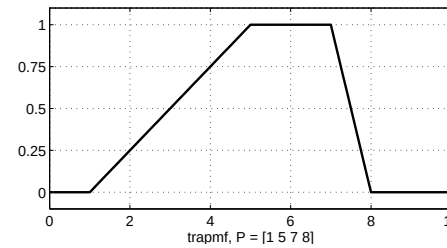
- piece-wise linear functions
- the Gaussian distribution function
- the sigmoid curve
- quadratic and cubic polynomial curves

For detailed information on any of the membership functions mentioned next, see the corresponding reference page. By convention, all membership functions have the letters `mf` at the end of their names.

The simplest membership functions are formed using straight lines. Of these, the simplest is the *triangular* membership function, and it has the function name `trimf`. This function is nothing more than a collection of three points forming a triangle. The *trapezoidal* membership function, `trapmf`, has a flat top and really is just a truncated triangle curve. These straight line membership functions have the advantage of simplicity.



trimf



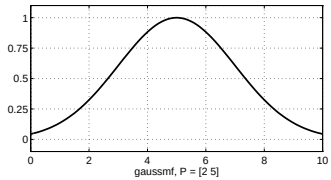
trapmf

Two membership functions are built on the *Gaussian* distribution curve: a simple Gaussian curve and a two-sided composite of two different Gaussian curves. The two functions are `gaussmf` and `gauss2mf`.

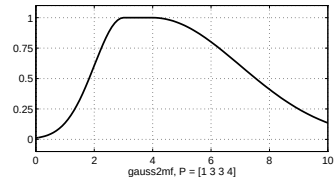
The *generalized bell* membership function is specified by three parameters and has the function name `gbellmf`. The bell membership function has one more parameter

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

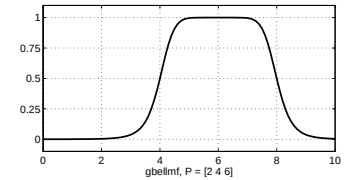
than the Gaussian membership function, so it can approach a non-fuzzy set if the free parameter is tuned. Because of their smoothness and concise notation, Gaussian and bell membership functions are popular methods for specifying fuzzy sets. Both of these curves have the advantage of being smooth and nonzero at all points.



gaussmf

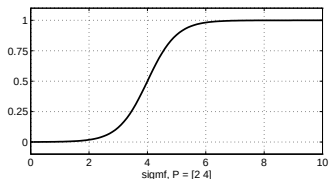


gauss2mf

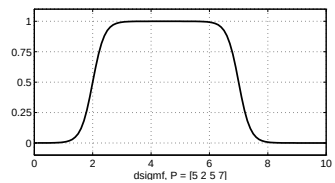


gbellmf

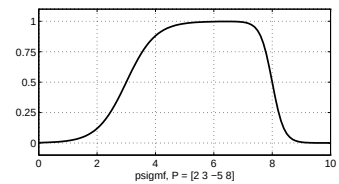
Although the Gaussian membership functions and bell membership functions achieve smoothness, they are unable to specify asymmetric membership functions, which are important in certain applications. Next, you define the *sigmoidal* membership function, which is either open left or right. Asymmetric and closed (i.e. not open to the left or right) membership functions can be synthesized using two sigmoidal functions, so in addition to the basic `sigmf`, you also have the difference between two sigmoidal functions, `dsigmf`, and the product of two sigmoidal functions `psigmf`.



sigmf



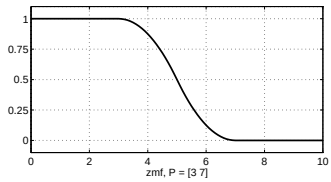
dsigmf



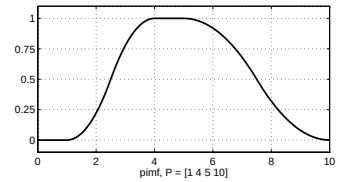
psigmf

Polynomial based curves account for several of the membership functions in the toolbox. Three related membership functions are the *Z, S*, and *Pi curves*, all named because of their shape. The function `zmf` is the asymmetrical polynomial curve open to the left, `smf` is the mirror-image function that opens to the right, and `pimf` is zero on both extremes with a rise in the middle.

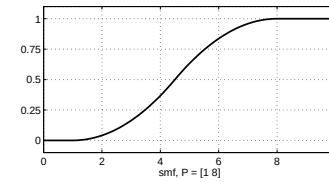
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



zmf



pimf



smf

There is a very wide selection to choose from when you're selecting a membership function. You can also create your own membership functions with the toolbox. However, if a list based on expanded membership functions seems too complicated, just remember that you could probably get along very well with just one or two types of membership functions, for example the triangle and trapezoid functions. The selection is wide for those who want to explore the possibilities, but expansive membership functions are not necessary for good fuzzy inference systems. Finally, remember that more details are available on all these functions in the reference section.

Summary of Membership Functions

- Fuzzy sets describe vague concepts (e.g., fast runner, hot weather, weekend days).
- A fuzzy set admits the possibility of partial membership in it. (e.g., Friday is sort of a weekend day, the weather is rather hot).
- The degree an object belongs to a fuzzy set is denoted by a membership value between 0 and 1. (e.g., Friday is a weekend day to the degree 0.8).
- A membership function associated with a given fuzzy set maps an input value to its appropriate membership value.

Logical Operations

Now that you understand the fuzzy inference, you need to see how fuzzy inference connects with logical operations.

The most important thing to realize about fuzzy logical reasoning is the fact that it is a superset of standard Boolean logic. In other words, if you keep the fuzzy values at their extremes of 1 (completely true), and 0 (completely false), standard logical operations will hold. As an example, consider the following standard truth tables.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

OR

A	not A
0	1
1	0

NOT

Now, because in fuzzy logic the truth of any statement is a matter of degree, can these truth tables be altered? The input values can be real numbers between 0 and 1. What function preserves the results of the AND truth table (for example) and also extend to all real numbers between 0 and 1?

One answer is the *min* operation. That is, resolve the statement *A* AND *B*, where *A* and *B* are limited to the range (0,1), by using the function *min*(*A*,*B*). Using the same reasoning, you can replace the OR operation with the *max* function, so that *A* OR *B* becomes equivalent to *max*(*A*,*B*). Finally, the operation NOT *A* becomes equivalent to the operation $1-A$. Notice how the previous truth table is completely unchanged by this substitution.

A	B	min(A,B)
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	max(A,B)
0	0	0
0	1	1
1	0	1
1	1	1

OR

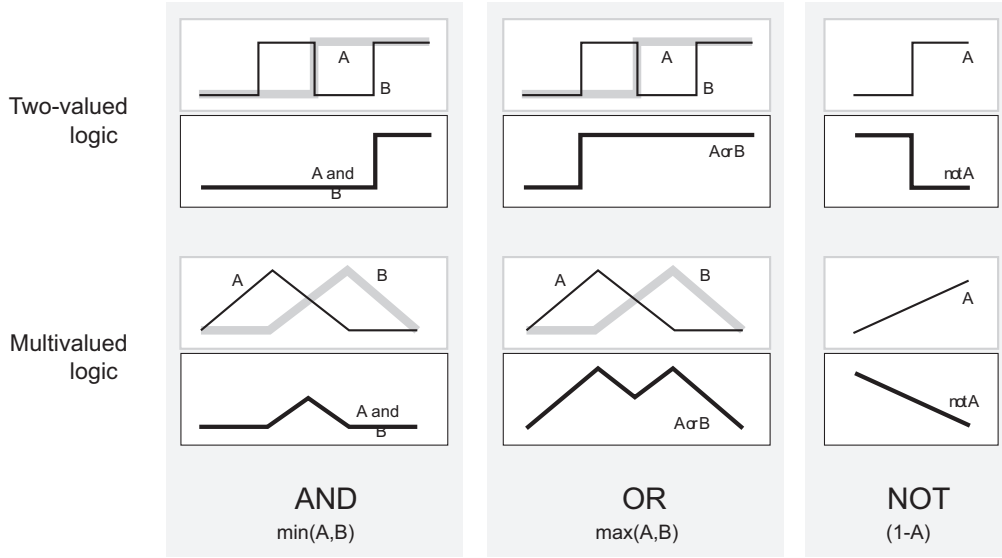
A	1 - A
0	1
1	0

NOT

Moreover, because there is a function behind the truth table rather than just the truth table itself, you can now consider values other than 1 and 0.

The next figure uses a graph to show the same information. In this figure, the truth table is converted to a plot of two fuzzy sets applied together to create one fuzzy set. The upper part of the figure displays plots corresponding to the preceding two-valued truth tables, while the lower part of the figure displays how the operations work over a continuously varying range of truth values *A* and *B* according to the fuzzy operations you have defined.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Given these three functions, you can resolve any construction using fuzzy sets and the fuzzy logical operation AND, OR, and NOT.

Additional Fuzzy Operators

In this case, you defined only one particular correspondence between two-valued and multivalued logical operations for AND, OR, and NOT. This correspondence is by no means unique.

In more general terms, you are defining what are known as the fuzzy intersection or conjunction (AND), fuzzy union or disjunction (OR), and fuzzy complement (NOT). The classical operators for these functions are: AND = *min*, OR = *max*, and NOT = additive complement. Typically, most fuzzy logic applications make use of these operations and leave it at that. In general, however, these functions are arbitrary to a surprising degree. Fuzzy Logic Toolbox software uses the classical operator for the fuzzy complement as shown in the previous figure, but also enables you to customize the AND and OR operators.

The intersection of two fuzzy sets *A* and *B* is specified in general by a binary mapping *T*, which aggregates two membership functions as follows:

$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x))$

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

For example, the binary operator T may represent the multiplication of $\mu_A(x)$ and $\mu_B(x)$. These fuzzy intersection operators, which are usually referred to as T-norm (Triangular norm) operators, meet the following basic requirements:

A *T-norm* operator is a binary mapping $T(.,.)$ satisfying
boundary: $T(0, 0) = 0$, $T(a, 1) = T(1, a) = a$
monotonicity: $T(a, b) <= T(c, d)$ if $a <= c$ and $b <= d$
commutativity: T(a, b) = T(b, a)
associativity: T(a, T(b, c)) = T(T(a, b), c)

The first requirement imposes the correct generalization to crisp sets. The second requirement implies that a decrease in the membership values in A or B cannot produce an increase in the membership value in A intersection B. The third requirement indicates that the operator is indifferent to the order of the fuzzy sets to be combined. Finally, the fourth requirement allows us to take the intersection of any number of sets in any order of pair-wise groupings.

Like fuzzy intersection, the fuzzy union operator is specified in general by a binary mapping S:

$$\mu_{A \cup B}(x) = S(\mu_A(x), \mu_B(x))$$

For example, the binary operator S can represent the addition of $\mu_A(x)$ and $\mu_B(x)$. These fuzzy union operators, which are often referred to as *T-conorm (or S-norm)* operators, must satisfy the following basic requirements:

A *T-conorm* (or *S-norm*) operator is a binary mapping $S(.,.)$ satisfying
boundary: $S(1, 1) = 1$, $S(a, 0) = S(0, a) = a$
monotonicity: $S(a, b) <= S(c, d)$ if $a <= c$ and $b <= d$
commutativity: $S(a, b) = S(b, a)$
associativity: $S(a, S(b, c)) = S(S(a, b), c)$

Several parameterized T-norms and dual T-conorms have been proposed in the past, such as those of Yager[10], Dubois and Prade [1], Schweizer and Sklar [7], and Sugeno [8], found in the Appendix A. Each of these provides a way to vary the gain on the function so that it can be very restrictive or very permissive.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

If-Then Rules

Fuzzy sets and fuzzy operators are the subjects and verbs of fuzzy logic. These if-then rule statements are used to formulate the conditional statements that comprise fuzzy logic.

A single fuzzy if-then rule assumes the form

if x is A then y is B

where A and B are linguistic values defined by fuzzy sets on the ranges (universes of discourse) X and Y, respectively. The if-part of the rule "x is A" is called the *antecedent* or premise, while the then-part of the rule "y is B" is called the *consequent* or conclusion. An example of such a rule might be
If service is good then tip is average

The concept *good* is represented as a number between 0 and 1, and so the antecedent is an interpretation that returns a single number between 0 and 1. Conversely, *average* is represented as a fuzzy set, and so the consequent is an assignment that assigns the entire fuzzy set B to the output variable y. In the if-then rule, the word *is* gets used in two entirely different ways depending on whether it appears in the antecedent or the consequent. In MATLAB terms, this usage is the distinction between a relational test using "==" and a variable assignment using the "=" symbol. A less confusing way of writing the rule would be

If service == good then tip = average

In general, the input to an if-then rule is the current value for the input variable (in this case, *service*) and the output is an entire fuzzy set (in this case, *average*). This set will later be *defuzzified,* assigning one value to the output. The concept of defuzzification is described in the next section.

Interpreting an if-then rule involves distinct parts: first evaluating the antecedent (which involves *fuzzifying* the input and applying any necessary *fuzzy operators*) and second applying that result to the consequent (known as *implication*). In the case of two-valued or binary logic, if-then rules do not present much difficulty. If the premise is true, then the conclusion is true. If you relax the restrictions of two-valued logic and let the antecedent be a fuzzy statement, how does this reflect on the conclusion? The answer is a simple one. if the antecedent is true to some degree of membership, then the consequent is also true to that same degree.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Thus:

in binary logic: $p \rightarrow q$ (p and q are either both true or both false.)
in fuzzy logic: $0.5\,p \rightarrow 0.5\,q$ (Partial antecedents provide partial implication.)

The antecedent of a rule can have multiple parts.

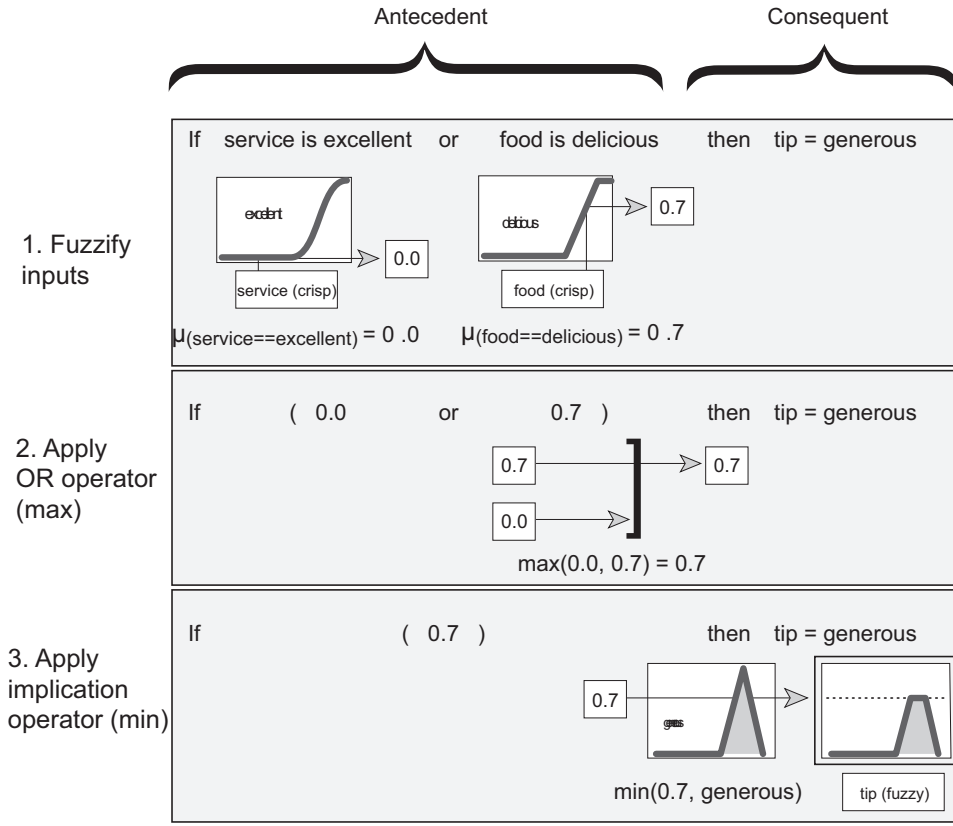
if sky is gray and wind is strong and barometer is falling, then ...

in which case all parts of the antecedent are calculated simultaneously and resolved to a single number using the logical operators described in the preceding section. The consequent of a rule can also have multiple parts.

if temperature is cold then hot water valve is open and cold water valve is shut

in which case all consequents are affected equally by the result of the antecedent. How is the consequent affected by the antecedent? The consequent specifies a fuzzy set be assigned to the output. The *implication function* then modifies that fuzzy set to the degree specified by the antecedent. The most common ways to modify the output fuzzy set are truncation using the `min` function (where the fuzzy set is truncated as shown in the following figure) or scaling using the `prod` function (where the output fuzzy set is squashed). Both are supported by the toolbox, but you use truncation for the examples in this section.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Summary of If-Then Rules

Interpreting if-then rules is a three-part process. This process is explained in detail in the next section:

1. **Fuzzify inputs**: Resolve all fuzzy statements in the antecedent to a degree of membership between 0 and 1. If there is only one part to the antecedent, then this is the degree of support for the rule.
2. **Apply fuzzy operator to multiple part antecedents**: If there are multiple parts to the antecedent, apply fuzzy logic operators and resolve the antecedent to a single number between 0 and 1. This is the degree of support for the rule.
3. **Apply implication method**: Use the degree of support for the entire rule to shape the output fuzzy set. The consequent of a fuzzy rule assigns an entire fuzzy set to

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

the output. This fuzzy set is represented by a membership function that is chosen to indicate the qualities of the consequent. If the antecedent is only partially true, (i.e., is assigned a value less than 1), then the output fuzzy set is truncated according to the implication method.

In general, one rule alone is not effective. Two or more rules that can play off one another are needed. The output of each rule is a fuzzy set. The output fuzzy sets for each rule are then aggregated into a single output fuzzy set. Finally the resulting set is defuzzified, or resolved to a single number. "Build Mamdani Systems (GUI)" on page 2-31 shows how the whole process works from beginning to end for a particular type of fuzzy inference system called a *Mamdani type*.

References

[1] Dubois, D. and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York, 1980.

[2] Kaufmann, A. and M.M. Gupta, *Introduction to Fuzzy Arithmetic*, V.N. Reinhold, 1985.

[3] Lee, C.-C., "Fuzzy logic in control systems: fuzzy logic controller-parts 1 and 2," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 2, pp 404-435, 1990.

[4] Mamdani, E.H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1-13, 1975.

[5] Mamdani, E.H., "Advances in the linguistic synthesis of fuzzy controllers," *International Journal of Man-Machine Studies*, Vol. 8, pp. 669-678, 1976.

[6] Mamdani, E.H., "Applications of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Transactions on Computers*, Vol. 26, No. 12, pp. 1182-1191, 1977.

[7] Schweizer, B. and A. Sklar, "Associative functions and abstract semi-groups," *Publ. Math Debrecen*, 10:69-81, 1963.

[8] Sugeno, M., "Fuzzy measures and fuzzy integrals: a survey," (M.M. Gupta, G. N. Saridis, and B.R. Gaines, editors) *Fuzzy Automata and Decision Processes*, pp. 89-102, North-Holland, NY, 1977.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

[9] Sugeno, M., *Industrial applications of fuzzy control*, Elsevier Science Pub. Co., 1985.

[10] Yager, R., "On a general class of fuzzy connectives," *Fuzzy Sets and Systems*, 4:235-242, 1980.

[11] Yager, R. and D. Filev, "Generation of Fuzzy Rules by Mountain Clustering," *Journal of Intelligent & Fuzzy Systems*, Vol. 2, No. 3, pp. 209-219, 1994.

[12] Zadeh, L.A., "Fuzzy sets," *Information and Control*, Vol. 8, pp. 338-353, 1965.

[13] Zadeh, L.A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 3, No. 1, pp. 28-44, Jan. 1973.

[14] Zadeh, L.A., "The concept of a linguistic variable and its application to approximate reasoning, Parts 1, 2, and 3," *Information Sciences*, 1975, 8:199-249, 8:301-357, 9:43-80.

[15] Zadeh, L.A., "Fuzzy Logic," *Computer*, Vol. 1, No. 4, pp. 83-93, 1988.

[16] Zadeh, L.A., "Knowledge representation in fuzzy logic," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, pp. 89-100, 1989.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Types of Fuzzy Inference Systems

You can implement two types of fuzzy inference systems in the toolbox:

- Mamdani
- Sugeno

These two types of inference systems vary somewhat in the way outputs are determined.

Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. Mamdani's method was among the first control systems built using fuzzy set theory. It was proposed in 1975 by Ebrahim Mamdani [4] as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani's effort was based on Lotfi Zadeh's 1973 paper on fuzzy algorithms for complex systems and decision processes [13]. Although the inference process described in the next few sections differs somewhat from the methods described in the original paper, the basic idea is much the same.

Mamdani-type inference, as defined for the toolbox, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. It is possible, and in many cases much more efficient, to use a single spike as the output membership function rather than a distributed fuzzy set. This type of output is sometimes known as a *singleton* output membership function, and it can be thought of as a pre-defuzzified fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of a two-dimensional function. Rather than integrating across the two-dimensional function to find the centroid, you use the weighted average of a few data points. Sugeno-type systems support this type of model. In general, Sugeno-type systems can be used to model any inference system in which the output membership functions are either linear or constant.

See the Bibliography for references to descriptions of these two types of fuzzy inference systems, [5], [4], [9].

Fuzzy inference systems have been successfully applied in fields such as automatic control, data classification, decision analysis, expert systems, and computer vision. Because of its multidisciplinary nature, fuzzy inference systems are associated with a number of names, such as fuzzy-rule-based systems, fuzzy expert systems, fuzzy modeling, fuzzy associative memory, fuzzy logic controllers, and simply (and ambiguously) fuzzy systems.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

More About

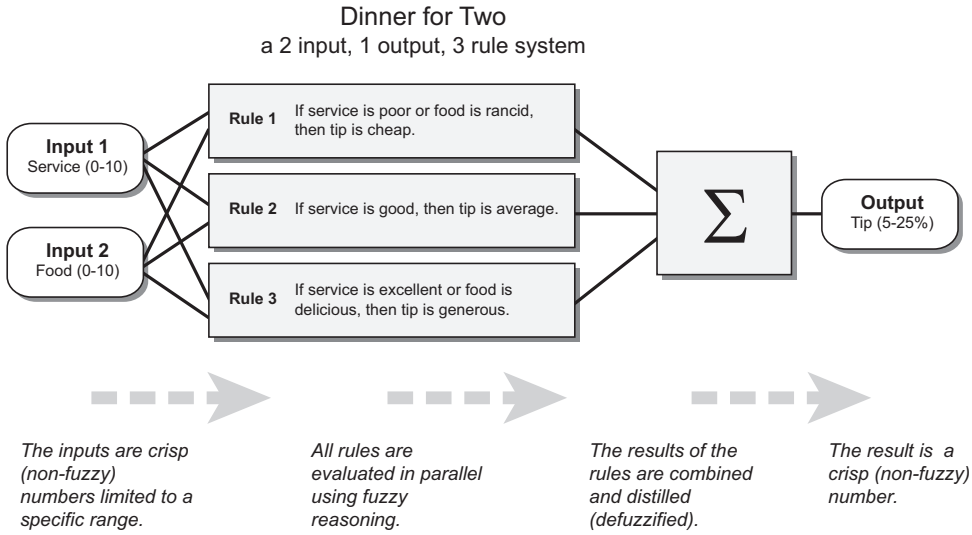
- "What Is Mamdani-Type Fuzzy Inference?" on page 2-30
- "What Is Sugeno-Type Fuzzy Inference?" on page 2-91
- "Comparison of Sugeno and Mamdani Systems" on page 2-97

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Fuzzy Inference Process

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. The process of fuzzy inference involves all of the pieces that are described in "Membership Functions" on page 2-6, "Logical Operations" on page 2-11, and "If-Then Rules" on page 2-15.

This section describes the fuzzy inference process and uses the example of the two-input, one-output, three-rule tipping problem "The Basic Tipping Problem" on page 2-33 that you saw in the introduction in more detail. The basic structure of this example is shown in the following diagram:



Information flows from left to right, from two inputs to a single output. The parallel nature of the rules is one of the more important aspects of fuzzy logic systems. Instead of sharp switching between modes based on breakpoints, logic flows smoothly from regions where the system's behavior is dominated by either one rule or another.

Fuzzy inference process comprises of five parts:

- Fuzzification of the input variables
- Application of the fuzzy operator (AND or OR) in the antecedent

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

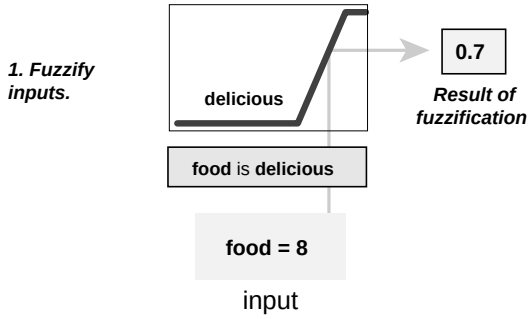
- Implication from the antecedent to the consequent
- Aggregation of the consequents across the rules
- Defuzzification

A fuzzy inference diagram displays all parts of the fuzzy inference process — from fuzzification through defuzzification.

Step 1. Fuzzify Inputs

The first step is to take the inputs and determine the degree to which they belong to each of the appropriate fuzzy sets via membership functions. In Fuzzy Logic Toolbox software, the input is always a crisp numerical value limited to the universe of discourse of the input variable (in this case the interval between 0 and 10) and the output is a fuzzy degree of membership in the qualifying linguistic set (always the interval between 0 and 1). Fuzzification of the input amounts to either a table lookup or a function evaluation.

This example is built on three rules, and each of the rules depends on resolving the inputs into a number of different fuzzy linguistic sets: service is poor, service is good, food is rancid, food is delicious, and so on. Before the rules can be evaluated, the inputs must be fuzzified according to each of these linguistic sets. For example, to what extent is the food really delicious? The following figure shows how well the food at the hypothetical restaurant (rated on a scale of 0 to 10) qualifies, (via its membership function), as the linguistic variable delicious. In this case, we rated the food as an 8, which, given your graphical definition of delicious, corresponds to $\mu = 0.7$ for the delicious membership function.



In this manner, each input is fuzzified over all the qualifying membership functions required by the rules.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Step 2. Apply Fuzzy Operator

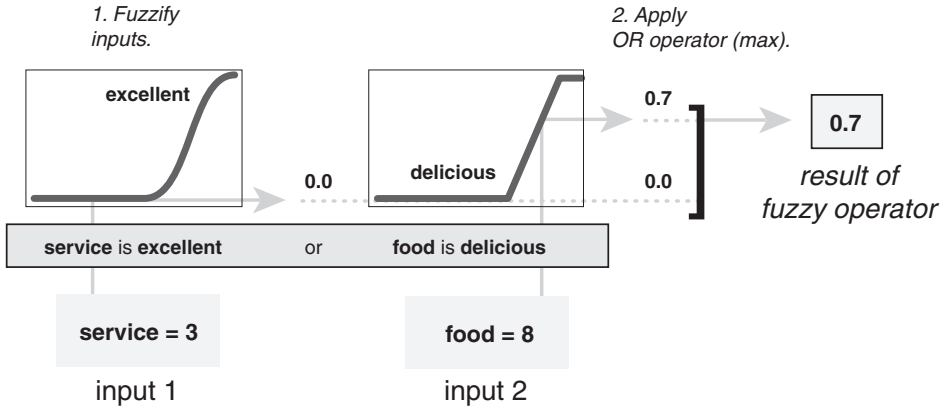
After the inputs are fuzzified, you know the degree to which each part of the antecedent is satisfied for each rule. If the antecedent of a given rule has more than one part, the fuzzy operator is applied to obtain one number that represents the result of the antecedent for that rule. This number is then applied to the output function. The input to the fuzzy operator is two or more membership values from fuzzified input variables. The output is a single truth value.

As is described in "Logical Operations" section, any number of well-defined methods can fill in for the AND operation or the OR operation. In the toolbox, two built-in AND methods are supported: *min* (minimum) and *prod* (product). Two built-in OR methods are also supported: *max* (maximum), and the probabilistic OR method *probor*. The probabilistic OR method (also known as the algebraic sum) is calculated according to the equation

$probor(a,b) = a + b - ab$

In addition to these built-in methods, you can create your own methods for AND and OR by writing any function and setting that to be your method of choice.

The following figure shows the OR operator *max* at work, evaluating the antecedent of the rule 3 for the tipping calculation. The two different pieces of the antecedent (service is excellent and food is delicious) yielded the fuzzy membership values 0.0 and 0.7 respectively. The fuzzy OR operator simply selects the maximum of the two values, 0.7, and the fuzzy operation for rule 3 is complete. The probabilistic OR method would still result in 0.7.

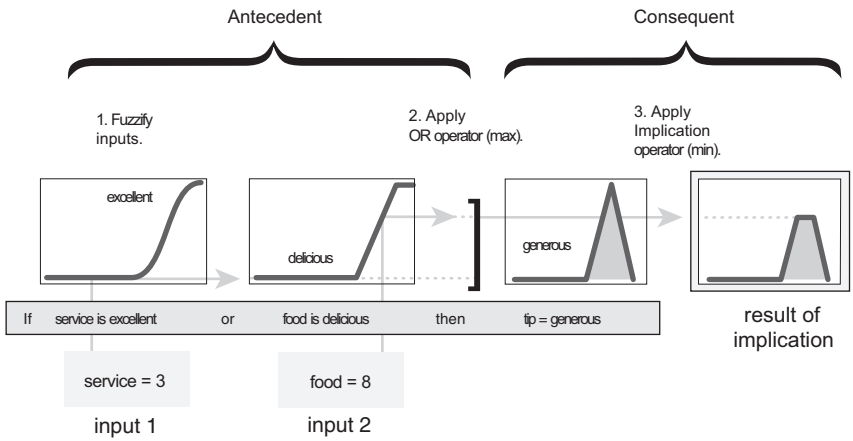


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Step 3. Apply Implication Method

Before applying the implication method, you must determine the rule's weight. Every rule has a *weight* (a number between 0 and 1), which is applied to the number given by the antecedent. Generally, this weight is 1 (as it is for this example) and thus has no effect at all on the implication process. From time to time you may want to weight one rule relative to the others by changing its weight value to something other than 1.

After proper weighting has been assigned to each rule, the implication method is implemented. A consequent is a fuzzy set represented by a membership function, which weights appropriately the linguistic characteristics that are attributed to it. The consequent is reshaped using a function associated with the antecedent (a single number). The input for the implication process is a single number given by the antecedent, and the output is a fuzzy set. Implication is implemented for each rule. Two built-in methods are supported, and they are the same functions that are used by the AND method: *min* (minimum), which truncates the output fuzzy set, and *prod* (product), which scales the output fuzzy set.



Step 4. Aggregate All Outputs

Because decisions are based on the testing of all of the rules in a FIS, the rules must be combined in some manner in order to make a decision. Aggregation is the process by which the fuzzy sets that represent the outputs of each rule are combined into a single fuzzy set. Aggregation only occurs once for each output variable, just prior to the

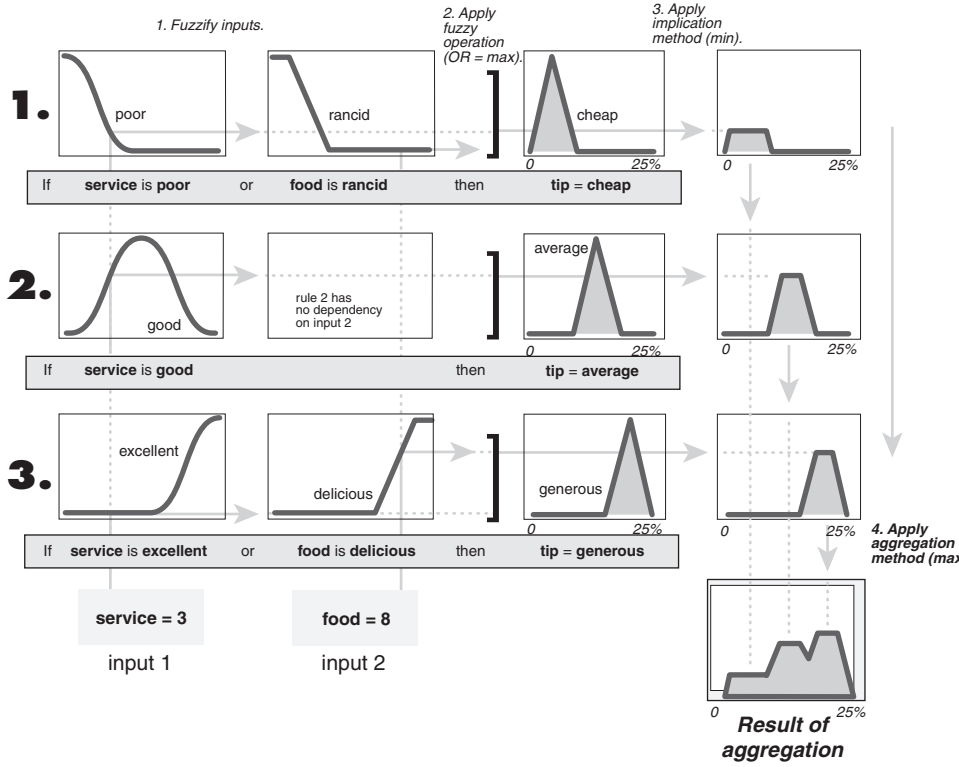
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

fifth and final step, defuzzification. The input of the aggregation process is the list of truncated output functions returned by the implication process for each rule. The output of the aggregation process is one fuzzy set for each output variable.

As long as the aggregation method is commutative (which it always should be), then the order in which the rules are executed is unimportant. Three built-in methods are supported:

- `max` (maximum)
- `probor` (probabilistic OR)
- `sum` (simply the sum of each rule's output set)

In the following diagram, all three rules have been placed together to show how the output of each rule is combined, or aggregated, into a single fuzzy set whose membership function assigns a weighting for every output (tip) value.

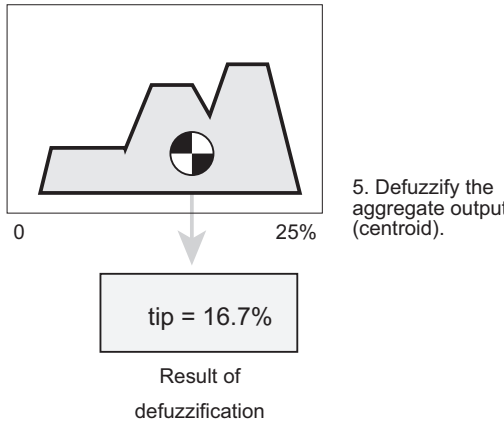


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Step 5. Defuzzify

The input for the defuzzification process is a fuzzy set (the aggregate output fuzzy set) and the output is a single number. As much as fuzziness helps the rule evaluation during the intermediate steps, the final desired output for each variable is generally a single number. However, the aggregate of a fuzzy set encompasses a range of output values, and so must be defuzzified in order to resolve a single output value from the set.

Perhaps the most popular defuzzification method is the centroid calculation, which returns the center of area under the curve. There are five built-in methods supported: centroid, bisector, middle of maximum (the average of the maximum value of the output set), largest of maximum, and smallest of maximum.

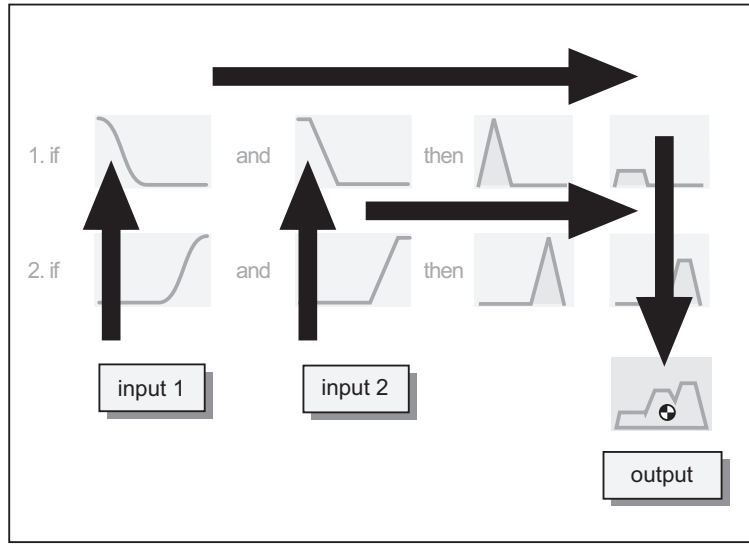


Fuzzy Inference Diagram

The fuzzy inference diagram is the composite of all the smaller diagrams presented so far in this section. It simultaneously displays all parts of the fuzzy inference process you have examined. Information flows through the fuzzy inference diagram as shown in the following figure.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

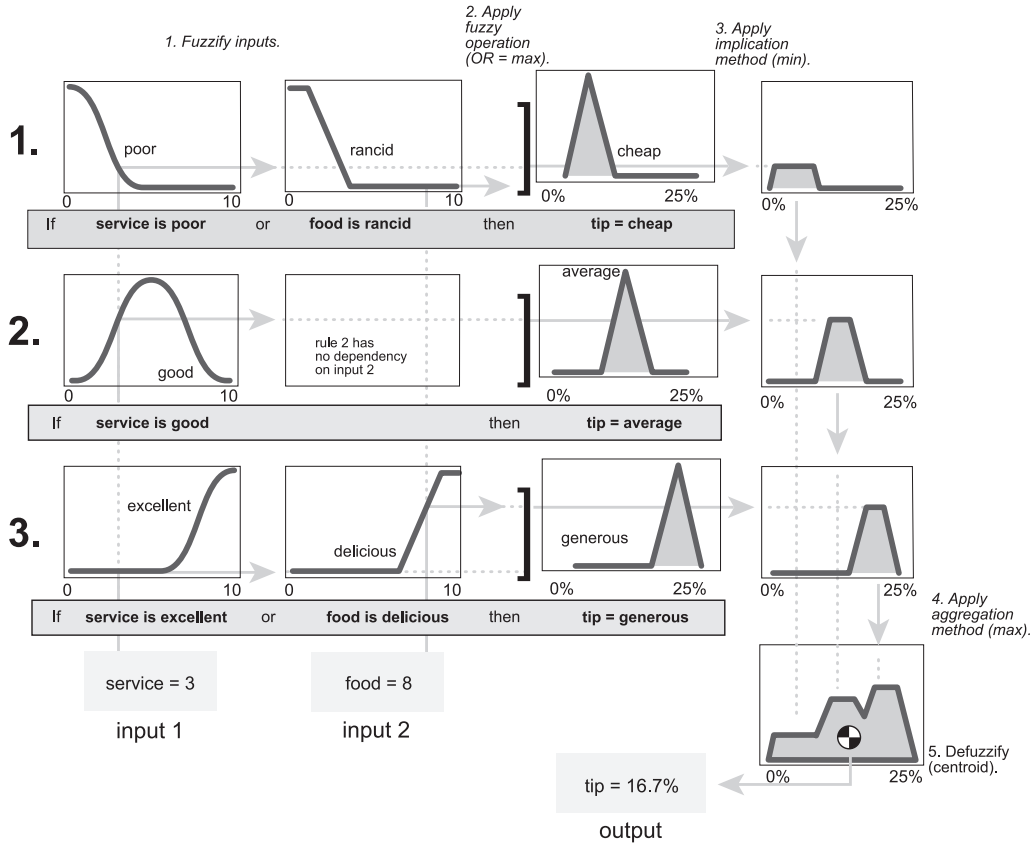
Interpreting the fuzzy inference diagram



In this figure, the flow proceeds up from the inputs in the lower left, then across each row, or rule, and then down the rule outputs to finish in the lower right. This compact flow shows everything at once, from linguistic variable fuzzification all the way through defuzzification of the aggregate output.

The following figure shows the actual full-size fuzzy inference diagram. There is a lot to see in a fuzzy inference diagram, but after you become accustomed to it, you can learn a lot about a system very quickly. For instance, from this diagram with these particular inputs, you can easily see that the implication method is truncation with the *min* function. The *max* function is being used for the fuzzy OR operation. Rule 3 (the bottom-most row in the diagram shown previously) is having the strongest influence on the output. and so on. The Rule Viewer described in “The Rule Viewer” on page 2-49 is a MATLAB implementation of the fuzzy inference diagram.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



More About

- "Membership Functions" on page 2-6
- "Logical Operations" on page 2-11
- "If-Then Rules" on page 2-15

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

What Is Mamdani-Type Fuzzy Inference?

Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. Mamdani's method was among the first control systems built using fuzzy set theory. It was proposed in 1975 by Ebrahim Mamdani [4] as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani's effort was based on Lotfi Zadeh's 1973 paper on fuzzy algorithms for complex systems and decision processes [13]. Although the inference process described in the next few sections differs somewhat from the methods described in the original paper, the basic idea is much the same.

Mamdani-type inference, as defined for the toolbox, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification.

Related Examples

- “Build Mamdani Systems (GUI)” on page 2-31
- “Build Mamdani Systems (Code)” on page 2-65
- “Build Mamdani Systems Using Custom Functions” on page 2-54

More About

- “Comparison of Sugeno and Mamdani Systems” on page 2-97
- “What Is Sugeno-Type Fuzzy Inference?” on page 2-91

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Build Mamdani Systems (GUI)

In this section...

"How to Build Mamdani Systems Using Fuzzy Logic Toolbox Graphical User Interface Tools" on page 2-31

"The Basic Tipping Problem" on page 2-33

"The Fuzzy Logic Designer" on page 2-34

"The Membership Function Editor" on page 2-39

"The Rule Editor" on page 2-46

"The Rule Viewer" on page 2-49

"The Surface Viewer" on page 2-51

"Importing and Exporting from the GUI Tools" on page 2-53

How to Build Mamdani Systems Using Fuzzy Logic Toolbox Graphical User Interface Tools

This example shows how to build a Fuzzy Inference System (FIS) for the tipping example, described in "The Basic Tipping Problem" on page 2-33, using the Fuzzy Logic Toolbox graphical user interface (GUI) tools.

You use the following graphical tools to build, edit, and view fuzzy inference systems:

- **Fuzzy Logic Designer** to handle the high-level issues for the system—How many input and output variables? What are their names?

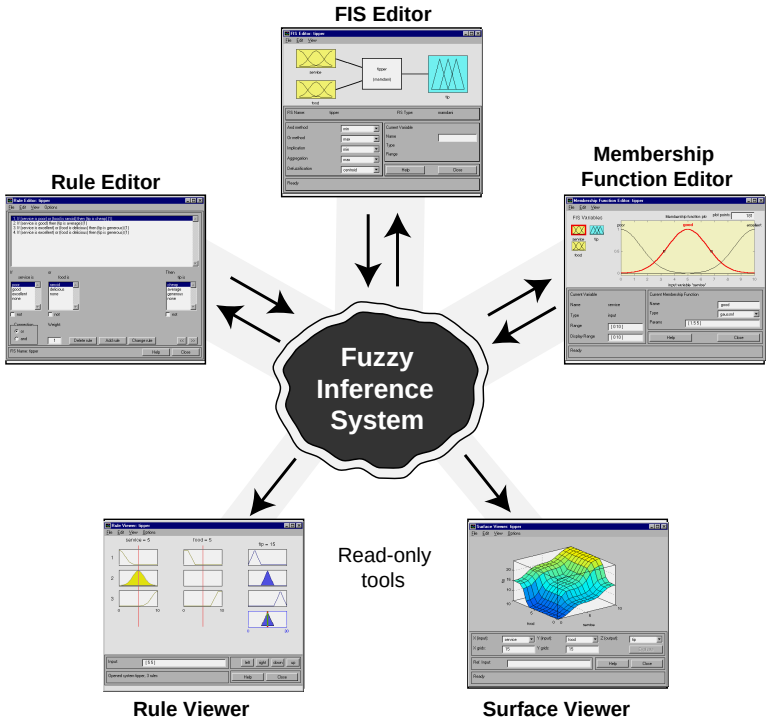
  Fuzzy Logic Toolbox software does not limit the number of inputs. However, the number of inputs may be limited by the available memory of your machine. If the number of inputs is too large, or the number of membership functions is too big, then it may also be difficult to analyze the FIS using the other tools.
- **Membership Function Editor** to define the shapes of all the membership functions associated with each variable
- **Rule Editor** to edit the list of rules that defines the behavior of the system.
- **Rule Viewer** to view the fuzzy inference diagram. Use this viewer as a diagnostic to see, for example, which rules are active, or how individual membership function shapes influence the results

www.MatlabSite.com | متلب سایت

مرجع آموزش برنامه‌نویسی متلب در ایران

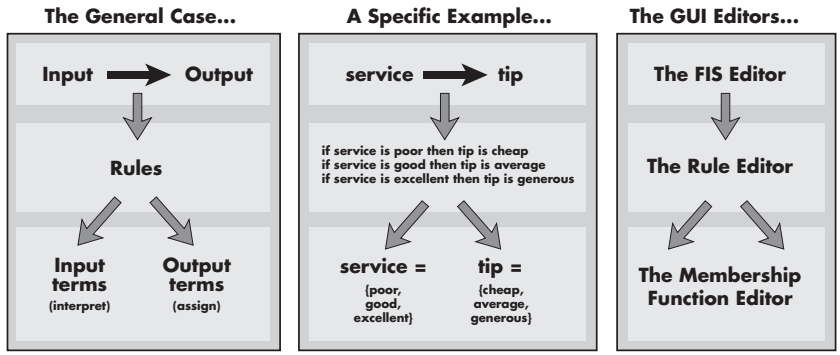
- **Surface Viewer** to view the dependency of one of the outputs on any one or two of the inputs—that is, it generates and plots an output surface map for the system.

These GUIs are dynamically linked, in that changes you make to the FIS using one of them, affect what you see on any of the other open GUIs. For example, if you change the names of the membership functions in the Membership Function Editor, the changes are reflected in the rules shown in the Rule Editor. You can use the GUIs to read and write variables both to the MATLAB workspace and to a file (the read-only viewers can still exchange plots with the workspace and save them to a file). You can have any or all of them open for any given system or have multiple editors open for any number of FIS systems.



The following figure shows how the main components of a FIS and the three editors fit together. The two viewers examine the behavior of the entire system.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



In addition to these five primary GUIs, the toolbox includes the graphical Neuro-Fuzzy Designer, which you use to build and analyze Sugeno-type adaptive neuro-fuzzy inference systems.

The Fuzzy Logic Toolbox GUIs do not support building FIS using data. If you want to use data to build a FIS, use one of the following techniques:

- genfis1, genfis2, or genfis3 commands to generate a Sugeno-type FIS. Then, select **File > Import** in the Fuzzy Logic Designer to import the FIS and perform fuzzy inference, as described in "The Fuzzy Logic Designer" on page 2-34.
- Neuro-adaptive learning techniques to model the FIS, as described in "Neuro-Adaptive Learning and ANFIS" on page 3-2.

If you wan to use MATLAB workspace variables, use the command-line interface instead of the Fuzzy Logic Designer. For an example, see "Building a System from Scratch" on page 2-70.

The Basic Tipping Problem

This example uses a two-input, one-output tipping problem based on tipping practices in the U.S.

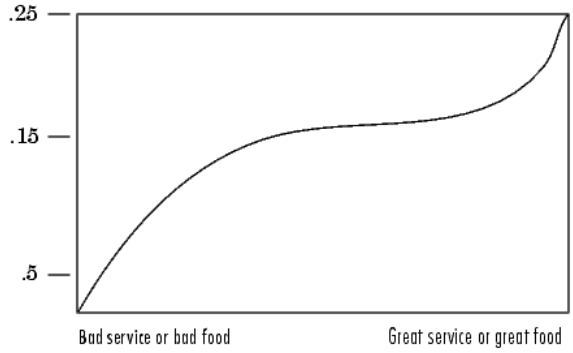
Given a number between 0 and 10 that represents the quality of service at a restaurant (where 10 is excellent), and another number between 0 and 10 that represents the quality of the food at that restaurant (again, 10 is excellent), what should the tip be?

The starting point is to write down the three golden rules of tipping:

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

1. *If the service is poor or the food is rancid, then tip is cheap.*
2. *If the service is good, then tip is average.*
3. *If the service is excellent or the food is delicious, then tip is generous.*

Assume that an average tip is 15%, a generous tip is 25%, and a cheap tip is 5%.



Obviously the numbers and the shape of the curve are subject to local traditions, cultural bias, and so on, but the three rules are generally universal.

Now that you know the rules and have an idea of what the output should look like, use the GUI tools to construct a fuzzy inference system for this decision process.

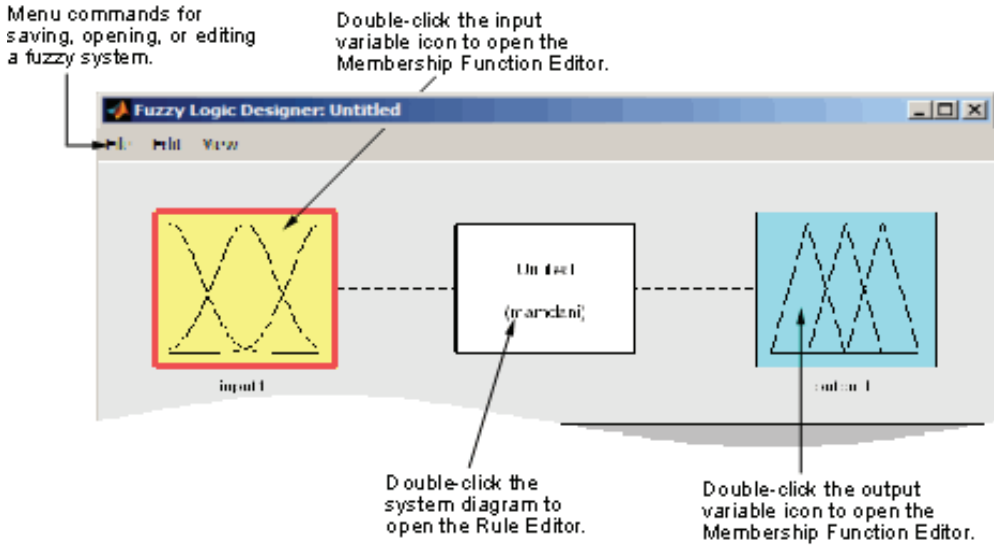
The Fuzzy Logic Designer

The Fuzzy Logic Designer displays information about a fuzzy inference system. To open the Fuzzy Logic Designer, type the following command at the MATLAB prompt:

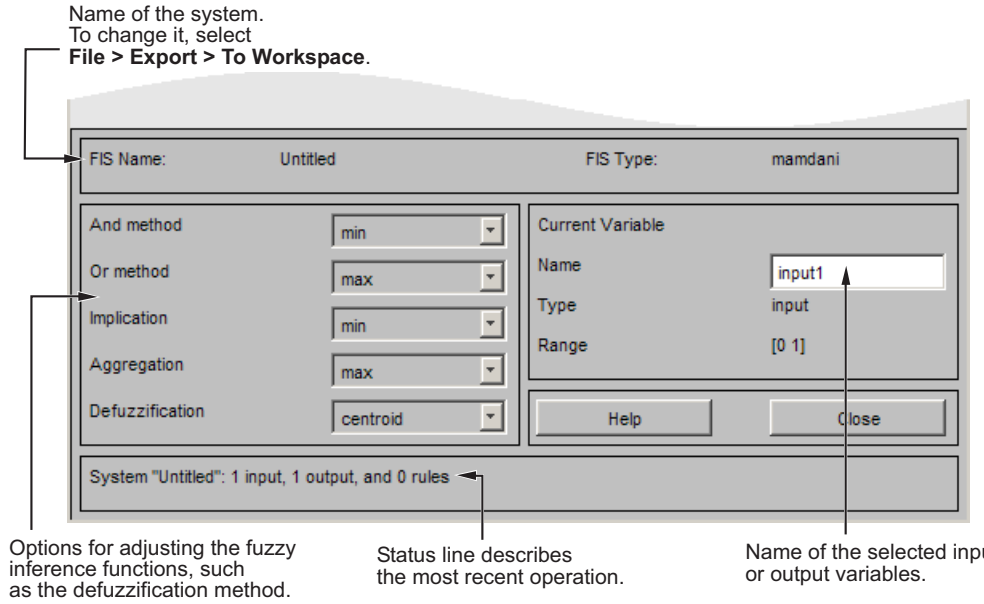
```
fuzzyLogicDesigner
```

The Fuzzy Logic Designer opens and displays a diagram of the fuzzy inference system with the names of each input variable on the left, and those of each output variable on the right, as shown in the next figure. The sample membership functions shown in the boxes are just icons and do not depict the actual shapes of the membership functions.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Below the diagram is the name of the system and the type of inference used.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

In this example, you use the default Mamdani-type inference. Another type of inference, called Sugeno-type inference, is also available. See “What Is Sugeno-Type Fuzzy Inference?” on page 2-91.

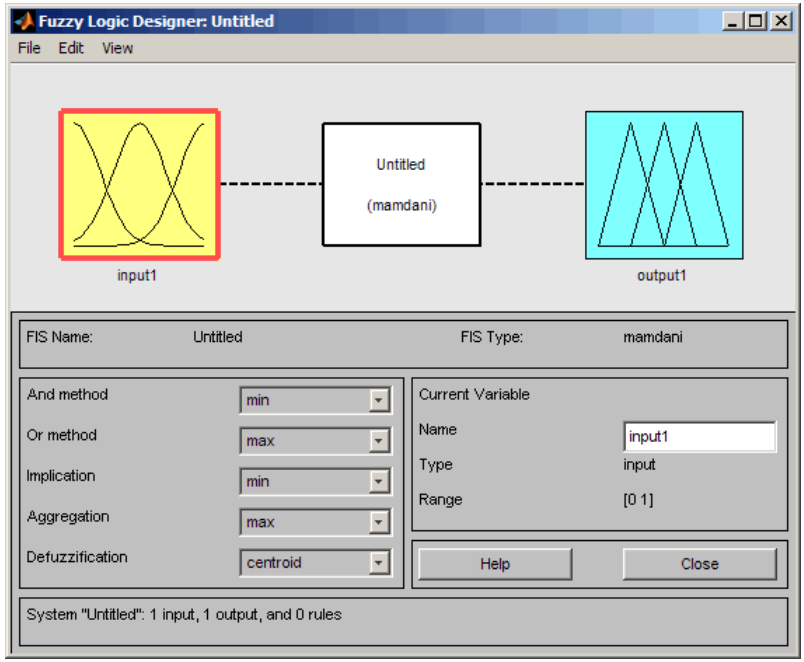
In the Fuzzy Logic Designer:

- The drop-down lists let you modify the fuzzy inference functions.
- The **Current Variable** area displays the name of either an input or output variable, its type, and default range.
- A status line at the bottom displays information about the most recent operation.

To build the Fuzzy Inference System described in “The Basic Tipping Problem” on page 2-33 from scratch, type the following command at the MATLAB prompt:

```
fuzzyLogicDesigner
```

The generic untitled Fuzzy Logic Designer opens, with one input **input1**, and one output **output1**.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Tip To open the Fuzzy Logic Designer with the prebuilt fuzzy inference system stored in `tipper.fis`, type

```
fuzzyLogicDesigner tipper
```

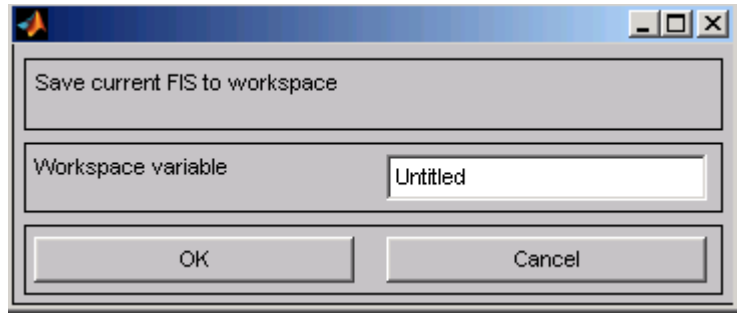
However, if you load the prebuilt system, you will not build rules or construct membership functions.

In this example, you construct a two-input, one output system. The two inputs are **service** and **food**. The one output is **tip**.

To add a second input variable and change the variable names to reflect these designations:

1. Select **Edit** > **Add variable** > **Input**.

   A second yellow box labeled **input2** appears.
2. Click the yellow box **input1**. This box is highlighted with a red outline.
3. Edit the **Name** field from `input1` to `service`, and press **Enter**.
4. Click the yellow box **input2**. This box is highlighted with a red outline.
5. Edit the **Name** field from `input2` to `food`, and press **Enter**.
6. Click the blue box **output1**.
7. Edit the **Name** field from `output1` to `tip`, and press **Enter**.
8. Select **File** > **Export** > **To Workspace**.

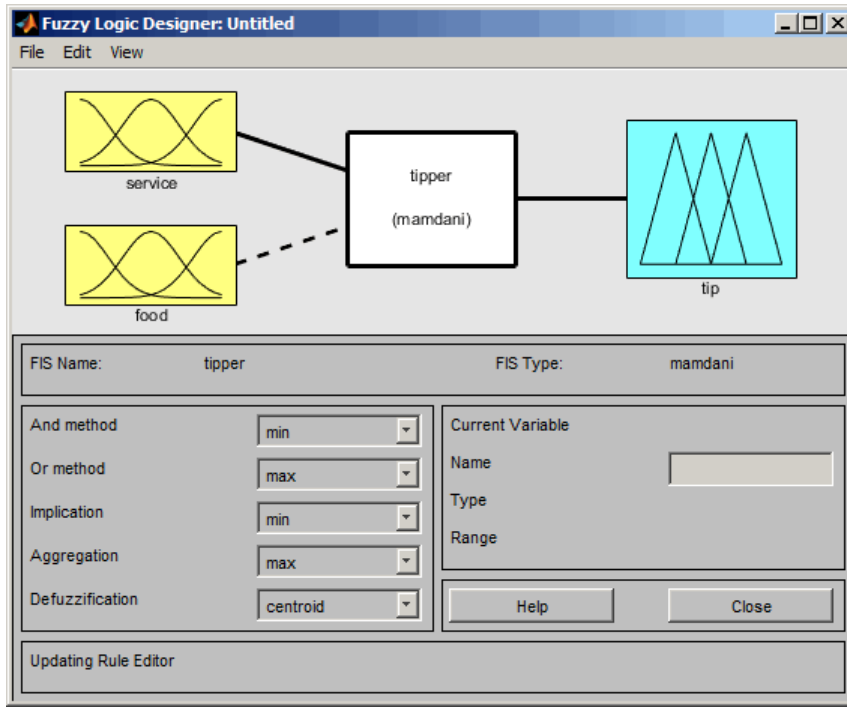


9. Enter the **Workspace variable** name `tipper`, and click **OK**.

   The diagram is updated to reflect the new names of the input and output variables. There is now a new variable in the workspace called `tipper` that contains all the

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

information about this system. By saving to the workspace with a new name, you also rename the entire system. Your window looks something like the following diagram.



Leave the inference options in the lower left in their default positions for now. You have entered all the information you need for this particular GUI. Next, define the membership functions associated with each of the variables. To do this, open the Membership Function Editor.

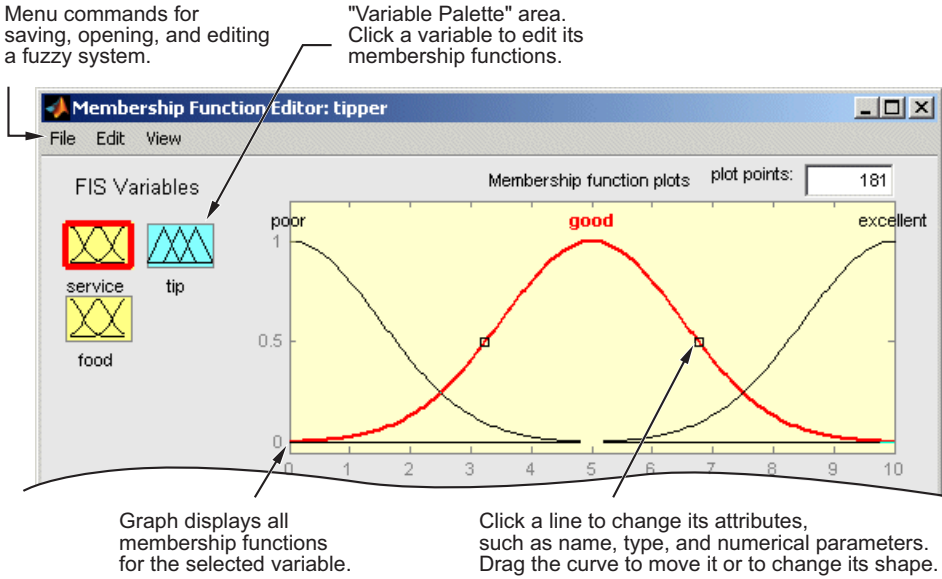
You can open the Membership Function Editor in one of three ways:

- Within the Fuzzy Logic Designer window, select **Edit** > **Membership Functions.**.
- Within the Fuzzy Logic Designer window, double-click the blue icon called **tip**.
- At the command line, type `mfedit`.

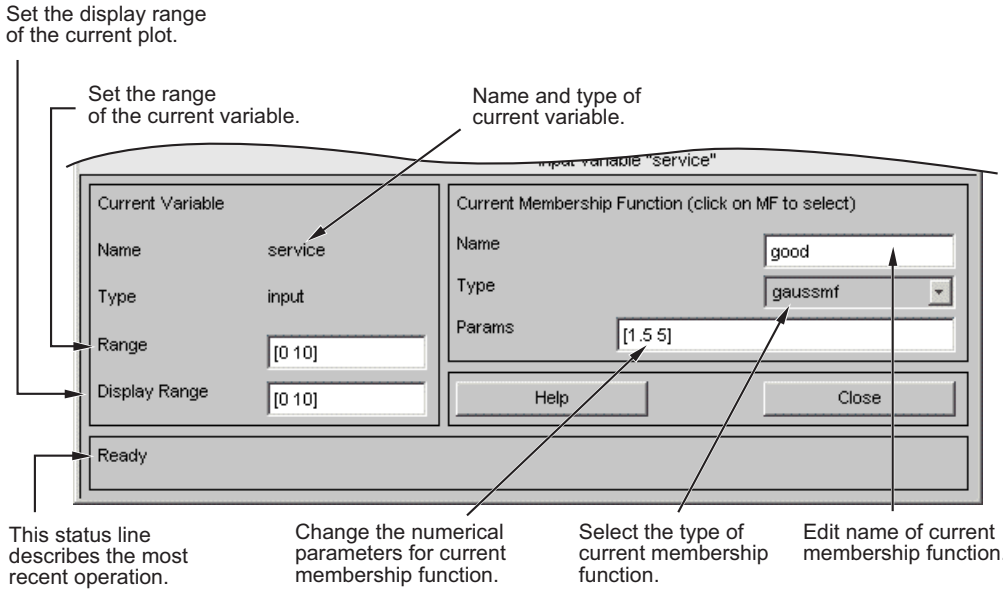
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The Membership Function Editor

The Membership Function Editor is the tool that lets you display and edit all of the membership functions associated with all of the input and output variables for the entire fuzzy inference system. The Membership Function Editor shares some features with the Fuzzy Logic Designer, as shown in the next figure. In fact, all of the five basic GUI tools have similar menu options, status lines, and **Help** and **Close** buttons.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



When you open the Membership Function Editor to work on a fuzzy inference system that does not already exist in the workspace, there are no membership functions associated with the variables that you defined with the Fuzzy Logic Designer.

On the upper-left side of the graph area in the Membership Function Editor is a "Variable Palette" that lets you set the membership functions for a given variable.

To set up the membership functions associated with an input or an output variable for the FIS, select a FIS variable in this region by clicking it.

Next select the **Edit** pull-down menu, and choose **Add MFs ..** A new window appears, which allows you to select both the membership function type and the number of membership functions associated with the selected variable. In the lower-right corner of the window are the controls that let you change the name, type, and parameters (shape), of the membership function, after it is selected.

The membership functions from the current variable are displayed in the main graph. These membership functions can be manipulated in two ways. You can first use the mouse to select a particular membership function associated with a given variable quality, (such as poor, for the variable, service), and then drag the membership function from side to side. This action affects the mathematical description of the quality

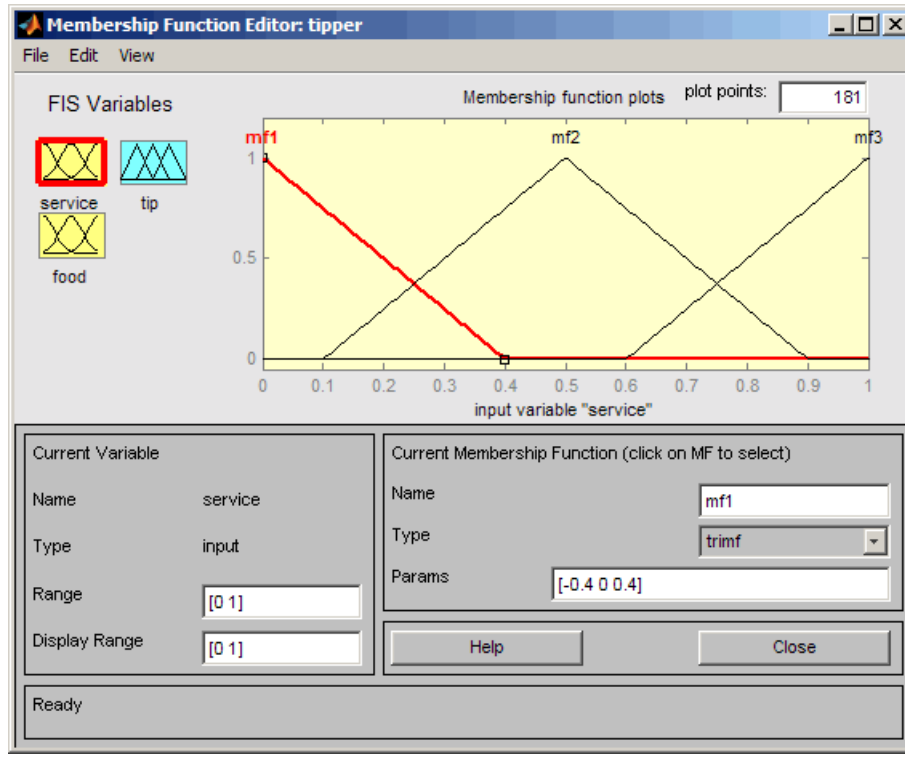
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

associated with that membership function for a given variable. The selected membership function can also be tagged for dilation or contraction by clicking on the small square drag points on the membership function, and then dragging the function with the mouse toward the *outside*, for dilation, or toward the *inside*, for contraction. This action changes the parameters associated with that membership function.

Below the Variable Palette is some information about the type and name of the current variable. There is a text field in this region that lets you change the limits of the current variable's range (universe of discourse) and another that lets you set the limits of the current plot (which has no real effect on the system).

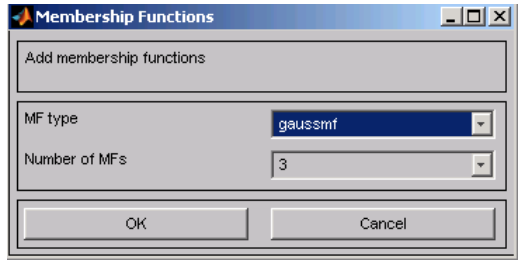
The process of specifying the membership functions for the two input tipping example, `tipper`, is as follows:

1 Double-click the input variable `service` to open the Membership Function Editor.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

2. In the Membership Function Editor, enter `[0 10]` in the **Range** and the **Display Range** fields.
3. Create membership functions for the input variable `service`.
   a. Select **Edit** > **Remove All MFs** to remove the default membership functions for the input variable `service`.
   b. Select **Edit** > **Add MFs.** to open the Membership Functions dialog box.
   c. In the Membership Functions dialog box, select `gaussmf` as the **MF Type**.



   d. Verify that `3` is selected as the **Number of MFs**.
   e. Click **OK** to add three Gaussian curves to the input variable `service`.
4. Rename the membership functions for the input variable `service`, and specify their parameters.
   a. Click on the curve named `mf1` to select it, and specify the following fields in the **Current Membership Function (click on MF to select)** area:
      - In the **Name** field, enter `poor`.
      - In the **Params** field, enter `[1.5 0]`.

        The two inputs of **Params** represent the standard deviation and center for the Gaussian curve.

      Tip To adjust the shape of the membership function, type in the desired parameters or use the mouse, as described previously.

   b. Click on the curve named `mf2` to select it, and specify the following fields in the **Current Membership Function (click on MF to select)** area:
      - In the **Name** field, enter `good`.

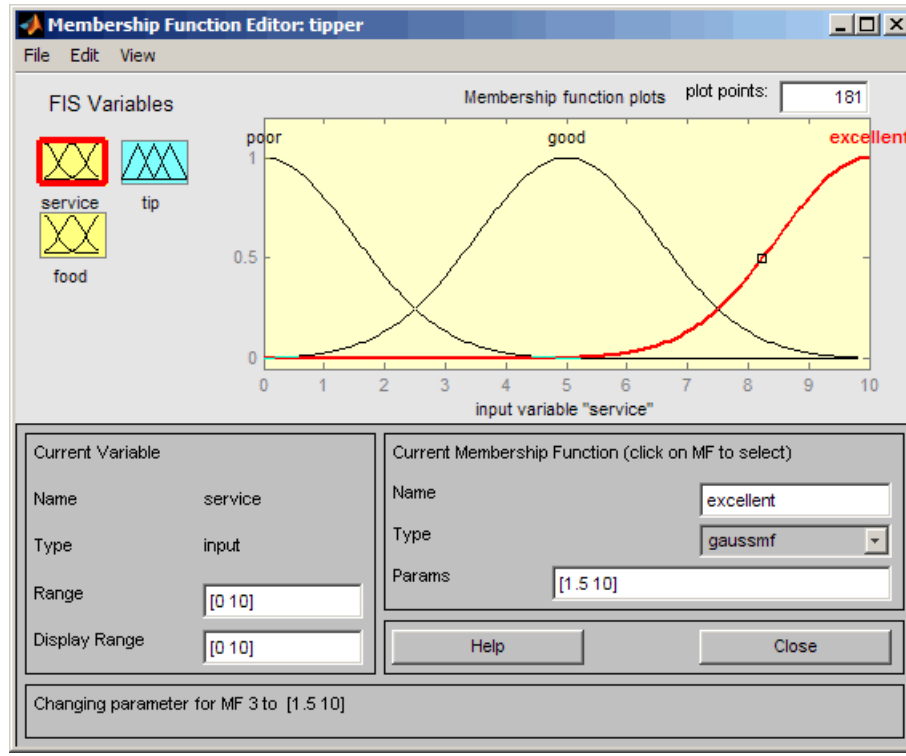
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- In the **Params** field, enter `[1.5 5]`.

c Click on the curve named `mf3`, and specify the following fields in the **Current Membership Function (click on MF to select)** area:

- In the **Name** field, enter `excellent`.
- In the **Params** field, enter `[1.5 10]`.

The Membership Function Editor window looks similar to the following figure.



5 In the **FIS Variables** area, click the input variable `food` to select it.

6 Enter `[0 10]` in the **Range** and the **Display Range** fields.

7 Create the membership functions for the input variable `food`.

a Select **Edit** > **Remove All MFs** to remove the default Membership Functions for the input variable `food`.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

b Select **Edit** > **Add MFs** to open the Membership Functions dialog box.

c In the Membership Functions dialog box, select `trapmf` as the **MF Type**.

d Select `2` in the **Number of MFs** drop-down list.

e Click **OK** to add two trapezoidal curves to the input variable `food`.

8 Rename the membership functions for the input variable `food`, and specify their parameters:

a In the **FIS Variables** area, click the input variable `food` to select it.

b Click on the curve named `mf1`, and specify the following fields in the **Current Membership Function (click on MF to select)** area:

- In the **Name** field, enter `rancid`.
- In the **Params** field, enter `[0 0 1 3]`.

c Click on the curve named `mf2` to select it, and enter `delicious` in the **Name** field.

Reset the associated parameters if desired.

9 Click on the output variable `tip` to select it.

10 Enter `[0 30]` in the **Range** and the **Display Range** fields to cover the output range.

The inputs ranges from 0 to 10, but the output is a tip between 5% and 25%.

11 Rename the default triangular membership functions for the output variable `tip`, and specify their parameters.

a Click the curve named `mf1` to select it, and specify the following fields in the **Current Membership Function (click on MF to select)** area:

- In the **Name** field, enter `cheap`.
- In the **Params** field, enter `[0 5 10]`.

b Click the curve named `mf2` to select it, and specify the following fields in the **Current Membership Function (click on MF to select)** area:

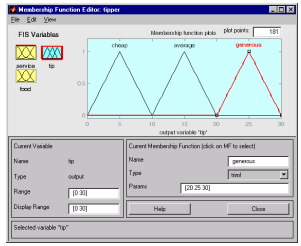
- In the **Name** field, enter `average`.
- In the **Params** field, enter `[10 15 20]`.

c Click the curve named `mf3` to select it, and specify the following:

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- In the **Name** field, enter `generous`.
- In the **Params** field, enter `[20 25 30]`.

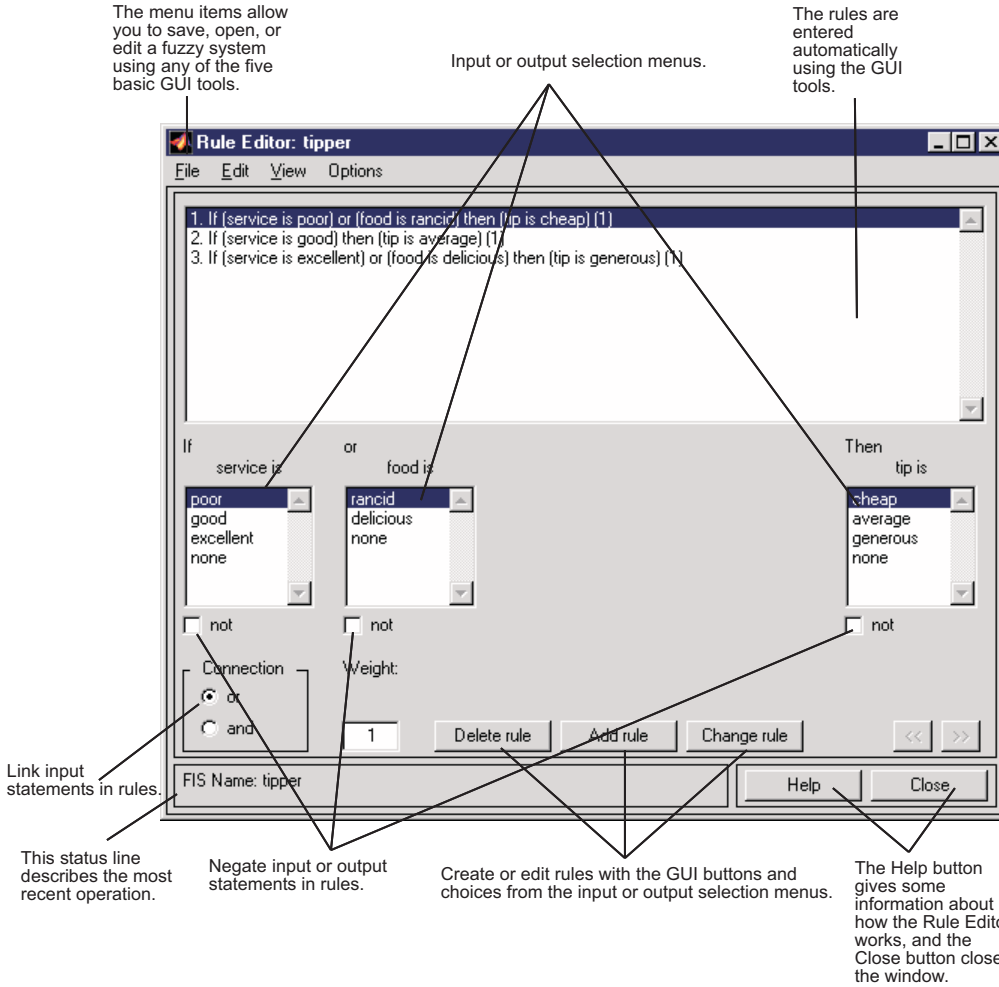
The Membership Function Editor looks similar to the following figure.



Now that the variables have been named and the membership functions have appropriate shapes and names, you can enter the rules. To call up the Rule Editor, go to the **Edit** menu and select **Rules**, or type `ruleedit` at the command line.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The Rule Editor



Constructing rules using the graphical Rule Editor interface is fairly self evident. Based on the descriptions of the input and output variables defined with the Fuzzy Logic Designer, the Rule Editor allows you to construct the rule statements automatically, From the GUI, you can:

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- Create rules by selecting an item in each input and output variable box, selecting one **Connection** item, and clicking **Add Rule**. You can choose `none` as one of the variable qualities to exclude that variable from a given rule and choose `not` under any variable name to negate the associated quality.
- Delete a rule by selecting the rule and clicking **Delete Rule**.
- Edit a rule by changing the selection in the variable box and clicking **Change Rule**.
- Specify weight to a rule by typing in a desired number between `0` and `1` in **Weight**. If you do not specify the weight, it is assumed to be unity `(1)`.

Similar to those in the Fuzzy Logic Designer and the Membership Function Editor, the Rule Editor has the menu bar and the status line. The menu items allow you to open, close, save and edit a fuzzy system using the five basic GUI tools. From the menu, you can also:

- Set the format for the display by selecting **Options** > **Format**.
- Set the language by selecting **Options** > **Language**.

You can access information about the Rule Editor by clicking **Help** and close the GUI using **Close**.

To insert the first rule in the Rule Editor, select the following:

- `poor` under the variable **service**
- `rancid` under the variable **food**
- The **or** radio button, in the **Connection** block
- `cheap`, under the output variable, **tip**.

Then, click **Add rule**.

The resulting rule is
1. If (service is poor) or (food is rancid) then (tip is cheap) (1)

The numbers in the parentheses represent weights.

Follow a similar procedure to insert the second and third rules in the Rule Editor to get

1. *If (service is poor) or (food is rancid) then (tip is cheap) (1)*
2. *If (service is good) then (tip is average) (1)*
3. *If (service is excellent) or (food is delicious) then (tip is generous) (1)*

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Tip To change a rule, first click on the rule to be changed. Next make the desired changes to that rule, and then click **Change rule**. For example, to change the first rule to
1. If (service not poor) or (food not rancid) then (tip is not cheap) (1)

Select the **not** check box under each variable, and then click **Change rule**.

The **Format** pop-up menu from the **Options** menu indicates that you are looking at the verbose form of the rules. Try changing it to `symbolic`. You will see
1. (service==poor) | (food==rancid) => (tip=cheap) (1)
2. (service==good) => (tip=average) (1)
3. (service==excellent) | (food==delicious) => (tip=generous) (1)

There is not much difference in the display really, but it is slightly more language neutral, because it does not depend on terms like *if* and *then*. If you change the format to indexed, you see an extremely compressed version of the rules.
1 1, 1 (1) : 2
2 0, 2 (1) : 1
3 2, 3 (1) : 2

This is the version of the rules that the machine deals with.

- The first column in this structure corresponds to the input variables.
- The second column corresponds to the output variable.
- The third column displays the weight applied to each rule.
- The fourth column is shorthand that indicates whether this is an OR (2) rule or an AND (1) rule.
- The numbers in the first two columns refer to the index number of the membership function.

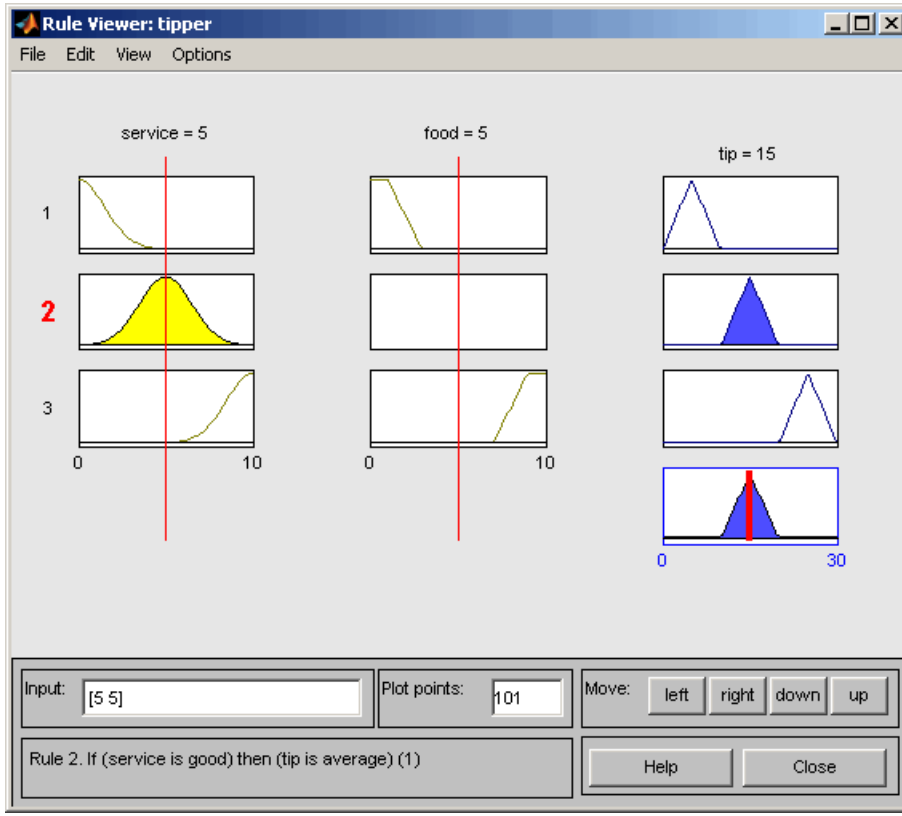
A literal interpretation of rule 1 is “If input 1 is MF1 (the first membership function associated with input 1) or if input 2 is MF1, then output 1 should be MF1 (the first membership function associated with output 1) with the weight 1.

The symbolic format does not consider the terms, *if*, *then*, and so on. The indexed format doesn't even bother with the names of your variables. Obviously the functionality of your system doesn't depend on how well you have named your variables and membership functions. The whole point of naming variables descriptively is, as always, making the system easier for you to interpret. Thus, unless you have some special purpose in mind, it is probably be easier for you to continue with the **verbose** format.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

At this point, the fuzzy inference system has been completely defined, in that the variables, membership functions, and the rules necessary to calculate tips are in place. Now, look at the fuzzy inference diagram presented at the end of the previous section and verify that everything is behaving the way you think it should. You can use the Rule Viewer, the next of the GUI tools we'll look at. From the **View** menu, select **Rules**.

The Rule Viewer



The Rule Viewer displays a roadmap of the whole fuzzy inference process. It is based on the fuzzy inference diagram described in the previous section. You see a single figure window with 10 plots nested in it. The three plots across the top of the figure represent the antecedent and consequent of the first rule. Each rule is a row of plots, and each column is a variable. The rule numbers are displayed on the left of each row. You can click on a rule number to view the rule in the status line.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- The first two columns of plots (the six yellow plots) show the membership functions referenced by the antecedent, or the if-part of each rule.
- The third column of plots (the three blue plots) shows the membership functions referenced by the consequent, or the then-part of each rule.

  Notice that under **food**, there is a plot which is blank. This corresponds to the characterization of `none` for the variable **food** in the second rule.
- The fourth plot in the third column of plots represents the aggregate weighted decision for the given inference system.

  This decision will depend on the input values for the system. The defuzzified output is displayed as a bold vertical line on this plot.

The variables and their current values are displayed on top of the columns. In the lower left, there is a text field **Input** in which you can enter specific input values. For the two-input system, you will enter an input vector, `[9 8]`, for example, and then press **Enter**. You can also adjust these input values by clicking on any of the three plots for each input. This will move the red index line horizontally, to the point where you have clicked. Alternatively, you can also click and drag this line in order to change the input values. When you release the line, (or after manually specifying the input), a new calculation is performed, and you can see the whole fuzzy inference process take place:

- Where the index line representing service crosses the membership function line "service is poor" in the upper-left plot determines the degree to which rule one is activated.
- A yellow patch of color under the actual membership function curve is used to make the fuzzy membership value visually apparent.

Each of the characterizations of each of the variables is specified with respect to the input index line in this manner. If you follow rule 1 across the top of the diagram, you can see the consequent "tip is cheap" has been truncated to exactly the same degree as the (composite) antecedent—this is the implication process in action. The aggregation occurs down the third column, and the resultant aggregate plot is shown in the single plot appearing in the lower right corner of the plot field. The defuzzified output value is shown by the thick line passing through the aggregate fuzzy set.

You can shift the plots using **left**, **right**, **down**, and **up**. The menu items allow you to save, open, or edit a fuzzy system using any of the five basic GUI tools.

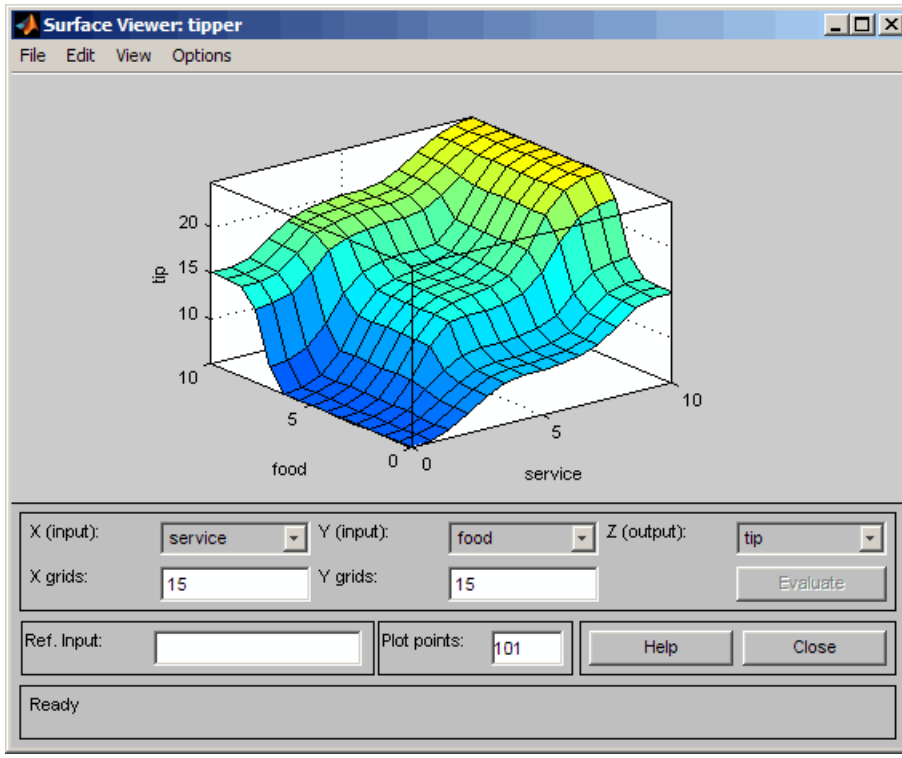
The Rule Viewer allows you to interpret the entire fuzzy inference process at once. The Rule Viewer also shows how the shape of certain membership functions influences

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

the overall result. Because it plots every part of every rule, it can become unwieldy for particularly large systems, but, for a relatively small number of inputs and outputs, it performs well (depending on how much screen space you devote to it) with up to 30 rules and as many as 6 or 7 variables.

The Rule Viewer shows one calculation at a time and in great detail. In this sense, it presents a sort of micro view of the fuzzy inference system. If you want to see the entire output surface of your system—the entire span of the output set based on the entire span of the input set—you need to open up the Surface Viewer. This viewer is the last of the five basic Fuzzy Logic Toolbox GUI tools. To open the Surface Viewer, select **Surface** from the **View** menu.

The Surface Viewer



Upon opening the Surface Viewer, you see a three-dimensional curve that represents the mapping from food and service quality to tip amount. Because this curve represents a

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

two-input one-output case, you can see the entire mapping in one plot. When we move beyond three dimensions overall, we start to encounter trouble displaying the results.

Accordingly, the Surface Viewer is equipped with drop-down menus **X (input):**, **Y (input):** and **Z (output):** that let you select any two inputs and any one output for plotting. Below these menus are two input fields **X grids:** and **Y grids:** that let you specify how many x-axis and y-axis grid lines you want to include. This capability allows you to keep the calculation time reasonable for complex problems.

If you want to create a smoother plot, use the **Plot points** field to specify the number of points on which the membership functions are evaluated in the input or output range. By default, the value of this field is 101.

Clicking **Evaluate** initiates the calculation, and the plot is generated after the calculation is complete. To change the *x*-axis or *y*-axis grid after the surface is in view, change the appropriate input field, and press **Enter**. The surface plot is updated to reflect the new grid settings.

The Surface Viewer has a special capability that is very helpful in cases with two (or more) inputs and one output: you can grab the axes, using the mouse and reposition them to get a different three-dimensional view on the data.

The **Ref. Input** field is used in situations when there are more inputs required by the system than the surface is mapping. You can edit this field to explicitly set inputs not specified in the surface plot.

Suppose you have a four-input one-output system and would like to see the output surface. The Surface Viewer can generate a three-dimensional output surface where any two of the inputs vary, but two of the inputs must be held constant because computer monitors cannot display a five-dimensional shape. In such a case, the input is a four-dimensional vector with `NaN`s holding the place of the varying inputs while numerical values indicates those values that remain fixed. A NaN is the IEEE® symbol for Not a Number.

The menu items allow you to open, close, save and edit a fuzzy system using the five basic GUI tools. You can access information about the Surface Viewer by clicking **Help** and close the GUI using **Close**.

This concludes the quick walk-through of each of the main GUI tools. For the tipping problem, the output of the fuzzy system matches your original idea of the shape of the fuzzy mapping from service to tip fairly well. In hindsight, you might say, “Why bother? I

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

could have just drawn a quick lookup table and been done an hour ago!" However, if you are interested in solving an entire class of similar decision-making problems, fuzzy logic may provide an appropriate tool for the solution, given its ease with which a system can be quickly modified.

Importing and Exporting from the GUI Tools

When you save a fuzzy system to a file, you are saving an ASCII text FIS file representation of that system with the file suffix `.fis`. This text file can be edited and modified and is simple to understand. When you save your fuzzy system to the MATLAB workspace, you are creating a variable (whose name you choose) that acts as a MATLAB structure for the FIS system. FIS files and FIS structures represent the same system.

Note If you do not save your FIS to a file, but only save it to the MATLAB workspace, you cannot recover it for use in a new MATLAB session.

---

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Build Mamdani Systems Using Custom Functions

In this section...

"How to Build Fuzzy Inference Systems Using Custom Functions in the Designer" on page 2-54

"Specifying Custom Membership Functions" on page 2-55

"Specifying Custom Inference Functions" on page 2-60

How to Build Fuzzy Inference Systems Using Custom Functions in the Designer

When you build a fuzzy inference system, as described in "Fuzzy Inference Process" on page 2-22, you can replace the built-in membership functions or inference functions, or both with custom functions. In this section, you learn how to build a fuzzy inference system using custom functions in the GUI. To learn how to build the system using custom functions at the command line, see "Specifying Custom Membership and Inference Functions" on page 2-72 in "Build Mamdani Systems (Code)" on page 2-65.

To build a fuzzy inference system using custom functions in the designer:

1. Open the Fuzzy Logic Designer by typing `fuzzyLogicDesigner` at the MATLAB prompt.
2. Specify the number of inputs and outputs of the fuzzy system, as described in "The Fuzzy Logic Designer" on page 2-34.
3. Create custom membership functions, and replace the built-in membership functions with them, as described in "Specifying Custom Membership Functions" on page 2-55.

   Membership functions define how each point in the input space is mapped to a membership value between 0 and 1.
4. Create rules using the Rule Editor, as described in "The Rule Editor" on page 2-46.

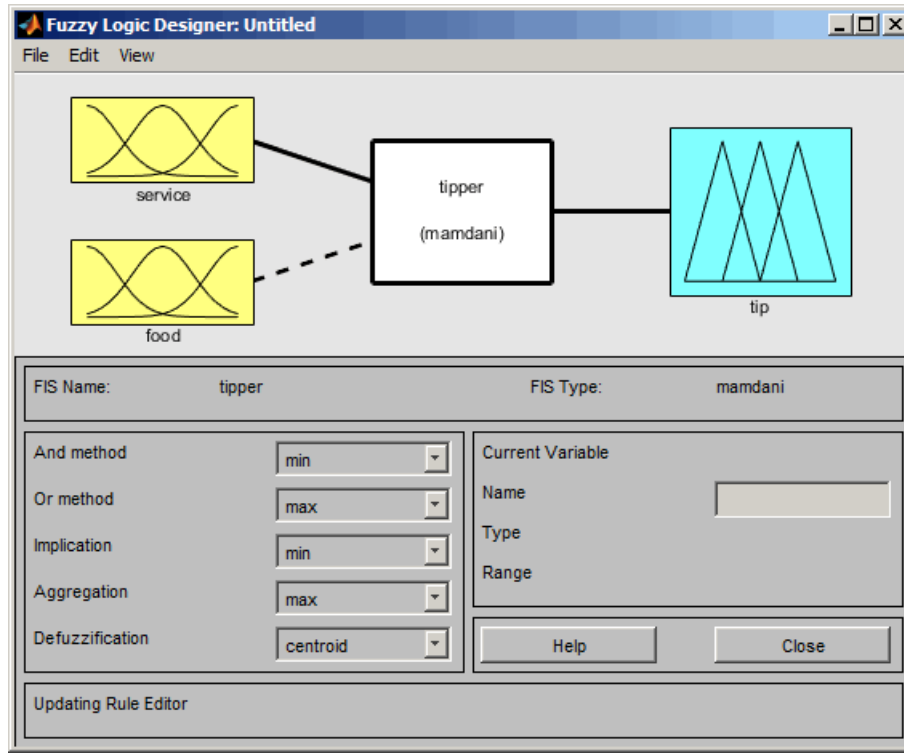
   Rules define the logical relationship between the inputs and the outputs.
5. Create custom inference functions, and replace the built in inference functions with them, as described in "Specifying Custom Inference Functions" on page 2-60.

www.MatlabSite.com | متلب سایت

مرجع آموزش برنامه‌نویسی متلب در ایران

Inference methods include the AND, OR, implication, aggregation and defuzzification methods. This action generates the output values for the fuzzy system.

The next figure shows the tipping problem example where the built-in **Implication** and **Defuzzification** functions are replaced with a custom implication function, `customimp`, and custom defuzzification function, `customdefuzz`, respectively.



6 Select **View** > **Surface** to view the output of the fuzzy inference system in the Surface Viewer, as described in "The Surface Viewer" on page 2-51.

Specifying Custom Membership Functions

You can create custom membership functions, and use them in the fuzzy inference process. The values of these functions must lie between 0 and 1. You must save the

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

custom membership functions in your current working folder. To learn how to build fuzzy systems using custom membership functions, see "How to Build Fuzzy Inference Systems Using Custom Functions in the Designer" on page 2-54.

To create a custom membership function, and replace the built-in membership function:

1 Create a MATLAB function, and save it in your current working folder.

   To learn how to create MATLAB functions, see "Scripts vs. Functions" in the MATLAB documentation.

   The following code is an example of a multi-step custom membership function, `custmf1`, that depends on eight parameters between `0` and `10`.

```
% Function to generate a multi-step custom membership function
% using 8 parameters for the input argument x
function out = custmf1(x, params)
for i=1:length(x)
if x(i)<params(1)
 y(i)=params(1);
elseif x(i)<params(2)
 y(i)=params(2);
elseif x(i)<params(3)
 y(i)=params(3);
elseif x(i)<params(4)
 y(i)=params(4);
elseif x(i)<params(5)
 y(i)=params(5);
elseif x(i)<params(6)
 y(i)=params(6);
elseif x(i)<params(7)
 y(i)=params(7);
elseif x(i)<params(8)
 y(i)=params(8);
else
 y(i)=0;
end
end
out=.1*y'; % scaling the output to lie between 0 and 1
```

   Note: Custom membership functions can include a maximum of 16 parameters for the input argument.

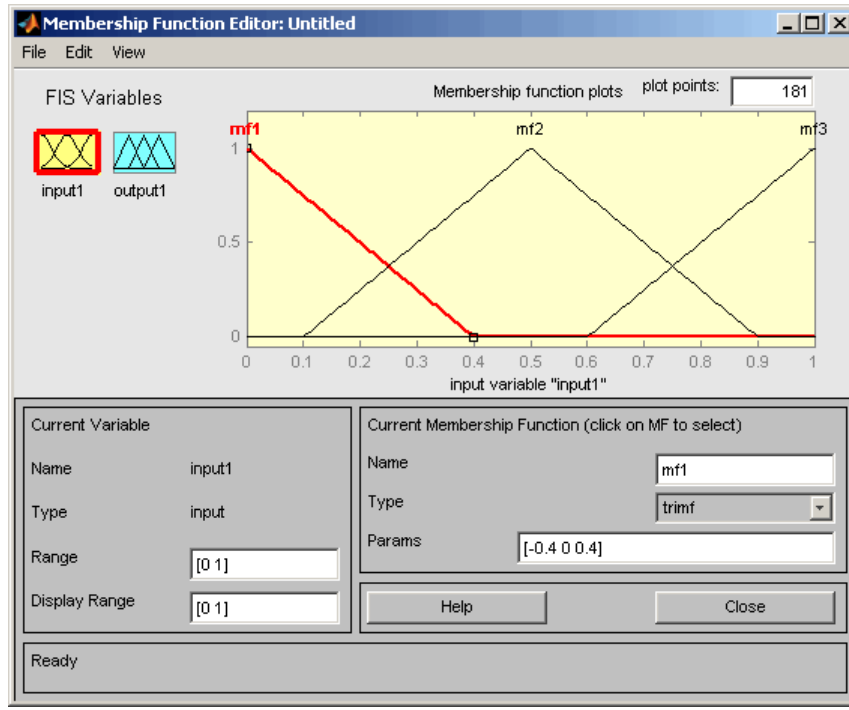
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

2 Open the Neuro–Fuzzy Designer by typing `fuzzyLogicDesigner` at the MATLAB prompt, if you have not done so already.

The Fuzzy Logic Designer opens with the default FIS name, `Untitled`, and contains one input **input1**, and one output **output1**.

3 In the Fuzzy Logic Designer, select **Edit** > **Membership Functions** to open the Membership Function Editor.

Three triangular-shaped membership functions for **input1** are displayed by default.

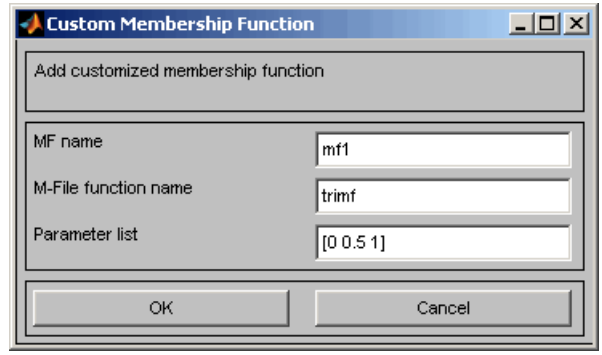


4 To replace the default membership function with a custom function in the Membership Function Editor:

a Select **Edit** > **Remove All MFs** to remove the default membership functions for **input1**.

b Select **Edit** > **Add Custom MF** to open the Custom Membership Function dialog box.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



5 To specify a custom function in the Custom Membership Function dialog box:

a Specify a name for the custom membership function in the **MF name** field.

Note: When adding additional custom membership functions, you must specify a different **MF name** for each function.

---

b Specify the name of the custom membership function file in the **M-file function name** field.

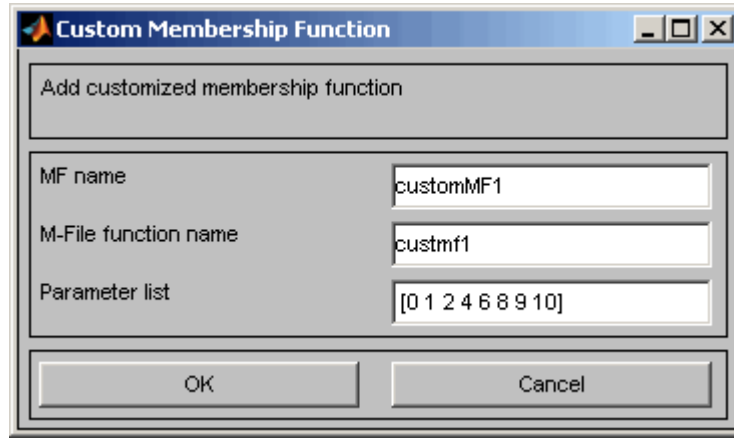
c Specify a vector of parameters in the **Parameter list** field.

These values determine the shape and position of the membership function, and the function is evaluated using these parameter values.

Note: The length of the parameter vector must be greater than or equal to the number of parameters in the custom membership function.

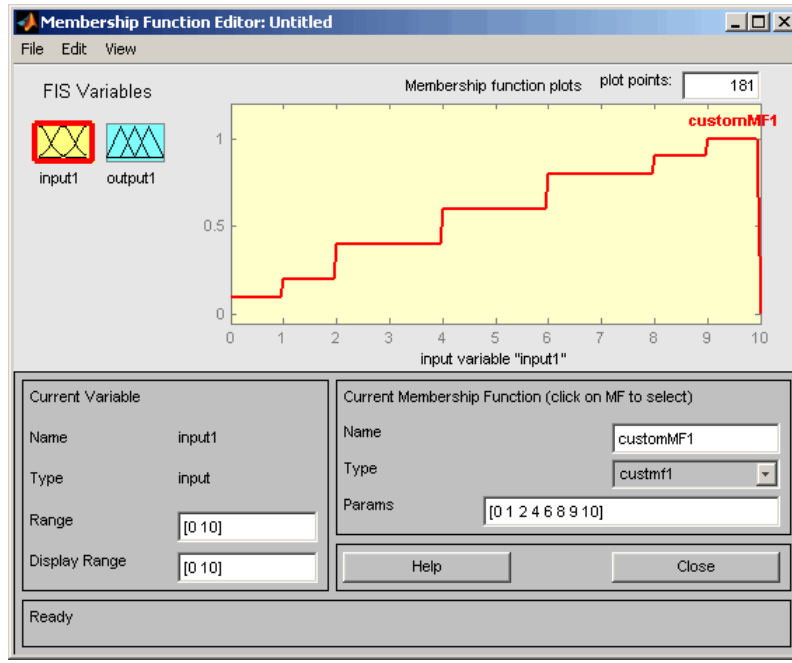
Using the `custmf1` example in step 1, the Custom Membership Function dialog box looks similar to the following figure.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



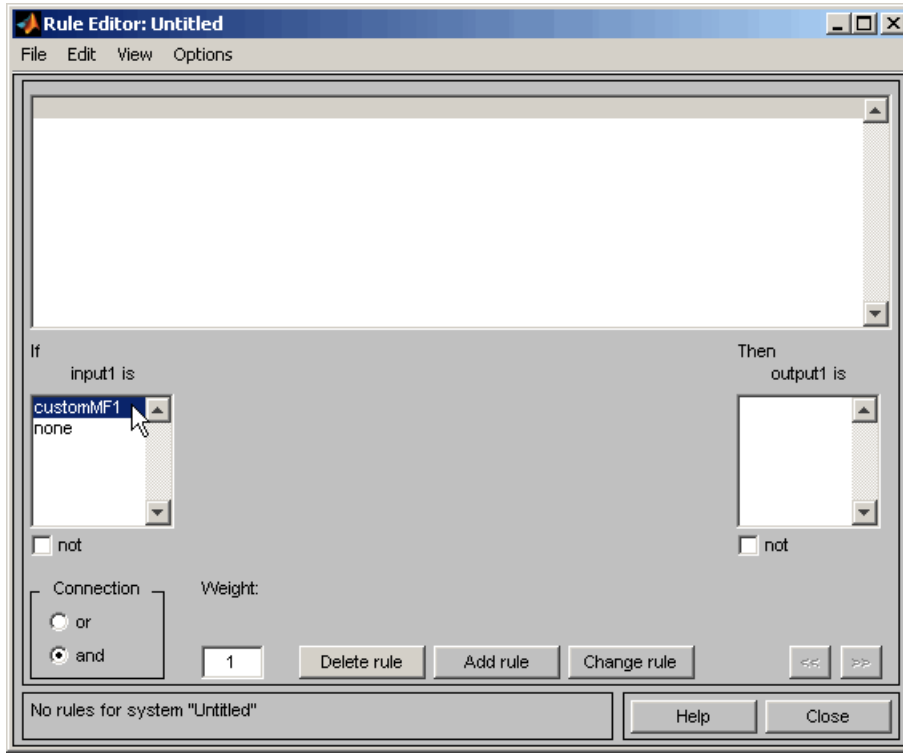
d Click **OK** to add the custom membership function.

The Membership Function Editor displays the custom membership function plot.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

This action also adds the custom membership function to the Rule Viewer, and is now available for creating rules for the fuzzy inference process. To view the custom function in the Rule Viewer, select **Edit** > **Rules** in either the Fuzzy Logic Designer or the Membership Function Editor.



6 To add custom membership functions for **output1**, select it in the Membership Function Editor, and repeat steps 4 and 5.

Specifying Custom Inference Functions

You can replace the built-in AND, OR, implication, aggregation, and defuzzification inference methods with custom functions. After you create the custom inference function, save it in your current working folder. To learn how to build fuzzy systems using custom inference functions, see the "How to Build Fuzzy Inference Systems Using Custom Functions in the Designer" on page 2-54 section.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

You must follow a few guidelines when creating custom inference functions. The guidelines for creating and specifying the functions for building fuzzy inference systems are described in the following sections.

- "Guidelines for Creating Custom AND and OR Functions" on page 2-61
- "Guidelines for Creating Custom Implication Functions" on page 2-62
- "Guidelines for Creating Custom Aggregation Functions" on page 2-63
- "Guidelines for Creating Custom Defuzzification Functions" on page 2-63
- "Steps for Specifying Custom Inference Functions" on page 2-63

Guidelines for Creating Custom AND and OR Functions

The custom AND and OR inference functions must operate column-wise on a matrix, in the same way as the MATLAB functions `max`, `min`, or `prod`.

For a row or column matrix `x`, `min(x)` returns the minimum element.

```
x=[1 2 3 4];
min(x)
ans =
     1
```

For a matrix `x`, `min(x)` returns a row vector containing the minimum element from each column.

```
x=[1 2 3 4;5 6 7 8;9 10 11 12];
min(x)
ans =
     1     2     3     4
```

For N-D arrays, `min(x)` operates along the first non-singleton dimension.

The function `min(x,y)` returns an array that is same size as `x` and `y` with the minimum elements from `x` or `y`. Either of the input arguments can be a scalar. Functions such as `max`, and `prod` operate in a similar manner.

In the toolbox, the AND implication methods perform an element by element matrix operation, similar to the MATLAB function `min(x,y)`.

```
a=[1 2; 3 4];
b=[2 2; 2 2];
min(a,b)
ans =
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
    1     2
    2     2
```

The OR implication methods perform an element by element matrix operation, similar to the MATLAB function `max(x,y)`.

Guidelines for Creating Custom Implication Functions

Custom implication functions must operate in the same way as the MATLAB functions `max`, `min`, or `prod`.

The following is an example of a custom implication function:

```
function y=custom_imp(x1,x2)

if nargin==1
    y=prod(x1).^2;
else
    y=(x1.*x2).^2;
end
```

An implication function must support either one or two inputs because the software calls the function in two ways:

- To calculate the output fuzzy set values using the firing strength of all the rules and the corresponding output membership functions. In this case, the software calls the implication function using two inputs, similar to the following example:

  ```
  impvals = custom_imp(w,outputmf)
  ```

  - `w` — Firing strength of multiple rules, specified as an *nr*-by-*ns* matrix. Here, *nr* is the number of rules and *ns* is the number of samples of the output membership functions.

    `w(:,j) = w(:,1)` for all *j*. `w(i,1)` is the firing strength of the ith rule.
  - `outputmf` — Output membership function values, specified as an *nr*-by-*ns* matrix. Here, *nr* is the number of rules and *ns* is the number of samples of the output membership functions.

    `outputmf(i,:)` contains the data of the ith output membership function.
- To calculate the output fuzzy value using the firing strength of a single rule and the corresponding output membership function, for a given sample. In this case,

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

the software calls the implication function using one input, similar to the following example:

```
impval = custom_imp([w outputmf])
```

`w` and `outputmf` are scalar values representing the firing strength of a rule and the corresponding output membership function value, for a given sample.

Guidelines for Creating Custom Aggregation Functions

The custom aggregation functions must operate in the same way as the MATLAB functions `max`, `min`, or `prod` and must be of the form `y = custom_agg(x)`.

x is an *nv*-by-*nr* matrix, which is the list of truncated output functions returned by the implication method for each rule. *nv* is the number of output variables, and *nr* is the number of rules. The output of the aggregation method is one fuzzy set for each output variable.

The following is an example of a custom aggregation function:

```
function aggfun = custom_agg(x)
aggfun=(sum(x)/2).^0.5;
```

Guidelines for Creating Custom Defuzzification Functions

The custom defuzzification functions must be of the form `y = custom_defuzz(xmf,ymf)`, where *(xmf,ymf)* is a finite set of membership function values. *xmf* is the vector of values in the membership function input range. *ymf* is the value of the membership function at *xmf*.

The following is an example of a custom defuzzification function:

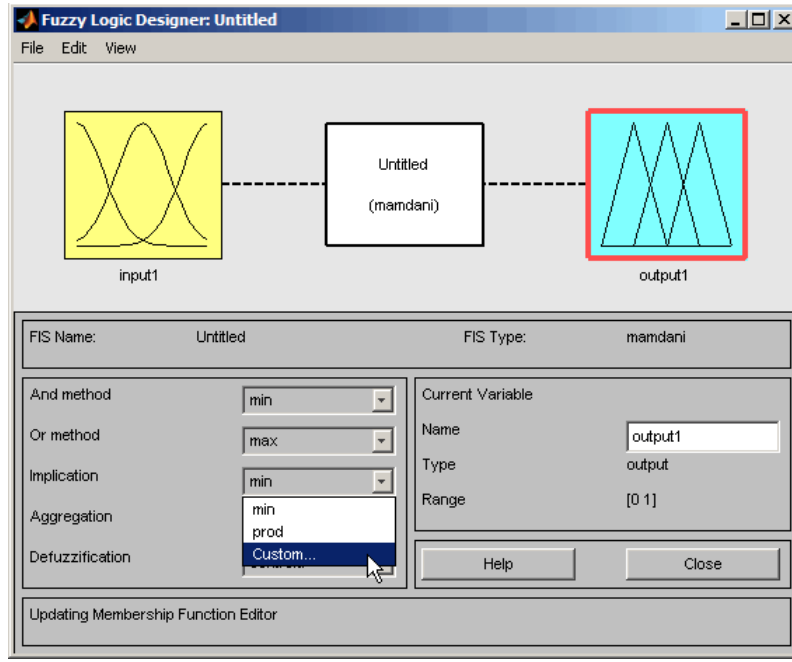
```
function defuzzfun= custom_defuzz(xmf,ymf);
total_area=sum(ymf);
defuzzfun=sum(ymf*xmf)/total_area;
```

Steps for Specifying Custom Inference Functions

After you create and save a custom inference function, use the following steps to specify the function in the fuzzy inference process:

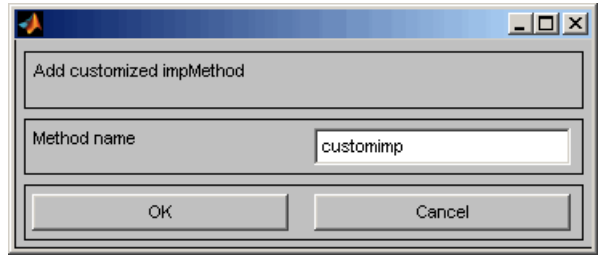
1. In the lower-left panel of the Fuzzy Logic Designer, select `Custom` from the drop-down menu corresponding to the inference method for which you want to specify the custom function.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



This action opens a dialog box where you specify the name of the custom inference function.

2 In the **Method name** field, specify the name of the custom inference function, and click **OK**.



The custom function replaces the built-in function when building the fuzzy inference system.

3 To specify custom functions for other inference methods, repeat steps 1 and 2.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Build Mamdani Systems (Code)

In this section...

"Tipping Problem from the Command Line" on page 2-65
"System Display Functions" on page 2-67
"Building a System from Scratch" on page 2-70
"FIS Evaluation" on page 2-73
"The FIS Structure" on page 2-73

Tipping Problem from the Command Line

The tipping problem is one of many Fuzzy Logic Toolbox examples of fuzzy inference systems. The FIS is always cast as a MATLAB structure. To load this system, type:

```
a = readfis('tipper.fis')
```

This command returns the following result:

```
a =
            name: 'tipper'
            type: 'mamdani'
       andMethod: 'min'
        orMethod: 'max'
    defuzzMethod: 'centroid'
       impMethod: 'min'
       aggMethod: 'max'
           input: [1x2 struct]
          output: [1x1 struct]
            rule: [1x3 struct]
```

The labels on the left of this listing represent the various components of the MATLAB structure associated with `tipper.fis`. To access the various components of this structure, type the component name after `a` at the MATLAB prompt. For example:

```
a.type
```

returns the following result:

```
ans =
mamdani
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The function

```
getfis(a)
```

returns almost the same structure information that typing a, alone does:

```
Name      = tipper
      Type      = mamdani
      NumInputs = 2
      InLabels  =
            service
            food
      NumOutputs = 1
      OutLabels =
            tip
      NumRules = 3
      AndMethod = min
      OrMethod = max
      ImpMethod = min
      AggMethod = max
      DefuzzMethod = centroid
```

Some of these fields are not part of the structure a. Thus, you cannot get information by typing a.Inlabels, but you can get it by typing

```
getfis(a,'Inlabels')
```

Similarly, you can obtain structure information using getfis in this manner.

```
getfis(a,'input',1)
getfis(a,'output',1)
getfis(a,'input',1,'mf',1)
```

The structure.field syntax also generates this information. For more information on the syntax for MATLAB structures and cell arrays, see “Create a Structure Array” and “Create a Cell Array” in the MATLAB documentation.

For example, type

```
a.input
```

or

```
a.input(1).mf(1)
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The function `getfis` is loosely modeled on the Handle Graphics® function `get`. The function `setfis` acts as the reciprocal to `getfis`. It allows you to change any property of a FIS. For example, if you wanted to change the name of this system, you could type

```
a = setfis(a,'name','gratuity');
```

However, because `a` is already a MATLAB structure, you can set this information more simply by typing

```
a.name = 'gratuity';
```

Now the FIS structure `a` has been changed to reflect the new name. If you want a little more insight into this FIS structure, try

```
showfis(a)
```

This syntax returns a printout listing all the information about `a`. This function is intended more for debugging than anything else, but it shows all the information recorded in the FIS structure

Because the variable, `a`, designates the fuzzy tipping system, you can display any of the GUIs for the tipping system directly from the command line. Any of the following functions will display the tipping system with the associated GUI:

- `fuzzyLogicDesigner(a)` displays the Fuzzy Logic Designer.
- `mfedit(a)` displays the Membership Function Editor.
- `ruleedit(a)` displays the Rule Editor.
- `ruleview(a)` displays the Rule Viewer.
- `surfview(a)` displays the Surface Viewer.

If, in addition, `a` is a Sugeno-type FIS, then `neuroFuzzyDesigner(a)` displays the Neuro–Fuzzy Designer.

When you open any of these GUIs, you can access any of the other GUIs using the pull-down menu rather than the command line.

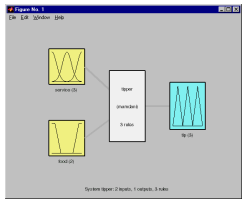
System Display Functions

There are three functions designed to give you a high-level view of your fuzzy inference system from the command line: `plotfis`, `plotmf`, and `gensurf`. The first of these

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

displays the whole system as a block diagram much as it would appear on the Fuzzy Logic Designer.

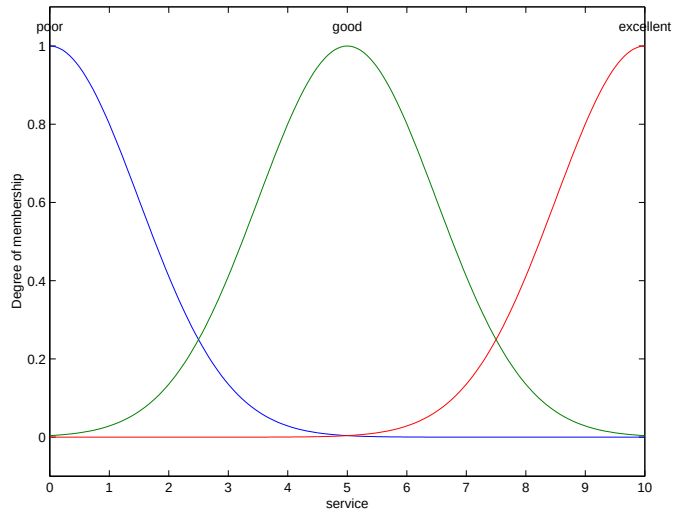
```
plotfis(a)
```



After closing any open MATLAB figures or GUI windows, the function `plotmf` plots all the membership functions associated with a given variable as follows.

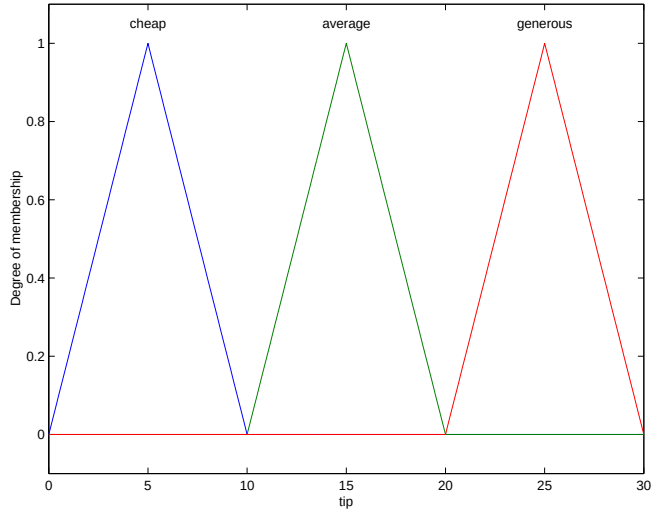
```
plotmf(a,'input',1)
```

returns the following plots




```
plotmf(a,'output',1)
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

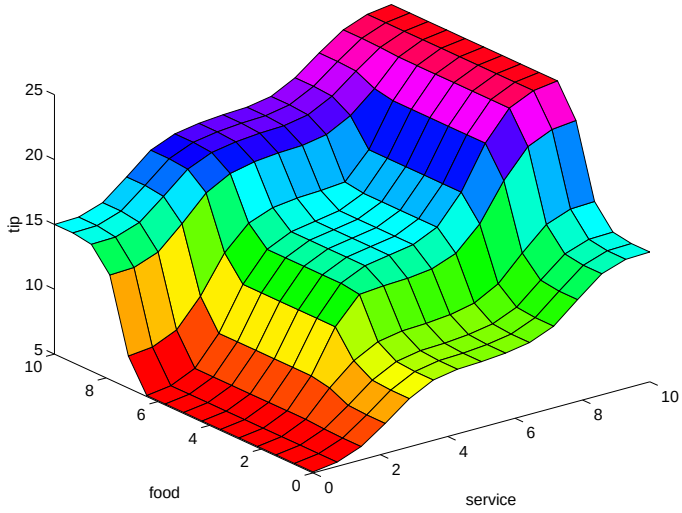


These plots appear in the Membership Function Editor GUI, or in an open MATLAB figure, if `plotmf` is called while either of these is open.

Finally, the function `gensurf` plots any one or two inputs versus any one output of a given system. The result is either a two-dimensional curve, or a three-dimensional surface. When there are three or more inputs, `gensurf` must be generated with all but two inputs fixed, as is described in `gensurf`.

```
gensurf(a)
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Building a System from Scratch

You can build a fuzzy inference system using Fuzzy Logic Toolbox commands as an alternative to the GUI tools. For example, to build the tipping system entirely from the command line, you use the commands `newfis`, `addvar`, `addmf`, and `addrule`.

Probably the most difficult part of this process is learning the shorthand that the fuzzy inference systems use for building rules. Use the command line function, `addrule` to do this.

Each variable, input, or output, has an index number, and each membership function has an index number. The rules are built from statements such as the following:

If input1 is MF1 or input2 is MF3, then output1 is MF2 (weight = 0.5)

This rule is turned into a structure according to the following logic. If there are *m* inputs to a system and *n* outputs, then the first *m* vector entries of the rule structure correspond to inputs 1 through *m*.

- The entry in column 1 is the index number for the membership function associated with input 1.
- The entry in column 2 is the index number for the membership function associated with input 2, and so on.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- The next n columns work the same way for the outputs.
- Column $m + n + 1$ is the weight associated with that rule (typically 1) and column $m + n + 2$ specifies the connective used (where AND = 1 and OR = 2).

The structure associated with the preceding rule is

```
1 3 2 0.5 2
```

This sample code shows one way you can build the entire tipping system from the command line, using the MATLAB structure syntax.

```
a=newfis('tipper');
a.input(1).name='service';
a.input(1).range=[0 10];
a.input(1).mf(1).name='poor';
a.input(1).mf(1).type='gaussmf';
a.input(1).mf(1).params=[1.5 0];
a.input(1).mf(2).name='good';
a.input(1).mf(2).type='gaussmf';
a.input(1).mf(2).params=[1.5 5];
a.input(1).mf(3).name='excellent';
a.input(1).mf(3).type='gaussmf';
a.input(1).mf(3).params=[1.5 10];
a.input(2).name='food';
a.input(2).range=[0 10];
a.input(2).mf(1).name='rancid';
a.input(2).mf(1).type='trapmf';
a.input(2).mf(1).params=[-2 0 1 3];
a.input(2).mf(2).name='delicious';
a.input(2).mf(2).type='trapmf';
a.input(2).mf(2).params=[7 9 10 12];
a.output(1).name='tip';
a.output(1).range=[0 30];
a.output(1).mf(1).name='cheap'
a.output(1).mf(1).type='trimf';
a.output(1).mf(1).params=[0 5 10];
a.output(1).mf(2).name='average';
a.output(1).mf(2).type='trimf';
a.output(1).mf(2).params=[10 15 20];
a.output(1).mf(3).name='generous';
a.output(1).mf(3).type='trimf';
a.output(1).mf(3).params=[20 25 30];
a.rule(1).antecedent=[1 1];
a.rule(1).consequent=[1];
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
a.rule(1).weight=1;
a.rule(1).connection=2;
a.rule(2).antecedent=[2 0];
a.rule(2).consequent=[2];
a.rule(2).weight=1;
a.rule(2).connection=1;
a.rule(3).antecedent=[3 2];
a.rule(3).consequent=[3];
a.rule(3).weight=1;
a.rule(3).connection=2
```

Tip You can also build the FIS structure using MATLAB workspace variables. For example, to specify the range of the input, type:

```
r1= [0 10]
a.input(1).range = r1;
```

Alternatively, you can build the entire tipping system from the command line using Fuzzy Logic Toolbox commands. These commands are in the `mktipper.m` file.

```
a=newfis('tipper');
a=addvar(a,'input','service',[0 10]);
a=addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a=addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a=addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
a=addvar(a,'input','food',[0 10]);
a=addmf(a,'input',2,'rancid','trapmf',[-2 0 1 3]);
a=addmf(a,'input',2,'delicious','trapmf',[7 9 10 12]);
a=addvar(a,'output','tip',[0 30]);
a=addmf(a,'output',1,'cheap','trimf',[0 5 10]);
a=addmf(a,'output',1,'average','trimf',[10 15 20]);
a=addmf(a,'output',1,'generous','trimf',[20 25 30]);
ruleList=[ ...
1 1 1 1 2
2 0 2 1 1
3 2 3 1 2 ];
a=addrule(a,ruleList);
```

Specifying Custom Membership and Inference Functions

You can create custom membership and inference functions as described in “Specifying Custom Membership Functions” on page 2-55, and “Specifying Custom Inference

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Functions" on page 2-60, and specify them for building fuzzy inference systems at the command line.

To include a custom membership function, specify the name of the custom membership function, as shown in the following example:

```
a=addmf(a,'input',1,'customMF1','custmf1',[0 1 2 4 6 8 9 10]);
```

To include a custom inference function, specify the name of the custom inference function, as shown in the following example:

```
a.defuzzMethod='customdefuzz';
```

FIS Evaluation

To evaluate the output of a fuzzy system for a given input, use the function `evalfis`. For example, the following script evaluates `tipper` at the input, [1 2].

```
a = readfis('tipper');
evalfis([1 2], a)
ans =
    5.5586
```

This function can also be used for multiple collections of inputs, because different input vectors are represented in different parts of the input structure.

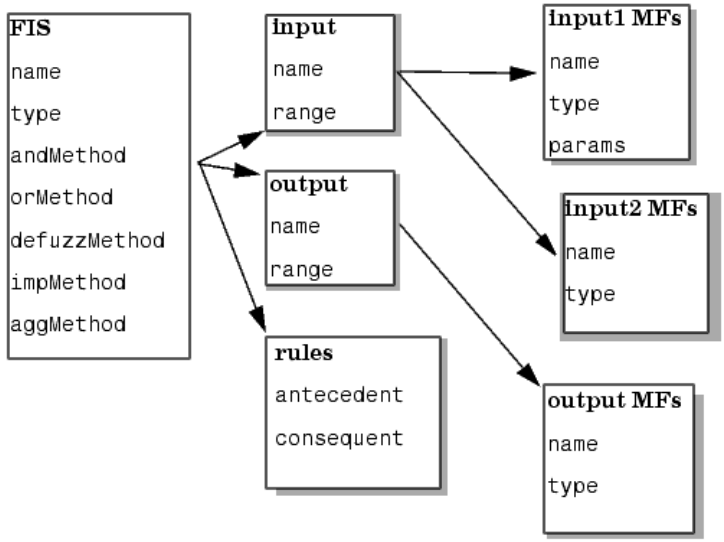
```
evalfis([3 5; 2 7], a)
ans =
    12.2184
     7.7885
```

The FIS Structure

The FIS structure is the MATLAB object that contains all the fuzzy inference system information. This structure is stored inside each GUI tool. Access functions such as `getfis` and `setfis` make it easy to examine this structure.

All the information for a given fuzzy inference system is contained in the FIS structure, including variable names, membership function definitions, and so on. This structure can itself be thought of as a hierarchy of structures, as shown in the following diagram.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



You can generate a listing of information on the FIS using the `showfis` command, as shown in the following code sample:

```
showfis(a)
1.  Name             tipper
2.  Type             mamdani
3.  Inputs/Outputs   [ 2 1 ]
4.  NumInputMFs      [ 3 2 ]
5.  NumOutputMFs     3
6.  NumRules         3
7.  AndMethod        min
8.  OrMethod         max
9.  ImpMethod        min
10. AggMethod        max
11. DefuzzMethod     centroid
12. InLabels         service
13.                  food
14. OutLabels        tip
15. InRange          [ 0 10 ]
16.                  [ 0 10 ]
17. OutRange         [ 0 30 ]
18. InMFLabels       poor
19.                  good
20.                  excellent
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
21.                  rancid
22.                  delicious
23. OutMFLabels      cheap
24.                  average
25.                  generous
26. InMFTypes        gaussmf
27.                  gaussmf
28.                  gaussmf
29.                  trapmf
30.                  trapmf
31. OutMFTypes       trimf
32.                  trimf
33.                  trimf
34. InMFParams       [ 1.5 0 0 0 ]
35.                  [ 1.5 5 0 0 ]
36.                  [ 1.5 10 0 0 ]
37.                  [ 0 0 1 3 ]
38.                  [ 7 9 10 10 ]
39. OutMFParams      [ 0 5 10 0 ]
40.                  [ 10 15 20 0 ]
41.                  [ 20 25 30 0 ]
42. Rule Antecedent  [ 1 1 ]
43.                  [ 2 0 ]
44.                  [ 3 2 ]
42. Rule Consequent  1
43.                  2
44.                  3
42. Rule Weight      1
43.                  1
44.                  1
42. Rule Connection  2
43.                  1
44.                  2
```

The list of command-line functions associated with FIS construction includes `getfis`, `setfis`, `showfis`, `addvar`, `addmf`, `addrule`, `rmvar`, and `rmmf`.

Saving FIS Files

A specialized text file format is used for saving fuzzy inference systems. The functions `readfis` and `writefis` are used for reading and writing these files.

If you prefer, you can modify the FIS by editing its `.fis` text file rather than using any of the GUIs. You should be aware, however, that changing one entry may oblige you to

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

change another. For example, if you delete a membership function using this method, you also need to make certain that any rules requiring this membership function are also deleted.

The rules appear in indexed format in a `.fis` text file. The following sample shows the file `tipper.fis`.

```
[System]
Name='tipper'
Type='mamdani'
NumInputs=2
NumOutputs=1
NumRules=3
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'

[Input1]
Name='service'
Range=[0 10]
NumMFs=3
MF1='poor':'gaussmf',[1.5 0]
MF2='good':'gaussmf',[1.5 5]
MF3='excellent':'gaussmf',[1.5 10]

[Input2]
Name='food'
Range=[0 10]
NumMFs=2
MF1='rancid':'trapmf',[0 0 1 3]
MF2='delicious':'trapmf',[7 9 10 10]

[Output1]
Name='tip'
Range=[0 30]
NumMFs=3
MF1='cheap':'trimf',[0 5 10]
MF2='average':'trimf',[10 15 20]
MF3='generous':'trimf',[20 25 30]

[Rules]
1 1, 1 (1) : 2
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

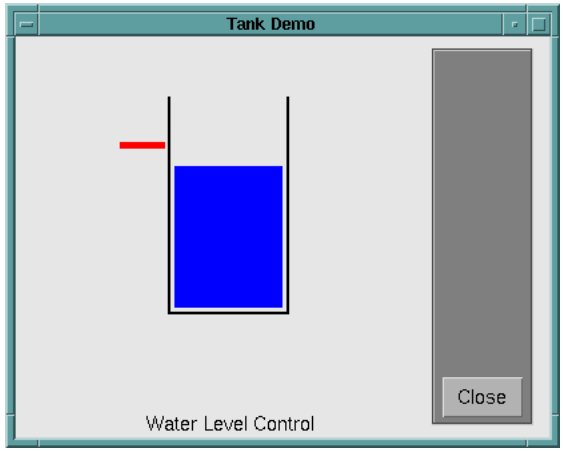
```
2 0, 2 (1) : 1
3 2, 3 (1) : 2
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Simulate Fuzzy Inference Systems in Simulink

Fuzzy Logic Toolbox software is designed to work in Simulink environment. After you have created your fuzzy system using the GUI tools or some other method, you are ready to embed your system directly into a simulation.

Picture a tank with a pipe flowing in and a pipe flowing out. You can change the valve controlling the water that flows in, but the outflow rate depends on the diameter of the outflow pipe (which is constant) and the pressure in the tank (which varies with the water level). The system has some very nonlinear characteristics.



A controller for the water level in the tank needs to know the current water level and it needs to be able to set the valve. Your controller's input is the water level error (desired water level minus actual water level), and its output is the rate at which the valve is opening or closing. A first pass at writing a fuzzy controller for this system might be the following:

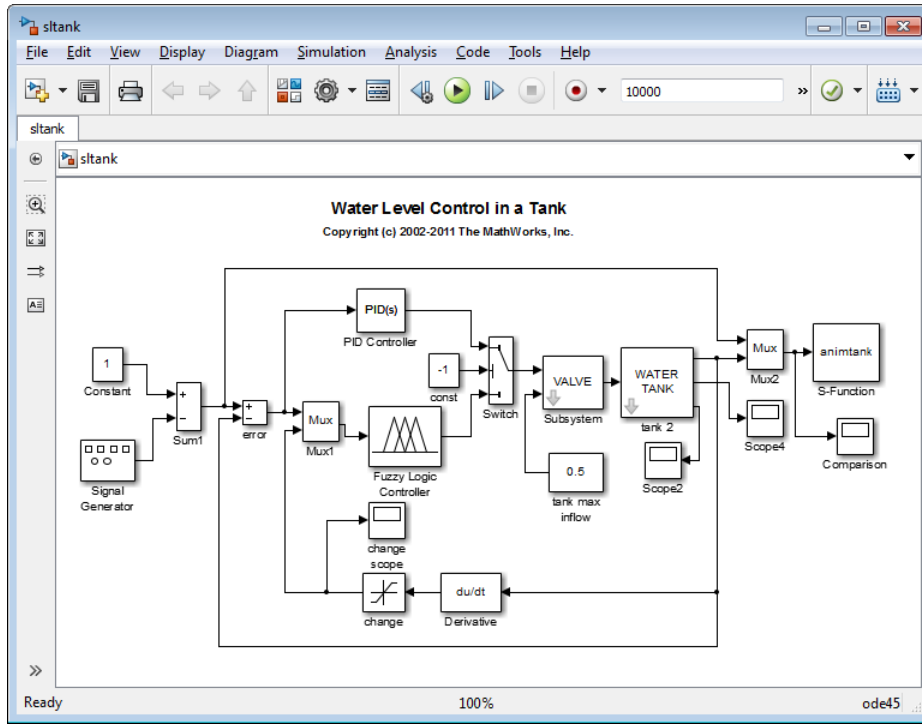
1. If (level is okay) then (valve is no_change) (1)
2. If (level is low) then (valve is open_fast) (1)
3. If (level is high) then (valve is close_fast) (1)

You can take fuzzy systems directly into Simulink and test them out in a simulation environment. A Simulink block diagram for this system is shown in the following figure. It contains a Simulink block called the Fuzzy Logic Controller block. The Simulink block diagram for this system is `sltank`. Typing

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

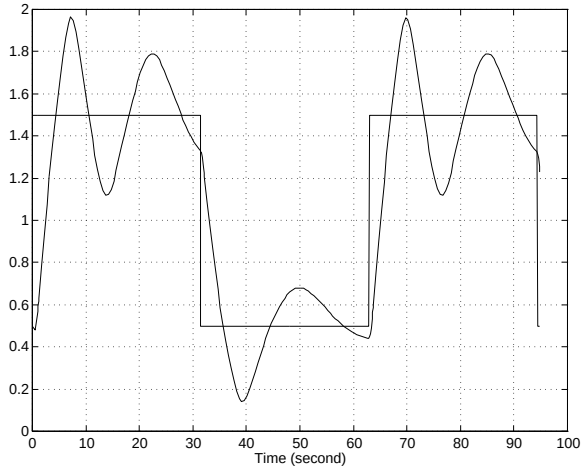
```
sltank
```

at the command line, causes the system to appear. At the same time, the file `tank.fis` is loaded into the FIS structure `tank`.



Some experimentation shows that three rules are not sufficient, because the water level tends to oscillate around the desired level. See the following plot:

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



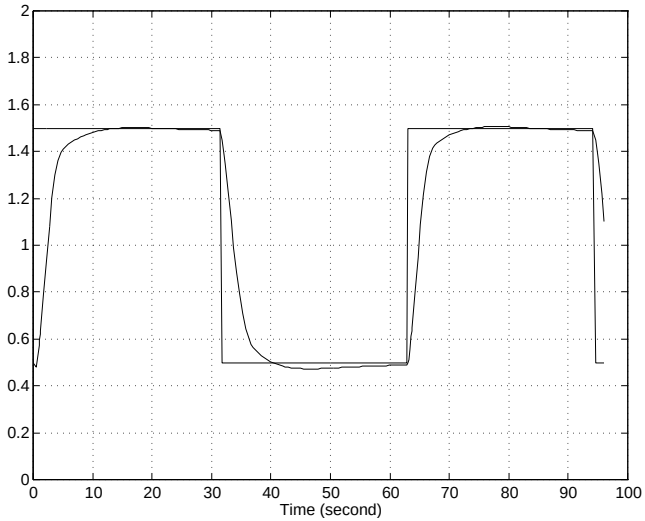
You need to add another input, the water level's rate of change, to slow down the valve movement when it gets close to the right level.

4. If (level is good) and (rate is negative), then (valve is close_slow) (1)

5. If (level is good) and (rate is positive), then (valve is open_slow) (1)

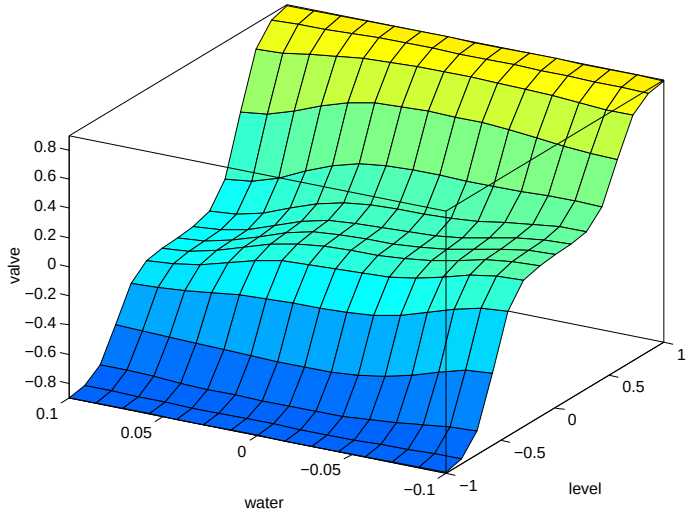
`sltank` is built with these five rules. With all five rules in operations, you can examine the step response by simulating this system. You do so by clicking **Simulation** > **Run**, and clicking the Comparison block. The result looks similar to the following plot.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



One interesting feature of the water tank system is that the tank empties much more slowly than it fills up because of the specific value of the outflow diameter pipe. You can deal with this by setting the `close_slow` valve membership function to be slightly different from the `open_slow` setting. A PID controller does not have this capability. The valve command versus the water level change rate (depicted as *water*) and the relative water level change (depicted as *level*) surface looks like this. If you look closely, you can see a slight asymmetry to the plot.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Because MATLAB software supports so many tools such as Control System Toolbox, and Neural Network Toolbox software, you can, for example, easily make a comparison of a fuzzy controller versus a linear controller or a neural network controller.

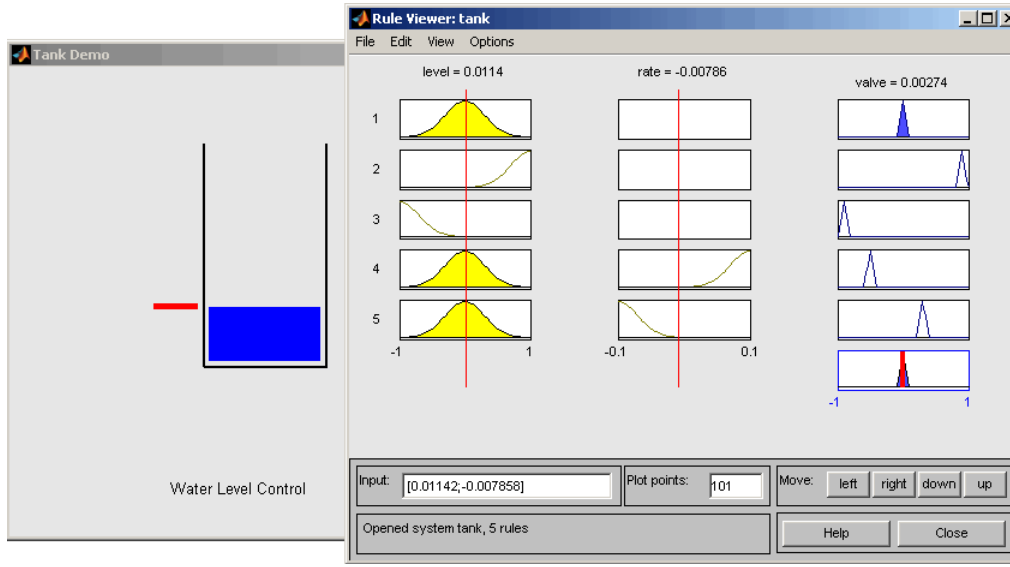
For a example of how the Rule Viewer can be used to interact with a Fuzzy Logic Toolbox block in a Simulink model, type

```
sltankrule
```

This example contains a block called the Fuzzy Controller With Rule Viewer block.

In this example, the Rule Viewer opens when you start the simulation. This Rule Viewer provides an animation of how the rules are fired during the water tank simulation. The windows that open when you simulate `sltankrule` are depicted as follows.

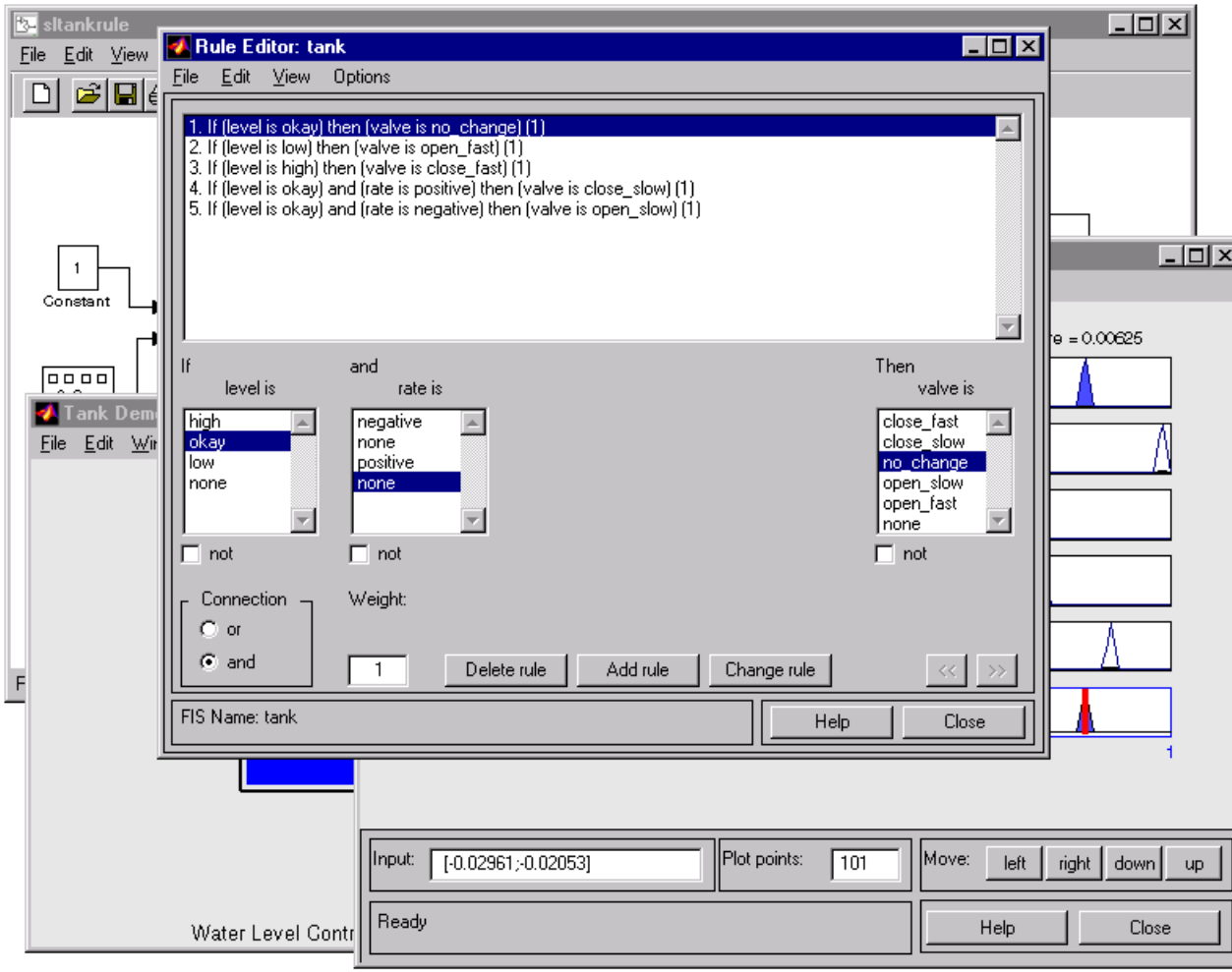
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



The Rule Viewer that opens during the simulation can be used to access the Membership Function Editor, the Rule Editor, or any of the other GUIs, (see "The Membership Function Editor" on page 2-39, or "The Rule Editor" on page 2-46, for more information).

For example, you may want to open the Rule Editor to change one of your rules. To do so, select **Rules** under the **Edit** menu of the open Rule Viewer. Now, you can view or edit the rules for this Simulink model.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



If you stop the simulation prior to selecting any of these editors, you should change your FIS. Remember to save any changes you make to your FIS to the workspace before you restart the simulation.

See Also

Fuzzy Logic Controller | Fuzzy Logic Controller with Ruleviewer

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

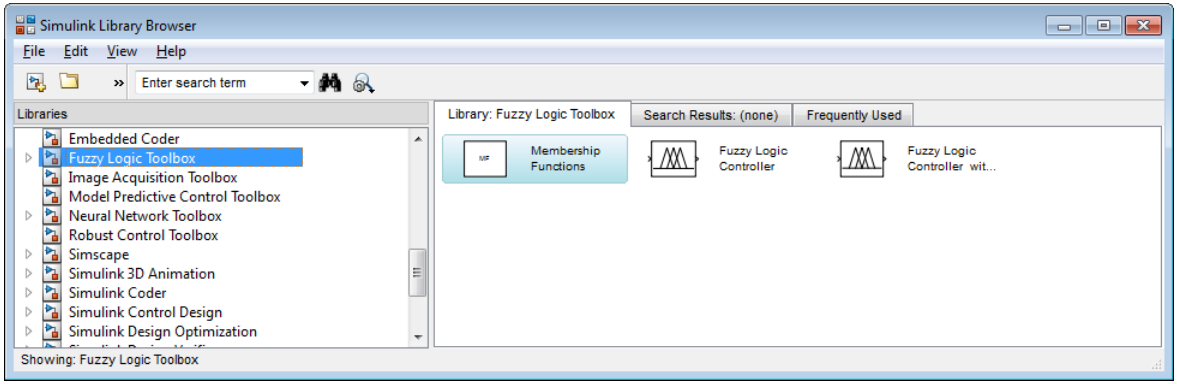
Related Examples

- "Build Your Own Fuzzy Simulink Models"

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Build Your Own Fuzzy Simulink Models

To build your own Simulink systems that use fuzzy logic, use the blocks in the Fuzzy Logic Toolbox library. To open the library, open the Simulink Library Browser. In the **Libraries** pane, expand the **Fuzzy Logic Toolbox** node.



The Fuzzy Logic Toolbox library contains the Fuzzy Logic Controller and Fuzzy Logic Controller with Rule Viewer blocks. It also includes a Membership Functions sublibrary that contains Simulink blocks for the built-in membership functions.

To add a block from the library, drag the block into the Simulink model window. You can get help on a specific block by clicking **Help**.

About the Fuzzy Logic Controller Block

For most fuzzy inference systems, the Fuzzy Logic Controller block automatically generates a hierarchical block diagram representation of your FIS. This automatic model generation ability is called the *Fuzzy Wizard*. The block diagram representation only uses built-in Simulink blocks and, therefore, allows for efficient code generation.

The Fuzzy Wizard cannot handle FIS with custom membership functions or with AND, OR, IMP, and AGG functions outside of the following list:

- `orMethod: max`
- `andMethod: min,prod`
- `impMethod: min,prod`

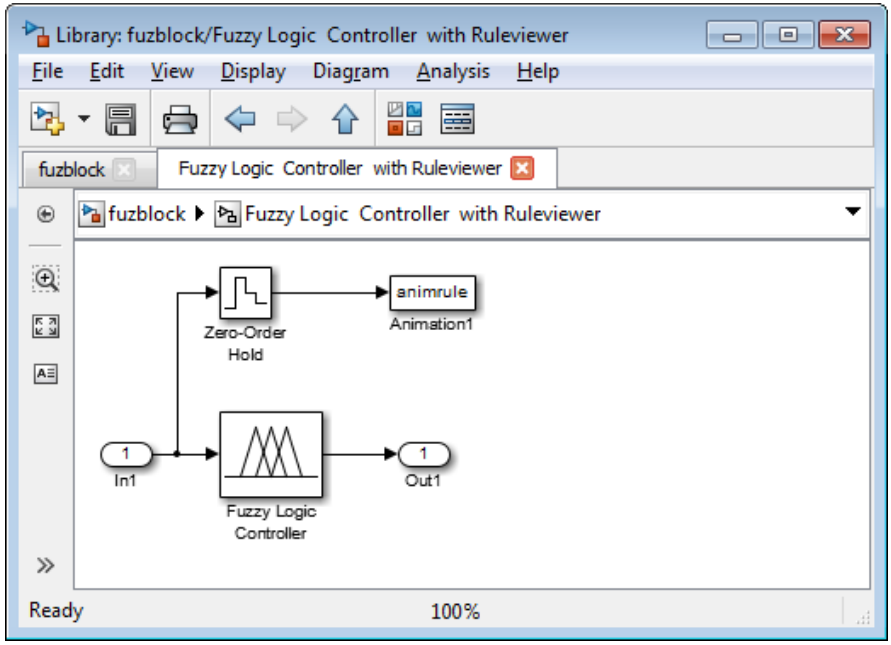
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- `aggMethod: max`

In these cases, the Fuzzy Logic Controller block uses the S-function `sffis` to simulate the FIS.

About the Fuzzy Logic Controller with Ruleviewer Block

The Fuzzy Logic Controller with Rule Viewer block is an extension of the Fuzzy Logic Controller block. It allows you to visualize how rules are fired during simulation. Right-click the Fuzzy Controller With Rule Viewer block, and select **Mask** > **Look Under Mask** to view the underlying subsystem.



Initializing Fuzzy Logic Controller Blocks

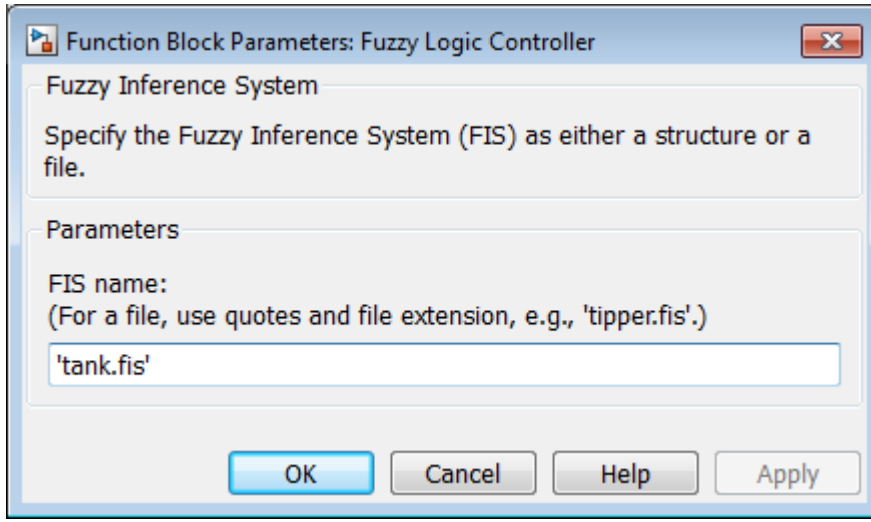
You can initialize a Fuzzy Logic Controller or Fuzzy Logic Controller with Ruleviewer block using a fuzzy inference system saved as a `.fis` file or a structure. To learn how to save your fuzzy inference system, see "Importing and Exporting from the GUI Tools" on page 2-53.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

To initialize a Fuzzy Logic Controller block, use the following steps:

1 Double-click the block to open the **Function Block Parameters: Fuzzy Logic Controller** dialog box.

2 In **FIS name**, enter the name of the structure variable or the name of the `.fis` file.



Note: When entering the name of the `.fis` file in the blocks, you must enclose it in single quotes.

You can similarly initialize the Fuzzy Logic Controller with Ruleviewer block. In this case, enter the name of the structure variable in **FIS matrix**.

Example: Cart and Pole Simulation

The cart and pole simulation is an example of a FIS model auto-generated by the Fuzzy Logic Controller block.

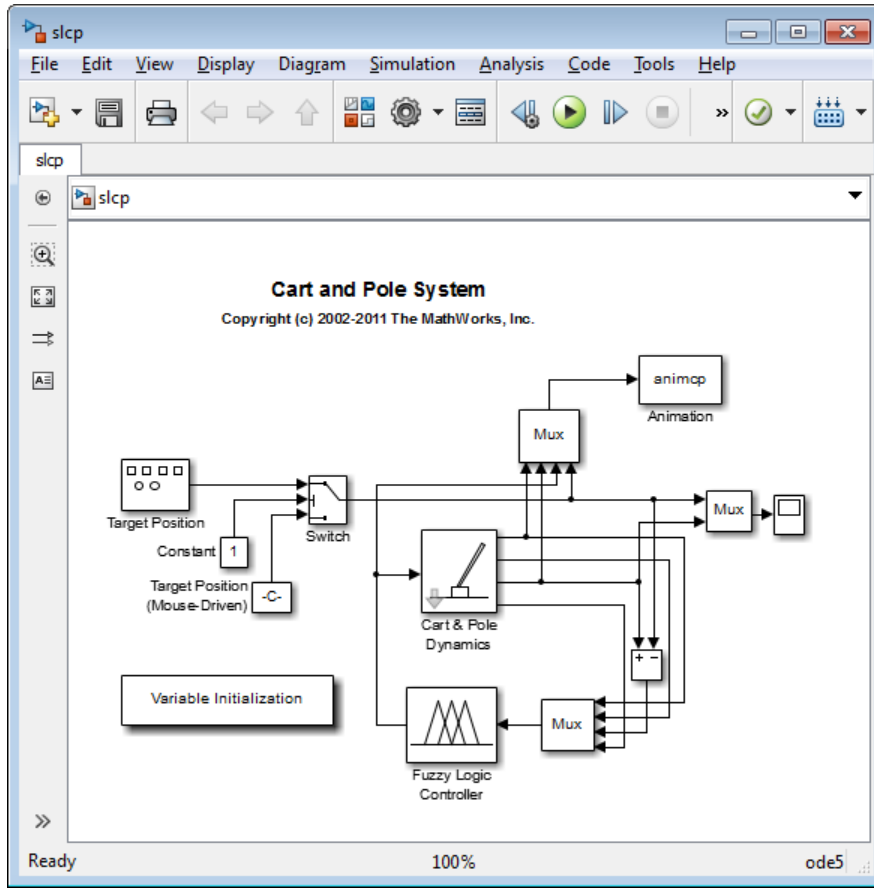
Type

```
slcp
```

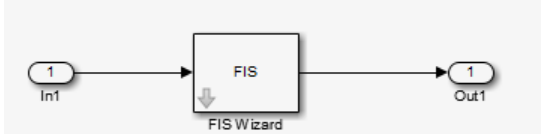
at the MATLAB prompt to open the simulation.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

This model appears.



Right-click on the Fuzzy Logic Controller block, and select **Mask** > **Look Under Mask** from the right-click menu. The following subsystem opens.



Follow the same procedure to look under the mask of the FIS Wizard subsystem to see the implementation of your FIS. The Fuzzy Logic Controller block uses built-in

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Simulink blocks to implement your FIS. Although the models can grow complex, this representation is better suited than the S-function `sffis` for efficient code generation.

See Also

Fuzzy Logic Controller | Fuzzy Logic Controller with Ruleviewer

Related Examples

- "Simulate Fuzzy Inference Systems in Simulink"

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

What Is Sugeno-Type Fuzzy Inference?

This topic discusses the so-called Sugeno, or Takagi-Sugeno-Kang, method of fuzzy inference. Introduced in 1985 [9], it is similar to the Mamdani method in many respects. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator, are exactly the same. The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant.

A typical rule in a Sugeno fuzzy model has the form

If Input 1 = x and Input 2 = y, then Output is $z = ax + by + c$

For a zero-order Sugeno model, the output level z is a constant *(a=b =0)*.

The output level z_i of each rule is weighted by the firing strength w_i of the rule. For example, for an AND rule with Input 1 = x and Input 2 = y, the firing strength is

$$w_i = AndMethod(F_1(x), F_2(y))$$

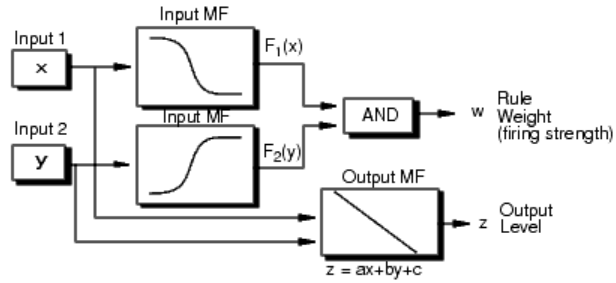
where $F_{1,2}$ *(.)* are the membership functions for Inputs 1 and 2.

The final output of the system is the weighted average of all rule outputs, computed as

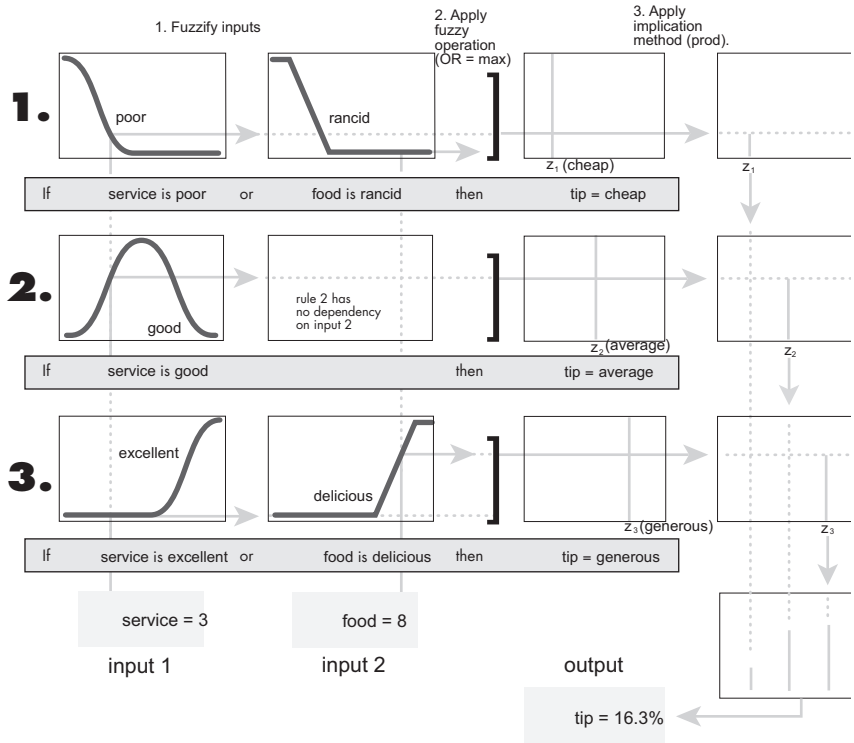
$$\text{Final Output} = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i}$$

where N is the number of rules.

A Sugeno rule operates as shown in the following diagram.



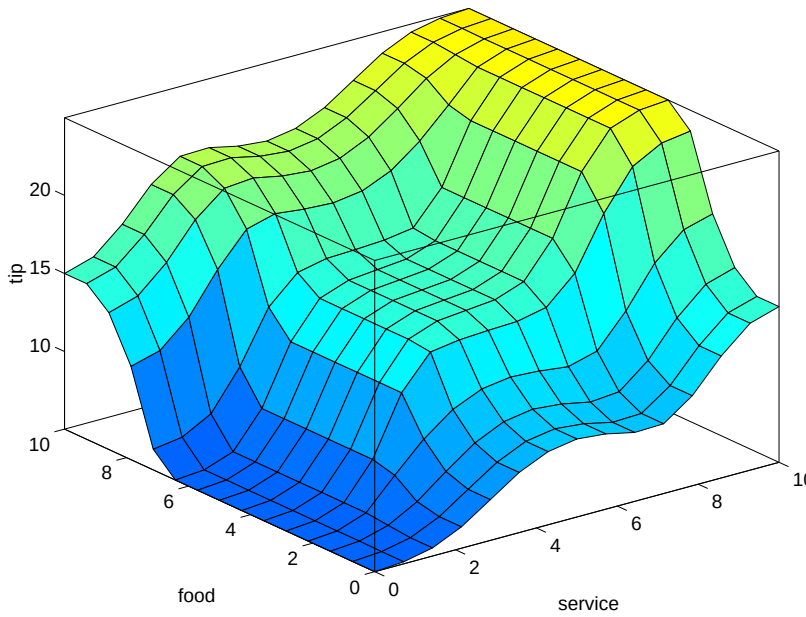
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



The preceding figure shows the fuzzy tipping model developed in previous sections of this manual adapted for use as a Sugeno system. Fortunately, it is frequently the case that singleton output functions are completely sufficient for the needs of a given problem. As an example, the system `tippersg.fis` is the Sugeno-type representation of the now-familiar tipping model. If you load the system and plot its output surface, you will see that it is almost the same as the Mamdani system you have previously seen.

```
a = readfis('tippersg');
gensurf(a)
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



The easiest way to visualize first-order Sugeno systems is to think of each rule as defining the location of a moving singleton. That is, the singleton output spikes can move around in a linear fashion in the output space, depending on what the input is. This also tends to make the system notation very compact and efficient. Higher-order Sugeno fuzzy models are possible, but they introduce significant complexity with little obvious merit. Sugeno fuzzy models whose output membership functions are greater than first order are not supported by Fuzzy Logic Toolbox software.

Because of the linear dependence of each rule on the input variables, the Sugeno method is ideal for acting as an interpolating supervisor of multiple linear controllers that are to be applied, respectively, to different operating conditions of a dynamic nonlinear system. For example, the performance of an aircraft may change dramatically with altitude and Mach number. Linear controllers, though easy to compute and well suited to any given flight condition, must be updated regularly and smoothly to keep up with the changing state of the flight vehicle. A Sugeno fuzzy inference system is extremely well suited to the task of smoothly interpolating the linear gains that would be applied across the input space; it is a natural and efficient gain scheduler. Similarly, a Sugeno system is suited for modeling nonlinear systems by interpolating between multiple linear models.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

To see a specific example of a system with linear output membership functions, consider the one input one output system stored in `sugeno1.fis`.

```
fismat = readfis('sugeno1');
getfis(fismat,'output',1)
```

This syntax returns:

```
Name = output
   NumMFs = 2
   MFLabels =
    line1
    line2
   Range = [0 1]
```

The output variable has two membership functions.

```
getfis(fismat,'output',1,'mf',1)
```

This syntax returns:

```
Name = line1
   Type = linear
   Params =
    -1    -1
```

```
getfis(fismat,'output',1,'mf',2)
```

This syntax returns:

```
 Name = line2
   Type = linear
   Params =
    1    -1
```

Further, these membership functions are linear functions of the input variable. The membership function `line1` is defined by the equation

$$output = (-1) \times input + (-1)$$

and the membership function `line2` is defined by the equation

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

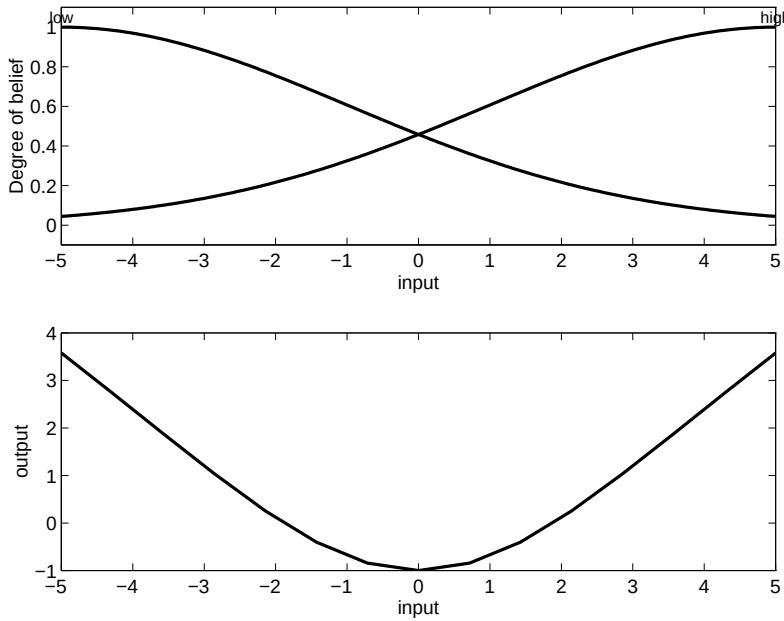
$$output = (1) \times input + (-1)$$

The input membership functions and rules define which of these output functions are expressed and when:

```
showrule(fismat)
ans =
1. If (input is low) then (output is line1) (1)
2. If (input is high) then (output is line2) (1)
```

The function `plotmf` shows us that the membership function `low` generally refers to input values less than zero, while `high` refers to values greater than zero. The function `gensurf` shows how the overall fuzzy system output switches smoothly from the line called `line1` to the line called `line2`.

```
subplot(2,1,1), plotmf(fismat,'input',1)
subplot(2,1,2),gensurf(fismat)
```



As this example shows, Sugeno-type system gives you the freedom to incorporate linear systems into your fuzzy systems. By extension, you could build a fuzzy system that

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

switches between several optimal linear controllers as a highly nonlinear system moves around in its operating space.

See Also

`gensurf` | `readfis`

More About

- "Comparison of Sugeno and Mamdani Systems" on page 2-97
- "What Is Mamdani-Type Fuzzy Inference?" on page 2-30

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Comparison of Sugeno and Mamdani Systems

Because it is a more compact and computationally efficient representation than a Mamdani system, the Sugeno system lends itself to the use of adaptive techniques for constructing fuzzy models. These adaptive techniques can be used to customize the membership functions so that the fuzzy system best models the data.

Note You can use the MATLAB command-line function `mam2sug` to convert a Mamdani system into a Sugeno system (not necessarily with a single output) with constant output membership functions. It uses the centroid associated with all of the output membership functions of the Mamdani system.

The following are some final considerations about the two different methods.

Advantages of the Sugeno Method

- It is computationally efficient.
- It works well with linear techniques (e.g., PID control).
- It works well with optimization and adaptive techniques.
- It has guaranteed continuity of the output surface.
- It is well suited to mathematical analysis.

Advantages of the Mamdani Method

- It is intuitive.
- It has widespread acceptance.
- It is well suited to human input.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

About the Fuzzy Inference Engine

Fuzzy Logic Toolbox software provides a stand-alone C code fuzzy inference engine. You can use the engine as an alternative tool to simulate the outputs of your fuzzy inference system (FIS), without using the MATLAB environment. You can perform the following tasks using the fuzzy inference engine:

- Perform fuzzy inference using a FIS structure file and an input data file.

  To learn more about how to create a FIS structure file, see "Build Mamdani Systems (GUI)" on page 2-31.
- Customize the fuzzy inference engine to include your own membership functions.
- Embed the executable code in other external applications.

The stand-alone fuzzy inference engine consists of two C code source files `fismain.c` and `fis.c` in the *matlabroot*`\toolbox\fuzzy\fuzzy` folder.

The `fismain.c` file contains only the `main()` function and you can easily modify it to adapt to other applications. It is ANSI Ccompatible, and you can compile it with any ANSI Ccompiler.

The `fis.c` file contains all the necessary functions to perform the fuzzy inference process:

- This file provides the 11 Fuzzy Logic Toolbox membership functions with their default settings.
- You can add a new membership function or new reasoning mechanism by updating the `fis.c` file.

The `fismain` executable code, generated after compiling the source codes, reads an input data file and a FIS structure file to simulate the output. The syntax for calling `fismain` is similar to its MEX-file counterpart `evalfis`, except that all matrices are replaced with files. To learn more about `evalfis`, see the `evalfis` function reference page.

Related Examples

- "Build Mamdani Systems (GUI)" on page 2-31
- "Simulate Systems on Windows Platforms" on page 2-99
- "Simulate Systems on UNIX Platforms" on page 2-103

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Simulate Systems on Windows Platforms

This example shows how to simulate a fuzzy inference system on a Windows® platform using the stand-alone fuzzy inference engine. In this example, you use the LCC C compiler shipped with MATLAB to compile the fuzzy inference engine source codes.

1. Open a DOS Command Window and change the folder to *matlabroot*`\toolbox\fuzzy\fuzzy`.

   Tip You can find the root folder of your MATLAB installation by typing `matlabroot` at the MATLAB command prompt.

2. In the DOS Command Window, type the following command:

   ```
   lcc -c fismain.c
   ```

   This command creates the `fismain.obj` object file in the *matlabroot*`\toolbox\fuzzy\fuzzy` folder.

   Note: You may encounter the following error when executing the `lcc` command:

   ```
   'lcc' is not recognized as an internal or external command,
   operable program or batch file.
   ```

   To learn how to troubleshoot this error, see the "Including the `lcc` Command in the PATH Environment Variable" on page 2-101 section.

3. In the DOS Command Window, type the following command:

   ```
   lcclnk fismain.obj.
   ```

   This command creates an executable file `fismain.exe` in the *matlabroot*`\toolbox\fuzzy\fuzzy` folder.

   Note: You may encounter the following error when executing the `lcclnk` command:

   ```
   'lcclnk' is not recognized as an internal or external command,
   operable program or batch file.
   ```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

To learn how to troubleshoot this error, see the "Including the `lcclnk` Command in the PATH Environment Variable" on page 2-101 section.

4 Open MATLAB desktop, and change to your current working folder using the **Current Folder** field in the MATLAB desktop toolbar.

5 At the MATLAB prompt, type the following command to create an input data file:

```
[x, y] = meshgrid(-5:5, -5:5);
input_data = [x(:) y(:)];
save fis_in input_data -ascii
```

This command saves the input data as a 121-by-2 matrix in the `fis_in` ASCII file in your current working folder. Each row of the matrix represents an input vector.

6 Copy the FIS structure file `mam21.fis` from the *matlabroot*`\toolbox\fuzzy\fuzdemos` folder to your current working folder.

7 Copy the stand-alone executable file `fismain.exe` from the *matlabroot*`\toolbox\fuzzy\fuzzy` folder to your current working folder.

8 In the DOS Command Window, change the folder to your current working folder, and type the following command to call the stand-alone executable code:

```
fismain fis_in mam21.fis
```

This command uses the data file `fis_in` and the FIS structure file `mam21.fis`, and generates 121 outputs on your screen.

Tip You can also direct the outputs to a file using the following command in the DOS Command Window:

```
fismain fis_in mam21.fis > fis_out
```

This command saves the output data `fis_out` as a 121-by-1 matrix in your current working folder. Each row of the output matrix represents an output vector.

9 To verify that the fuzzy inference engine output matches the MATLAB MEX-file `evalfis.m` output, type the following command at the MATLAB prompt:

```
fismat = readfis('mam21');
matlab_out = evalfis(input_data, fismat);
load fis_out
max(max(matlab_out - fis_out))
```

This command returns the following result:

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
ans =

  4.9583e-013
```

The difference results from the relative precision between the outputs.

Including the `lcc` Command in the PATH Environment Variable

When executing the `lcc` command to create the `fismain.obj` object file, you get the following error if the command and/or the path for the `<include>` header files are not in the `PATH` environment variable:

```
'lcc' is not recognized as an internal or external command, operable
program or batch file.
```

To include the command in the `PATH` environment variable, type the following in the DOS Command Window:

matlabroot`\sys\lcc\bin\lcc -I`*matlabroot*`\sys\lcc\include fismain.c`

Press **Enter** when prompted to `Please enter the path for the <include> header files`.

Including the `lcclnk` Command in the PATH Environment Variable

When executing the `lcclnk` command to create the `fismain.exe` file, you get the following error if the command is not in the `PATH` environment variable:

```
'lcclnk' is not recognized as an internal or external command,
operable program or batch file.
```

To include the command in the `PATH` environment variable, type the following in the DOS Command Window:

matlabroot`\sys\lcc\bin\lcclnk fismain.obj`

See Also

"Simulate Systems on UNIX Platforms" on page 2-103

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

More About

- "About the Fuzzy Inference Engine" on page 2-98

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Simulate Systems on UNIX Platforms

This example shows how to simulate a fuzzy inference system on a UNIX® platform using the stand-alone fuzzy inference engine.

1. Open a UNIX Command Window and change the folder to *matlabroot*`\toolbox\fuzzy\fuzzy`.

   Tip You can find the root folder of your MATLAB installation by typing `matlabroot` at the MATLAB command prompt.

2. In the UNIX Command Window, type the following command:

   ```
   cc -O -o fismain fismain.c -lm
   ```

   This command creates the `fismain` file in the *matlabroot*`\toolbox\fuzzy\fuzzy` folder.

   The `fis.c` file is included in the `fismain.c` file; you do not have to compile it separately.

3. At the MATLAB prompt, create an input data file using the following command:

   ```
   [x, y] = meshgrid(-5:5, -5:5);
   input_data = [x(:) y(:)];
   save fis_in input_data -ascii
   ```

   This command saves the input data as a 121-by-2 matrix in the ASCII file `fis_in` in your current working folder. Each row of the matrix represents an input vector.

   Tip You can find your current working folder in the **Current Folder** field in the MATLAB desktop toolbar.

4. Copy the FIS structure file `mam21.fis` from the *matlabroot*`\toolbox\fuzzy\fuzdemos` folder to your current working folder.
5. Copy the `fismain` file from the *matlabroot*`\toolbox\fuzzy\fuzzy` folder to your current working folder.
6. In the UNIX Command Window, change the folder to your current working folder, and type the following executable command:

   ```
   fismain fis_in mam21.fis
   ```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

This command uses the data file `fis_in` and the FIS structure file `mam21.fis` and generates 121 outputs on your screen.

Tip You can also direct the outputs to another file using the following command in the DOS Command Window:

```
fismain fis_in mam21.fis > fis_out
```

This command saves the output data `fis_out` as a 121-by-1 matrix in your current working folder. Each row of the output matrix represents an output vector.

---

7 To verify that the fuzzy inference engine output matches the MATLAB MEX-file `evalfis.m` output, type the following command at the MATLAB prompt:

```
fismat = readfis('mam21');
matlab_out = evalfis(input_data, fismat);
load fis_out
max(max(matlab_out - fis_out))
```

This command returns the following result:

```
ans =

  4.9583e-013
```

The difference results from the relative precision between the outputs.

More About

- "About the Fuzzy Inference Engine" on page 2-98

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

3

Adaptive Neuro-Fuzzy Modeling

- "Neuro-Adaptive Learning and ANFIS" on page 3-2
- "Comparison of anfis and Neuro-Fuzzy Designer Functionality" on page 3-7
- "Train Adaptive Neuro-Fuzzy Inference Systems (GUI)" on page 3-13
- "Test Data Against Trained System (GUI)" on page 3-18
- "Save Training Error Data to MATLAB Workspace" on page 3-31
- "Predict Chaotic Time-Series" on page 3-39
- "Modeling Inverse Kinematics in a Robotic Arm" on page 3-46

www.MatlabSite.com | متلب سایت

مرجع آموزش برنامه‌نویسی متلب در ایران

Neuro-Adaptive Learning and ANFIS

In this section...

"When to Use Neuro-Adaptive Learning" on page 3-2
"Model Learning and Inference Through ANFIS" on page 3-3
"References" on page 3-5

When to Use Neuro-Adaptive Learning

The basic structure of Mamdani fuzzy inference system is a model that maps input characteristics to input membership functions, input membership functions to rules, rules to a set of output characteristics, output characteristics to output membership functions, and the output membership functions to a single-valued output or a decision associated with the output. Such a system uses fixed membership functions that are chosen arbitrarily and a rule structure that is essentially predetermined by the user's interpretation of the characteristics of the variables in the model.

`anfis` and the Neuro-Fuzzy Designer apply fuzzy inference techniques to data modeling. As you have seen from the other fuzzy inference GUIs, the shape of the membership functions depends on parameters, and changing these parameters change the shape of the membership function. Instead of just looking at the data to choose the membership function parameters, you choose membership function parameters automatically using these Fuzzy Logic Toolbox applications.

Suppose you want to apply fuzzy inference to a system for which you already have a collection of input/output data that you would like to use for modeling, model-following, or some similar scenario. You do not necessarily have a predetermined model structure based on characteristics of variables in your system.

In some modeling situations, you cannot discern what the membership functions should look like simply from looking at data. Rather than choosing the parameters associated with a given membership function arbitrarily, these parameters could be chosen so as to tailor the membership functions to the input/output data in order to account for these types of variations in the data values. In such cases, you can use the Fuzzy Logic Toolbox *neuro-adaptive* learning techniques incorporated in the `anfis` command.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Model Learning and Inference Through ANFIS

The neuro-adaptive learning method works similarly to that of neural networks. Neuro-adaptive learning techniques provide a method for the fuzzy modeling procedure to *learn* information about a data set. Fuzzy Logic Toolbox software computes the membership function parameters that best allow the associated fuzzy inference system to track the given input/output data. The Fuzzy Logic Toolbox function that accomplishes this membership function parameter adjustment is called `anfis`. The `anfis` function can be accessed either from the command line or through the Neuro-Fuzzy Designer. Because the functionality of the command line function `anfis` and the Neuro-Fuzzy Designer is similar, they are used somewhat interchangeably in this discussion, except when specifically describing the GUI.

What Is ANFIS?

The acronym ANFIS derives its name from *adaptive neuro-fuzzy inference system*. Using a given input/output data set, the toolbox function `anfis` constructs a fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using either a back propagation algorithm alone or in combination with a least squares type of method. This adjustment allows your fuzzy systems to learn from the data they are modeling.

FIS Structure and Parameter Adjustment

A network-type structure similar to that of a neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map.

The parameters associated with the membership functions changes through the learning process. The computation of these parameters (or their adjustment) is facilitated by a gradient vector. This gradient vector provides a measure of how well the fuzzy inference system is modeling the input/output data for a given set of parameters. When the gradient vector is obtained, any of several optimization routines can be applied in order to adjust the parameters to reduce some error measure. This error measure is usually defined by the sum of the squared difference between actual and desired outputs. `anfis` uses either back propagation or a combination of least squares estimation and back propagation for membership function parameter estimation.

Know Your Data

The modeling approach used by `anfis` is similar to many system identification techniques. First, you hypothesize a parameterized model structure (relating inputs to

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

membership functions to rules to outputs to membership functions, and so on). Next, you collect input/output data in a form that will be usable by `anfis` for training. You can then use `anfis` to *train* the FIS model to emulate the training data presented to it by modifying the membership function parameters according to a chosen error criterion.

In general, this type of modeling works well if the training data presented to `anfis` for training (estimating) membership function parameters is fully representative of the features of the data that the trained FIS is intended to model. In some cases however, data is collected using noisy measurements, and the training data cannot be representative of all the features of the data that will be presented to the model. In such situations, model validation is helpful.

Model Validation Using Testing and Checking Data Sets

Model validation is the process by which the input vectors from input/output data sets on which the FIS was not trained, are presented to the trained FIS model, to see how well the FIS model predicts the corresponding data set output values.

One problem with model validation for models constructed using adaptive techniques is selecting a data set that is both representative of the data the trained model is intended to emulate, yet sufficiently distinct from the training data set so as not to render the validation process trivial.

If you have collected a large amount of data, hopefully this data contains all the necessary representative features, so the process of selecting a data set for checking or testing purposes is made easier. However, if you expect to be presenting noisy measurements to your model, it is possible the training data set does not include all of the representative features you want to model.

The testing data set lets you check the generalization capability of the resulting fuzzy inference system. The idea behind using a checking data set for model validation is that after a certain point in the training, the model begins overfitting the training data set. In principle, the model error for the checking data set tends to decrease as the training takes place up to the point that overfitting begins, and then the model error for the checking data suddenly increases. Overfitting is accounted for by testing the FIS trained on the training data against the checking data, and choosing the membership function parameters to be those associated with the minimum checking error if these errors indicate model overfitting.

Usually, these training and checking data sets are collected based on observations of the target system and are then stored in separate files.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

In the first example, two similar data sets are used for checking and training, but the checking data set is corrupted by a small amount of noise. This example illustrates of the use of the Neuro-Fuzzy Designer with checking data to reduce the effect of model overfitting. In the second example, a training data set that is presented to `anfis` is sufficiently different than the applied checking data set. By examining the checking error sequence over the training period, it is clear that the checking data set is not good for model validation purposes. This example illustrates the use of the Neuro-Fuzzy Designer to compare data sets.

References

[1] Jang, J.-S. R., "Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm," *Proc. of the Ninth National Conf. on Artificial Intelligence (AAAI-91)*, pp. 762-767, July 1991.

[2] Jang, J.-S. R., "ANFIS: Adaptive-Network-based Fuzzy Inference Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, pp. 665-685, May 1993.

[3] Jang, J.-S. R. and N. Gulley, "Gain scheduling based fuzzy controller design," *Proc. of the International Joint Conference of the North American Fuzzy Information Processing Society Biannual Conference, the Industrial Fuzzy Control and Intelligent Systems Conference, and the NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic*, San Antonio, Texas, Dec. 1994.

[4] Jang, J.-S. R. and C.-T. Sun, "Neuro-fuzzy modeling and control, *Proceedings of the IEEE*, March 1995.

[5] Jang, J.-S. R. and C.-T. Sun, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, 1997.

[6] Wang, L.-X., *Adaptive fuzzy systems and control: design and stability analysis*, Prentice Hall, 1994.

[7] Widrow, B. and D. Stearns, *Adaptive Signal Processing*, Prentice Hall, 1985.

Related Examples

- "Train Adaptive Neuro-Fuzzy Inference Systems (GUI)" on page 3-13
- "Test Data Against Trained System (GUI)" on page 3-18

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- "Save Training Error Data to MATLAB Workspace" on page 3-31
- "Predict Chaotic Time-Series" on page 3-39

More About

- "Comparison of anfis and Neuro-Fuzzy Designer Functionality" on page 3-7

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Comparison of anfis and Neuro-Fuzzy Designer Functionality

This topic discusses the arguments and range components of the command line function anfis and the analogous functionality of the Neuro-Fuzzy Designer.

The command anfis takes at least two and at most six input arguments. The general format is

```
[fismat1,trnError,ss,fismat2,chkError] = ...
anfis(trnData,fismat,trnOpt,dispOpt,chkData,method);
```

where trnOpt (training options), dispOpt (display options), chkData (checking data), andmethod (training method), are optional. All output arguments are also optional.

When you open the Neuro-Fuzzy Designer using neuroFuzzyDesigner, only the training data set must exist prior to implementing anfis. In addition, the step-size is fixed when the adaptive neuro-fuzzy system is trained using this GUI tool.

Training Data

The training data, trnData, is a required argument to anfis, as well as to the Neuro-Fuzzy Designer. Each row of trnData is a desired input/output pair of the target system you want to model Each row starts with an input vector and is followed by an output value. Therefore, the number of rows of trnData is equal to the number of training data pairs, and, because there is only one output, the number of columns of trnData is equal to the number of inputs plus one.

Input FIS Structure

You can obtain the input FIS structure, fismat, from any of the fuzzy editors:

- The Fuzzy Logic Designer
- The Membership Function Editor
- The Rule Editor from the Neuro-Fuzzy Designer (which allows a FIS structure to be loaded from a file or the MATLAB workspace)
- The command line function, genfis1 (for which you only need to give numbers and types of membership functions)

The FIS structure contains both the model structure, (which specifies such items as the number of rules in the FIS, the number of membership functions for each input, etc.), and the parameters, (which specify the shapes of membership functions).

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

There are two *methods* that `anfis` learning employs for updating membership function parameters:

- Backpropagation for all parameters (a steepest descent method)
- A hybrid method consisting of backpropagation for the parameters associated with the input membership functions, and least squares estimation for the parameters associated with the output membership functions

As a result, the training error decreases, at least locally, throughout the learning process. Therefore, the more the initial membership functions resemble the optimal ones, the easier it will be for the model parameter training to converge. Human expertise about the target system to be modeled may aid in setting up these initial membership function parameters in the FIS structure.

The `genfis1` function produces a FIS structure based on a fixed number of membership functions. This structure invokes the so-called *curse of dimensionality*, and causes excessive propagation of the number of rules when the number of inputs is moderately large, that is, more than four or five. Fuzzy Logic Toolbox software offers a method that provides for some dimension reduction in the fuzzy inference system: you can generate a FIS structure using the clustering algorithm discussed in "Subtractive Clustering" on page 4-3. To use the clustering algorithm, you must select the **Sub. Clustering** option in the **Generate FIS** portion of the Neuro-Fuzzy Designer before the FIS is generated. This subtractive clustering method partitions the data into groups called clusters, and generates a FIS with the minimum number rules required to distinguish the fuzzy qualities associated with each of the clusters.

Training Options

The Neuro-Fuzzy Designer tool allows you to choose your desired error tolerance and number of training epochs.

Training option `trnOpt` for the command line `anfis` is a vector that specifies the stopping criteria and the step-size adaptation strategy:

- `trnOpt(1)`: number of training epochs, default = 10
- `trnOpt(2)`: error tolerance, default = 0
- `trnOpt(3)`: initial step-size, default = 0.01
- `trnOpt(4)`: step-size decrease rate, default = 0.9
- `trnOpt(5)`: step-size increase rate, default = 1.1

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

If any element of trnOpt is an NaN or missing, then the default value is taken. The training process stops if the designated epoch number is reached or the error goal is achieved, whichever comes first.

Usually, the step-size profile is a curve that increases initially, reaches some maximum, and then decreases for the remainder of the training. You achieve this ideal step-size profile by adjusting the initial step-size and the increase and decrease rates (trnOpt(3) - trnOpt(5)). The default values are set up to cover a wide range of learning tasks. For any specific application, you may want to modify these step-size options in order to optimize the training. However, there are no user-specified step-size options for training the adaptive neuro-fuzzy inference system generated using the Neuro-Fuzzy Designer.

Display Options

Display options apply only to the command-line function anfis.

For the command line anfis, the display options *argument*, dispOpt, is a vector of either 1s or 0s that specifies what information to display, (print in the MATLAB command window), before, during, and after the training process. A 1 is used to denote *print this option*, whereas a 0 denotes *do not print this option*:

- dispOpt(1): display ANFIS information, default = 1
- dispOpt(2): display error (each epoch), default = 1
- dispOpt(3): display step-size (each epoch), default = 1
- dispOpt(4): display final results, default = 1

The default mode displays all available information. If any element of dispOpt is NaN or missing, the default value is used.

Method

Both the Neuro-Fuzzy Designer and the command line anfis apply either a backpropagation form of the steepest descent method for membership function parameter estimation, or a combination of backpropagation and the least-squares method to estimate membership function parameters. The choices for this argument are hybrid or backpropagation. These method choices are designated in the command line function, anfis, by 1 and 0, respectively.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Output FIS Structure for Training Data

`fismat1` is the output FIS structure corresponding to a minimal training error. This FIS structure is the one that you use to represent the fuzzy system when there is no checking data used for model cross-validation. This data also represents the FIS structure that is saved by the Neuro-Fuzzy Designer when the checking data option is not used.

When you use the checking data option, the output saved is that associated with the minimum checking error.

Training Error

The training error is the difference between the training data output value, and the output of the fuzzy inference system corresponding to the same training data input value, (the one associated with that training data output value). The training error `trnError` records the root mean squared error (RMSE) of the training data set at each epoch. `fismat1` is the snapshot of the FIS structure when the training error measure is at its minimum. The Neuro-Fuzzy Designer plots the training error versus epochs curve as the system is trained.

Step-Size

You cannot control the step-size options with the Neuro-Fuzzy Designer. Using the command line `anfis`, the step-size array `ss` records the step-size during the training. Plotting `ss` gives the step-size profile, which serves as a reference for adjusting the initial step-size and the corresponding decrease and increase rates. The step-size (`ss`) for the command-line function `anfis` is updated according to the following guidelines:

- If the error undergoes four consecutive reductions, increase the step-size by multiplying it by a constant (`ssinc`) greater than one.
- If the error undergoes two consecutive combinations of one increase and one reduction, decrease the step-size by multiplying it by a constant (`ssdec`) less than one.

The default value for the initial step-size is 0.01; the default values for `ssinc` and `ssdec` are 1.1 and 0.9, respectively. All the default values can be changed via the training option for the command line `anfis`.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Checking Data

The checking data, chkData, is used for testing the generalization capability of the fuzzy inference system at each epoch. The checking data has the same format as that of the training data, and its elements are generally distinct from those of the training data.

The checking data is important for learning tasks for which the input number is large, and/or the data itself is noisy. A fuzzy inference system needs to track a given input/output data set well. Because the model structure used for anfis is fixed, there is a tendency for the model to overfit the data on which is it trained, especially for a large number of training epochs. If overfitting does occur, the fuzzy inference system may not respond well to other independent data sets, especially if they are corrupted by noise. A validation or checking data set can be useful for these situations. This data set is used to cross-validate the fuzzy inference model. This cross-validation requires applying the checking data to the model and then seeing how well the model responds to this data.

When the checking data option is used with anfis, either via the command line, or using the Neuro-Fuzzy Designer, the checking data is applied to the model at each training epoch. When the command line anfis is invoked, the model parameters that correspond to the minimum checking error are returned via the output argument fismat2. The FIS membership function parameters computed using the Neuro-Fuzzy Designer when both training and checking data are loaded are associated with the training epoch that has a minimum checking error.

The use of the minimum checking data error epoch to set the membership function parameters assumes

- The checking data is similar enough to the training data that the checking data error decreases as the training begins.
- The checking data increases at some point in the training after the data overfitting occurs.

Depending on the behavior of the checking data error, the resulting FIS may or may not be the one you need to use. Refer to “Checking Data Does Not Validate Model” on page 3-26.

Output FIS Structure for Checking Data

The output of the command line anfis, fismat2, is the output FIS structure with the minimum checking error. This FIS structure is the one that you should use for further calculation if checking data is used for cross validation.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Checking Error

The checking error is the difference between the checking data output value, and the output of the fuzzy inference system corresponding to the same checking data input value, which is the one associated with that checking data output value. The checking error `chkError` records the RMSE for the checking data at each epoch. `fismat2` is the snapshot of the FIS structure when the checking error is at its minimum. The Neuro-Fuzzy Designer plots the checking error versus epochs curve as the system is trained.

Related Examples

- "Train Adaptive Neuro-Fuzzy Inference Systems (GUI)" on page 3-13
- "Test Data Against Trained System (GUI)" on page 3-18
- "Save Training Error Data to MATLAB Workspace" on page 3-31
- "Predict Chaotic Time-Series" on page 3-39

More About

- "Neuro-Adaptive Learning and ANFIS" on page 3-2

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Train Adaptive Neuro-Fuzzy Inference Systems (GUI)

This example shows how to create, train, and test Sugeno-type fuzzy systems using the Neuro-Fuzzy Designer.

To start the GUI, type the following command at the MATLAB prompt:

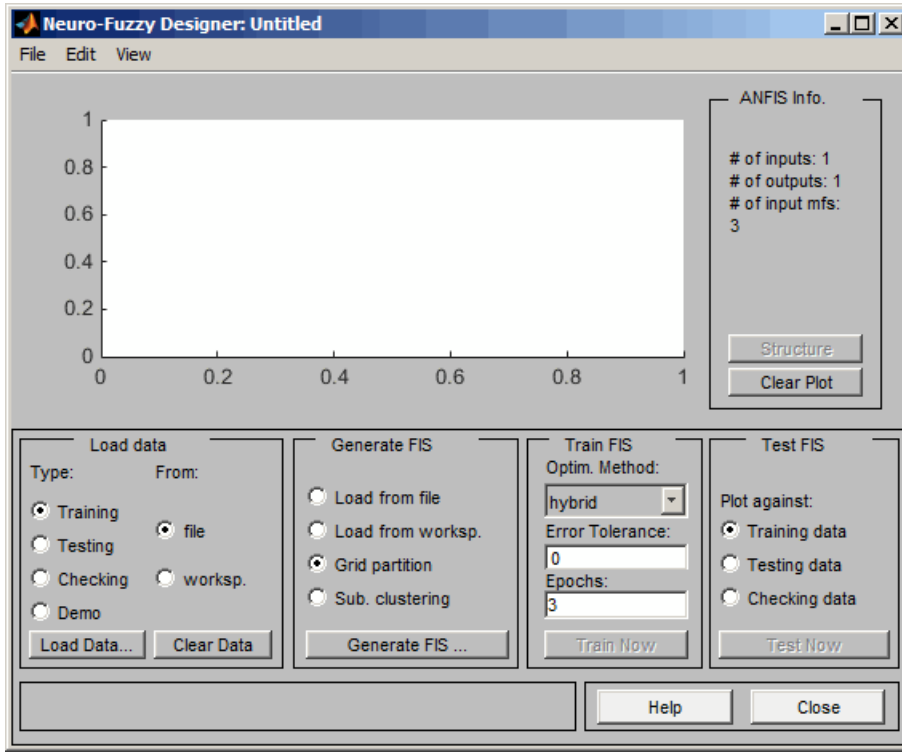
```
neuroFuzzyDesigner
```

The Neuro-Fuzzy Designer window shown in the following figure includes four distinct areas to support a typical workflow. The designer lets you perform the following tasks:

1. "Loading, Plotting, and Clearing the Data" on page 3-14
2. "Generating or Loading the Initial FIS Structure" on page 3-15
3. "Training the FIS" on page 3-15
4. "Validating the Trained FIS" on page 3-16

Access the online help topics by clicking **Help** in the Neuro-Fuzzy Designer.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Loading, Plotting, and Clearing the Data

To train a FIS, you must begin by loading a **Training** data set that contains the desired input/output data of the system to be modeled. Any data set you load must be an array with the data arranged as column vectors, and the output data in the last column.

You can also load **Testing** and **Checking** data in the designer. For more information on testing and checking data sets, see "Model Validation Using Testing and Checking Data Sets" on page 3-4.

To load a data set using the **Load data** portion of the designer:

1. Specify the data **Type**.
2. Select the data from a **file** or the MATLAB **worksp**.
3. Click **Load Data**.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

After you load the data, it displays in the plot. The training, testing and checking data are annotated in blue as *circles*, *diamonds*, and *pluses* respectively.

To clear a specific data set from the designer:

1. In the **Load data** area, select the data **Type**.
2. Click **Clear Data**.

This action also removes the corresponding data from the plot.

Generating or Loading the Initial FIS Structure

Before you start the FIS training, you must specify an initial FIS model structure. To specify the model structure, perform one of the following tasks:

- Load a previously saved Sugeno-type FIS structure from a file or the MATLAB workspace.
- Generate the initial FIS model by choosing one of the following partitioning techniques:
  - **Grid partition**— Generates a single-output Sugeno-type FIS by using grid partitioning on the data.
  - **Sub. clustering** — Generates an initial model for ANFIS training by first applying subtractive clustering on the data.

To view a graphical representation of the initial FIS model structure, click **Structure**.

Training the FIS

After loading the training data and generating the initial FIS structure, you can start training the FIS.

Tip If you want to save the training error generated during ANFIS training to the MATLAB workspace, see "Save Training Error Data to MATLAB Workspace" on page 3-31.

The following steps show you how to train the FIS.

1. In **Optim. Method**, choose **hybrid** or **backpropaga** as the optimization method.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The optimization methods train the membership function parameters to emulate the training data.

Note: The **hybrid** optimization method is a combination of least-squares and backpropagation gradient descent method.

---

2. Enter the number of training **Epochs** and the training **Error Tolerance** to set the stopping criteria for training.

   The training process stops whenever the maximum epoch number is reached or the training error goal is achieved.

3. Click **Train Now** to train the FIS.

   This action adjusts the membership function parameters and displays the error plots.

Examine the error plots to determine overfitting during the training. If you notice the checking error increasing over iterations, it indicates model overfitting. For examples on model overfitting, see "Checking Data Helps Model Validation" on page 3-18 and "Checking Data Does Not Validate Model" on page 3-26.

Validating the Trained FIS

After the FIS is trained, validate the model using a **Testing** or **Checking** data that differs from the one you used to train the FIS. To validate the trained FIS:

1. Select the validation data set and click **Load Data**.
2. Click **Test Now**.

   This action plots the test data against the FIS output (shown in red) in the plot.

For more information on the use of testing data and checking data for model validation, see "Model Validation Using Testing and Checking Data Sets" on page 3-4.

Related Examples

- "Test Data Against Trained System (GUI)" on page 3-18
- "Save Training Error Data to MATLAB Workspace" on page 3-31

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

More About

- "Neuro-Adaptive Learning and ANFIS" on page 3-2
- "Comparison of anfis and Neuro-Fuzzy Designer Functionality" on page 3-7

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Test Data Against Trained System (GUI)

In this section...

"Checking Data Helps Model Validation" on page 3-18
"Checking Data Does Not Validate Model" on page 3-26

Checking Data Helps Model Validation

In this section, we look at an example that loads similar training and checking data sets. The checking data set is corrupted by noise.

1. "Loading Data" on page 3-18
2. "Initializing and Generating Your FIS" on page 3-21
3. "Viewing Your FIS Structure" on page 3-23
4. "ANFIS Training" on page 3-24
5. "Testing Your Data Against the Trained FIS" on page 3-25

Loading Data

To work both of the following examples, you load the training data sets (`fuzex1trnData` and `fuzex2trnData`) and the checking data sets (`fuzex1chkData` and `fuzex2chkData`), into the Neuro-Fuzzy Designer from the workspace. You may also substitute your own data sets.

To load the data sets from the workspace into the Neuro-Fuzzy Designer:

1. Type the following commands at the MATLAB command line to load the data sets from the folder `fuzzydemos` into the MATLAB workspace:

   ```
   load fuzex1trnData.dat
   load fuzex2trnData.dat
   load fuzex1chkData.dat
   load fuzex2chkData.dat
   ```

2. Open the Neuro-Fuzzy Designer by typing `neuroFuzzyDesigner` in the MATLAB command line.
3. To load the training data set from the workspace:

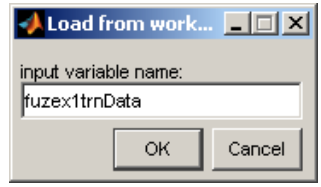
   a. In the **Load data** portion of the designer, select the following options:

      - **Type: Training**

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

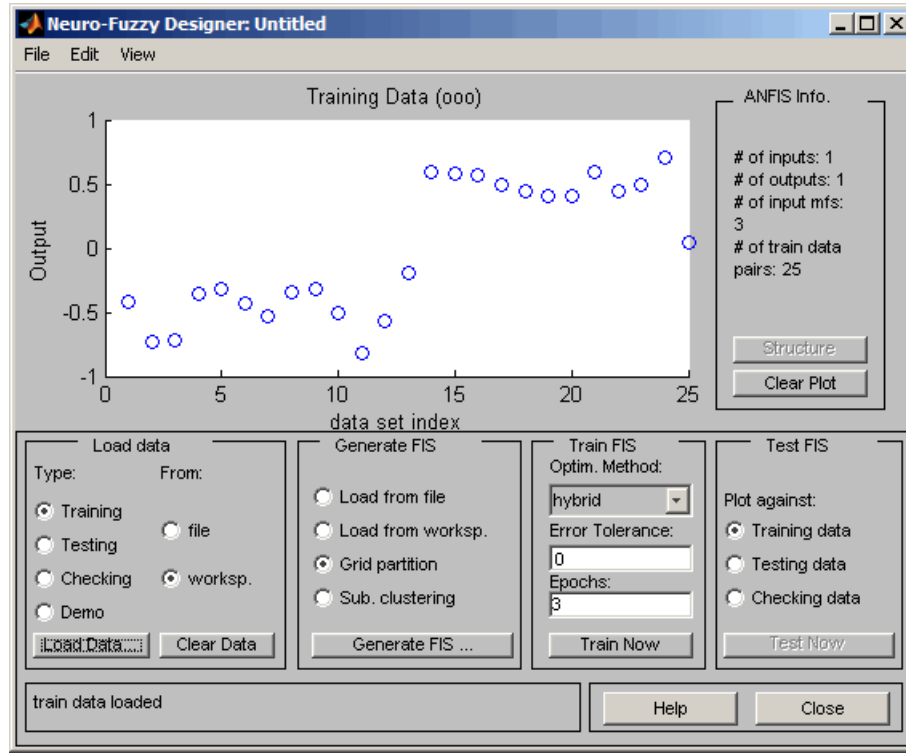
- **From: worksp.**

b Click **Load Data** to open the Load from workspace dialog box.



c Type `fuzex1trnData` as shown in the following figure, and click **OK**.

The training data set is used to train a fuzzy system by adjusting the membership function parameters that best model this data, and appears in the plot in the center of the GUI as a set of *circles*.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The horizontal axis is marked **data set index**. This index indicates the row from which that input data value was obtained (whether or not the input is a vector or a scalar).

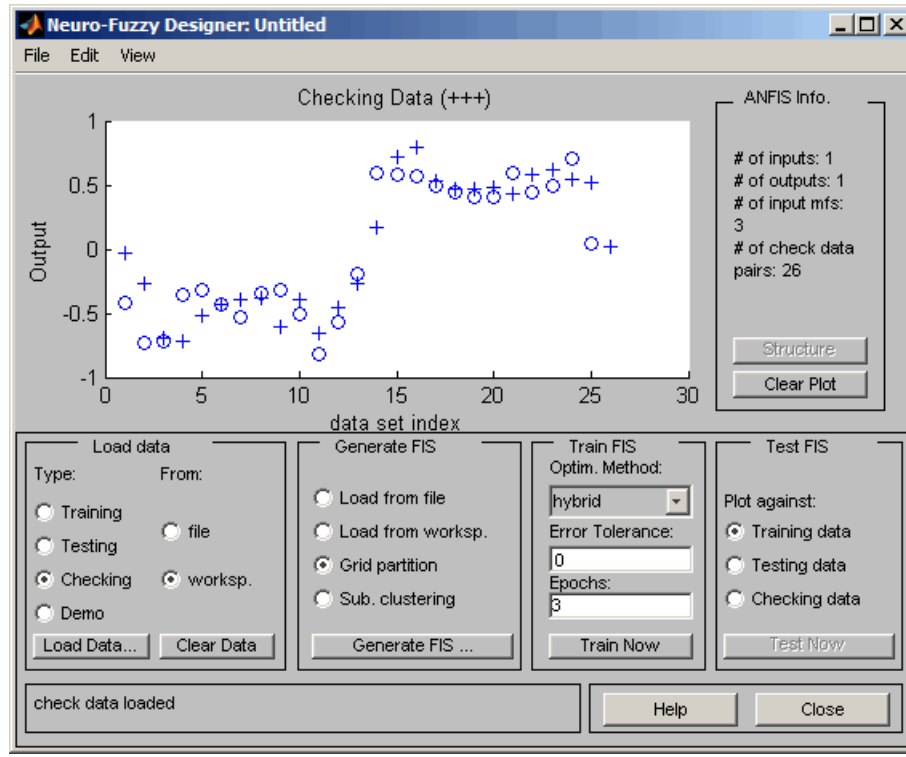
4 To load the checking data set from the workspace:

a In the **Load data** portion of the GUI, select **Checking** in the **Type** column.

b Click **Load Data** to open the Load from workspace dialog box.

c Type `fuzex1chkData` as the variable name and click **OK**.

The checking data appears in the GUI plot as *pluses* superimposed on the training data.



The next step is to specify an initial fuzzy inference system for `anfis` to train.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Initializing and Generating Your FIS

You can either initialize the FIS parameters to your own preference, or if you do not have any preference for how you want the initial membership functions to be parameterized, you can let `anfis` initialize the parameters for you, as described in the following sections:

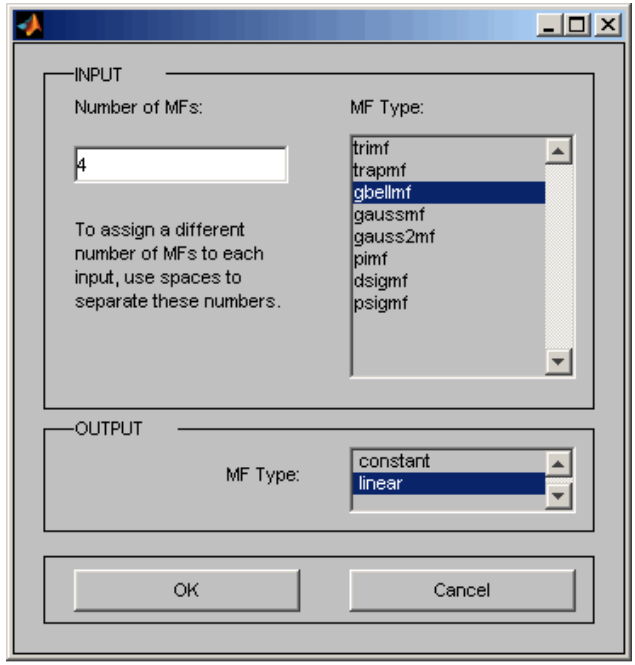
- "Automatic FIS Structure Generation" on page 3-21
- "Specifying Your Own Membership Functions for ANFIS" on page 3-22

Automatic FIS Structure Generation

To initialize your FIS using `anfis`:

1. Choose **Grid partition,** the default partitioning method. The two partition methods, grid partitioning and subtractive clustering, are described in `genfis1` and `genfis2` respectively.
2. Click on the **Generate FIS** button. Clicking this button displays a menu from which you can choose the number of membership functions, **MFs**, and the type of input and output membership functions. There are only two choices for the output membership function: `constant` and `linear`. This limitation of output membership function choices is because `anfis` only operates on Sugeno-type systems.
3. Fill in the entries as shown in the following figure, and click **OK.**

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



You can also implement this FIS generation from the command line using the command `genfis1` (for grid partitioning) or `genfis2` (for subtractive clustering).

Specifying Your Own Membership Functions for ANFIS

You can choose your own preferred membership functions with specific parameters to be used by `anfis` as an initial FIS for training.

To define your own FIS structure and parameters:

1. Open the **Membership functions** menu item from the **Edit** menu.
2. Add your desired membership functions (the custom membership option will be disabled for `anfis`). The output membership functions must either be all constant or all linear. For carrying out this and the following step, see "The Fuzzy Logic Designer" on page 2-34 and "The Membership Function Editor" on page 2-39.
3. Select the **Rules** menu item in the **Edit** menu, and use the Rule Editor to generate the rules (see "The Rule Editor" on page 2-46).

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

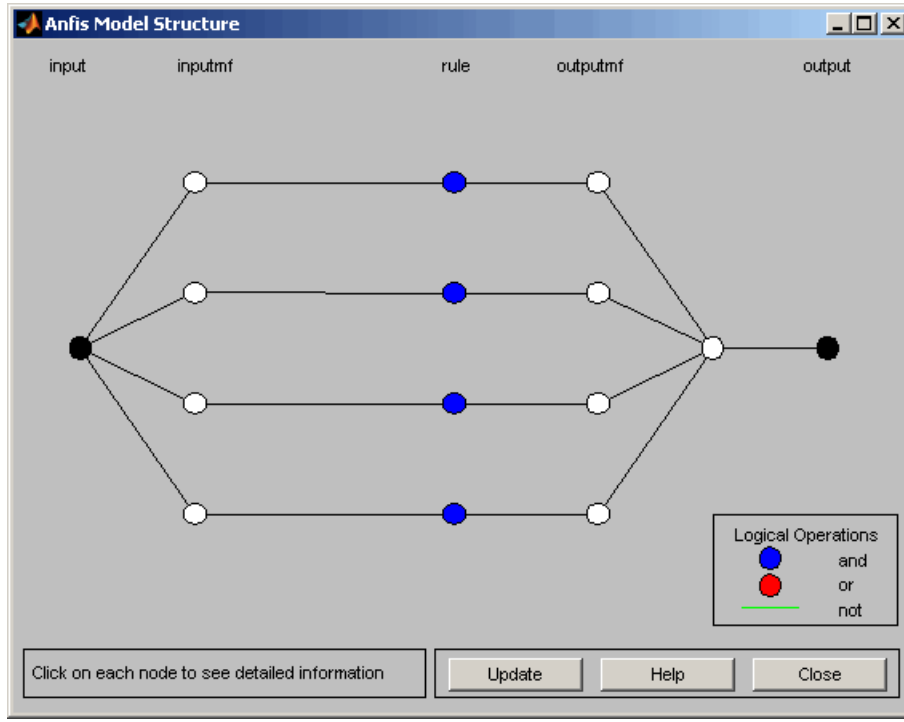
4 Select the **FIS Properties** menu item from the **Edit** menu. Name your FIS, and save it to either the workspace or to file.

5 Click the **Close** button to return to the Neuro-Fuzzy Designer to train the FIS.

6 To load an existing FIS for ANFIS initialization, in the **Generate FIS** portion of the designer, click **Load from worksp.** or **Load from file**. You load your FIS from a file if you have saved a FIS previously that you would like to use. Otherwise you load your FIS from the workspace.

Viewing Your FIS Structure

After you generate the FIS, you can view the model structure by clicking the **Structure** button in the middle of the right side of the editor. A new editor appears, as follows.



The branches in this graph are color coded. Color coding of branches characterize the rules and indicate whether or not *and*, *not*, or *or* are used in the rules. The input is represented by the left-most node and the output by the right-most node. The

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

node represents a normalization factor for the rules. Clicking on the nodes indicates information about the structure.

You can view the membership functions or the rules by opening either the Membership Function Editor, or the Rule Editor from the **Edit** menu.

ANFIS Training

The two `anfis` parameter optimization method options available for FIS training are `hybrid` (the default, mixed least squares and backpropagation) and `backpropa` (backpropagation). **Error Tolerance** is used to create a training stopping criterion, which is related to the error size. The training will stop after the training data error remains within this tolerance. This is best left set to 0 if you are unsure how your training error may behave.

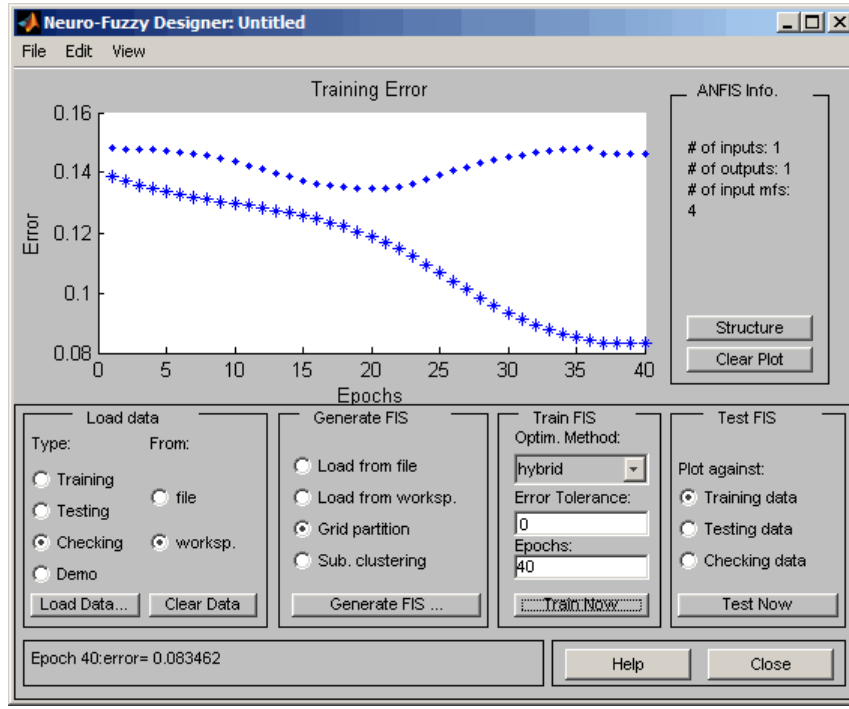
Note: If you want to save the training error data generated during ANFIS training to the MATLAB workspace, you must train the FIS at the command line. For an example, "Save Training Error Data to MATLAB Workspace" on page 3-31.

To start the training:

1. Leave the optimization method at `hybrid`.
2. Set the number of training epochs to 40, under the **Epochs** listing on the GUI (the default value is 3).
3. Select **Train Now**.

   The following window appears on your screen.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

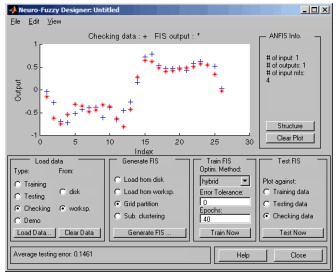


The plot shows the checking error as ♦ ♦ on the top . The training error appears as * * on the bottom. The checking error decreases up to a certain point in the training, and then it increases. This increase represents the point of model overfitting. `anfis` chooses the model parameters associated with the minimum checking error (just prior to this jump point). This example shows why the checking data option of `anfis` is useful.

Testing Your Data Against the Trained FIS

To test your FIS against the checking data, select **Checking data** in the **Test FIS** portion of the Neuro-Fuzzy Designer, and click **Test Now**. When you test the checking data against the FIS, it looks satisfactory.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Loading More Data with anfis

If you load data into `anfis` after clearing previously loaded data, you must make sure that the newly loaded data sets have the same number of inputs as the previously loaded ones did. Otherwise, you must start a new `neuroFuzzyDesigner` session from the command line.

Checking Data Option and Clearing Data

If you do not want to use the checking data option of `anfis`, then do not load any checking data before you train the FIS. If you decide to retrain your FIS with no checking data, you can unload the checking data in one of two ways:

- Select the **Checking** option button in the **Load data** portion of the Neuro-Fuzzy Designer, and then click **Clear Data** to unload the checking data.
- Close the Neuro-Fuzzy Designer, and go to the MATLAB command line, and retype `neuroFuzzyDesigner`. In this case you must reload the training data.

After clearing the data, you must regenerate your FIS. After the FIS is generated, you can use your first training experience to decide on the number of training epochs you want for the second round of training.

Checking Data Does Not Validate Model

This example examines what happens when the training and checking data sets are sufficiently different. To see how the Neuro-Fuzzy Designer can be used to learn something about data sets and how they differ:

1. Clear the Neuro-Fuzzy Designer:
   - Clear both the training and checking data.

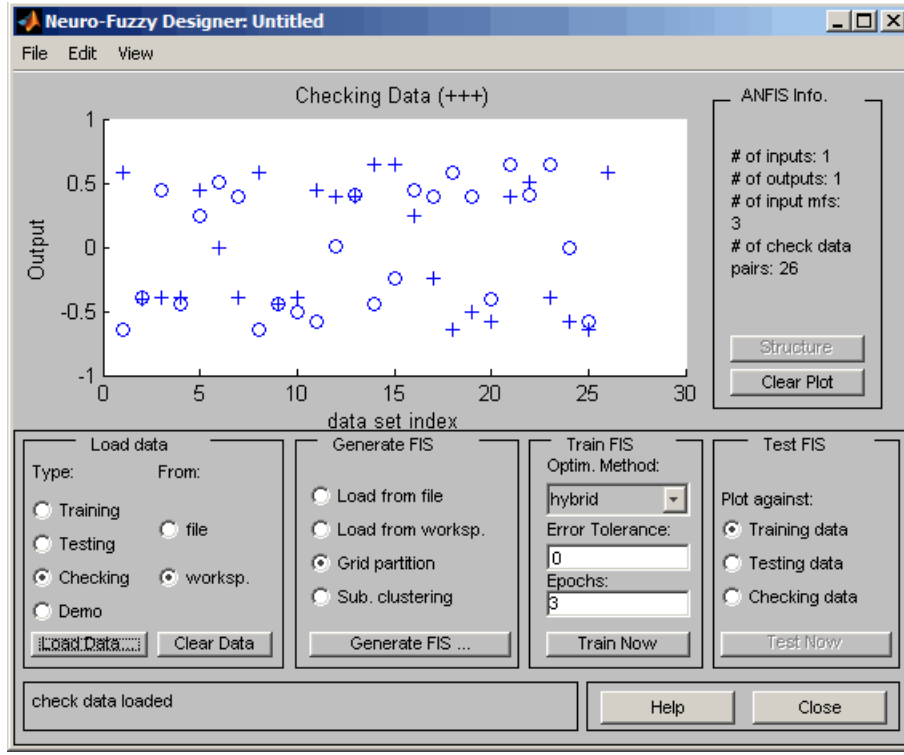
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- (optional) Click the **Clear Plot** button on the right.

2 Load `fuzex2trnData` and `fuzex2chkData` (respectively, the training data and checking data) from the MATLAB workspace just as you did in the previous example.

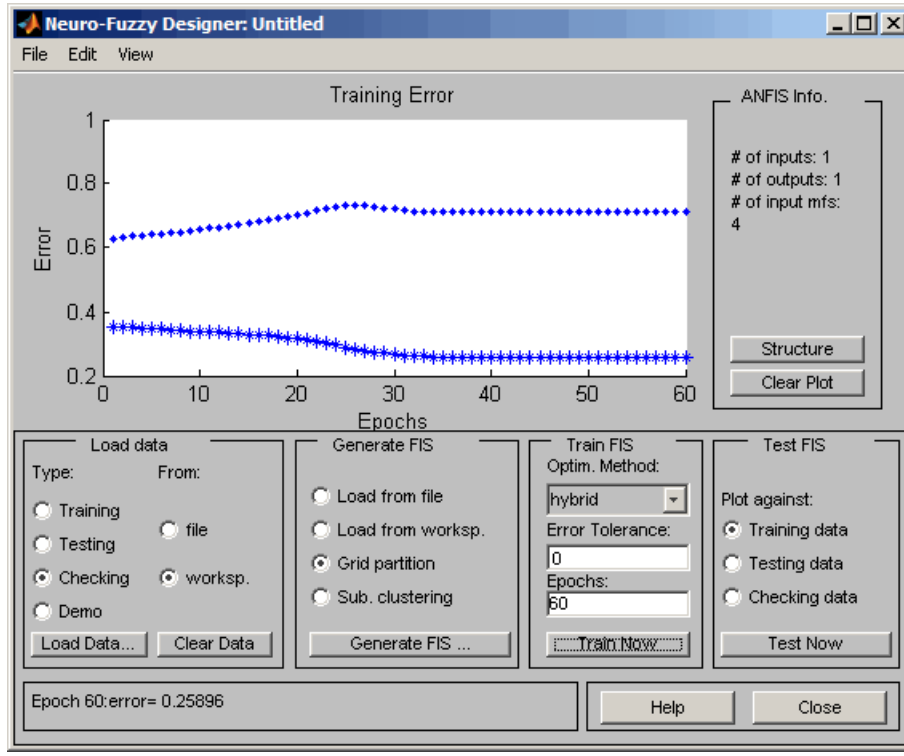
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

You should see a plot similar to the one in the following figure. The training data appears as *circles* superimposed with the checking data, appearing as *pluses*.



Train the FIS for this system exactly as you did in the previous example, except now choose 60 **Epochs** before training. You should get the following plot, showing the checking error as ♦ ♦ on top and the training error as * * on the bottom.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



In this case, the checking error is quite large. It appears that the minimum checking error occurs within the first epoch. Using the checking data option with `anfis` automatically sets the FIS parameters to be those associated with the minimum checking error. Clearly this set of membership functions is not the best choice for modeling the training data.

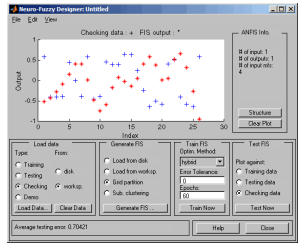
This example illustrates the problem discussed earlier wherein the checking data set presented to `anfis` for training was sufficiently different from the training data set. As a result, the trained FIS did not capture the features of this data set very well. It is important to know the features of your data set well when you select your training and checking data. When you do not know the features of your data, you can analyze the checking error plots to see whether or not the checking data performed sufficiently well with the trained model.

In this example, the checking error is sufficiently large to indicate that either you need to select more data for training or modify your membership function choices (both the

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

number of membership functions and the type). Otherwise, the system can be retrained without the checking data, if you think the training data sufficiently captures the features you are trying to represent.

To complete this example, test the trained FIS model against the checking data. To do so, select **Checking data** in the **Test FIS** portion of the GUI, and click **Test Now**. The following plot in the GUI indicates that there is quite a discrepancy between the checking data output and the FIS output.



Related Examples

- "Train Adaptive Neuro-Fuzzy Inference Systems (GUI)" on page 3-13
- "Save Training Error Data to MATLAB Workspace" on page 3-31

More About

- "Neuro-Adaptive Learning and ANFIS" on page 3-2
- "Comparison of anfis and Neuro-Fuzzy Designer Functionality" on page 3-7

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Save Training Error Data to MATLAB Workspace

When working in the Neuro-Fuzzy Designer, you can export your initial FIS structure to the MATLAB workspace and then save the ANFIS training error values in the workspace.

The following example shows how to save the training error generated during ANFIS training to the MATLAB workspace:

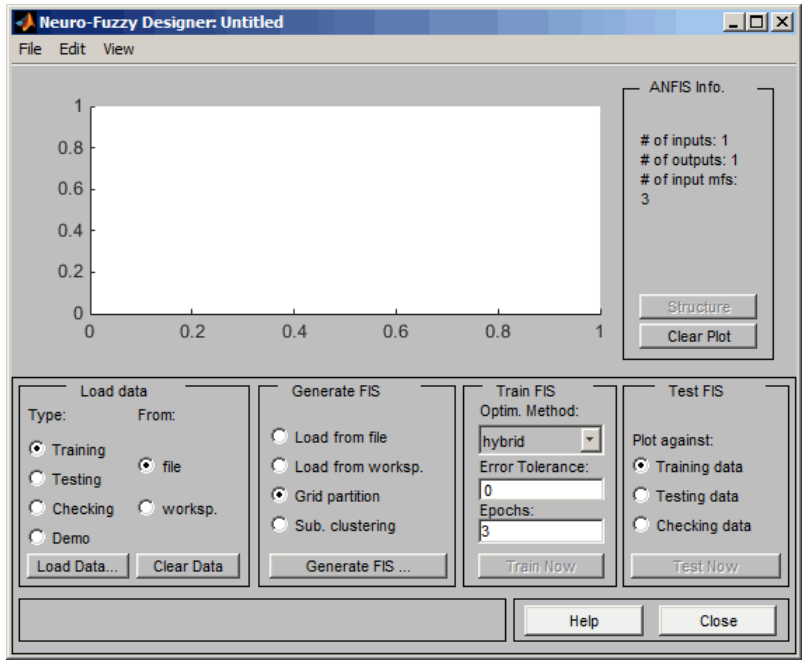
1. Load the training and checking data in the MATLAB workspace by typing the following commands at the MATLAB prompt:

   ```
   load fuzex1trnData.dat
   load fuzex1chkData.dat
   ```

2. Open the Neuro-Fuzzy Designer by typing the following command:

   ```
   neuroFuzzyDesigner
   ```

   The Neuro-Fuzzy Designer opens, as shown in the next figure.



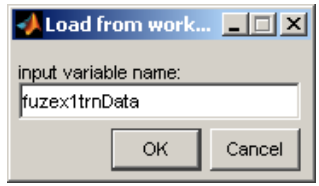
3. Load the training data from the MATLAB workspace into the Neuro-Fuzzy Designer:

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

a In the **Load data** panel of the Neuro-Fuzzy Designer, verify that **Training** is selected in the **Type** column.

b Select **worksp.** in the **From** column.

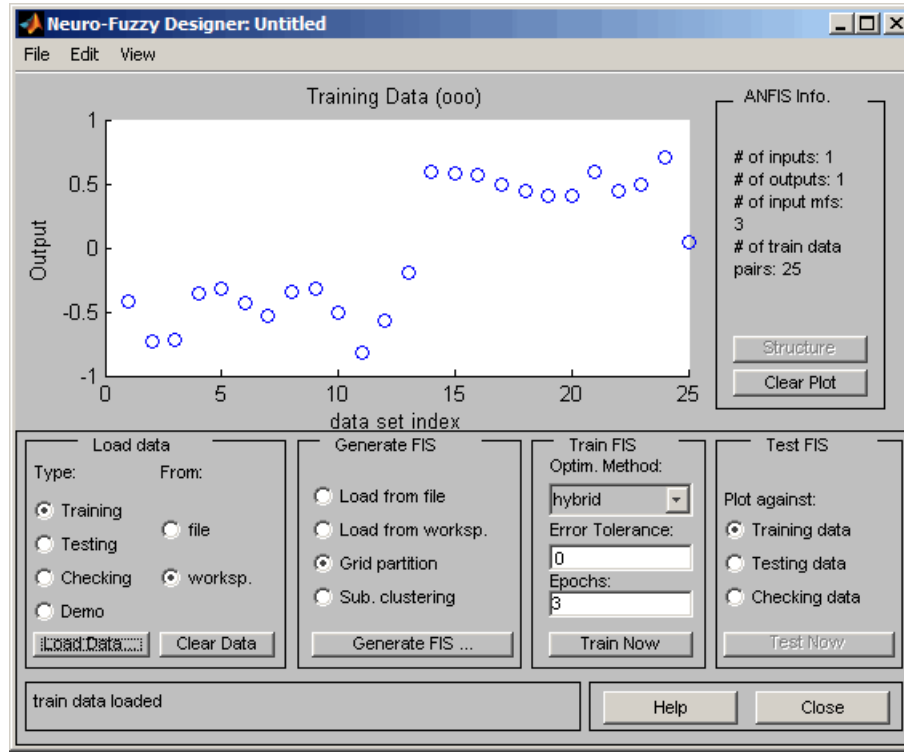
c Click **Load Data** to open the Load from workspace dialog box.



d Type `fuzex1trnData`, and click **OK**.

The Neuro-Fuzzy Designer displays the training data in the plot as a set of circles (○).

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

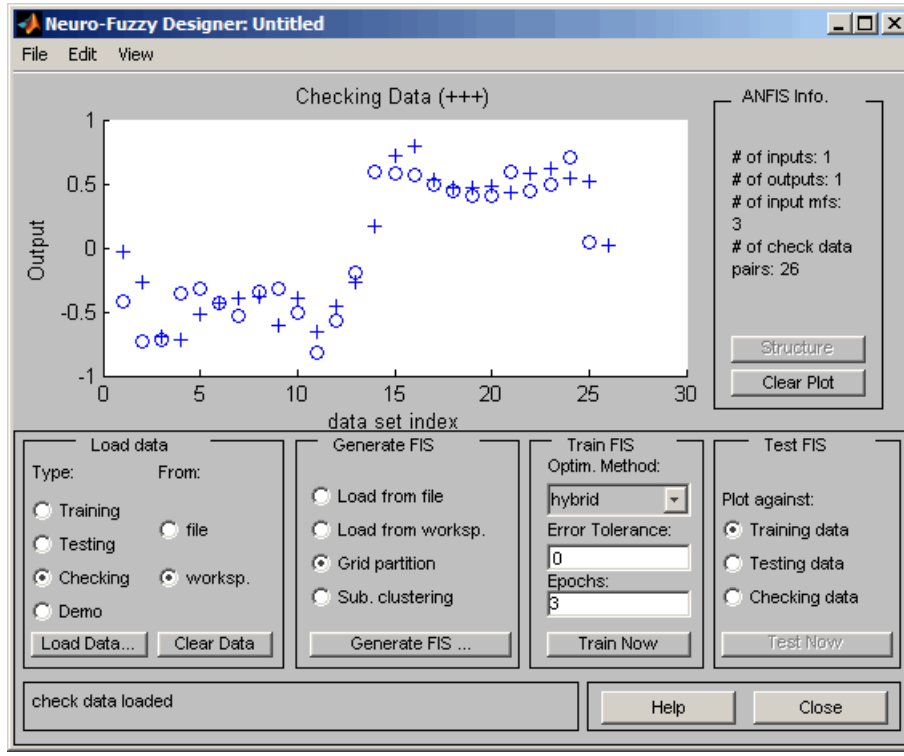


4 Load the checking data from the MATLAB workspace into the Neuro-Fuzzy Designer:

- **a** In the **Load data** panel of the Neuro-Fuzzy Designer, select **Checking** in the **Type** column.
- **b** Click **Load Data** to open the Load from workspace dialog box.
- **c** Type `fuzex1chkData` as the variable name, and click **OK**.

  The Neuro-Fuzzy Designer displays the checking data as plus signs (+) superimposed on the training data.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



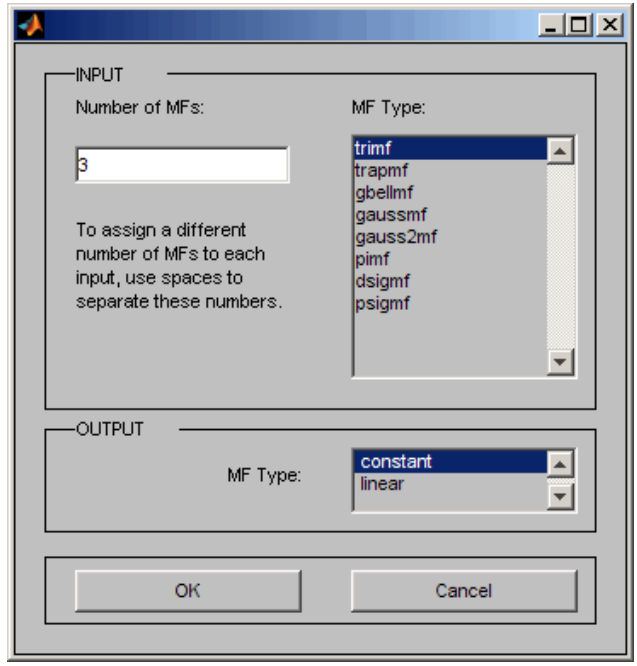
5 Generate an initial FIS:

a In the **Generate FIS** panel, verify that **Grid partition** option is selected.

b Click **Generate FIS**.

This action opens a dialog box where you specify the structure of the FIS.

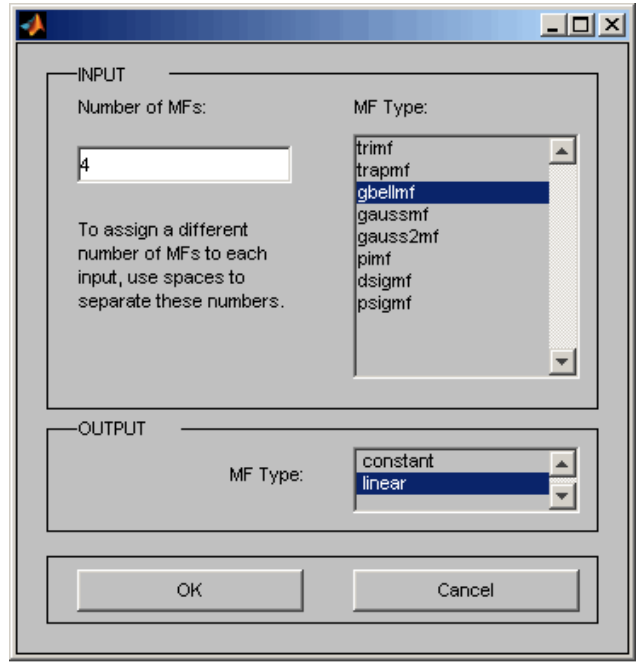
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



c In the dialog box, specify the following:

- Enter 4 in the **Number of MFs** field.
- Select `gbellmf` as the **Membership Type** for the input.
- Select `linear` as the **Membership Type** for the output.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

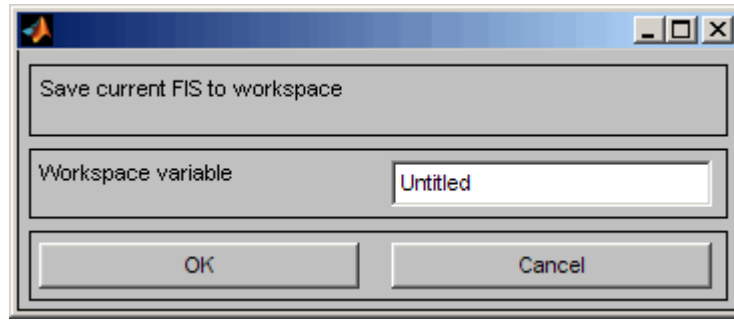


d Click **OK** to generate the FIS and close the dialog box.

6 Export the initial FIS to the MATLAB workspace:

a In the Neuro-Fuzzy Designer, select **File** > **Export** > **To Workspace**.

This action opens a dialog box where you specify the MATLAB variable name.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

b Enter `initfis` in the **Workspace variable** field.

c Click **OK** to close the dialog box.

A variable named `initfis` now appears in the MATLAB workspace.

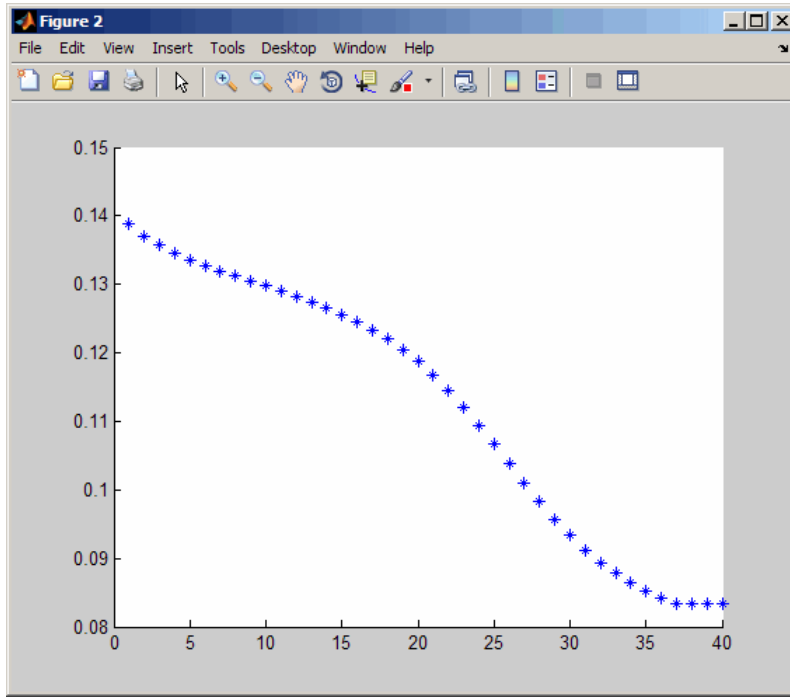
7 Train the FIS for 40 epochs by typing the following command at the MATLAB prompt:

```
figure;hold on;
fismat=initfis;
for ct=1:40,
[fismat,error]=anfis(fuzex1trnData,fismat,...
        2,NaN,fuzex1chkData,1);
plot(ct,error(1),'b*');
end
```

To improve accuracy when you train the FIS, the code uses the results of the current iteration returned by the `anfis` command as the initial conditions for the next iteration. The output argument `error` contains the root mean squared errors representing the training data error. For more information, see the anfis reference page.

The plot of the training error versus the number of epochs appears in the next figure.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Related Examples

- "Train Adaptive Neuro-Fuzzy Inference Systems (GUI)" on page 3-13
- "Test Data Against Trained System (GUI)" on page 3-18

More About

- "Neuro-Adaptive Learning and ANFIS" on page 3-2
- "Comparison of anfis and Neuro-Fuzzy Designer Functionality" on page 3-7

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Predict Chaotic Time-Series

This example shows how to use the command line features of `anfis` on a chaotic time-series prediction example.

Generating an FIS using the ANFIS Editor GUI is quite simple.

However, you need to be cautious about implementing the checking data validation feature of `anfis`. You must check that the checking data error does what it is supposed to. Otherwise, you need to retrain the FIS.

This example uses `anfis` to predict a time series that is generated by the following Mackey-Glass (MG) time-delay differential equation.

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} = 0.1x(t)$$

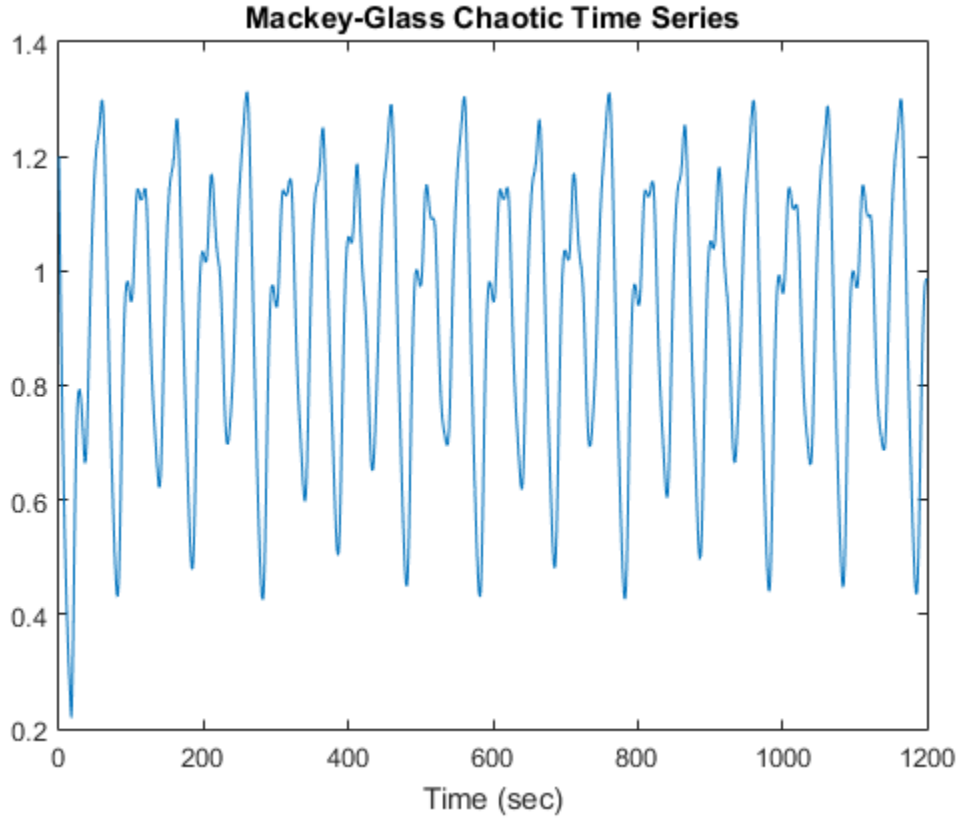
This time series is chaotic with no clearly defined period. The series does not converge or diverge, and the trajectory is highly sensitive to initial conditions. This benchmark problem is used in the neural network and fuzzy modeling research communities.

To obtain the time series value at integer points, the fourth-order Runge-Kutta method was used to find the numerical solution to the previous MG equation. It was assumed that $x(0) = 1.2$, $\tau = 17$, and $x(t) = 0$ for $t < 0$. The result was saved in the file `mgdata.dat`.

Plot the MG time series.

```
load mgdata.dat
time  =  mgdata(:, 1); x  =  mgdata(:, 2);
figure(1), plot(time, x);
title('Mackey-Glass Chaotic Time Series')
xlabel('Time (sec)')
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



In time-series prediction, you need to use known values of the time series up to the point in time, t, to predict the value at some point in the future, $t + P$. The standard method for this type of prediction is to create a mapping from D sample data points, sampled every Δ units in time, ($x(t-(D-1)\Delta), \ldots, x(t-\Delta), x(t)$), to a predicted future value $x = (t + P)$. Following the conventional settings for predicting the MG time series, set $D = 4$ and $\Delta = P = 6$. For each t, the input training data for `anfis` is a four-column vector of the following form.

$$w(t) = [x(t-19), x(t-12), x(t-6), x(t)]$$

The output training data corresponds to the trajectory prediction.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

$$s(t) = x(t+6)$$

For each t, ranging in values from 118 to 1117, the training input/output data is a structure whose first component is the four-dimensional input w, and whose second component is the output s. There are 1000 input/output data values. You use the first 500 data values for the `anfis` training (these become the training data set), while the others are used as checking data for validating the identified fuzzy model. This division of data values results in two 500-point data structures, `trnData` and `chkData`.

The following code generates this data:

```
for t=118:1117,
Data(t-117,:) = [x(t-18) x(t-12) x(t-6) x(t) x(t+6)];
end
trnData=Data(1:500,:);
chkData=Data(501:end,:);
```

To start the training, you need a FIS structure that specifies the structure and initial parameters of the FIS for learning. The `genfis1` function handles this specification.

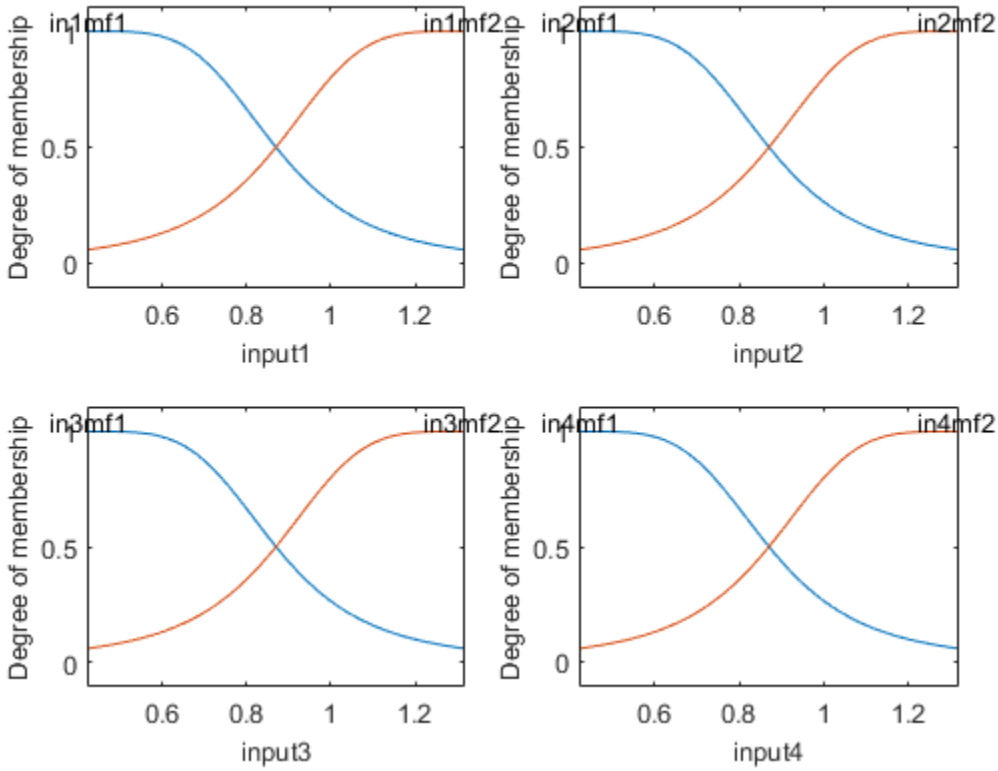
```
fismat = genfis1(trnData);
```

Because you did not specify numbers and types of membership functions used in the FIS, default values are assumed. These defaults provide two generalized bell membership functions on each of the four inputs, eight altogether. The generated FIS structure contains 16 fuzzy rules with 104 parameters. To achieve good generalization capability, it is important that the number of training data points be several times larger than the number parameters being estimated. In this case, the ratio between data and parameters is about five (500/104).

The function `genfis1` generates initial membership functions that are equally spaced and cover the whole input space. Plot the input membership functions.

```
figure(2)
subplot(2,2,1)
plotmf(fismat, 'input', 1)
subplot(2,2,2)
plotmf(fismat, 'input', 2)
subplot(2,2,3)
plotmf(fismat, 'input', 3)
subplot(2,2,4)
plotmf(fismat, 'input', 4)
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Start the training.

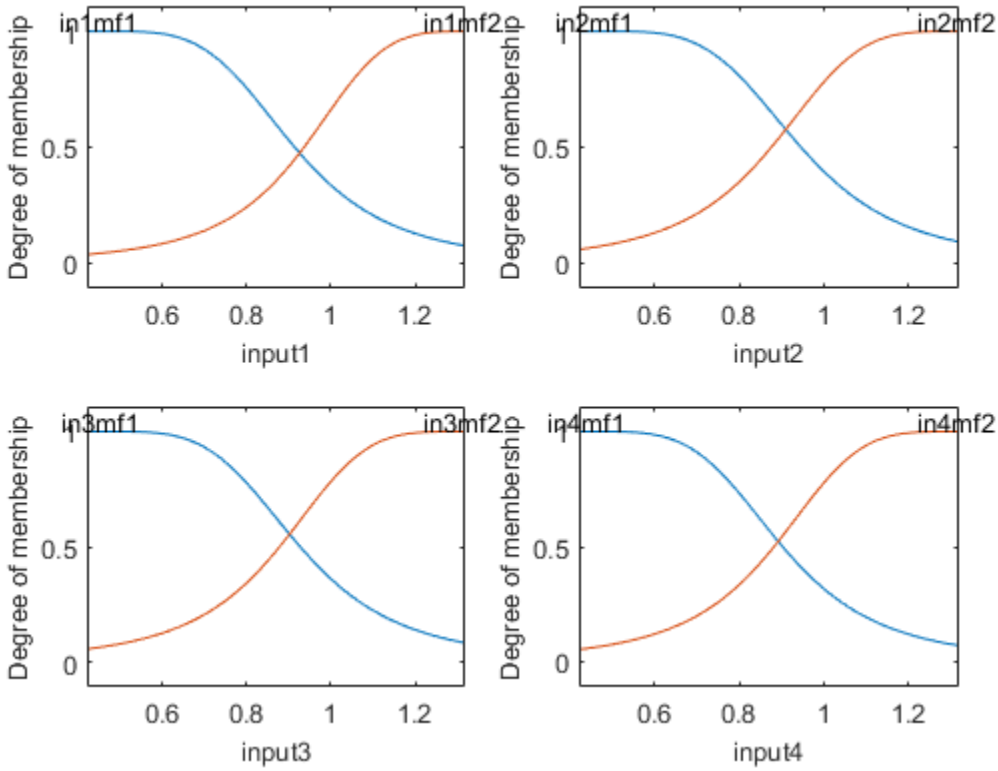
```
[fismat1,error1,ss,fismat2,error2] = ...
   anfis(trnData,fismat,[],[0 0 0 0],chkData);
```

Because the checking data option of `anfis` is invoked, the final FIS you choose is the one associated with the minimum checking error. This result is stored in `fismat2`. Plots these new membership functions.

```
figure(3)
subplot(2,2,1)
plotmf(fismat2, 'input', 1)
subplot(2,2,2)
plotmf(fismat2, 'input', 2)
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

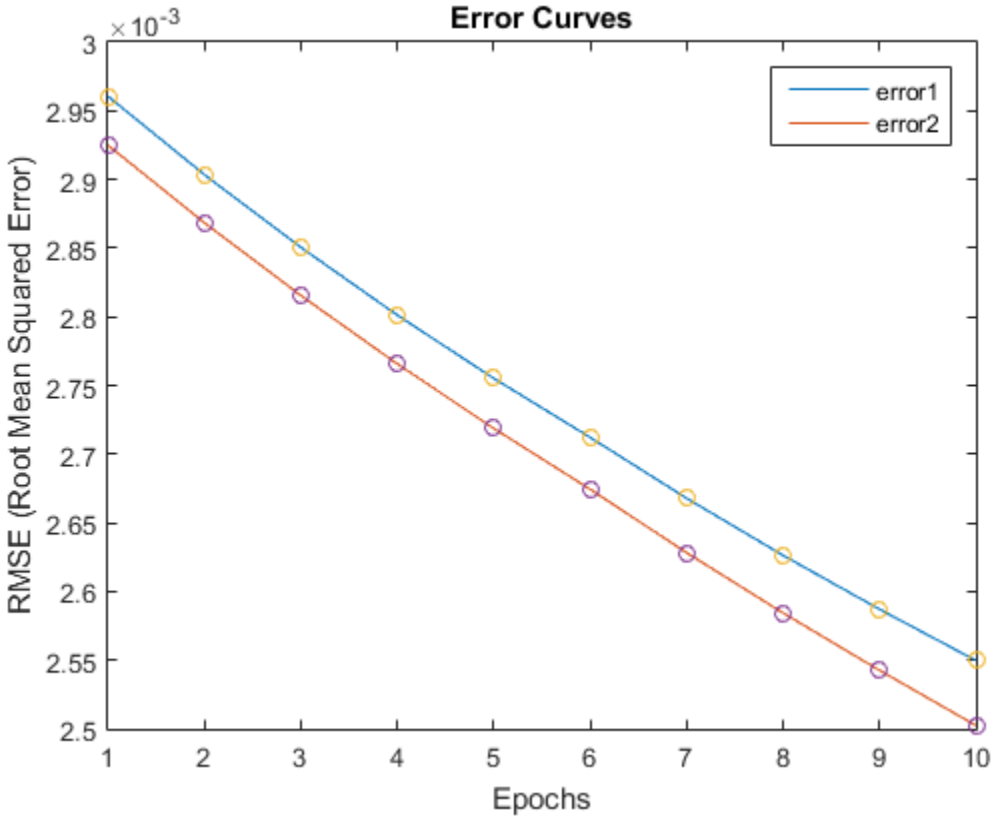
```
subplot(2,2,3)
plotmf(fismat2, 'input', 3)
subplot(2,2,4)
plotmf(fismat2, 'input', 4)
```



Plot the error signals.

```
figure(4)
plot([error1 error2]);
hold on; plot([error1 error2], 'o');
legend('error1','error2');
xlabel('Epochs');
ylabel('RMSE (Root Mean Squared Error)');
title('Error Curves');
```

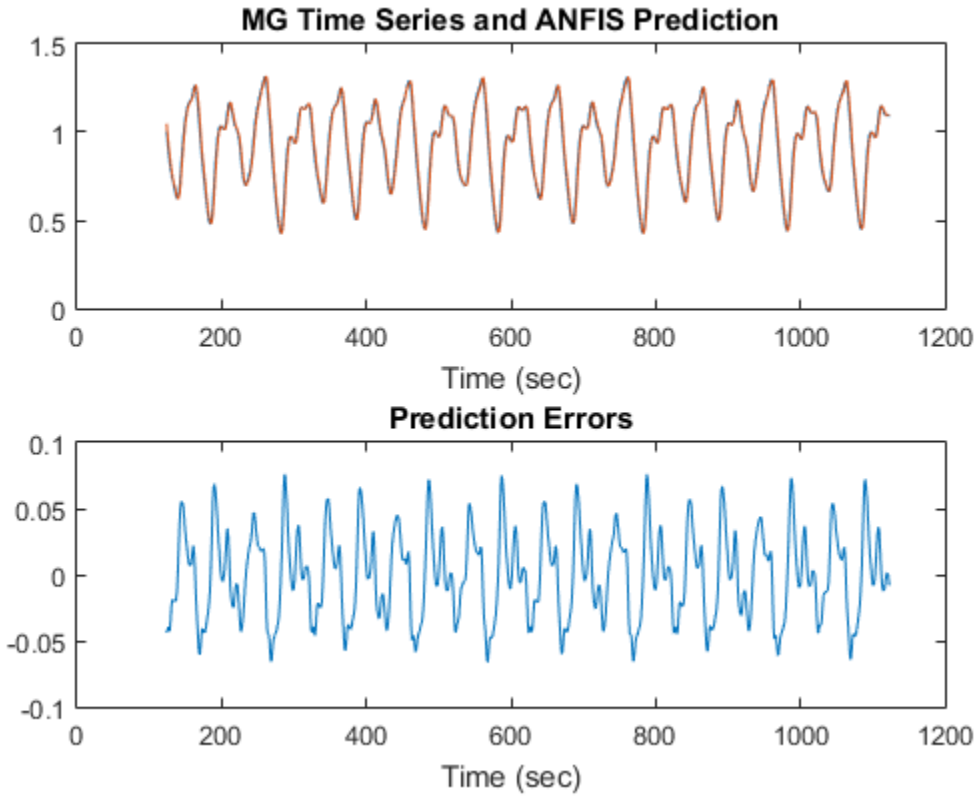

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



In addition to these error plots, you may want to plot the FIS output versus the training or checking data. To compare the original MG time series and the fuzzy prediction side by side, try:

```
figure(5)
anfis_output = evalfis([trnData(:,1:4); chkData(:,1:4)], ...
    fismat2);
index = 125:1124;
subplot(211), plot(time(index), [x(index) anfis_output]);
xlabel('Time (sec)');
title('MG Time Series and ANFIS Prediction');
subplot(212), plot(time(index), x(index) - anfis_output);
xlabel('Time (sec)');
title('Prediction Errors');
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



The difference between the original MG time series and the values estimated using `anfis` is very small, Thus, you can only see one curve in the first plot. The prediction error appears in the second plot with a much finer scale. You trained for only 10 epochs. If you apply more extensive training, you get better performance.

See Also

`anfis` | `evalfis` | `genfis1`

More About

- "Neuro-Adaptive Learning and ANFIS" on page 3-2
- "Comparison of anfis and Neuro-Fuzzy Designer Functionality" on page 3-7

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Modeling Inverse Kinematics in a Robotic Arm

This example shows how to use a fuzzy system to model the inverse kinematics in a two-joint robotic arm.

What Is Inverse Kinematics?

Kinematics is the science of motion. In a two-joint robotic arm, given the angles of the joints, the kinematics equations give the location of the tip of the arm. Inverse kinematics refers to the reverse process. Given a desired location for the tip of the robotic arm, what should the angles of the joints be so as to locate the tip of the arm at the desired location. There is usually more than one solution and can at times be a difficult problem to solve.

This is a typical problem in robotics that needs to be solved to control a robotic arm to perform tasks it is designated to do. In a 2-dimensional input space, with a two-joint robotic arm and given the desired co-ordinate, the problem reduces to finding the two angles involved. The first angle is between the first arm and the ground (or whatever it is attached to). The second angle is between the first arm and the second arm.

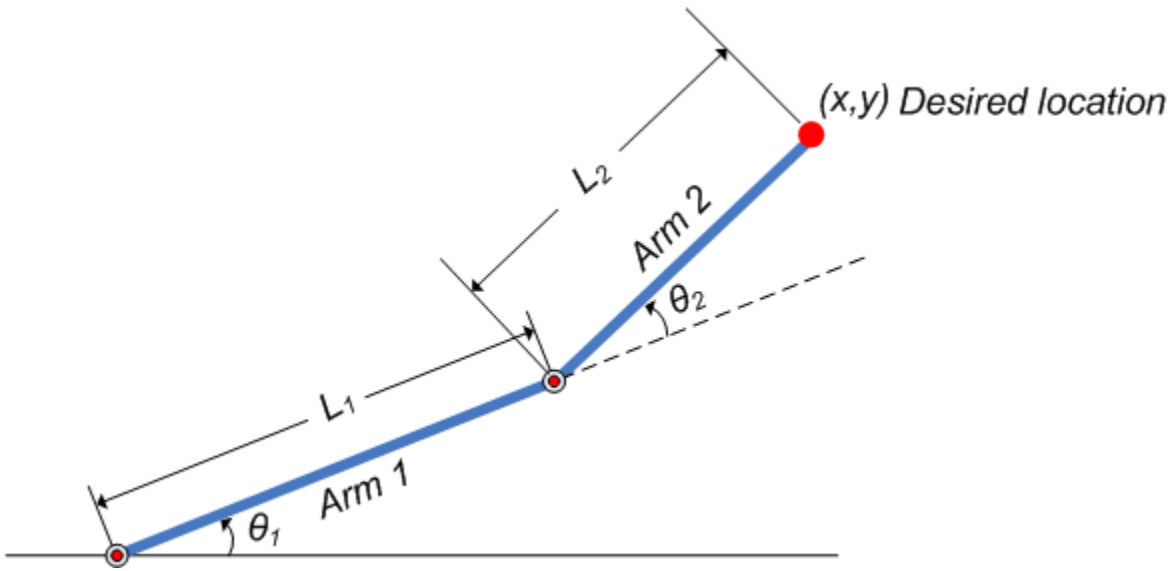


Figure 1: Illustration showing the two-joint robotic arm with the two angles, `theta1` and `theta2`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Why Use Fuzzy Logic?

For simple structures like the two-joint robotic arm, it is possible to mathematically deduce the angles at the joints given the desired location of the tip of the arm. However with more complex structures (eg: n-joint robotic arms operating in a 3-dimensional input space) deducing a mathematical solution for the inverse kinematics may prove challenging.

Using fuzzy logic, we can construct a Fuzzy Inference System that deduces the inverse kinematics if the forward kinematics of the problem is known, hence sidestepping the need to develop an analytical solution. Also, the fuzzy solution is easily understandable and does not require special background knowledge to comprehend and evaluate it.

In the following section, a broad outline for developing such a solution is described, and later, the detailed steps are elaborated.

Overview of Fuzzy Solution

Since the forward kinematics formulae for the two-joint robotic arm are known, x and y co-ordinates of the tip of the arm are deduced for the entire range of angles of rotation of the two joints. The co-ordinates and the angles are saved to be used as training data to train ANFIS (Adaptive Neuro-Fuzzy Inference System) network.

During training the ANFIS network learns to map the co-ordinates `(x,y)` to the angles `(theta1, theta2)`. The trained ANFIS network is then used as a part of a larger control system to control the robotic arm. Knowing the desired location of the robotic arm, the control system uses the trained ANFIS network to deduce the angular positions of the joints and applies force to the joints of the robotic arm accordingly to move it to the desired location.

What Is ANFIS?

ANFIS stands for Adaptive Neuro-Fuzzy Inference System. It is a hybrid neuro-fuzzy technique that brings learning capabilities of neural networks to fuzzy inference systems. The learning algorithm tunes the membership functions of a Sugeno-type Fuzzy Inference System using the training input-output data.

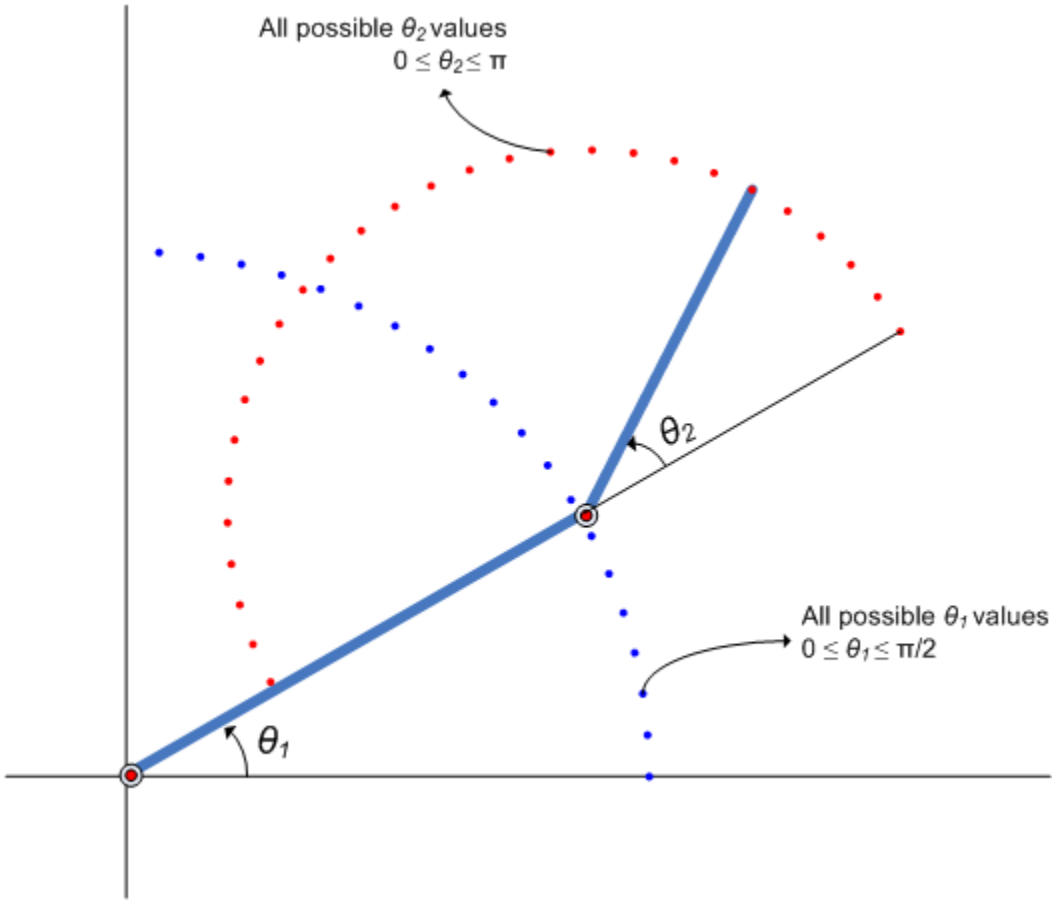
In this case, the input-output data refers to the "coordinates-angles" dataset. The coordinates act as input to the ANFIS and the angles act as the output. The learning algorithm "teaches" the ANFIS to map the co-ordinates to the angles through a process called training. At the end of training, the trained ANFIS network would have learned the input-output map and be ready to be deployed into the larger control system solution.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Data Generation

Let `theta1` be the angle between the first arm and the ground. Let `theta2` be the angle between the second arm and the first arm (Refer to Figure 1 for illustration). Let the length of the first arm be `l1` and that of the second arm be `l2`.

Let us assume that the first joint has limited freedom to rotate and it can rotate between 0 and 90 degrees. Similarly, assume that the second joint has limited freedom to rotate and can rotate between 0 and 180 degrees. (This assumption takes away the need to handle some special cases which will confuse the discourse). Hence, `0<=theta1<=pi/2` and `0<=theta2<=pi`.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Figure 2: Illustration showing all possible `theta1` and `theta2` values.

Now, for every combination of `theta1` and `theta2` values the x and y coordinates are deduced using forward kinematics formulae.

The following code snippet shows how data is generated for all combination of `theta1` and `theta2` values and saved into a matrix to be used as training data. The reason for saving the data in two matrices is explained in the following section.

```
l1 = 10; % length of first arm
l2 = 7; % length of second arm

theta1 = 0:0.1:pi/2; % all possible theta1 values
theta2 = 0:0.1:pi; % all possible theta2 values

[THETA1, THETA2] = meshgrid(theta1, theta2); % generate a grid of theta1 and theta2 val

X = l1 * cos(THETA1) + l2 * cos(THETA1 + THETA2); % compute x coordinates
Y = l1 * sin(THETA1) + l2 * sin(THETA1 + THETA2); % compute y coordinates

data1 = [X(:) Y(:) THETA1(:)]; % create x-y-theta1 dataset
data2 = [X(:) Y(:) THETA2(:)]; % create x-y-theta2 dataset
```

Click here for unvectorized code

The following plot shows all the X-Y data points generated by cycling through different combinations of `theta1` and `theta2` and deducing x and y co-ordinates for each. The plot can be generated by using the code-snippet shown below. The plot is illustrated further for easier understanding.

```
  plot(X(:), Y(:), 'r.');
  axis equal;
  xlabel('X','fontsize',10)
  ylabel('Y','fontsize',10)
  title('X-Y co-ordinates generated for all theta1 and theta2 combinations using forwar
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

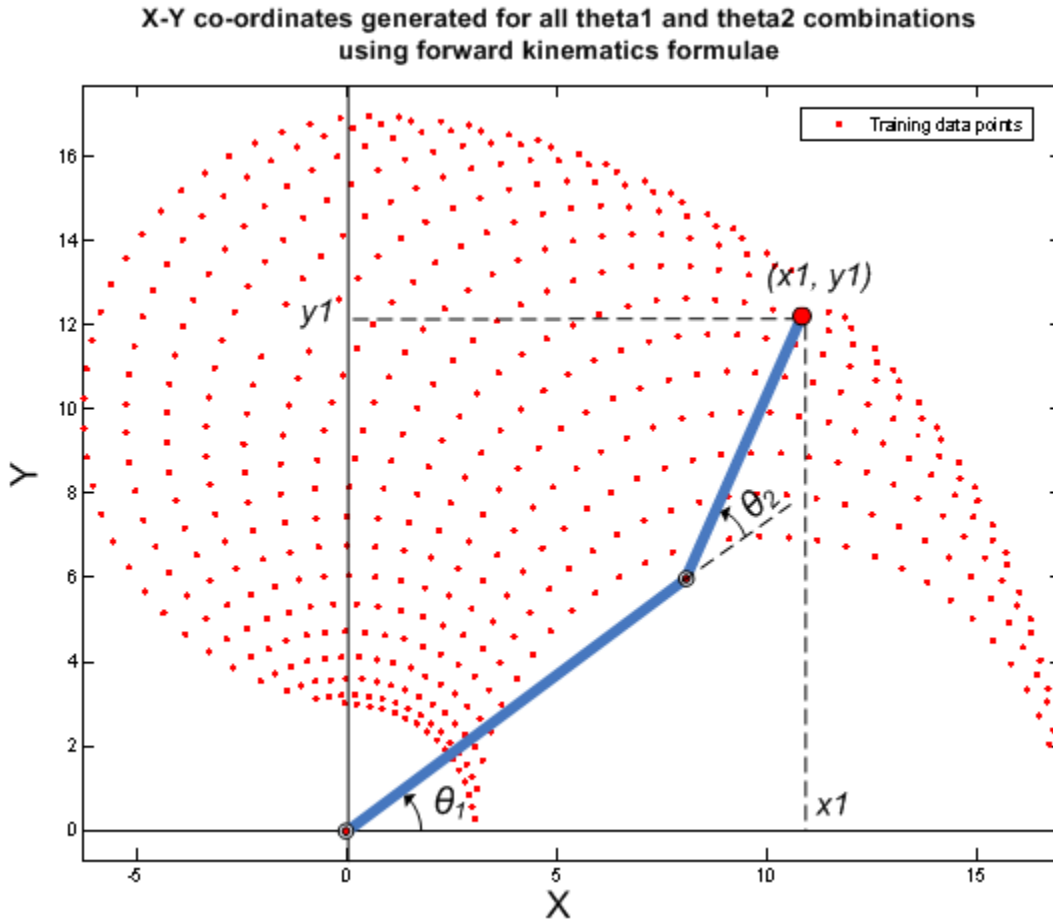


Figure 3: X-Y co-ordinates generated for all `theta1` and `theta2` combinations using forward kinematics formulae

Building ANFIS Networks

One approach to building an ANFIS solution for this problem, is to build two ANFIS networks, one to predict `theta1` and the other to predict `theta2`.

In order for the ANFIS networks to be able to predict the angles they have to be trained with sample input-output data. The first ANFIS network will be trained with X and Y coordinates as input and corresponding `theta1` values as output. The matrix `data1`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

contains the `x-y-theta1` dataset required to train the first ANFIS network. Therefore `data1` will be used as the dataset to train the first ANFIS network.

Similarly, the second ANFIS network will be trained with X and Y coordinates as input and corresponding `theta2` values as output. The matrix `data2` contains the `x-y-theta2` dataset required to train the second ANFIS network. Therefore `data2` will be used as the dataset to train the second ANFIS network.

`anfis` is the function that is used to train an ANFIS network. There are several syntaxes to the function. If called with the following syntax, `anfis` automatically creates a Sugeno-type FIS and trains it using the training data passed to the function.

The following code may take a couple of minutes to run:

```
fprintf('-->%s\n','Start training first ANFIS network. It may take one minute dependin
anfis1 = anfis(data1, 7, 150, [0,0,0,0]); % train first ANFIS network
fprintf('-->%s\n','Start training second ANFIS network. It may take one minute dependi
anfis2 = anfis(data2, 6, 150, [0,0,0,0]); % train second ANFIS network
```

```
-->Start training first ANFIS network. It may take one minute depending on your compute
-->Start training second ANFIS network. It may take one minute depending on your comput
```

The first parameter to `anfis` is the training data, the second parameter is the number of membership functions used to characterize each input and output, the third parameter is the number of training epochs and the last parameter is the options to display progress during training. The values for number of epochs and the number of membership functions have been arrived at after a fair amount of experimentation with different values.

The toolbox comes with GUI's that helps build and experiment with ANFIS networks.

`anfis1` and `anfis2` represent the two trained ANFIS networks that will be deployed in the larger control system.

Once the training is complete, the two ANFIS networks would have learned to approximate the angles (`theta1, theta2`) as a function of the coordinates (`x, y`). One advantage of using the fuzzy approach is that the ANFIS network would now approximate the angles for coordinates that are similar but not exactly the same as it was trained with. For example, the trained ANFIS networks are now capable of approximating the angles for coordinates that lie between two points that were included in the training dataset. This will allow the final controller to move the arm smoothly in the input space.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

We now have two trained ANFIS networks which are ready to be deployed into the larger system that will utilize these networks to control the robotic arms.

Validating ANFIS Networks

Having trained the networks, an important follow up step is to validate the networks to determine how well the ANFIS networks would perform inside the larger control system.

Since this example problem deals with a two-joint robotic arm whose inverse kinematics formulae can be derived, it is possible to test the answers that the ANFIS networks produce with the answers from the derived formulae.

Let's assume that it is important for the ANFIS networks to have low errors within the operating range `0<x<2` and `8<y<10`.

```
x = 0:0.1:2; % x coordinates for validation
y = 8:0.1:10; % y coordinates for validation
```

The `theta1` and `theta2` values are deduced mathematically from the x and y coordinates using inverse kinematics formulae.

```
[X, Y] = meshgrid(x,y);

c2 = (X.^2 + Y.^2 - l1^2 - l2^2)/(2*l1*l2);
s2 = sqrt(1 - c2.^2);
THETA2D = atan2(s2, c2); % theta2 is deduced

k1 = l1 + l2.*c2;
k2 = l2*s2;
THETA1D = atan2(Y, X) - atan2(k2, k1); % theta1 is deduced
```

Click here for unvectorized code

`THETA1D` and `THETA2D` are the variables that hold the values of `theta1` and `theta2` deduced using the inverse kinematics formulae.

`theta1` and `theta2` values predicted by the trained anfis networks are obtained by using the command `evalfis` which evaluates a FIS for the given inputs.

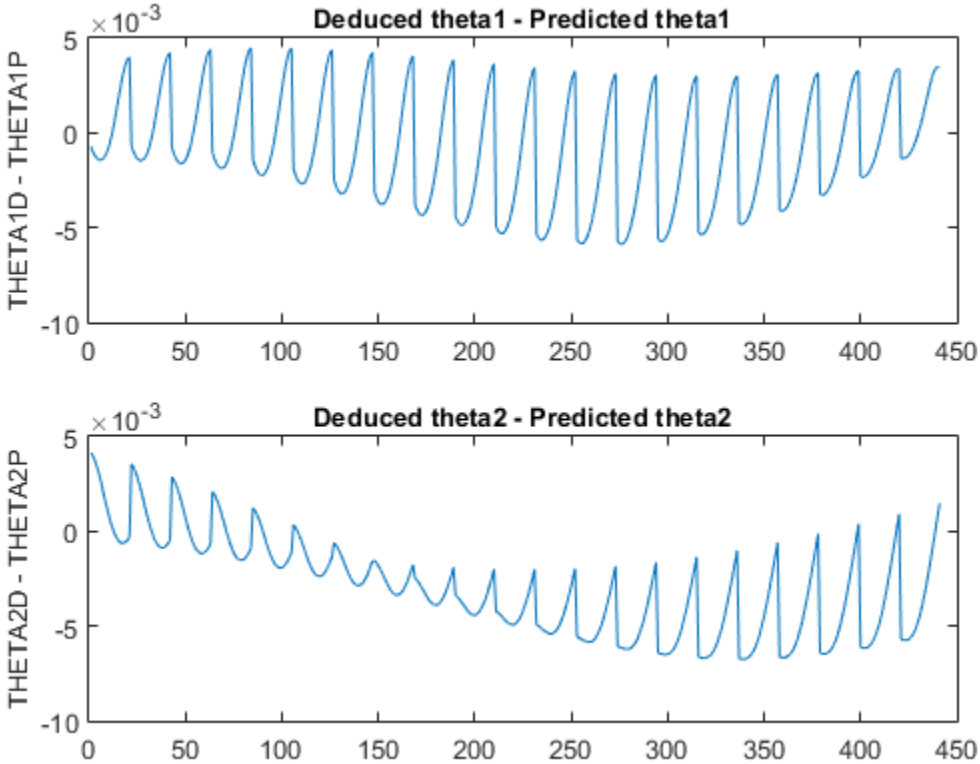
Here, `evalfis` is used to find out the FIS outputs for the same x-y values used earlier in the inverse kinematics formulae.

```
XY = [X(:) Y(:)];
THETA1P = evalfis(XY, anfis1); % theta1 predicted by anfis1
THETA2P = evalfis(XY, anfis2); % theta2 predicted by anfis2
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Now, we can see how close the FIS outputs are with respect to the deduced values.

```
theta1diff = THETA1D(:) - THETA1P;
theta2diff = THETA2D(:) - THETA2P;

subplot(2,1,1);
plot(theta1diff);
ylabel('THETA1D - THETA1P','fontsize',10)
title('Deduced theta1 - Predicted theta1','fontsize',10)

subplot(2,1,2);
plot(theta2diff);
ylabel('THETA2D - THETA2P','fontsize',10)
title('Deduced theta2 - Predicted theta2','fontsize',10)
```



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The errors are in the `1e-3` range which is a fairly good number for the application it is being used in. However this may not be acceptable for another application, in which case the parameters to the `anfis` function may be tweaked until an acceptable solution is arrived at. Also, other techniques like input selection and alternate ways to model the problem may be explored.

Building a Solution Around the Trained ANFIS Networks

Now given a specific task, such as robots picking up an object in an assembly line, the larger control system will use the trained ANFIS networks as a reference, much like a lookup table, to determine what the angles of the arms must be, given a desired location for the tip of the arm. Knowing the desired angles and the current angles of the joints, the system will apply force appropriately on the joints of the arms to move them towards the desired location.

The invkine command launches a GUI that shows how the two trained ANFIS networks perform when asked to trace an ellipse.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

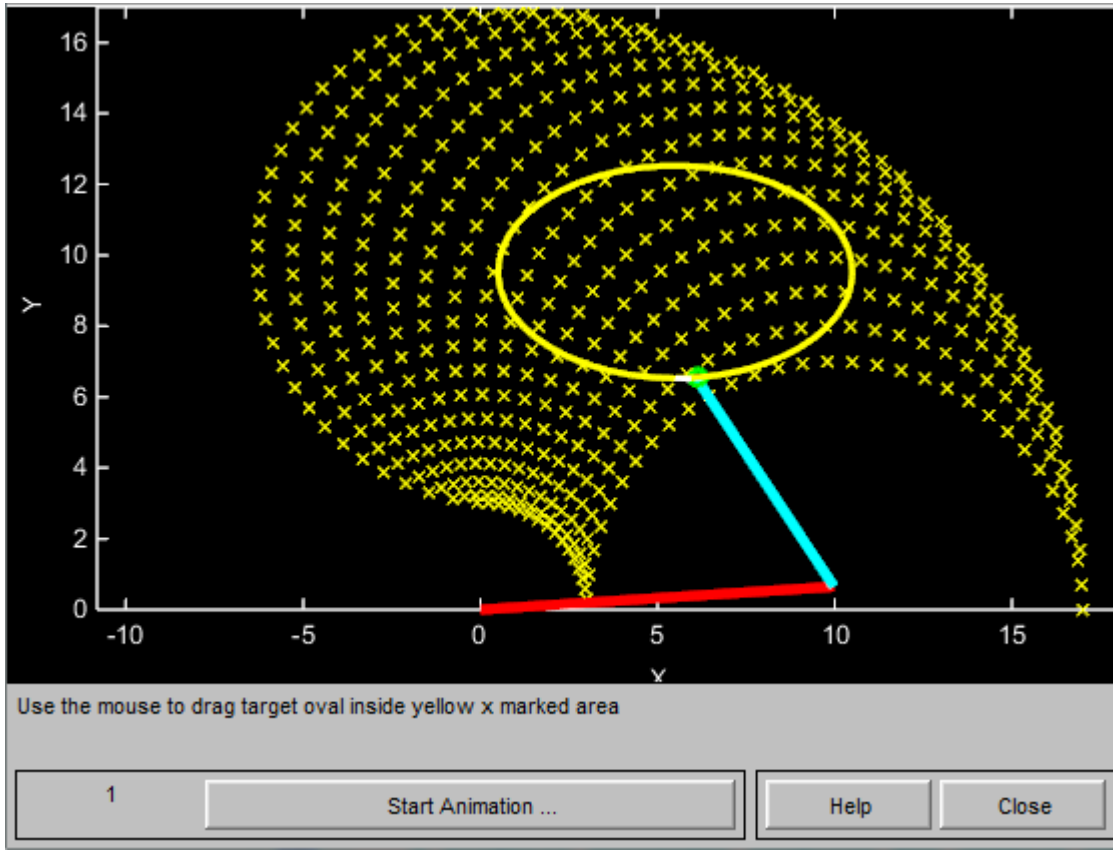


Figure 4: GUI for Inverse Kinematics Modeling.

The two ANFIS networks used in the example have been pre-trained and are deployed into a larger system that controls the tip of the two-joint robot arm to trace an ellipse in the input space.

The ellipse to be traced can be moved around. Move the ellipse to a slightly different location and observe how the system responds by moving the tip of the robotic arm from its current location to the closest point on the new location of the ellipse. Also observe that the system responds smoothly as long as the ellipse to be traced lies within the 'x' marked spots which represent the data grid that was used to train the networks. Once the ellipse is moved outside the range of data it was trained with, the ANFIS networks respond unpredictably. This emphasizes the importance of having relevant and

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

representative data for training. Data must be generated based on the expected range of operation to avoid such unpredictability and instability issues.

Conclusion

This example illustrated using ANFIS to solve an inverse kinematics problem. Fuzzy logic has also found numerous other applications in other areas of technology like non-linear control, automatic control, signal processing, system identification, pattern recognition, time series prediction, data mining, financial applications etc.,

Explore other demos and the documentation for more insight into fuzzy logic and its applications.

Glossary

ANFIS - Adaptive Neuro-Fuzzy Inference System. a technique for automatically tuning Sugeno-type inference systems based on training data.

membership functions - a function that specifies the degree to which a given input belongs to a set or is related to a concept.

input space - it is a term used to define the range of all possible values

FIS - Fuzzy Inference System. The overall name for a system that uses fuzzy reasoning to map an input space to an output space.

epochs - 1 epoch of training represents one complete presentation of all the samples/datapoints/rows of the training dataset to the FIS. The inputs of each sample are presented and the FIS outputs are computed which are compared with the desired outputs to compute the error between the two. The parameters of the membership functions are then tuned to reduce the error between the desired output and the actual FIS output.

See Also

`anfis` | `evalfis`

More About

- "Neuro-Adaptive Learning and ANFIS" on page 3-2
- "Comparison of anfis and Neuro-Fuzzy Designer Functionality" on page 3-7

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

4

Data Clustering

- "Fuzzy Clustering" on page 4-2
- "Cluster Quasi-Random Data Using Fuzzy C-Means Clustering" on page 4-4
- "Model Suburban Commuting Using Subtractive Clustering" on page 4-8
- "Data Clustering Using the Clustering Tool" on page 4-20

www.MatlabSite.com | متلب سایت

مرجع آموزش برنامه‌نویسی متلب در ایران

Fuzzy Clustering

In this section...

"What Is Data Clustering?" on page 4-2
"Fuzzy C-Means Clustering" on page 4-2
"Subtractive Clustering" on page 4-3
"References" on page 4-3

What Is Data Clustering?

Clustering of numerical data forms the basis of many classification and system modeling algorithms. The purpose of clustering is to identify natural groupings of data from a large data set to produce a concise representation of a system's behavior.

Fuzzy Logic Toolbox tools allow you to find clusters in input-output training data. You can use the cluster information to generate a Sugeno-type fuzzy inference system that best models the data behavior using a minimum number of rules. The rules partition themselves according to the fuzzy qualities associated with each of the data clusters. Use the command-line function, `genfis2` to automatically accomplish this type of FIS generation.

Fuzzy C-Means Clustering

Fuzzy c-means (FCM) is a data clustering technique wherein each data point belongs to a cluster to some degree that is specified by a membership grade. This technique was originally introduced by Jim Bezdek in 1981[1] as an improvement on earlier clustering methods. It provides a method that shows how to group data points that populate some multidimensional space into a specific number of different clusters.

Fuzzy Logic Toolbox command line function `fcm` starts with an initial guess for the cluster centers, which are intended to mark the mean location of each cluster. The initial guess for these cluster centers is most likely incorrect. Additionally, `fcm` assigns every data point a membership grade for each cluster. By iteratively updating the cluster centers and the membership grades for each data point, `fcm` iteratively moves the cluster centers to the right location within a data set. This iteration is based on minimizing an objective function that represents the distance from any given data point to a cluster center weighted by that data point's membership grade.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The command line function `fcm` outputs a list of cluster centers and several membership grades for each data point. You can use the information returned by `fcm` to help you build a fuzzy inference system by creating membership functions to represent the fuzzy qualities of each cluster.

Subtractive Clustering

If you do not have a clear idea how many clusters there should be for a given set of data, *Subtractive clustering*, [2], is a fast, one-pass algorithm for estimating the number of clusters and the cluster centers in a set of data. The cluster estimates, which are obtained from the `subclust` function, can be used to initialize iterative optimization-based clustering methods (`fcm`) and model identification methods (like `anfis`). The `subclust` function finds the clusters by using the subtractive clustering method.

The `genfis2` function builds upon the `subclust` function to provide a fast, one-pass method to take input-output training data and generate a Sugeno-type fuzzy inference system that models the data behavior.

References

[1] Bezdec, J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms,* Plenum Press, New York, 1981.

[2] Chiu, S., "Fuzzy Model Identification Based on Cluster Estimation," *Journal of Intelligent & Fuzzy Systems*, Vol. 2, No. 3, Spet. 1994.

Related Examples

- "Cluster Quasi-Random Data Using Fuzzy C-Means Clustering" on page 4-4
- "Model Suburban Commuting Using Subtractive Clustering" on page 4-8
- "Data Clustering Using the Clustering Tool" on page 4-20

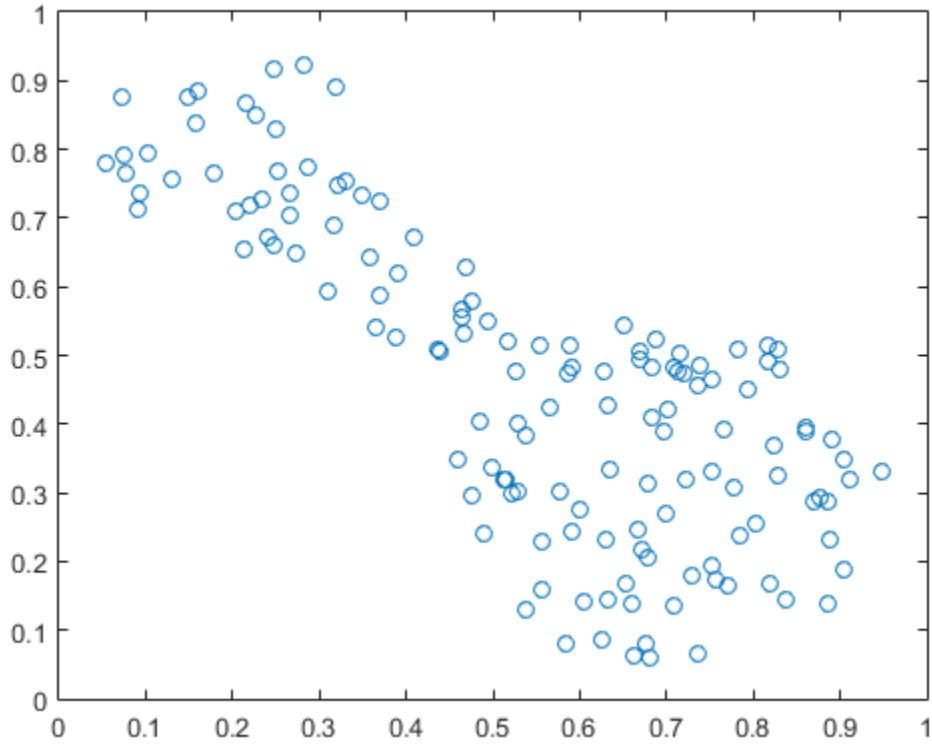
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Cluster Quasi-Random Data Using Fuzzy C-Means Clustering

This example shows how FCM clustering works using quasi-random two-dimensional data.

Load the data set and plot it.

```
load fcmdata.dat
plot(fcmdata(:,1),fcmdata(:,2),'o')
```



Next, invoke the command-line function, `fcm`, to find two clusters in this data set until the objective function is no longer decreasing much at all.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
[center,U,objFcn] = fcm(fcmdata,2);
```

```
Iteration count = 1, obj. fcn = 8.970479
Iteration count = 2, obj. fcn = 7.197402
Iteration count = 3, obj. fcn = 6.325579
Iteration count = 4, obj. fcn = 4.586142
Iteration count = 5, obj. fcn = 3.893114
Iteration count = 6, obj. fcn = 3.810804
Iteration count = 7, obj. fcn = 3.799801
Iteration count = 8, obj. fcn = 3.797862
Iteration count = 9, obj. fcn = 3.797508
Iteration count = 10, obj. fcn = 3.797444
Iteration count = 11, obj. fcn = 3.797432
Iteration count = 12, obj. fcn = 3.797430
```

`center` contains the coordinates of the two cluster centers, `U` contains the membership grades for each of the data points, and `objFcn` contains a history of the objective function across the iterations.

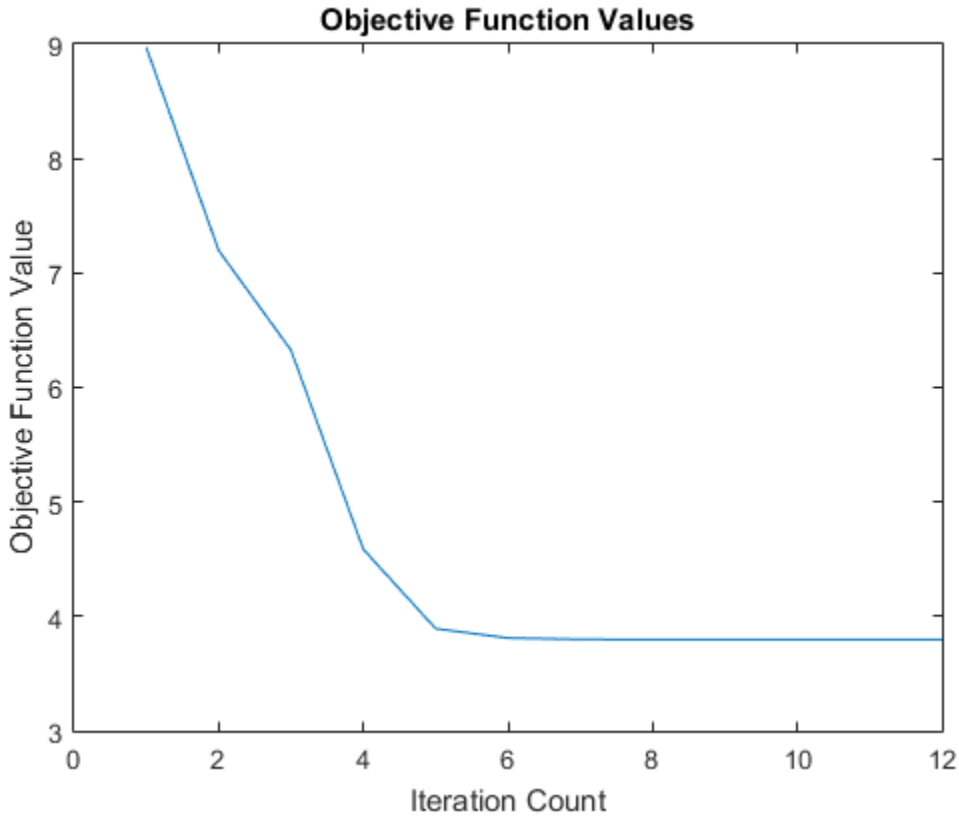
The `fcm` function is an iteration loop built on top of the following routines:

- `initfcm` - initializes the problem
- `distfcm` - performs Euclidean distance calculation
- `stepfcm` - performs one iteration of clustering

To view the progress of the clustering, plot the objective function.

```
figure
plot(objFcn)
title('Objective Function Values')
xlabel('Iteration Count')
ylabel('Objective Function Value')
```

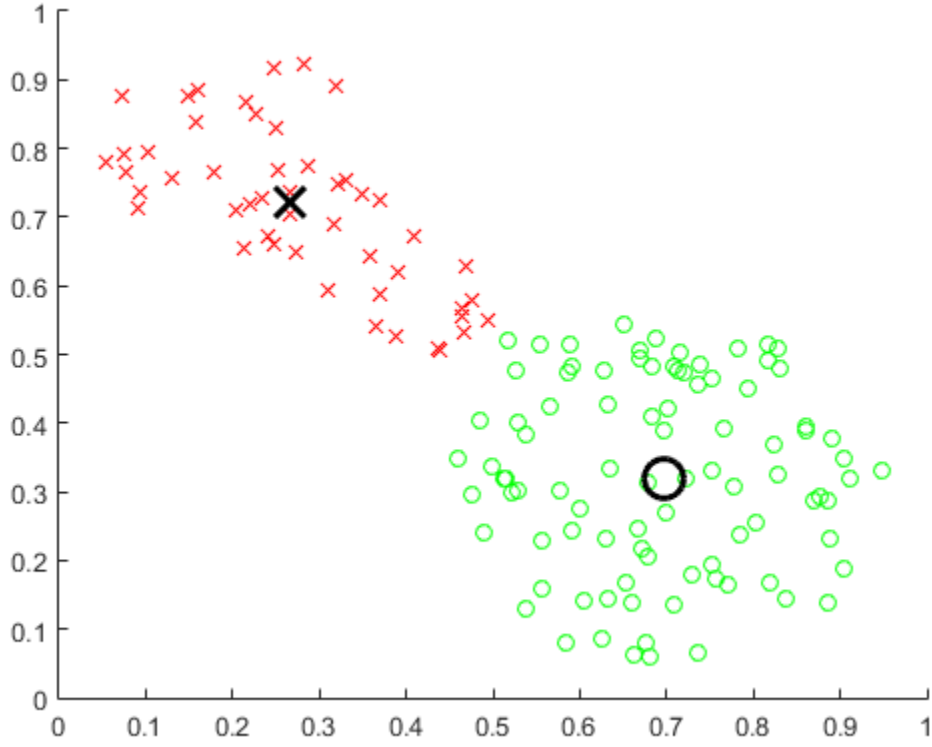

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Finally, plot the two cluster centers found by the `fcm` function. The large characters in the plot indicate cluster centers.

```
maxU = max(U);
index1 = find(U(1,:) == maxU);
index2 = find(U(2,:) == maxU);
figure
line(fcmdata(index1,1), fcmdata(index1,2), 'linestyle',...
                        'none','marker', 'o','color','g');
line(fcmdata(index2,1),fcmdata(index2,2),'linestyle',...
                        'none','marker', 'x','color','r');
hold on
plot(center(1,1),center(1,2),'ko','markersize',15,'LineWidth',2)
plot(center(2,1),center(2,2),'kx','markersize',15,'LineWidth',2)
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Note: Every time you run this example, the `fcm` function initializes with different initial conditions. This behavior swaps the order in which the cluster centers are computed and plotted.

More About

- "Fuzzy Clustering" on page 4-2

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

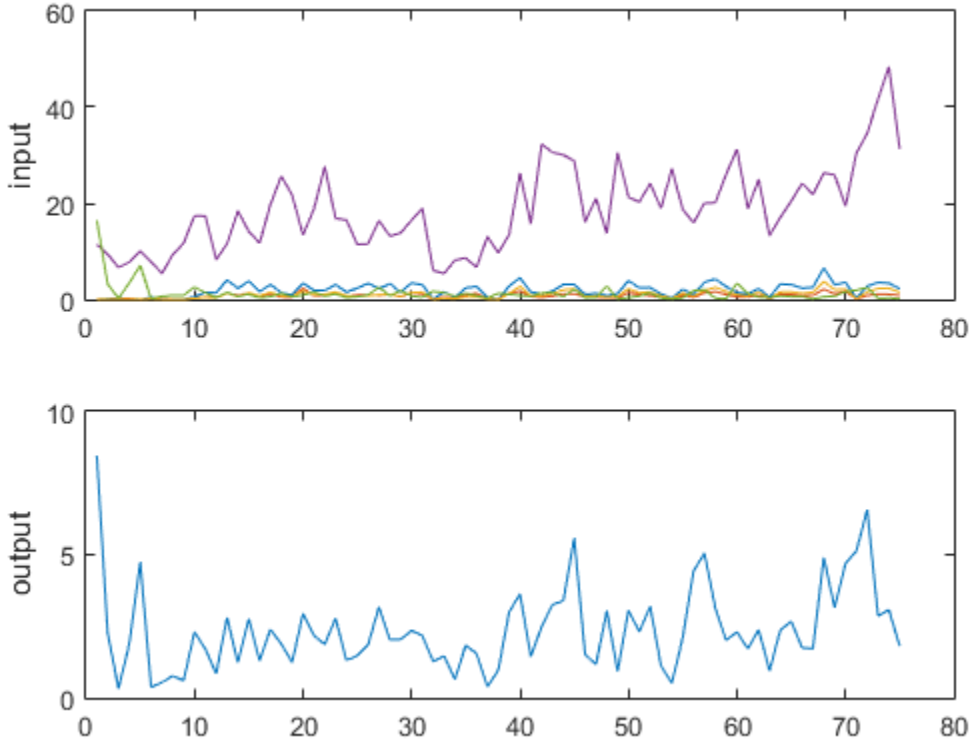
Model Suburban Commuting Using Subtractive Clustering

This example shows how to apply the `genfis2` function to model the relationship between the number of automobile trips generated from an area and the area's demographics. Demographic and trip data are from 100 traffic analysis zones in New Castle County, Delaware. Five demographic factors are considered: population, number of dwelling units, vehicle ownership, median household income, and total employment. Hence, the model has five input variables and one output variable.

Load and plot the data.

```
mytripdata
subplot(2,1,1); plot(datin); ylabel('input');
subplot(2,1,2); plot(datout);ylabel('output');
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



The `mytripdata` function creates several variables in the workspace. Of the original 100 data points, use 75 data points as training data (`datin` and `datout`)and 25 data points as checking data, (as well as for test data to validate the model). The checking data input/output pairs are denoted by `chkdatin` and `chkdatout`.

Use the `genfis2` function to generate a model from data using clustering. `genfis2` requires you to specify a cluster radius. The cluster radius indicates the range of influence of a cluster when you consider the data space as a unit hypercube. Specifying a small cluster radius usually yields many small clusters in the data, and results in many rules. Specifying a large cluster radius usually yields a few large clusters in the data, and results in fewer rules. The cluster radius is specified as the third argument of `genfis2`. The following code calls the `genfis2` function using a cluster radius of 0.5.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
fismat=genfis2(datin,datout,0.5);
```

The genfis2 function is a fast, one-pass method that does not perform any iterative optimization. A FIS structure is returned; the model type for the FIS structure is a first order Sugeno model with three rules.

Use the following commands to verify the model. Here, trnRMSE is the root mean square error of the system generated by the training data.

```
fuzout=evalfis(datin,fismat);
trnRMSE=norm(fuzout-datout)/sqrt(length(fuzout))
```

```
trnRMSE =

    0.5276
```

Next, apply the test data to the FIS to validate the model. In this example, the checking data is used for both checking and testing the FIS parameters. Here, chkRMSE is the root mean square error of the system generated by the checking data.

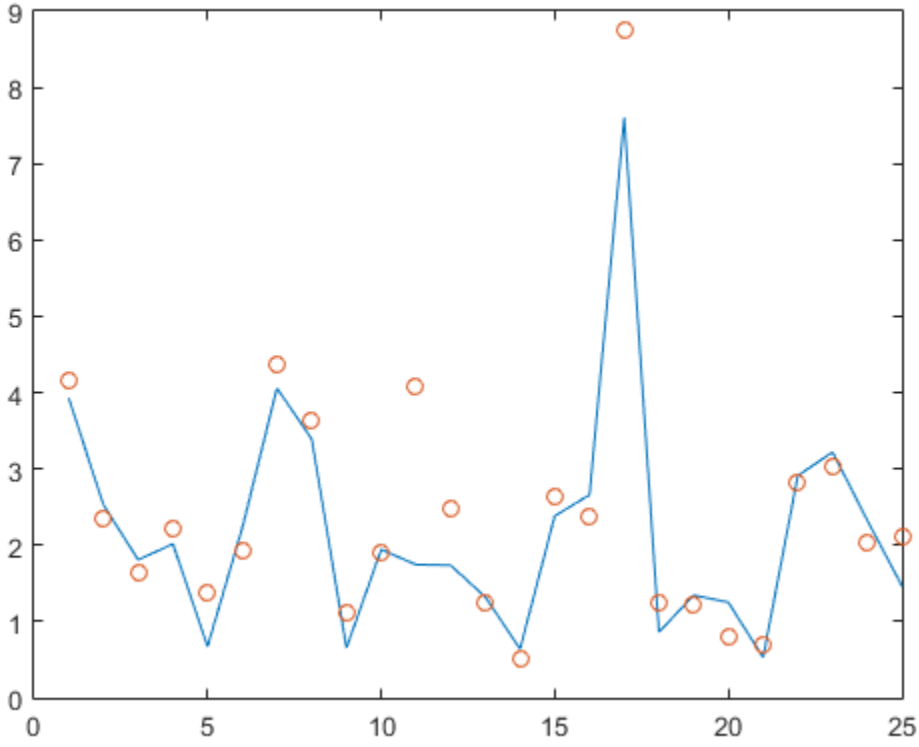
```
chkfuzout=evalfis(chkdatin,fismat);
chkRMSE=norm(chkfuzout-chkdatout)/sqrt(length(chkfuzout))
```

```
chkRMSE =

    0.6179
```

Plot the output of the model, chkfuzout, against the checking data, chkdatout.

```
figure
plot(chkdatout)
hold on
plot(chkfuzout,'o')
hold off
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



The model output and checking data are shown as circles and solid blue line, respectively. The plot shows that the model does not perform well on the checking data.

At this point, you can use the optimization capability of `anfis` to improve the model. First, try using a relatively short anfis training (20 epochs) without implementing the checking data option, and then test the resulting FIS model against the testing data. To perform the optimization, type the following command:

```
fismat2=anfis([datin datout],fismat,[20 0 0.1])
```

```
ANFIS info:
 Number of nodes: 44
 Number of linear parameters: 18
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
 Number of nonlinear parameters: 30
 Total number of parameters: 48
 Number of training data pairs: 75
 Number of checking data pairs: 0
 Number of fuzzy rules: 3


Start training ANFIS ...

   1   0.527607
   2   0.513727
   3   0.492996
   4   0.499985
   5   0.490585
   6   0.492924
   7   0.48733
Step size decreases to 0.090000 after epoch 7.
   8   0.485036
   9   0.480813
  10   0.475097
Step size increases to 0.099000 after epoch 10.
  11   0.469759
  12   0.462516
  13   0.451177
  14   0.447856
Step size increases to 0.108900 after epoch 14.
  15   0.444356
  16   0.433904
  17   0.433739
  18   0.420408
Step size increases to 0.119790 after epoch 18.
  19   0.420512
  20   0.420275

Designated epoch number reached --> ANFIS training completed at epoch 20.


fismat2 =

            name: 'sug51'
            type: 'sugeno'
       andMethod: 'prod'
        orMethod: 'probor'
    defuzzMethod: 'wtaver'
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
       impMethod: 'prod'
       aggMethod: 'max'
           input: [1x5 struct]
          output: [1x1 struct]
            rule: [1x3 struct]
```

Here, 20 is the number of epochs, 0 is the training error goal, and 0.1 is the initial step size.

After the training is done, validate the model.

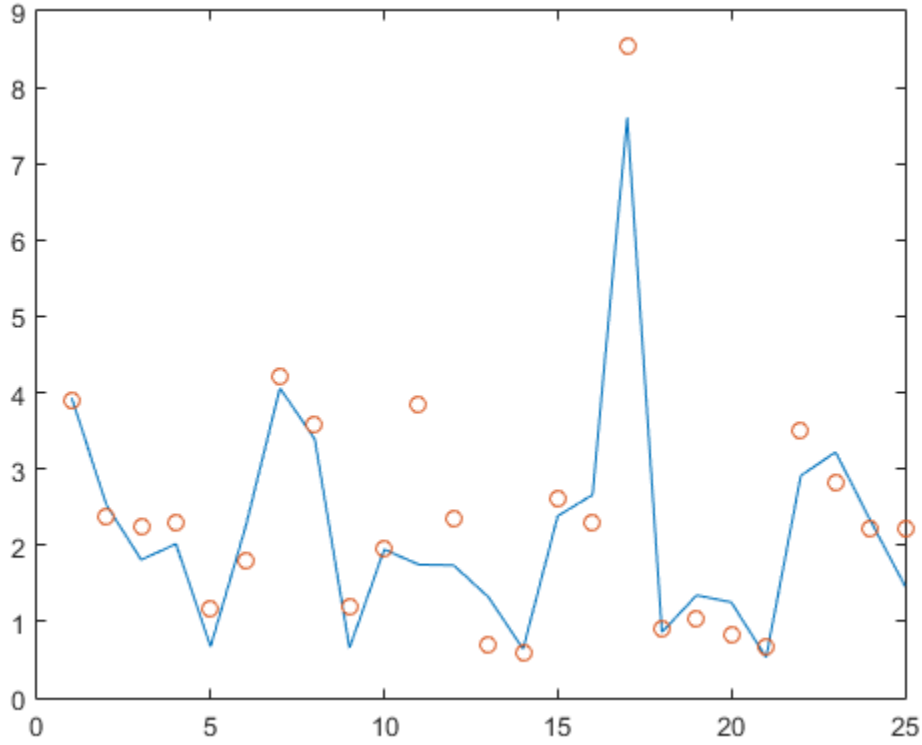
```
fuzout2=evalfis(datin,fismat2);
trnRMSE2=norm(fuzout2-datout)/sqrt(length(fuzout2))
chkfuzout2=evalfis(chkdatin,fismat2);
chkRMSE2=norm(chkfuzout2-chkdatout)/sqrt(length(chkfuzout2))
```

```
trnRMSE2 =

    0.4203


chkRMSE2 =

    0.5894
```

The model has improved a lot with respect to the training data, but only a little with respect to the checking data. Plot the improved model output obtained using `anfis` against the testing data.

```
figure
plot(chkdatout)
hold on
plot(chkfuzout2,'o')
hold off
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



The model output and checking data are shown as circles and solid blue line, respectively. This plot shows that `genfis2` can be used as a stand-alone, fast method for generating a fuzzy model from data, or as a preprocessor to `anfis` for determining the initial rules. An important advantage of using a clustering method to find rules is that the resultant rules are more tailored to the input data than they are in a FIS generated without clustering. This reduces the problem of an excessive propagation of rules when the input data has a high dimension.

Overfitting Overfitting can be detected when the checking error starts to increase while the training error continues to decrease.

To check the model for overfitting, use `anfis` with the checking data option to train the model for 200 epochs. Here, `fismat3` is the FIS structure when the training error

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

reaches a minimum. `fismat4` is the snapshot FIS structure taken when the checking data error reaches a minimum.

```
[fismat3,trnErr,stepSize,fismat4,chkErr]= ...
   anfis([datin datout],fismat,[200 0 0.1],[0 0 0 0], ...
   [chkdatin chkdatout]);
```

This command returns a list of output arguments. The output arguments show a history of the step sizes, the RMSE using the training data, and the RMSE using the checking data for each training epoch.

After the training completes, validate the model.

```
fuzout4=evalfis(datin,fismat4);
trnRMSE4=norm(fuzout4-datout)/sqrt(length(fuzout4))
chkfuzout4=evalfis(chkdatin,fismat4);
chkRMSE4=norm(chkfuzout4-chkdatout)/sqrt(length(chkfuzout4))
```

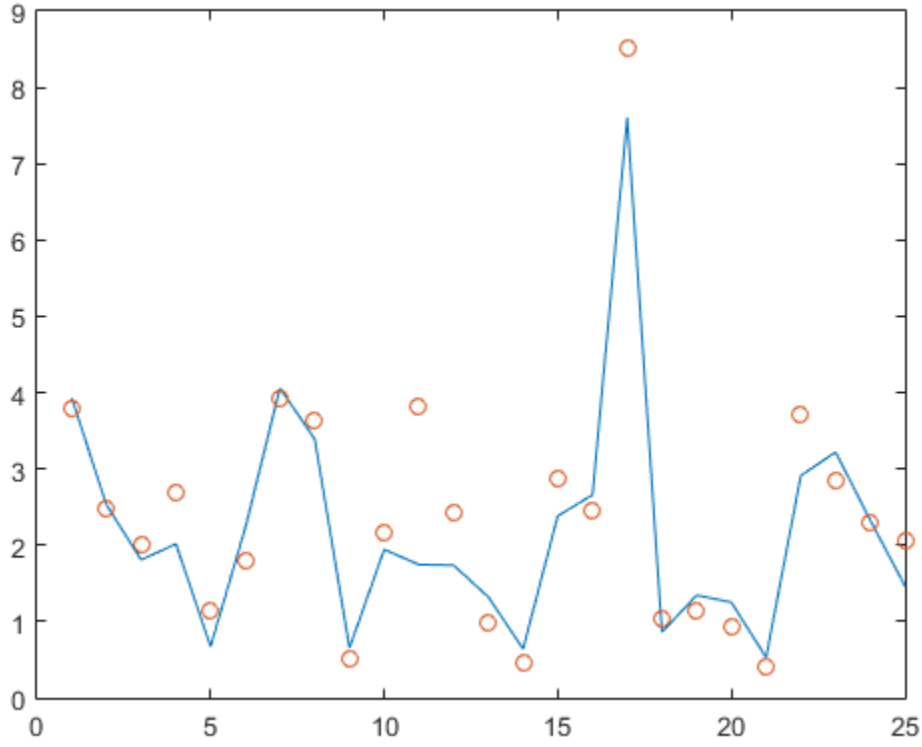
```
trnRMSE4 =

    0.3405


chkRMSE4 =

    0.5820
```

The error with the training data is the lowest thus far, and the error with the checking data is also slightly lower than before. This result suggests perhaps there is an overfit of the system to the training data. Overfitting occurs when you fit the fuzzy system to the training data so well that it no longer does a very good job of fitting the checking data. The result is a loss of generality.

View the improved model output. Plot the model output against the checking data.

```
figure
plot(chkdatout)
hold on
plot(chkfuzout4,'o')
hold off
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

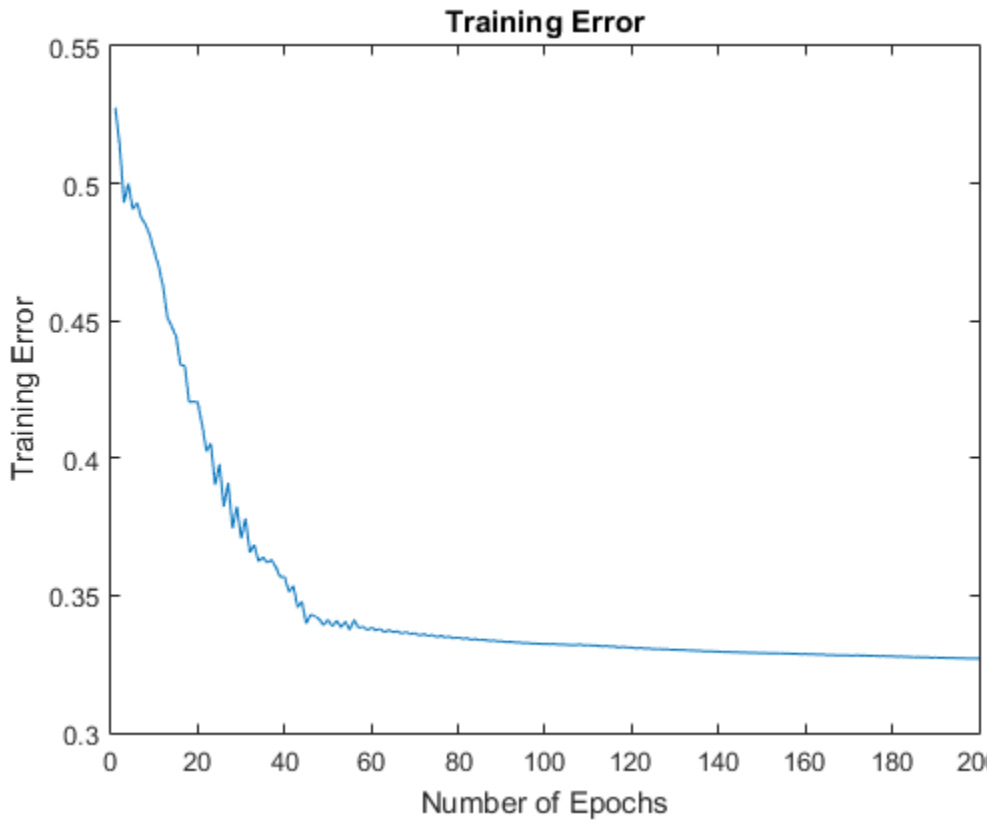


The model output and checking data are shown as circles and solid blue line, respectively.

Next, plot the training error, `trnErr`.

```
figure
plot(trnErr)
title('Training Error')
xlabel('Number of Epochs')
ylabel('Training Error')
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

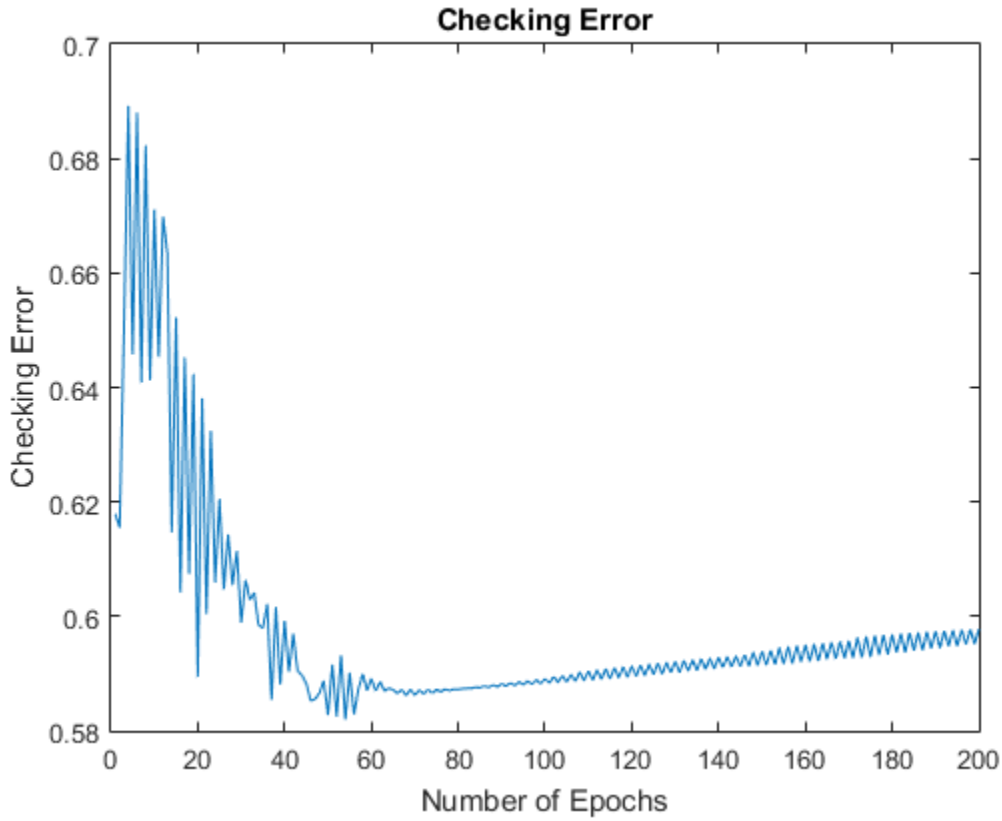


This plot shows that the training error settles at about the 60th epoch point.

Plot the checking error, `chkErr`.

```
figure
plot(chkErr)
title('Checking Error')
xlabel('Number of Epochs')
ylabel('Checking Error')
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

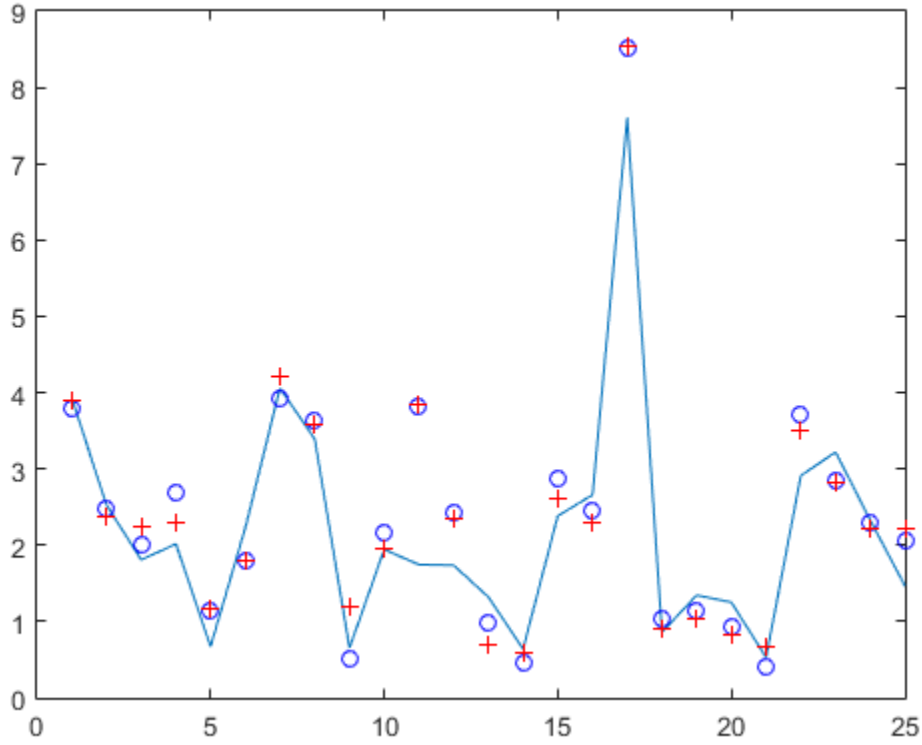


The plot shows that the smallest value of the checking data error occurs at the 52nd epoch, after which it increases slightly even as `anfis` continues to minimize the error against the training data all the way to the 200th epoch. Depending on the specified error tolerance, the plot also indicates the model's ability to generalize the test data.

You can also compare the output of `fismat2` and `fistmat4` against the checking data, `chkdatout`.

```
figure
plot(chkdatout)
hold on
plot(chkfuzout4,'ob')
plot(chkfuzout2,'+r')
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



More About

- "Fuzzy Clustering" on page 4-2

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

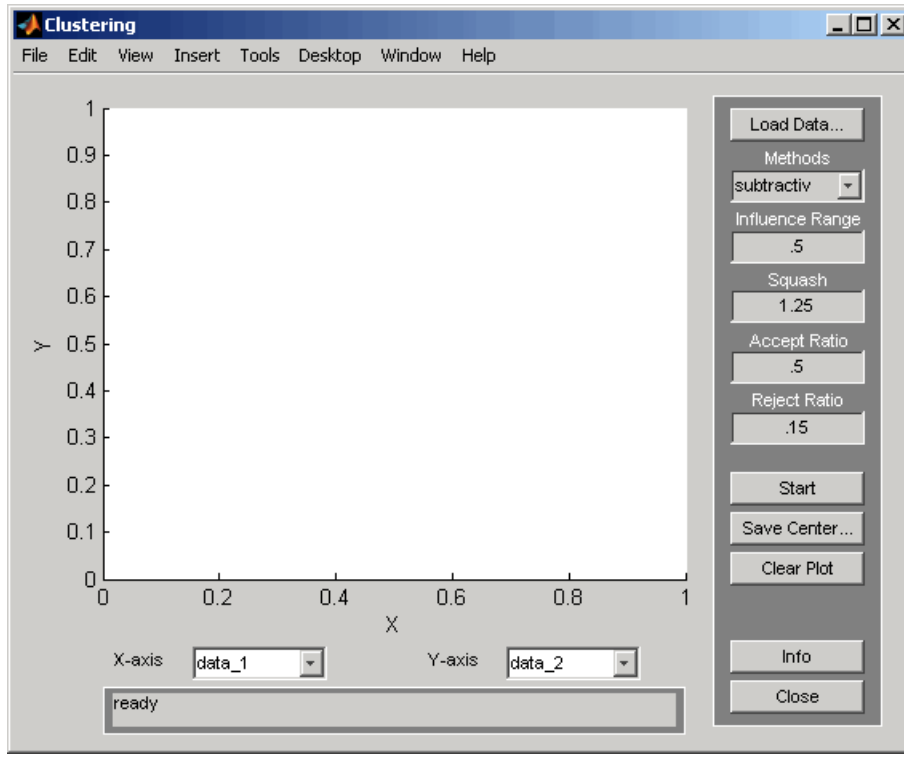
Data Clustering Using the Clustering Tool

The Clustering GUI Tool implements the fuzzy data clustering functions `fcm` and `subclust` and lets you perform clustering on the data. For more information on the clustering functions, see "Fuzzy Clustering" on page 4-2.

To start the GUI, type the following command at the MATLAB command prompt:

```
findcluster
```

The Clustering GUI Tool shown in the next figure.



This GUI lets you perform the following tasks:

1 Load and plot the data.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

2. Start the clustering.
3. Save the cluster center.

Access the online help topics by clicking **Info** or using the **Help** menu in the Clustering GUI.

Loading and Plotting the Data

To load a data set in the GUI, perform either of the following actions:

- Click **Load Data**, and select the file containing the data.
- Open the GUI with a data set directly by invoking `findcluster` with the data set as the argument, in the MATLAB Command Window.

  The data set must have the extension `.dat`. For example, to load the data set, `clusterdemo.dat`, type `findcluster('clusterdemo.dat')`.

The Clustering GUI Tool works on multidimensional data sets, but displays only two of those dimensions on the plot. To select other dimensions in the data set for plotting, you can use the drop-down lists under **X-axis** and **Y-axis**.

Starting the Clustering

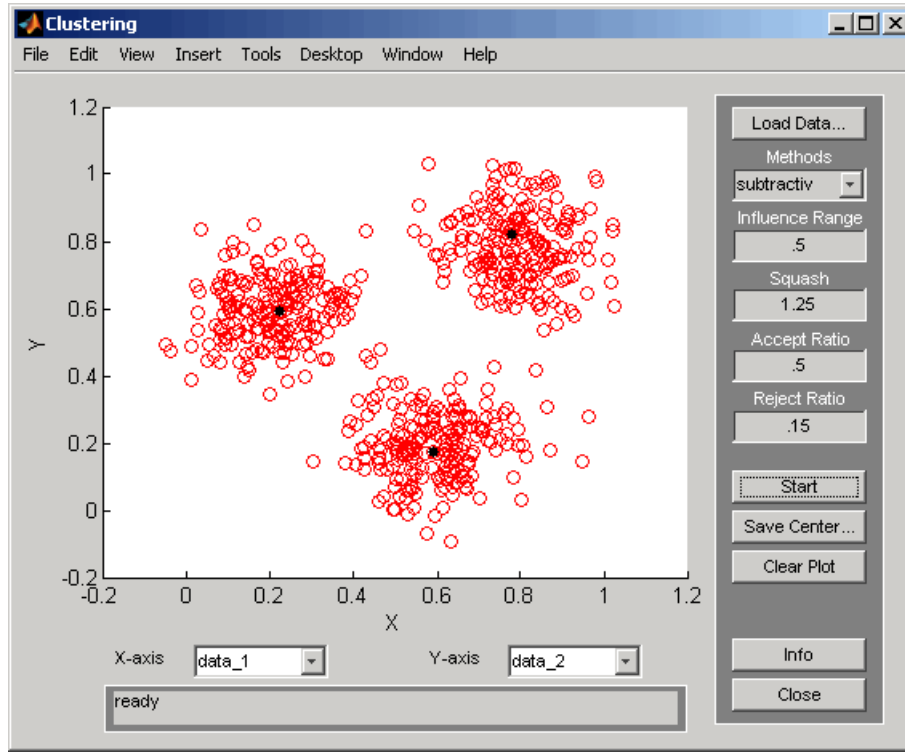
To start clustering the data:

1. Choose the clustering function `fcm` (fuzzy C-Means clustering) or `subtractiv` (subtractive clustering) from the drop-down menu under **Methods**.
2. Set options for the selected method using the **Influence Range**, **Squash**, **Aspect Ratio**, and **Reject Ratio** fields.

   For more information on these methods and their options, refer to `fcm`, and `subclust` respectively.
3. Begin clustering by clicking **Start**.

   After clustering gets completed, the cluster centers appear in black as shown in the next figure.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



Saving the Cluster Center

To save the cluster centers, click **Save Center**.

More About

- "Fuzzy Clustering" on page 4-2

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

5

Functions — Alphabetical List

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

addmf

Add membership function to Fuzzy Inference System

Syntax

```
a = addmf(a,'varType',varIndex,'mfName','mfType',mfParams)
```

Description

A membership function can be added only to a variable in an existing MATLAB workspace FIS. Indices are assigned to membership functions in the order in which they are added, so the first membership function added to a variable is always known as membership function number one for that variable. You cannot add a membership function to input variable number two of a system if only one input has been defined.

The function requires six input arguments in this order:

1. A MATLAB variable name of an FIS structure in the workspace
2. A string representing the type of variable you want to add the membership function to (`'input'` or `'output'`)
3. The index of the variable you want to add the membership function to
4. A string representing the name of the new membership function
5. A string representing the type of the new membership function
6. The vector of parameters that specify the membership function

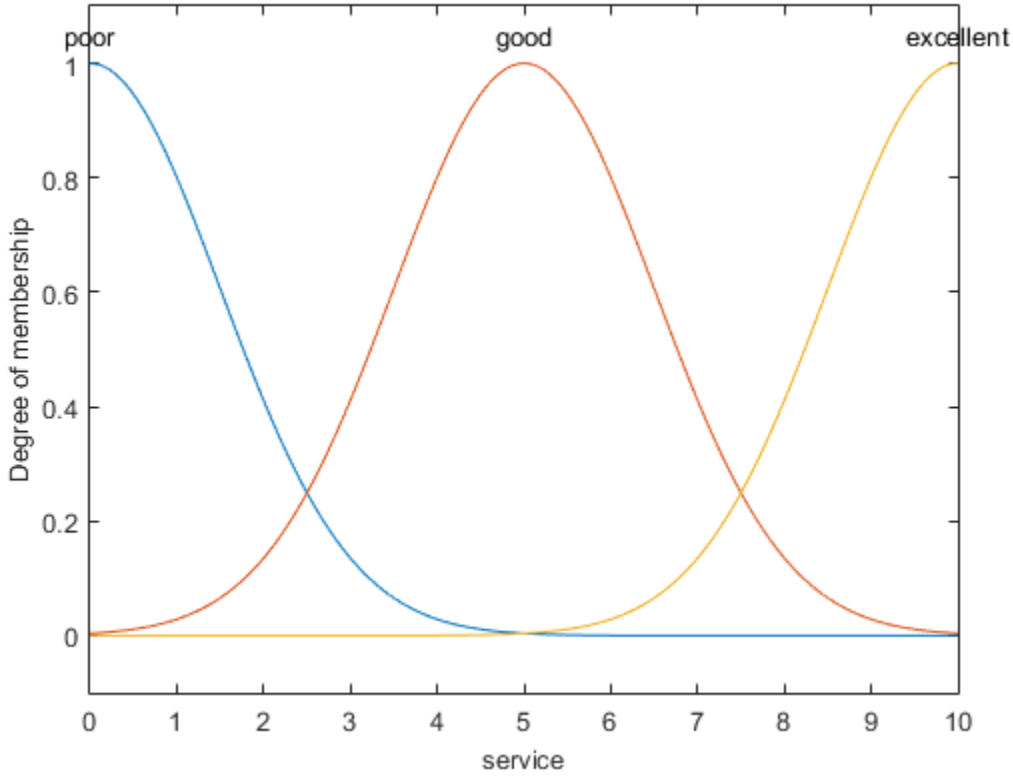
Examples

Add Membership Functions to FIS Inputs

```
a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
plotmf(a,'input',1)
```



See Also

addrule | addvar | plotmf | rmmf | rmvar

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

addrule

Add rule to Fuzzy Inference System

Syntax

```
a = addrule(a,ruleList)
```

Description

`addrule` has two arguments. The first argument is the MATLAB workspace variable FIS name. The second argument is a matrix of one or more rows, each of which represents a given rule. The format that the rule list matrix must take is very specific. If there are m inputs to a system and n outputs, there must be exactly $m + n + 2$ columns to the rule list.

The first m columns refer to the inputs of the system. Each column contains a number that refers to the index of the membership function for that variable.

The next n columns refer to the outputs of the system. Each column contains a number that refers to the index of the membership function for that variable.

The $m + n + 1$ column contains the weight that is to be applied to the rule. The weight must be a number between zero and one and is generally left as one.

The $m + n + 2$ column contains a 1 if the fuzzy operator for the rule's antecedent is AND. It contains a 2 if the fuzzy operator is OR.

Examples

Add Rules to Fuzzy Inference System

Create the fuzzy inference system. For this example, load the `tipper` FIS and clear the existing rules for the FIS.

```
a = readfis('tipper');
a.rule=[];
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Create the rule list.

```
rule1 = [1 1 1 1 1];
rule2 = [1 2 2 1 1];
ruleList = [rule1;rule2];
```

Add the rules list to the FIS.

```
a = addrule(a,ruleList);
```

Verify that the rules were added.

```
showrule(a)
```

```
ans =

1. If (service is poor) and (food is rancid) then (tip is cheap) (1)
2. If (service is poor) and (food is delicious) then (tip is average) (1)
```

Delete Rule from Fuzzy Inference System

Delete the second rule of a Fuzzy Inference System (FIS).

Load the FIS.

```
a = readfis('tipper');
```

View the rules of `a`.

```
showrule(a)
```

```
ans =

1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
2. If (service is good) then (tip is average) (1)
3. If (service is excellent) or (food is delicious) then (tip is generous) (1)
```

Delete the second rule.

```
a.rule(2)=[];
```

Verify that the second rule is deleted.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
showrule(a)
```

```
ans =

1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
2. If (service is excellent) or (food is delicious) then (tip is generous) (1)
```

See Also

addmf | addvar | parsrule | rmmf | rmvar | showrule

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

addvar

Add variable to Fuzzy Inference System

Syntax

```
a = addvar(a,'varType','varName',varBounds)
```

Description

`addvar` has four arguments in this order:

- The name of an FIS structure in the MATLAB workspace
- A string representing the type of the variable you want to add (`'input'` or `'output'`)
- A string representing the name of the variable you want to add
- The vector describing the limiting range values for the variable you want to add

Indices are applied to variables in the order in which they are added, so the first input variable added to a system is always known as input variable number one for that system. Input and output variables are numbered independently.

Examples

Add Variable to Fuzzy Inference System

```
a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
getfis(a,'input',1)
```

```
      Name =     service
      NumMFs =   0
      MFLabels =
      Range =    [0 10]

ans =
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
      Name: 'service'
    NumMFs: 0
     range: [0 10]
```

See Also

addmf | addrule | rmmf | rmvar

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

anfis

Training routine for Sugeno-type Fuzzy Inference System

Syntax

```
fis = anfis(trnData)
fis = anfis(trnData,initFIS)
fis = anfis(trnData,initFIS,trnOpt,dispOpt)
[fis,error] = anfis( ___ )
[fis,error,stepsize] = anfis( ___ )
[fis,error,stepsize,chkFis,chkErr] = anfis(trnData,initFIS,trnOpt,dispOpt,chkData)
[fis,error,stepsize,chkFis,chkErr] = anfis(trnData,initFIS,trnOpt,dispOpt,chkData,optMe
```

Description

`anfis` uses a hybrid learning algorithm to tune the parameters of a Sugeno-type fuzzy inference system (FIS). The algorithm uses a combination of the least-squares and back-propagation gradient descent methods to model a training data set. `anfis` also validates models using a checking data set to test for overfitting of the training data.

Input arguments for `anfis` are:

- `trnData` — Training data, specified as a matrix. For an FIS with `N` inputs, `trnData` has `N+1` columns, where the first `N` columns contain input data and the final column contains output data.
- `initFis` — FIS structure used to provide an initial set of membership functions for training, specified as one of the following:
  - Positive integer — Specifies the number of membership functions for all inputs and generates an initial FIS using `genfis1`.
  - Vector of positive integers — Specifies the number of membership functions for each input individually and generates an initial FIS using `genfis1`.
  - An FIS structure, generated using `genfis1` or `genfis2`, that satisfies these conditions:

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- First or zeroth order Sugeno-type system.
- Single output, obtained using weighted average defuzzification. All output membership functions must be the same type and be either linear or constant.
- No rule sharing. Different rules cannot use the same output membership function; that is the number of output membership functions must be equal to the number of rules.
- Unity weight for each rule.
- No custom membership functions or defuzzification methods.

If `initFis` is not specified, `anfis` uses `genfis1` to create a default initial FIS for training. The default FIS has two Gaussian membership functions for each input.

- `trnOpt` — Training options, specified as a vector of scalars that represent the following settings:

  - `trnOpt(1)` — Training epoch number (default: `10`)
  - `trnOpt(2)` — Training error goal (default: `0`)
  - `trnOpt(3)` — Initial step size (default: `0.01`)
  - `trnOpt(4)` — Step size decrease rate (default: `0.9`)
  - `trnOpt(5)` — Step size increase rate (default: `1.1`)

  When a training option is entered as `NaN`, the default options are used. If the length of `trnOpt` is less than five, the missing elements are set to their default values.

  The training process stops when it reaches the designated epoch number or achieves the training error goal.

- `dispOpt` — Display options that specify information to display in the Command Window during training, specified as a vector of integers that represent these settings:

  - `dispOpt(1)` — ANFIS information, such as numbers of input and output membership functions
  - `dispOpt(2)` — Error values
  - `dispOpt(3)` — Step size at each parameter update
  - `dispOpt(4)` — Final results

  Each display option is specified as:

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- `1` (default) — Display the corresponding information.
- `0` — Do not display the corresponding information.
- `NaN` — The default option is used.

If the length of `dispOpt` is less than four, the missing elements are set to their default values.

- `chkData` — Validation data used to prevent overfitting of the training data, specified as a matrix. This matrix is in the same format as `trnData`. When you supply `chkData` as an input argument, specify `chkFis` and `chkErr` as output arguments to access the validation results.
- `optMethod` — Optimization method used in membership function parameter training, specified as an integer with the following values:

  - `1` (default) — Hybrid method. This method is a combination of least-squares estimation and back-propagation.
  - `0` — Back-propogation method

  If any other value is specified, the default method is used.

Note: To use the default values, you can specify the optional arguments, `initFIS`, `trnOpt`, `dispOpt`, `chkData`, and `optMethod` as empty, `[]`.

Output arguments for `anfis` are:

- `fis` — FIS structure whose parameters are tuned using the training data, returned as a structure.
- `error` — Root mean squared training data errors at each training epoch, returned as an array of scalars.
- `stepsize` — Step sizes at each training epoch, returned as an array of scalars. If the error measure undergoes two consecutive combinations of an increase followed by a decrease, then `anfis` scales the step size by the decrease rate, `trnOpt(4)`. If the error measure undergoes four consecutive decreases, then `anfis` scales the step size by the increase rate, `trnOpt(5)`.
- `chkFis` — FIS structure that corresponds to the epoch at which `chkErr` is minimum. The function returns `chkFis` only when you supply `chkData` as an input argument.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- `chkErr` — Root mean squared checking data errors at each training epoch, returned as an array of scalars. The function returns `chkErr` only when you supply `chkData` as an input argument.

Examples

Train ANFIS with Custom Number of Training Epochs

Create single-input-single-output training data.

```
x = (0:0.1:10)';
y = sin(2*x)./exp(x/5);
trnData = [x y];
```

Define an FIS structure with five bell-shaped input membership functions.

```
numMFs = 5;
mfType = 'gbellmf';
in_fis = genfis1(trnData,numMFs,mfType);
```

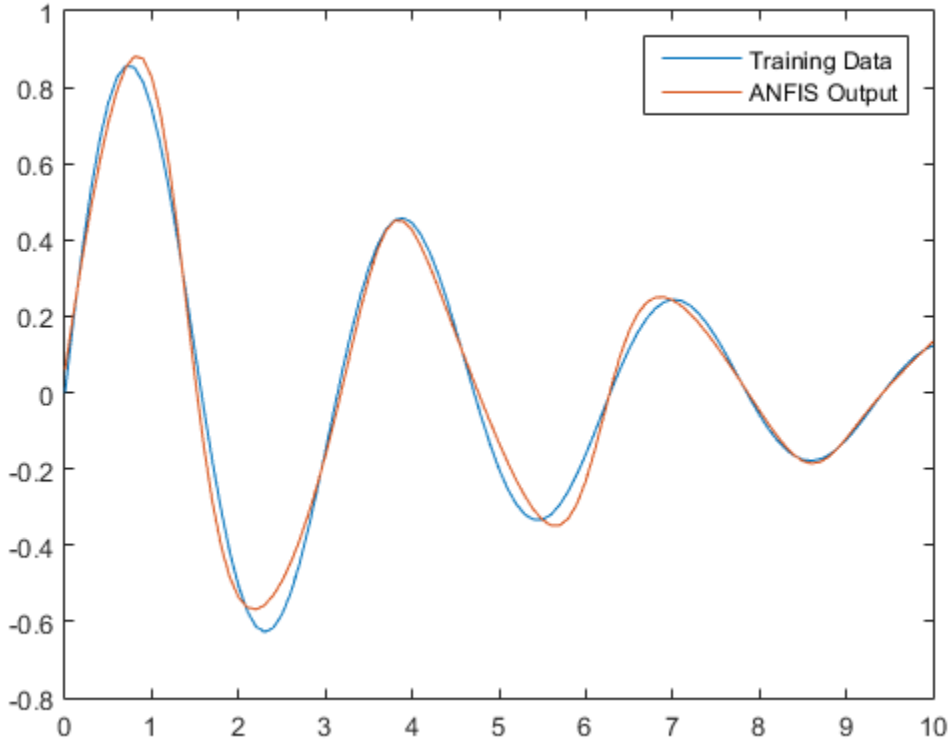
Train the FIS using 20 training epochs. Suppress the Command Window display.

```
epoch_n = 20;
dispOpt = zeros(1,4);
out_fis = anfis(trnData,in_fis,20,dispOpt);
```

Compare the ANFIS output with the training data.

```
plot(x,y,x,evalfis(x,out_fis));
legend('Training Data','ANFIS Output');
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



- "Predict Chaotic Time-Series"
- "Modeling Inverse Kinematics in a Robotic Arm"

More About

- "Neuro-Adaptive Learning and ANFIS"
- "Comparison of anfis and Neuro-Fuzzy Designer Functionality"

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

References

Jang, J.-S. R., “Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm,” *Proc. of the Ninth National Conf. on Artificial Intelligence (AAAI-91)*, pp. 762-767, July 1991.

Jang, J.-S. R., “ANFIS: Adaptive-Network-based Fuzzy Inference Systems,” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, pp. 665-685, May 1993.

See Also

`genfis1` | `genfis2`

Introduced before R2006a

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

convertfis

Convert Fuzzy Logic Toolbox Version 1.0 Fuzzy Inference System matrix to current-version Fuzzy Inference System structure

Syntax

```
fis_new=convertfis(fis_old)
```

Description

`convertfis` takes a Version 1.0 FIS matrix and converts it to an FIS structure compatible with the current version.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

defuzz

Defuzzify membership function

Syntax

```
out = defuzz(x,mf,type)
```

Description

`defuzz(x,mf,type)` returns a defuzzified value `out`, of a membership function `mf` positioned at associated variable value `x`, using one of several defuzzification strategies, according to the argument, `type`. The variable `type` can be one of the following:

- `centroid`: centroid of area
- `bisector`: bisector of area
- `mom`: mean value of maximum
- `som`: smallest (absolute) value of maximum
- `lom`: largest (absolute) value of maximum

If `type` is not one of those listed, Fuzzy Logic Toolbox software assumes it to be a user-defined function. `x` and `mf` are passed to this function to generate the defuzzified output.

Examples

Obtain Defuzzified Value

```
x = -10:0.1:10;
mf = trapmf(x,[-10 -8 -4 7]);
out = defuzz(x,mf,'centroid')
```

```
out =

   -3.2857
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

dsigmf

Difference between two sigmoidal functions membership function

Syntax

```
y = dsigmf(x,[a1 c1 a2 c2])
```

Description

The sigmoidal membership function used depends on the two parameters a and c and is given by

$$f(x;a,c)=\frac{1}{1+e^{-a(x-c)}}$$

The membership function `dsigmf` depends on four parameters, `a1`, `c1`, `a2`, and `c2`, and is the difference between two of these sigmoidal functions.

$f_1(x;\ a_1,\ c_1)$ - $f_2(x;\ a_2,\ c_2)$

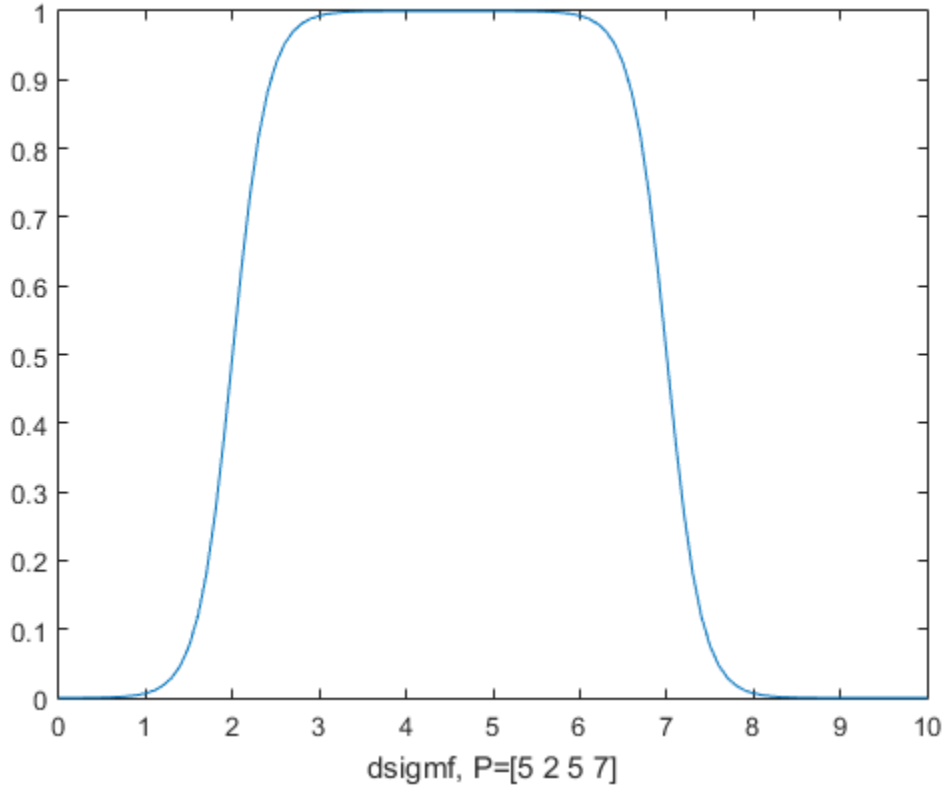
The parameters are listed in the order: $[a_1\ c_1\ a_2\ c_2]$.

Examples

Obtain Difference of Two Sigmoidal Functions

```
x=0:0.1:10;
y=dsigmf(x,[5 2 5 7]);
plot(x,y)
xlabel('dsigmf, P=[5 2 5 7]')
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`gaussmf` | `gauss2mf` | `gbellmf` | `evalmf` | `mf2mf` | `pimf` | `psigmf` | `sigmf` | `smf` | `trapmf` | `trimf` | `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

evalfis

Perform fuzzy inference calculations

Syntax

```
output= evalfis(input,fismat)
```

```
output= evalfis(input,fismat, numPts)
```

```
[output, IRR, ORR, ARR]= evalfis(input,fismat)
```

```
[output, IRR, ORR, ARR]= evalfis(input,fismat,numPts)
```

Description

`evalfis` has the following arguments:

- `input`: a number or a matrix specifying input values. If `input` is an M-by-N matrix, where N is number of input variables, then `evalfis` takes each row of `input` as an input vector and returns the M-by-L matrix to the variable, `output`, where each row is an output vector and L is the number of output variables.
- `fismat`: an FIS structure to be evaluated.
- `numPts`: an optional argument that represents the number of sample points on which to evaluate the membership functions over the input or output range. If this argument is not used, the default value of 101 points is used.

The range labels for `evalfis` are as follows:

- `output`: the output matrix of size M-by-L, where M represents the number of input values specified previously, and L is the number of output variables for the FIS.

The optional range variables for `evalfis` are only calculated when the `input` argument is a row vector, (only one set of inputs is applied). These optional range variables are

- `IRR`: the result of evaluating the input values through the membership functions. This matrix is of the size *numRules*-by-*N*, where *numRules* is the number of rules, and *N* is the number of input variables.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- `ORR`: the result of evaluating the output values through the membership functions. This matrix is of the size `numPts`-by-*numRules***L*, where *numRules* is the number of rules, and *L* is the number of outputs. The first *numRules* columns of this matrix correspond to the first output, the next *numRules* columns of this matrix correspond to the second output, and so forth.
- `ARR`: the `numPts`-by-*L* matrix of the aggregate values sampled at `numPts` along the output range for each output.

When it is invoked with only one range variable, this function computes the output vector, `output`, of the fuzzy inference system specified by the structure, `fismat`, for the input value specified by the number or matrix, `input`.

Examples

Evaluate FIS

```
fismat = readfis('tipper');
out = evalfis([2 1; 4 9],fismat)
```

```
out =

    7.0169
   19.6810
```

See Also

`ruleview` | `gensurf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

evalmf

Generic membership function evaluation

Syntax

```
y = evalmf(x,mfParams,mfType)
```

Description

`evalmf` evaluates any membership function, where x is the variable range for the membership function evaluation, `mfType` is a membership function from the toolbox, and `mfParams` are appropriate parameters for that function.

If you want to create your own custom membership function, `evalmf` still works, because it evaluates any membership function whose name it does not recognize.

Examples

Evaluate Membership Function

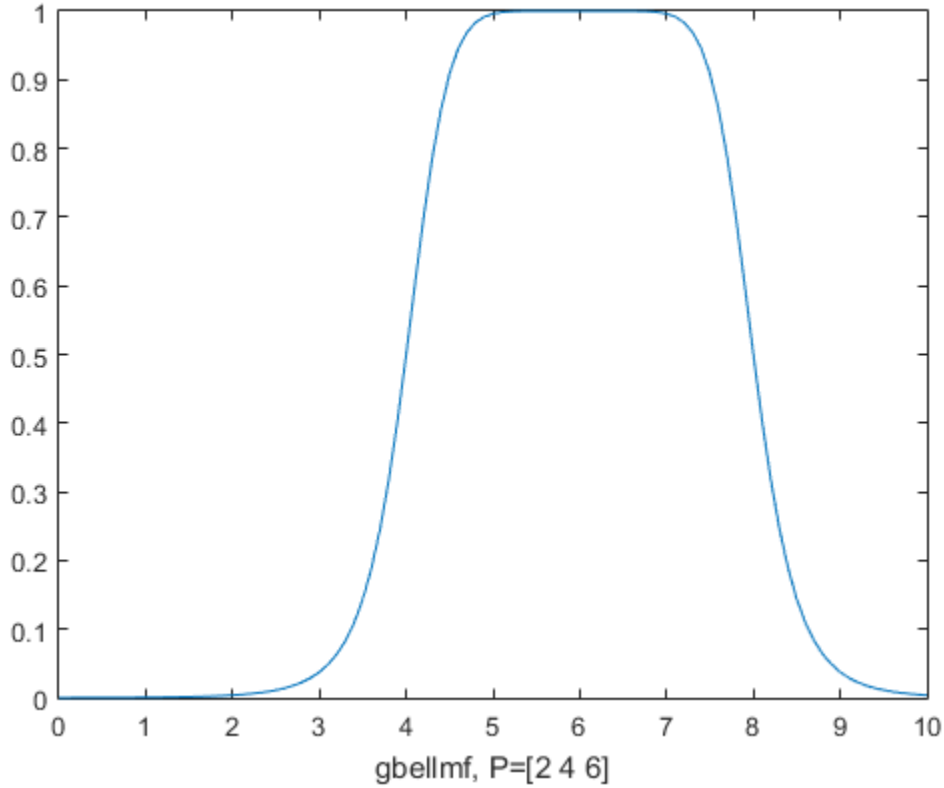
Evaluate a generalized bell-shaped membership function.

```
x=0:0.1:10;
mfparams = [2 4 6];
mftype = 'gbellmf';
y=evalmf(x,mfparams,mftype);
```

Plot the evaluation.

```
plot(x,y)
xlabel('gbellmf, P=[2 4 6]')
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`dsigmf` | `gaussmf` | `gauss2mf` | `gbellmf` | `evalmf` | `mf2mf` | `pimf` | `psigmf` | `sigmf` | `smf` | `trapmf` | `trimf` | `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

fcm

Fuzzy C-means clustering

Syntax

```
[center,U,obj_fcn] = fcm(data,cluster_n)
```

```
[center,U,obj_fcn] = fcm(data,cluster_n,options)
```

Description

`[center, U, obj_fcn] = fcm(data, cluster_n)` applies the fuzzy c-means clustering method to a given data set.

The input arguments of this function are

- `data`: data set to be clustered; each row is a sample data point
- `cluster_n`: number of clusters (greater than one)

The output arguments of this function are

- `center`: matrix of final cluster centers where each row provides the center coordinates
- `U`: final fuzzy partition matrix (or membership function matrix)
- `obj_fcn`: values of the objective function during iterations

`fcm(data,cluster_n,options)` uses an additional argument variable, `options`, to specify clustering parameters, introduce a stopping criteria, or set the iteration information display. Specify `options` as a vector:

- `options(1)`: Exponent for the partition matrix `U`. Default: `2.0`.
- `options(2)`: Maximum number of iterations. Default: `100`.
- `options(3)`: Minimum amount of improvement. Default: `1e-5`.
- `options(4)`: Information displayed during iteration. Default: `1`.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

If any entry of `options` is `NaN`, the default value for that option is used instead. The clustering process stops when the maximum number of iterations is reached or when the objective function improvement between two consecutive iterations is less than the minimum amount of improvement specified.

Examples

Apply Fuzzy C-Means Clustering Method to Data

Apply the Fuzzy C-Means clustering to training data.

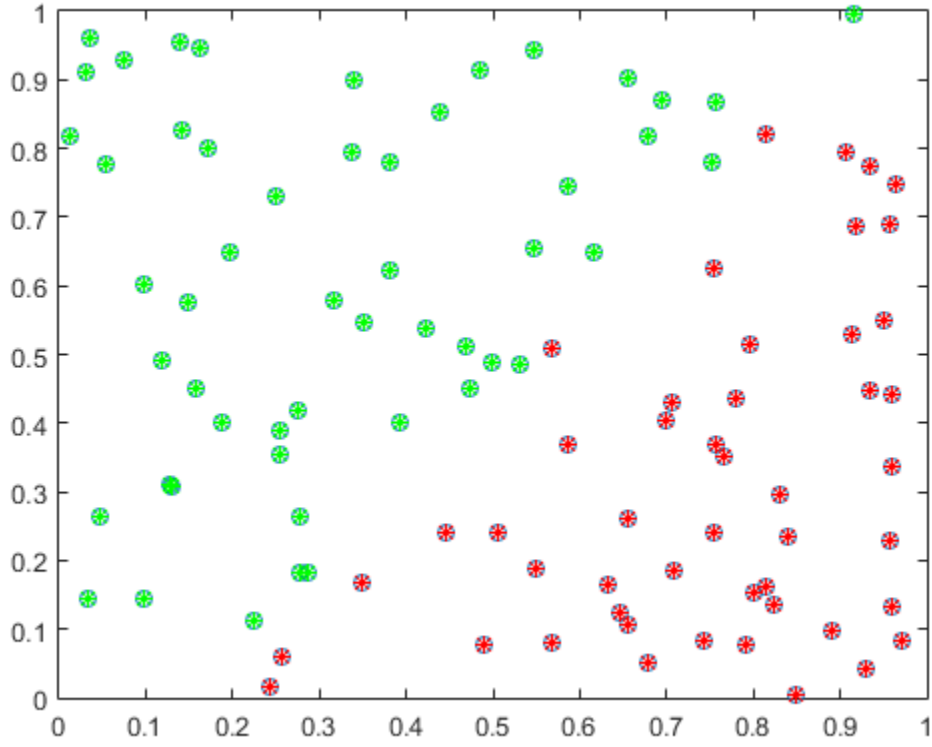
```
data = rand(100,2);
opts = [nan;nan;nan;0];
[center,U,obj_fcn] = fcm(data,2,opts);
```

The fourth entry of `opts` turns off the iteration display. The remaining entries specify the default values for the corresponding options.

Plot the results.

```
plot(data(:,1),data(:,2),'o');
maxU = max(U);
index1 = find(U(1,:)== maxU);
index2 = find(U(2,:)== maxU);
line(data(index1,1),data(index1,2),'linestyle','none',...
     'marker','*','color','g');
line(data(index2,1),data(index2,2),'linestyle','none',...
     'marker', '*','color','r');
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



More About

- "Fuzzy C-Means Clustering"

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

findcluster

Open Clustering tool

Syntax

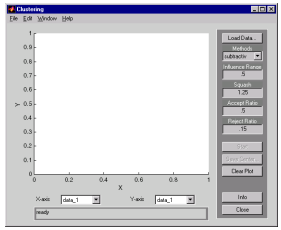
```
findcluster findcluster('file.dat')
```

Description

`findcluster` opens a GUI to implement either the fuzzy c-means (**fcm**), the fuzzy subtractive clustering (**subtractiv**) using the pull-down tab under **Method** on the GUI, or both. Data is entered using the **Load Data** button. The options for each of these methods are set to default values. These default values can be changed. See `fcm` reference page for a description of the options for fuzzy c-means. The `subclust` reference page provides a description of the options for fuzzy subclustering.

This tool works on multidimensional data sets, but only displays two of those dimensions. Use the pull-down tabs under **X-axis** and **Y-axis** to select which data dimension you want to view. For example, if you have data that is five-dimensional, this tool labels the data as data_1, data_2, data_3, data_4, data_5, in the order in which the data appears in the data set. **Start** to perform the clustering, and **Save Center** to save the cluster center.

When operating on a data set called `file.dat`, `findcluster` (`file.dat`) loads the data set automatically, plotting up to the first two dimensions of the data only. You can still choose which two dimensions of the data you want to cluster after the GUI appears.



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Examples

```
findcluster('clusterdemo.dat')
```

See Also

fcm | subclust

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

fuzarith

Perform fuzzy arithmetic

Syntax

```
C = fuzarith(X, A, B, operator)
```

Description

Using interval arithmetic, `C = fuzarith(X, A, B, operator)` returns a fuzzy set `C` as the result of applying the function represented by the string, `operator`, which performs a binary operation on the sampled convex fuzzy sets `A` and `B`. The elements of `A` and `B` are derived from convex functions of the sampled universe, `X`:

- `A`, `B`, and `X` are vectors of the same dimension.
- `operator` is one of the following strings: `'sum'`, `'sub'`, `'prod'`, and `'div'`.
- The returned fuzzy set `C` is a column vector with the same length as `X`.

Note: Fuzzy addition might generate the message `"divide by zero"` but this does not affect the accuracy of this function.

Examples

Perform Fuzzy Arithmetic

Specify Gaussian and Trapezoidal membership functions.

```
N = 101;
minx = -20;
maxx = 20;
x = linspace(minx,maxx,N);

A = trapmf(x,[-10 -2 1 3]);
B = gaussmf(x,[2 5]);
```

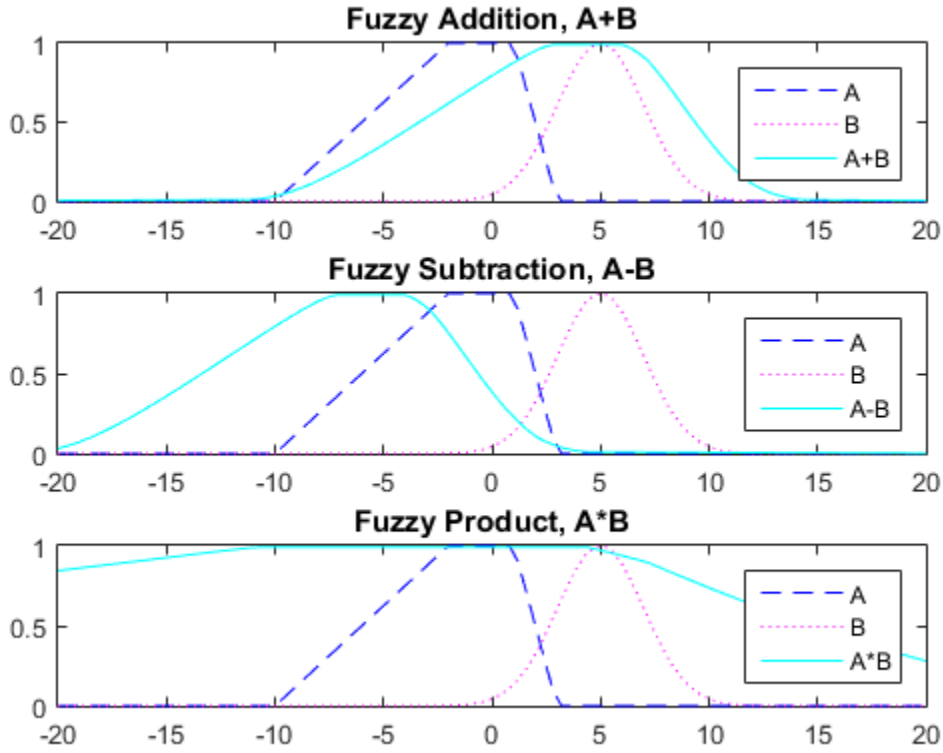
Evaluate the sum, difference, and product of `A` and `B`.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
Csum = fuzarith(x,A,B,'sum');
Csub = fuzarith(x,A,B,'sub');
Cprod = fuzarith(x,A,B,'prod');
```

Plot the results.

```
subplot(3,1,1); plot(x,A,'b--',x,B,'m:',x,Csum,'c');
title('Fuzzy Addition, A+B');
legend('A','B','A+B');
subplot(3,1,2); plot(x,A,'b--',x,B,'m:',x,Csub,'c');
title('Fuzzy Subtraction, A-B');
legend('A','B','A-B');
subplot(3,1,3); plot(x,A,'b--',x,B,'m:',x,Cprod,'c');
title('Fuzzy Product, A*B');
legend('A','B','A*B');
```



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

fuzzyLogicDesigner

Open Fuzzy Logic Designer app

Syntax

```
fuzzyLogicDesigner
fuzzyLogicDesigner(fis)
```

Description

`fuzzyLogicDesigner` opens the Fuzzy Logic Designer app, which allows you to edit the highest level features of the fuzzy inference system, such as the number of input and output variables, the defuzzyfication method, fuzzy implication and fuzzy inference.

`fuzzyLogicDesigner(fis)` opens a prebuilt fuzzy inference system.

The Fuzzy Logic Designer app is the high-level interface for any fuzzy logic inference system (FIS). This interface provides convenient access to all other editors with an emphasis on maximum flexibility for interaction with the fuzzy system.

Input Arguments

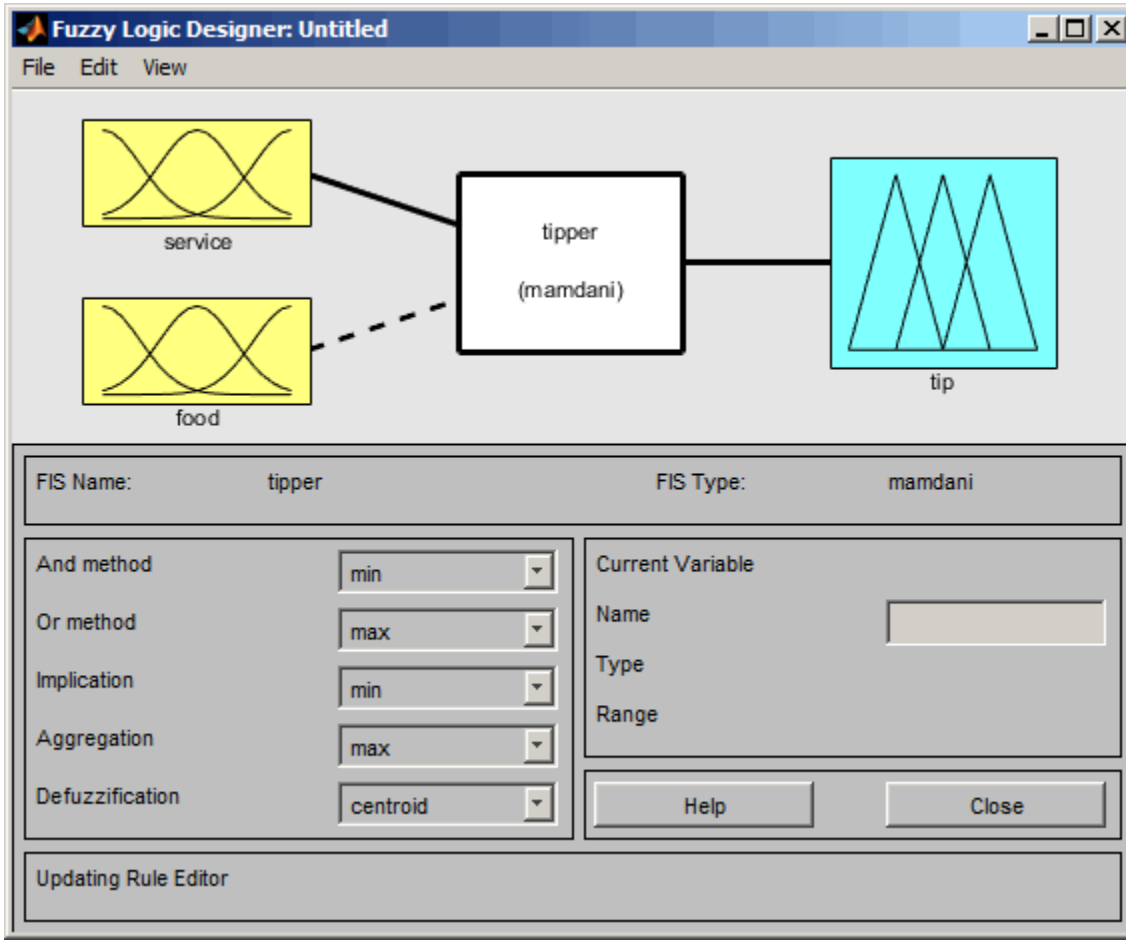
`fis` — Name of prebuilt fuzzy interface system

string

Name of the prebuilt fuzzy interface system, specified as a string. The system must be stored in a `.fis` file or a workplace variable

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The Designer



The Fuzzy Logic Designer shows the inputs, outputs, and a central fuzzy rule processor. Click one of the variable boxes to select a variable to edit. The box is highlighted in red and the Current Variable field displays the default settings of the selected variable. Double-click one of the variables to open the Membership Function Editor. Double-click the fuzzy rule processor to open the Rule Editor. If a variable exists but is not mentioned in the rule base, it is connected to the rule processor block with a dashed line.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Menu Items

The Fuzzy Logic Designer menu bar allows you to open related tools and open and save systems.

- In the **File** menu select

  New FIS > **Mamdani** to open a new `Untitled` Mamdani-style system with no variables and no rules.

  New FIS > **Sugeno** to open a new `Untitled` Sugeno-style system with no variables and no rules.

  Import > **From workspace** to load a system from a specified FIS structure variable in the MATLAB workspace.

  Import > **From file** to load a system from a specified `.fis` file.

  Export > **To workspace.** to save the system to an FIS structure variable in the MATLAB workspace.

  Export > **To file** to save the current system to a `.fis` file.

  Print to print the GUI display.

  Close to close the GUI.
- In the **Edit** menu select

  Undo to undo the most recent change.

  Add variable > **Input** to add another input to the current system.

  Add variable > **Output** to add another output to the current system.

  Remove Selected Variable to delete a selected variable.

  Membership functions to open the Membership Function Editor.

  Rules to open the Rule Editor.
- In the **View** menu select

  Rules to open the Rule Viewer.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Surface to open the Surface Viewer.

Inference Method Pop-up Menus

You can also use pop-up menus to change defuzzyfication, implication and inference settings:

- **And method**: Choose `min`, `prod`, or `Custom`, to specify a custom operation.
- **Or method**: Choose `max`, `probor` (probabilistic or), or `Custom`, for a custom operation.
- **Implication**: Choose `min`, `prod`, or `Custom`, for a custom operation. This selection is not available for Sugeno-style fuzzy inference.
- **Aggregation**: Choose `max`, `sum`, `probor`, or `Custom`, for a custom operation. This selection is not available for Sugeno-style fuzzy inference.
- **Defuzzification**: For Mamdani-style inference, choose `centroid`, `bisector`, `mom` (middle of maximum), `som` (smallest of maximum), `lom` (largest of maximum), or `Custom`, for a custom operation. For Sugeno-style inference, choose between `wtaver` (weighted average) or `wtsum` (weighted sum).

More About

- "The Fuzzy Logic Designer" on page 2-34

See Also

`mfedit` | `ruleedit` | `ruleview` | `surfview` | `neuroFuzzyDesigner`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

gauss2mf

Gaussian combination membership function

Syntax

```
y = gauss2mf(x,[sig1 c1 sig2 c2])
```

Description

The Gaussian function depends on two parameters *sig* and *c* as given by

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

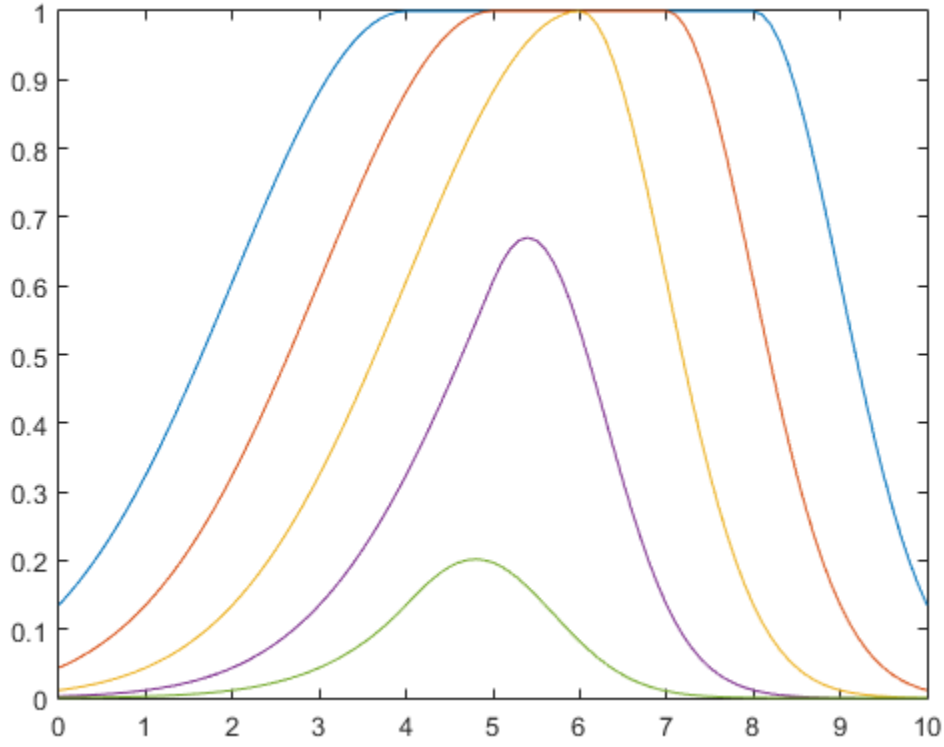
The function `gauss2mf` is a combination of two of these two parameters. The first function, specified by *sig1* and *c1*, determines the shape of the left-most curve. The second function specified by *sig2* and *c2* determines the shape of the right-most curve. Whenever *c1* < *c2*, the `gauss2mf` function reaches a maximum value of 1. Otherwise, the maximum value is less than one. The parameters are listed in the order:
[*sig1*, *c1*, *sig2*, *c2*] .

Examples

Gaussian Combination Membership Functions

```
x = [0:0.1:10]';
y1 = gauss2mf(x,[2 4 1 8]);
y2 = gauss2mf(x,[2 5 1 7]);
y3 = gauss2mf(x,[2 6 1 6]);
y4 = gauss2mf(x,[2 7 1 5]);
y5 = gauss2mf(x,[2 8 1 4]);
plot(x,[y1 y2 y3 y4 y5]);
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`dsigmf` | `gauss2mf` | `gbellmf` | `evalmf` | `mf2mf` | `pimf` | `psigmf` | `sigmf` | `smf` | `trapmf` | `trimf` | `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

gaussmf

Gaussian curve membership function

Syntax

```
y = gaussmf(x,[sig c])
```

Description

The symmetric Gaussian function depends on two parameters σ and c as given by

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

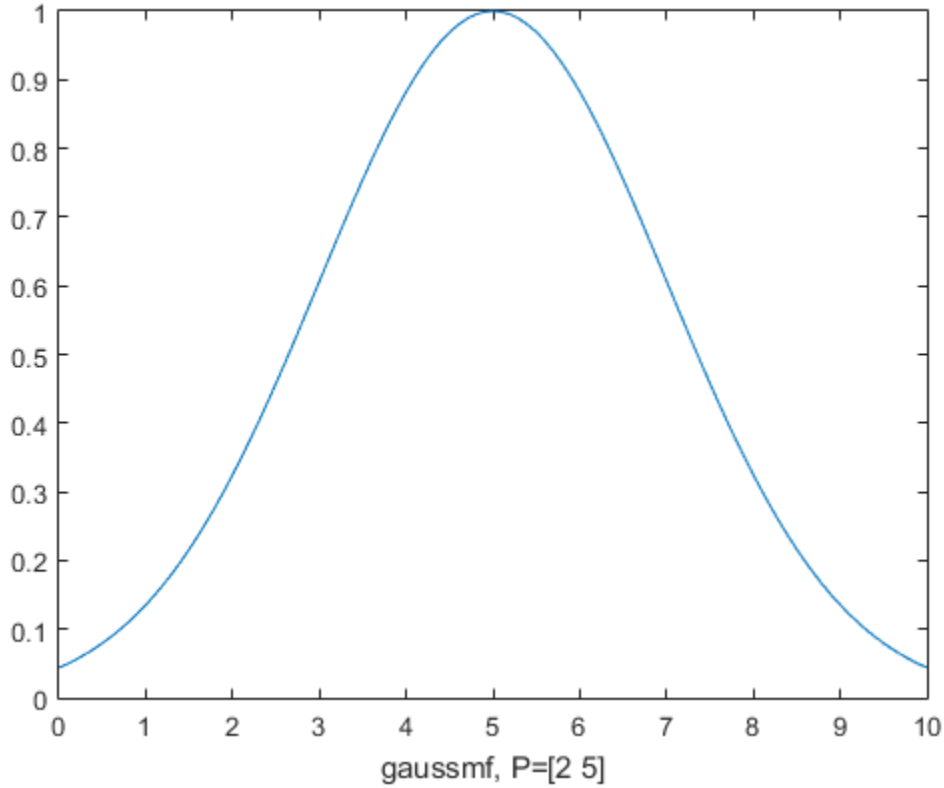
The parameters for gaussmf represent the parameters σ and c listed in order in the vector [`sig c`].

Examples

Gaussian Membership Function

```
x = 0:0.1:10;
y = gaussmf(x,[2 5]);
plot(x,y)
xlabel('gaussmf, P=[2 5]')
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`dsigmf` | `gaussmf` | `gbellmf` | `evalmf` | `mf2mf` | `pimf` | `psigmf` | `sigmf` | `smf` | `trapmf` | `trimf` | `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

gbellmf

Generalized bell-shaped membership function

Syntax

```
y = gbellmf(x,params)
```

Description

The generalized bell function depends on three parameters a, b, and c as given by

$$f(x;a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}$$

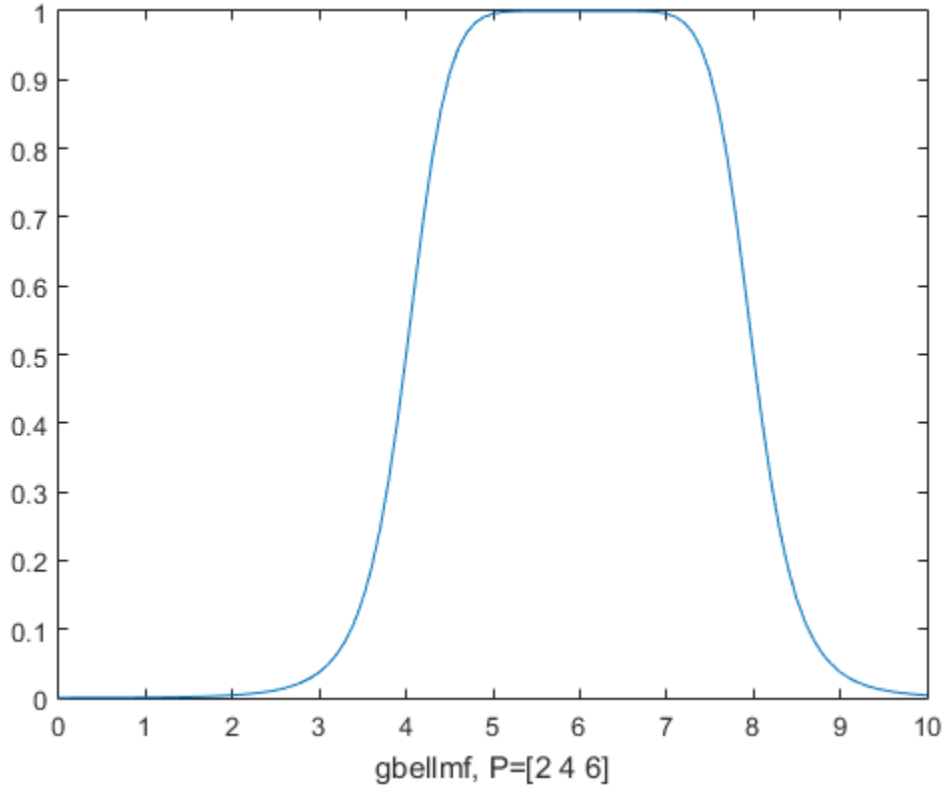
where the parameter b is usually positive. The parameter c locates the center of the curve. Enter the parameter vector `params`, the second argument for `gbellmf`, as the vector whose entries are a, b, and c, respectively.

Examples

Generalized Bell-Shaped Membership Function

```
x = 0:0.1:10;
y = gbellmf(x,[2 4 6]);
plot(x,y)
xlabel('gbellmf, P=[2 4 6]')
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`dsigmf` | `gaussmf` | `gauss2mf` | `evalmf` | `mf2mf` | `pimf` | `psigmf` | `sigmf` | `smf` | `trapmf` | `trimf` | `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

genfis1

Generate Fuzzy Inference System structure from data using grid partition

Syntax

```
fismat = genfis1(data)
```

```
fismat = genfis1(data,numMFs,inmftype,outmftype)
```

Description

`genfis1` generates a Sugeno-type FIS structure used as initial conditions (initialization of the membership function parameters) for `anfis` training.

`genfis1(data)` generates a single-output Sugeno-type fuzzy inference system using a grid partition on the data.

`genfis1(data,numMFs,inmftype,outmftype)` generates an FIS structure from a training data set, `data`, with the number and type of input membership functions and the type of output membership functions explicitly specified.

The arguments for `genfis1` are as follows:

- `data` is the training data matrix, which must be entered with all but the last columns representing input data, and the last column representing the single output.
- `numMFs` is a vector whose coordinates specify the number of membership functions associated with each input. If you want the same number of membership functions to be associated with each input, then specify `numMFs` as a single number.
- `inmftype` is a string array in which each row specifies the membership function type associated with each input. This can be a one-dimensional single string if the type of membership functions associated with each input is the same.
- `outmftype` is a string that specifies the membership function type associated with the output. There can only be one output, because this is a Sugeno-type system. The output membership function type must be either `linear` or `constant`. The number of membership functions associated with the output is the same as the number of rules generated by `genfis1`.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The default number of membership functions, `numMFs`, is 2; the default input membership function type is `'gbellmf'`; and the default output membership function type is `'linear'`. These are used whenever `genfis1` is invoked without the last three arguments.

The following table summarizes the default inference methods.

Inference Method	Default
AND	`prod`
OR	`max`
Implication	`prod`
Aggregation	`max`
Defuzzification	`wtaver`

Examples

Generate FIS Using Grid Partition

Generate a FIS using grid partition.

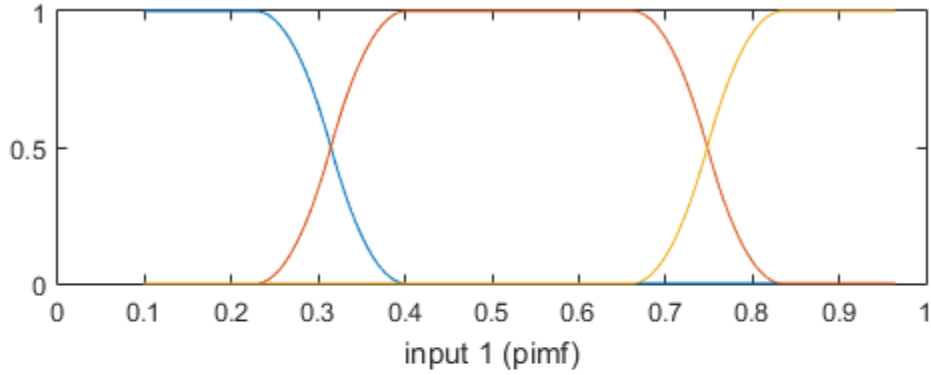
```
data = [rand(10,1) 10*rand(10,1)-5 rand(10,1)];
numMFs = [3 7];
mfType = char('pimf','trimf');
fismat = genfis1(data,numMFs,mfType);
```

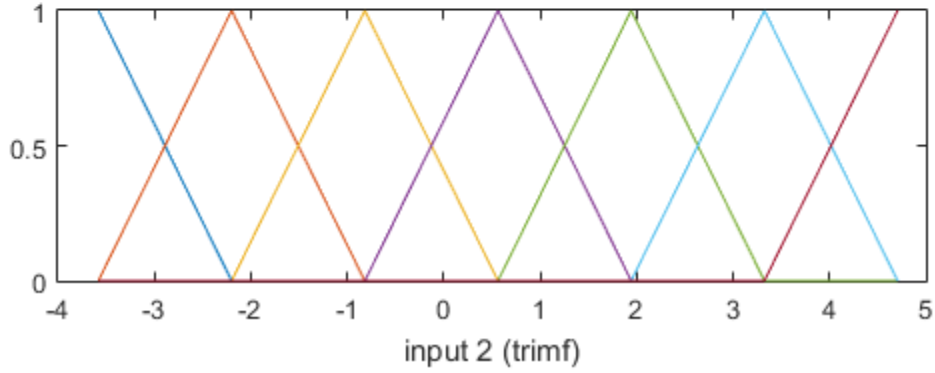
To see the contents of `fismat`, use `showfis(fismat)`.

Plot the FIS input membership functions.

```
[x,mf] = plotmf(fismat,'input',1);
subplot(2,1,1), plot(x,mf);
xlabel('input 1 (pimf)');
[x,mf] = plotmf(fismat,'input',2);
subplot(2,1,2), plot(x,mf);
xlabel('input 2 (trimf)');
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران





See Also

`anfis` | `genfis2` | `genfis3`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

genfis2

Generate Fuzzy Inference System structure from data using subtractive clustering

Syntax

```
fismat = genfis2(Xin,Xout,radii)
fismat = genfis2(Xin,Xout,radii,xBounds)
fismat = genfis2(Xin,Xout,radii,xBounds,options)
fismat = genfis2(Xin,Xout,radii,xBounds,options,user_centers)
```

Description

`genfis2` generates a Sugeno-type FIS structure using subtractive clustering and requires separate sets of input and output data as input arguments. When there is only one output, `genfis2` may be used to generate an initial FIS for `anfis` training. `genfis2` accomplishes this by extracting a set of rules that models the data behavior.

The rule extraction method first uses the `subclust` function to determine the number of rules and antecedent membership functions and then uses linear least squares estimation to determine each rule's consequent equations. This function returns an FIS structure that contains a set of fuzzy rules to cover the feature space.

The arguments for `genfis2` are as follows:

- `Xin` is a matrix in which each row contains the input values of a data point.
- `Xout` is a matrix in which each row contains the output values of a data point.
- `radii` is a vector that specifies a cluster center's range of influence in each of the data dimensions, assuming the data falls within a unit hyperbox.

  For example, if the data dimension is 3 (e.g., `Xin` has two columns and `Xout` has one column), `radii` = [0.5 0.4 0.3] specifies that the ranges of influence in the first, second, and third data dimensions (i.e., the first column of `Xin`, the second column of `Xin`, and the column of `Xout`) are 0.5, 0.4, and 0.3 times the width of the data space, respectively. If `radii` is a scalar value, then this scalar value is applied to all data dimensions, i.e., each cluster center has a spherical neighborhood of influence with the given radius.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- `xBounds` is a 2-by-*N* optional matrix that specifies how to map the data in `Xin` and `Xout` into a unit hyperbox, where *N* is the data (row) dimension. The first row of `xBounds` contains the minimum axis range values and the second row contains the maximum axis range values for scaling the data in each dimension.

  For example, `xBounds` = [-10 0 -1; 10 50 1] specifies that data values in the first data dimension are to be scaled from the range [-10 +10] into values in the range [0 1]; data values in the second data dimension are to be scaled from the range [0 50]; and data values in the third data dimension are to be scaled from the range [-1 +1]. If `xBounds` is an empty matrix or not provided, then `xBounds` defaults to the minimum and maximum data values found in each data dimension.
- `options` is an optional vector for specifying algorithm parameters to override the default values. These parameters are explained in the help text for `subclust`. Default values are in place when this argument is not specified.
- `user_centers` is an optional matrix for specifying custom cluster centers. `user_centers` has a size of `J`-by-`N` where `J` is the number of clusters and `N` is the total number of inputs and outputs.

The input membership function type is `'gaussmf'`, and the output membership function type is `'linear'`.

The following table summarizes the default inference methods.

Inference Method	Default
AND	`prod`
OR	`probor`
Implication	`prod`
Aggregation	`max`
Defuzzification	`wtaver`

Examples

Specify One Cluster Center Range of Influence For All Data Dimensions

Generate a FIS using subtractive clustering and specify the cluster center's range of influence.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
Xin = [7*rand(50,1) 20*rand(50,1)-10];
Xout = 5*rand(50,1);
fismat = genfis2(Xin,Xout,0.5);
```

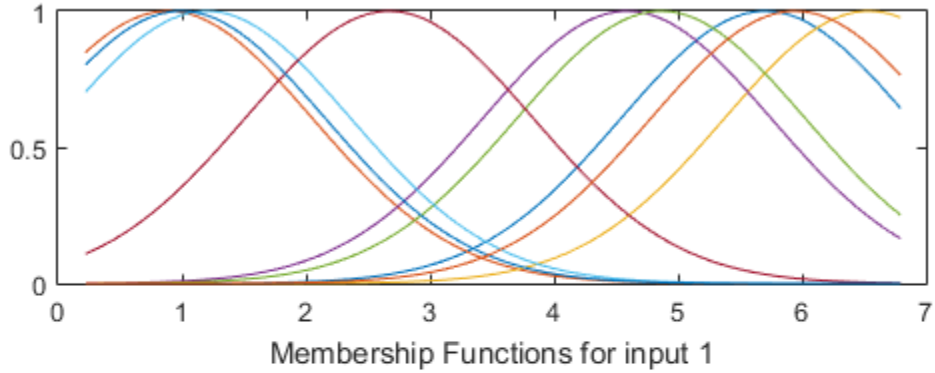
`fismat` specifies a range of influence of 0.5 for all data dimensions.

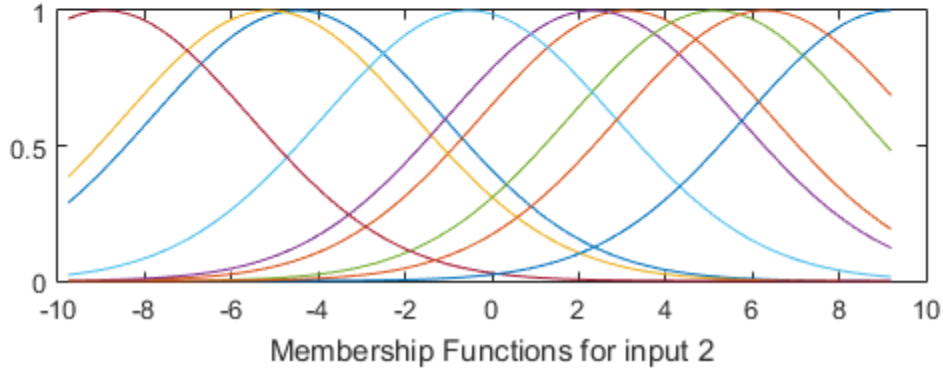
To see the contents of `fismat`, use `showfis(fismat)`.

Plot the input membership functions.

```
[x,mf] = plotmf(fismat,'input',1);
subplot(2,1,1), plot(x,mf);
xlabel('Membership Functions for input 1');
[x,mf] = plotmf(fismat,'input',2);
subplot(2,1,2), plot(x,mf);
xlabel('Membership Functions for input 2');
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران





Specify Cluster Center Range of Influence For Each Data Dimension

Suppose the input data has two columns, and the output data has one column. Specify 0.5 and 0.25 as the range of influence for the first and second input data columns. Specify 0.3 as the range of influence for the output data.

```
Xin = [7*rand(50,1) 20*rand(50,1)-10];
Xout = 5*rand(50,1);
fismat = genfis2(Xin,Xout,[0.5 0.25 0.3]);
```

Specify Data Hyperbox Scaling Range

Suppose the input data has two columns, and the output data has one column. Specify the scaling range for the inputs and outputs to normalize the data into the [0 1] range.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

The ranges for the first and second input data columns and the output data are: [-10 +10], [-5 +5], and [0 20].

```
Xin = [7*rand(50,1) 20*rand(50,1)-10];
Xout = 5*rand(50,1);
fismat = genfis2(Xin,Xout,0.5,[-10 -5 0; 10 5 20]);
```

Here, the third input argument, 0.5, specifies the range of influence for all data dimensions. The fourth input argument specifies the scaling range for the input and output data.

See Also

subclust | genfis1 | genfis3 | anfis

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

genfis3

Generate Fuzzy Inference System structure from data using FCM clustering

Syntax

```
fismat = genfis3(Xin,Xout)
```

```
fismat = genfis3(Xin,Xout,type)
```

```
fismat = genfis3(Xin,Xout,type,cluster_n)
```

```
fismat = genfis3(Xin,Xout,type,cluster_n,fcmoptions)
```

Description

`genfis3` generates an FIS using fuzzy c-means (FCM) clustering by extracting a set of rules that models the data behavior. The function requires separate sets of input and output data as input arguments. When there is only one output, you can use `genfis3` to generate an initial FIS for `anfis` training. The rule extraction method first uses the `fcm` function to determine the number of rules and membership functions for the antecedents and consequents.

`fismat = genfis3(Xin,Xout)` generates a Sugeno-type FIS structure (`fismat`) given input data `Xin` and output data `Xout`. The matrices `Xin` and `Xout` have one column per FIS input and output, respectively.

`fismat = genfis3(Xin,Xout,type)` generates an FIS structure of the specified `type`, where `type` is either `'mamdani'` or `'sugeno'`.

`fismat = genfis3(Xin,Xout,type,cluster_n)` generates an FIS structure of the specified `type` and allows you to specify the number of clusters (`cluster_n`) to be generated by FCM.

The number of clusters determines the number of rules and membership functions in the generated FIS. `cluster_n` must be an integer or `'auto'`. When `cluster_n` is `'auto'`, the function uses the `subclust` algorithm with a `radii` of 0.5 and the minimum and maximum values of `Xin` and `Xout` as `xBounds` to find the number of clusters. See `subclust` for more information.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

`fismat = genfis3(Xin,Xout,type,cluster_n,fcmoptions)` generates an FIS structure of the specified `type` and number of clusters and uses the specified `fcmoptions` for the FCM algorithm. If you omit `fcmoptions`, the function uses the default FCM values. See `fcm` for information about these parameters.

The input membership function type is `'gaussmf'`. By default, the output membership function type is `'linear'`. However, if you specify `type` as `'mamdani'`, then the output membership function type is `'gaussmf'`.

The following table summarizes the default inference methods.

Inference Method	Default
AND	`prod`
OR	`probor`
Implication	`prod`
Aggregation	`sum`
Defuzzification	`wtaver`

Examples

Generate Sugeno-Type FIS and Specify Number of Clusters

Obtain the input and output data.

```
Xin = [7*rand(50,1) 20*rand(50,1)-10];
Xout = 5*rand(50,1);
```

Generate a Sugeno-type FIS with 3 clusters.

```
opt = NaN(4,1);
opt(4) = 0;
fismat = genfis3(Xin,Xout,'sugeno',3,opt);
```

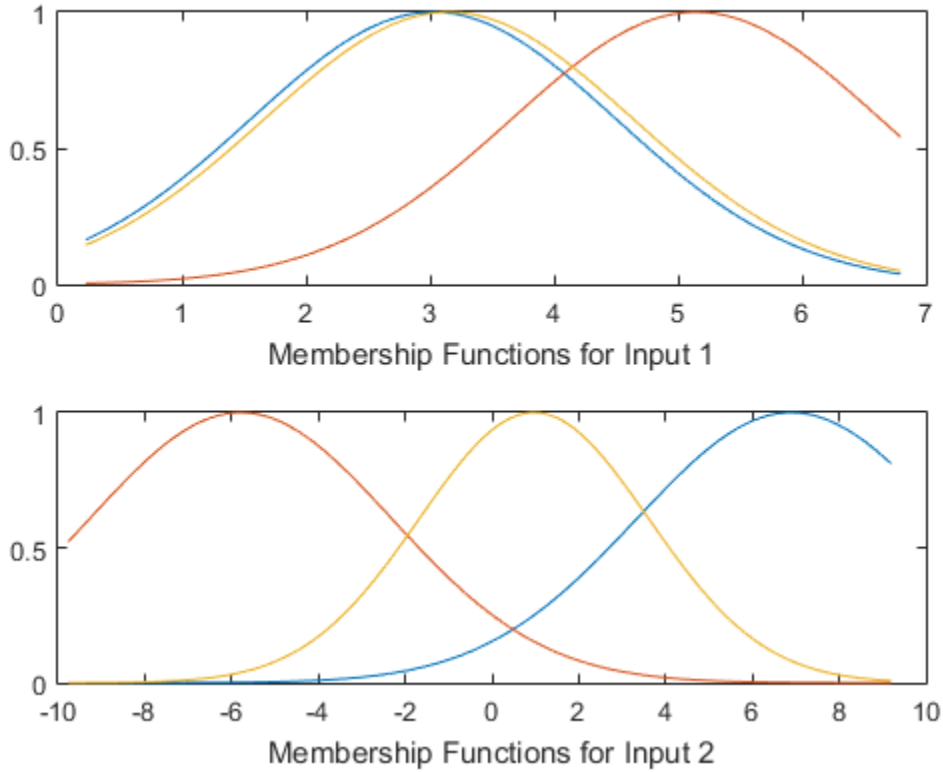
The fourth input argument specifies the number of clusters. The fifth input argument, `opt`, specifies the options for the FCM algorithm. The NaN entries of `opt` specify default option values. `opt(4)` turns off the display of iteration information at the MATLAB™ command line.

To see the contents of `fismat`, use `showfis(fismat)`.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Plot the input membership functions.

```
[x,mf] = plotmf(fismat,'input',1);
subplot(2,1,1), plot(x,mf);
xlabel('Membership Functions for Input 1');
[x,mf] = plotmf(fismat,'input',2);
subplot(2,1,2), plot(x,mf);
xlabel('Membership Functions for Input 2');
```



See Also

anfis | fcm | genfis1 | genfis2

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

gensurf

Generate Fuzzy Inference System output surface

Syntax

```
gensurf(fis)
gensurf(fis,inputs,output)
gensurf(fis,inputs,output,grids)
gensurf(fis,inputs,output,grids,refinput)
gensurf(fis,inputs,output,grids,refinput,numofpoints)
[x,y,z]=gensurf(...)
```

Description

`gensurf(fis)` generates a plot of the output surface of a given fuzzy inference system (`fis`) using the first two inputs and the first output.

`gensurf(fis,inputs,output)` generates a plot using the inputs (one or two) and output (only one is allowed) given, respectively, by the vector, `inputs`, and the scalar, `output`.

`gensurf(fis,inputs,output,grids)` allows you to specify the number of grids in the X (first, horizontal) and Y (second, vertical) directions. If `grids` is a two element vector, you can set the grids in the X and Y directions independently.

`gensurf(fis,inputs,output,grids,refinput)` allows you to specify a reference input, and can be used if there are more than two outputs. The length of the vector `refinput` is the same as the number of inputs:

- Enter `NaN`s for the entries of `refinput` corresponding to the inputs whose surface is being displayed.
- Enter real double scalars to fix the values of other inputs.

`gensurf(fis,inputs,output,grids,refinput,numofpoints)` allows you to specify the number of sample points on which to evaluate the membership functions in the input or output range. If `numofpoints` is not specified, a default value of 101 is used.

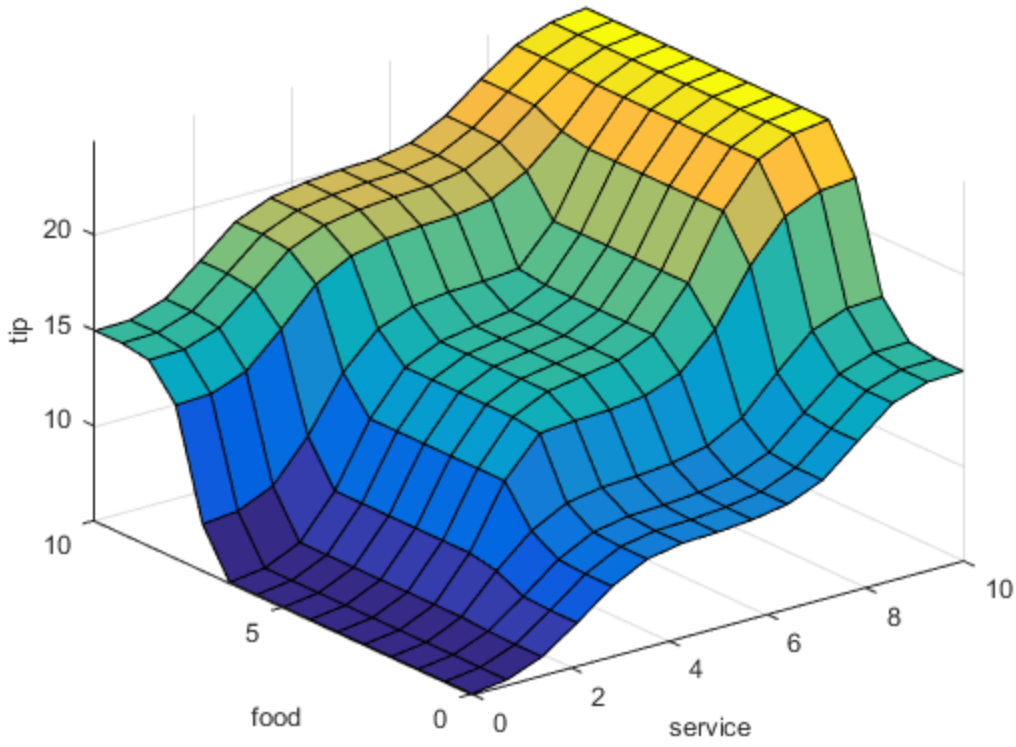
www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

`[x,y,z]=gensurf(...)` returns the variables that define the output surface and suppresses automatic plotting.

Examples

Generate FIS Output Surface

```
a = readfis('tipper');
gensurf(a)
```



Specify Reference Input for Surface Plot

```
a = gensurf(Temp,[1 2],1,[20 20],[nan nan 0.2]);
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

This equation generates the surface of a three-input FIS named `Temp` from its first two inputs to its first output, while fixing a reference value for the third input at .2.

See Also

`evalfis` | `surfview`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

getfis

Get fuzzy system properties

Syntax

```
getfis(a)
getfis(a,'fisprop')
getfis(a, 'vartype', varindex)
getfis(a,'vartype',varindex,'varprop')
getfis(a,'vartype',varindex,'mf',mfindex)
getfis(a,'vartype',varindex,'mf',mfindex,'mfprop')
```

Description

This function provides the fundamental access for the FIS structure. With this one function you can learn about every part of the fuzzy inference system.

The arguments for `getfis` are as follows:

- `a`: the name of a workspace variable FIS structure.
- `'fisprop'`: a string indicating the field you want to access. Possible fields include:

  ```
  'name'
  'type'
  'numinputs'
  'numoutputs'
  'numinputmfs'
  'numoutputmfs'
  'numrules'
  'andmethod'
  'ormethod'
  ```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
'impmethod'
'aggmethod'
'defuzzmethod'
'inlabels'
'outlabels'
'inrange'
'outrange'
'inmfs'
'outmfs'
'inmflabels'
'outmflabels'
'inmftypes'
'outmftypes'
'inmfparams'
'outmfparams'
'rulelist'
```

`'Name'` or `'NumInputs'`.

- `'vartype'`: a string indicating the type of variable you want (either `input` or `output`).
- `varindex`: an integer indicating the index of the variable you want (1, for input 1, for example).
- `'varprop'`: a string indicating the variable property you want. Possible strings are `name`, `range`, `nummfs`, and `mflabels`. See the examples for sample string names.
- `'mf'`: a required string that indicates you are searching for membership function information.
- `mfindex`: the index of the membership function for which you are seeking information.
- `'mfprop'`: a string indicating values for membership function properties. Possible strings are`name`, `type`, and `params`.

You can also access fuzzy system properties directly using MATLAB syntax for structures (including dot notation). See the examples that follow.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Examples

One input argument (`output` is the empty set)

```
a = readfis('tipper');
getfis(a)
         Name = tipper
        Type = mamdani
        NumInputs = 2
        InLabels =
                service
                food
        NumOutputs = 1
        OutLabels =
                tip
        NumRules = 3
        AndMethod = min
        OrMethod = max
        ImpMethod = min
        AggMethod = max
        DefuzzMethod = centroid
```

Two input arguments

```
getfis(a,'type')
ans =
mamdani
```

or

```
a.type
ans =
mamdani
```

Three input arguments (`output` is the empty set)

```
getfis(a,'input',1)
        Name = service
        NumMFs = 3
        MFLabels =
                poor
                good
                excellent
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
        Range = [0 10]
```

or

```
a.input(1)
ans =
     name: 'service'
    range: [0 10]
       mf: [1x3 struct]
```

Four input arguments

```
getfis(a,'input',1,'name')
ans =
service
```

or

```
a.input(1).name
ans =
service
```

Five input arguments

```
getfis(a,'input',1,'mf',2)
        Name = good
        Type = gaussmf
        Params =
    1.5000    5.0000
```

or

```
a.input(1).mf(2)
ans =
      name: 'good'
      type: 'gaussmf'
    params: [1.5000 5]
```

Six input arguments

```
getfis(a,'input',1,'mf',2,'name')
ans =
good
```

or

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
a.input(1).mf(2).name
ans =
good
```

See Also

setfis | showfis

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

mam2sug

Transform Mamdani Fuzzy Inference System into Sugeno Fuzzy Inference System

Syntax

```
sug_fis=mam2sug(mam_fis)
```

Description

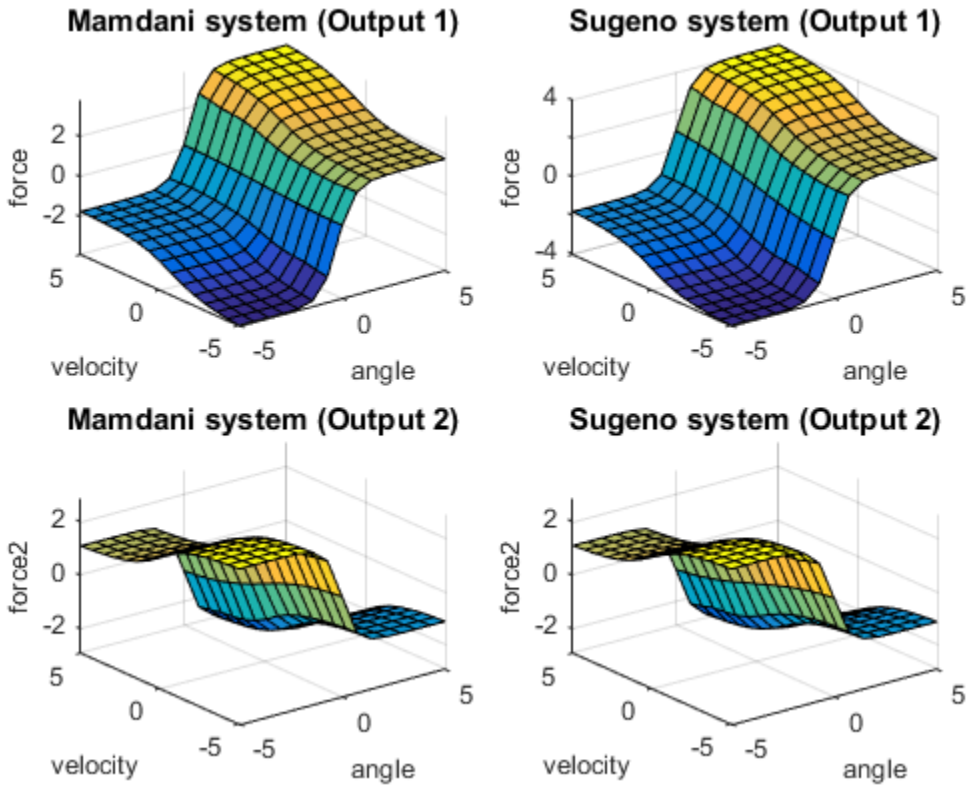
`mam2sug (mam_fis)` transforms a (not necessarily a single output) Mamdani FIS structure `mam_fis` into a Sugeno FIS structure `sug_fis`. The returned Sugeno system has constant output membership functions. These constants are determined by the centroids of the consequent membership functions of the original Mamdani system. The antecedent remains unchanged.

Examples

Transform Mamdani FIS into Sugeno FIS

```
mam_fismat = readfis('mam22.fis');
sug_fismat = mam2sug(mam_fismat);
subplot(2,2,1); gensurf(mam_fismat,[1 2],1);
title('Mamdani system (Output 1)');
subplot(2,2,2); gensurf(sug_fismat,[1 2],1);
title('Sugeno system (Output 1)');
subplot(2,2,3); gensurf(mam_fismat,[1 2],2);
title('Mamdani system (Output 2)');
subplot(2,2,4); gensurf(sug_fismat,[1 2],2);
title('Sugeno system (Output 2)');
```

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

mf2mf

Translate parameters between membership functions

Syntax

```
outParams = mf2mf(inParams,inType,outType)
```

Description

This function translates any built-in membership function type into another, in terms of its parameter set. In principle, `mf2mf` mimics the symmetry points for both the new and old membership functions.

Caution Occasionally this translation results in lost information, so that if the output parameters are translated back into the original membership function type, the transformed membership function does not look the same as it did originally.

The input arguments for `mf2mf` are as follows:

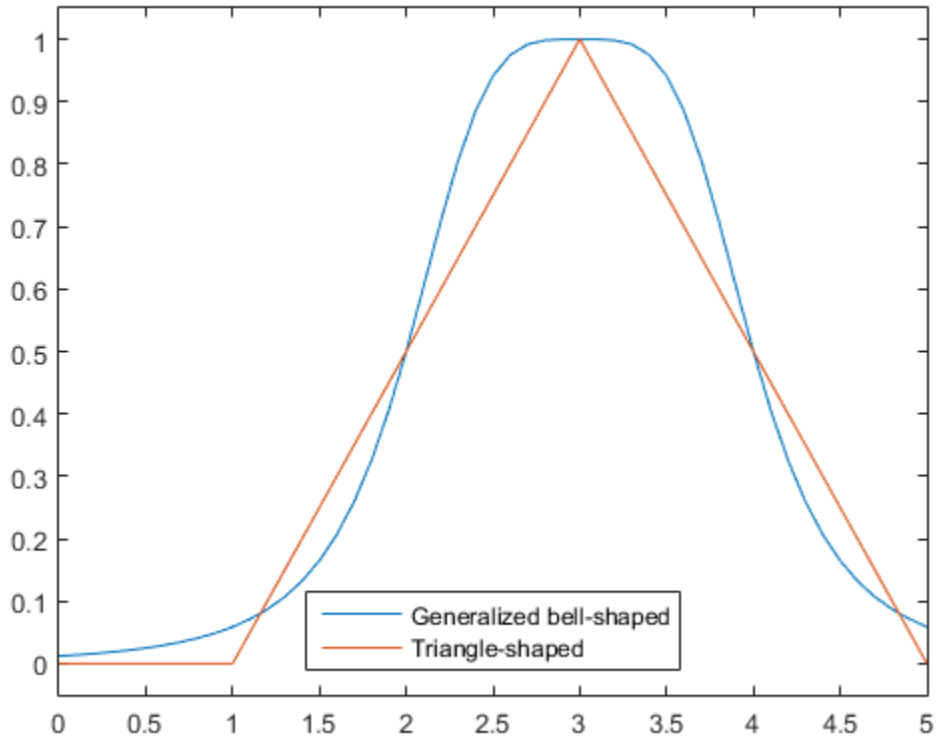
- `inParams`: the parameters of the membership function you are transforming
- `inType`: a string name for the type of membership function you are transforming
- `outType`: a string name for the new membership function you are transforming to

Examples

Translate Parameters Between Membership Functions

```
x = 0:0.1:5;
mfp1 = [1 2 3];
mfp2 = mf2mf(mfp1,'gbellmf','trimf');
plot(x,gbellmf(x,mfp1),x,trimf(x,mfp2))
legend('Generalized bell-shaped','Triangle-shaped','Location','South');
ylim([-0.05 1.05])
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`dsigmf` | `gaussmf` | `gauss2mf` | `gbellmf` | `evalmf` | `pimf` | `psigmf` | `sigmf` | `smf` | `trapmf` | `trimf` | `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

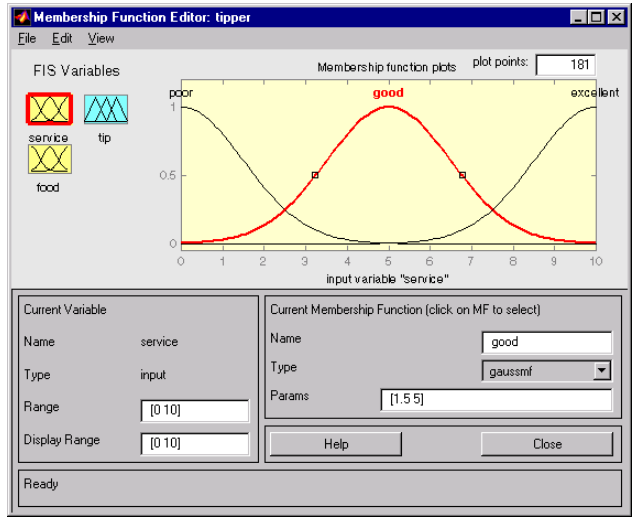
mfedit

Open Membership Function Editor

Syntax

```
mfedit('a')
mfedit(a)
mfedit
```

Description



`mfedit('a')` generates a membership function editor that allows you to inspect and modify all the membership functions for your FIS stored in the file, `a.fis`.

`mfedit(a)` operates on a MATLAB workspace variable, for an FIS structure, `a`.

`mfedit` alone opens the membership function editor with no FIS loaded.

For each membership function you can change the name, the type, and the parameters. Eleven built-in membership functions are provided for you to choose from, although of

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

course you can always create your own specialized versions. Refer to "The Membership Function Editor" on page 2-39 for more information about how to use `mfedit`.

Select the icon for the variable on the upper left side of the diagram (under **FIS Variables**) to display its associated membership functions in the plot region. Select membership functions by clicking once on them or their labels.

Menu Items

On the Membership Function Editor, there is a menu bar that allows you to open related GUI tools, open and save systems, and so on. The **File** menu for the Membership Function Editor is the same as the one found on the FIS Editor. Refer to "The Membership Function Editor" on page 2-39 for more information.

- Under **Edit**, select:

  Undo to undo the most recent change.

  Add MFs to add membership functions to the current variable.

  Add Custom MF to add a customized membership function to the current variable.

  Remove Selected MF to delete the current membership function.

  Remove All MFs to delete all membership functions of the current variable.

  FIS properties to invoke the FIS Editor.

  Rules to invoke the Rule Editor.
- Under **View**, select:

  Rules to invoke the Rule Viewer.

  Surface to invoke the Surface Viewer.

Membership Function Pop-up Menu

There are 11 built-in membership functions to choose from, and you also have the option of installing a customized membership function.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

See Also

`ruleedit` | `ruleview` | `surfview`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

neuroFuzzyDesigner

Open Neuro-Fuzzy Designer app

Syntax

```
neuroFuzzyDesigner
neuroFuzzyDesigner('a')
neuroFuzzyDesigner(a)
```

Description

`neuroFuzzyDesigner` opens the Neuro-Fuzzy Designer app which allows you to load a data set and train Adaptive Neuro-Fuzzy Inference Systems (ANFIS).

`neuroFuzzyDesigner('a')` opens the designer which allows you to implement ANFIS using an FIS structure stored as a file `a.fis`.

`neuroFuzzyDesigner(a)` uses an FIS structure stored as a variable.

Input Arguments

'a' — FIS structure file name
string

FIS structure file name specified as a string. This argument points to the `a.fis` file that contains an FIS structure.

a — FIS structure variable
string

FIS structure variable specified as a string. This argument points to the variable that contains an FIS structure, stored in the MATLAB workspace.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Menu Items

The menu bar on the Neuro-Fuzzy Designer app allows you to open related tools, open and save systems.

- In the **File** menu select

  New FIS > **Mamdani** to open a new `Untitled` Mamdani-style system with no variables and no rules.

  New FIS > **Sugeno** to open a new `Untitled` Sugeno-style system with no variables and no rules.

  Import > **From workspace** to load a system from a specified FIS structure variable in the MATLAB workspace.

  Import > **From file** to load a system from a specified `.fis` file.

  Export > **To workspace.** to save the system to an FIS structure variable in the MATLAB workspace.

  Export > **To file** to save the current system to a `.fis` file.

  Print to print the GUI display.

  Close to close the GUI.
- In the **Edit** menu select:

  Undo to undo the most recent change.

  FIS properties to open the FIS Editor.

  Membership functions to open the Membership Function Editor.

  Rules to open the Rule Editor.
- In the **View** menu select:

  Rules to open the Rule Viewer.

  Surface to open the Surface Viewer.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

More About

- "Neuro-Adaptive Learning and ANFIS" on page 3-2

See Also

`fuzzyLogicDesigner` | `mfedit` | `ruleedit` | `ruleview` | `surfview`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

newfis

Create new Fuzzy Inference System

Syntax

```
a = newfis(fisName,fisType,andMethod,orMethod,impMethod,aggMethod,defuzzMethod)
```

Description

This function creates new FIS structures. `newfis` has up to seven input arguments, and the output argument is an FIS structure. The seven input arguments are as follows:

- `fisName` is the string name of the FIS structure, `fisName.fis` you create.
- `fisType` is the type of FIS.
- `andMethod`, `orMethod`, `impMethod`, `aggMethod`, and `defuzzMethod`, respectively, provide the methods for AND, OR, implication, aggregation, and defuzzification.

Examples

The following example shows what the defaults are for each of the methods.

```
a=newfis('newsys');
getfis(a)
```

returns

```
        Name = newsys
        Type = mamdani
        NumInputs = 0
        InLabels =
        NumOutputs = 0
        OutLabels =
        NumRules   0
        AndMethod   min
        OrMethod   max
        ImpMethod   min
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
        AggMethod   max
        DefuzzMethod   centroid
ans =
     [newsys]
```

See Also

readfis | writefis

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

parsrule

Parse fuzzy rules

Syntax

```
fis2 = parsrule(fis,txtRuleList)
```

```
fis2 = parsrule(fis,txtRuleList,ruleFormat)
```

```
fis2 = parsrule(fis,txtRuleList,ruleFormat,lang)
```

Description

This function parses the text that defines the rules (`txtRuleList`) for a MATLAB workspace FIS variable, `fis`, and returns an FIS structure with the appropriate rule list in place. If the original input FIS structure, `fis`, has any rules initially, they are replaced in the new structure, `fis2`.

Three different rule formats (indicated by `ruleFormat`) are supported: `'verbose'`, `'symbolic'`, and `'indexed'`. The default format is `'verbose'`. When the optional language argument, `lang`, is used, the rules are parsed in verbose mode, assuming the key words are in the language, `lang`. This language must be either `''english'`, `'francais'`, or `'deutsch'`. The key language words in English are if, then, is, AND, OR, and NOT.

Examples

Add Single Rule to Fuzzy Inference System

Load a fuzzy inference system (FIS).

```
a = readfis('tipper');
```

Add a rule to the FIS.

```
ruleTxt = 'If service is poor then tip is generous';
a2 = parsrule(a,ruleTxt,'verbose');
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Verify the rules associated with the FIS.

```
showrule(a2)
```

```
ans =

1. If (service is poor) then (tip is generous) (1)
```

Add Multiple Rules to Fuzzy Inference System

Load a fuzzy inference system (FIS).

```
a = readfis('tipper');
```

Add multiple rules to the FIS.

```
rule1 = 'If service is poor or food is rancid then tip is cheap';
rule2 = 'If service is good then tip is average';
rule3 = 'If service is excellent or food is delicious then tip is generous';
rules = char(rule1,rule2,rule3);

a2 = parsrule(a,rules);
```

Verify the rules associated with the FIS.

```
showrule(a2)
```

```
ans =

1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
2. If (service is good) then (tip is average) (1)
3. If (service is excellent) or (food is delicious) then (tip is generous) (1)
```

See Also

addrule | ruleedit | showrule

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

pimf

Π-shaped membership function

Syntax

```
y = pimf(x,[a b c d])
```

Description

This spline-based curve is so named because of its Π shape. The membership function is evaluated at the points determined by the vector `x`. The parameters a and d locate the "feet" of the curve, while b and c locate its "shoulders." The membership function is a product of `smf` and `zmf` membership functions, and is given by:

$$f(x;a,b,c,d)=\begin{cases} 0, & x \le a \\ 2\left(\frac{x-a}{b-a}\right)^2, & a \le x \le \frac{a+b}{2} \\ 1-2\left(\frac{x-b}{b-a}\right)^2, & \frac{a+b}{2} \le x \le b \\ 1, & b \le x \le c \\ 1-2\left(\frac{x-c}{d-c}\right)^2, & c \le x \le \frac{c+d}{2} \\ 2\left(\frac{x-d}{d-c}\right)^2, & \frac{c+d}{2} \le x \le d \\ 0, & x \ge d \end{cases}$$

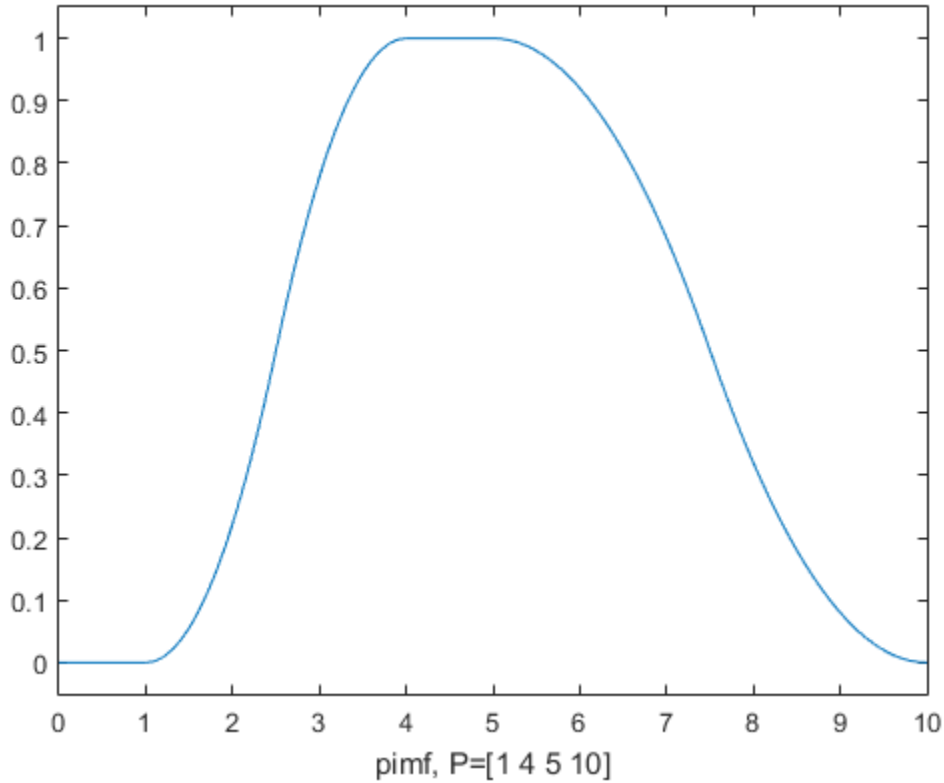
Examples

Pi-Shaped Membership Function

```
x = 0:0.1:10;
y = pimf(x,[1 4 5 10]);
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
plot(x,y)
xlabel('pimf, P=[1 4 5 10]')
ylim([-0.05 1.05])
```



See Also

dsigmf | evalmf | gauss2mf | gaussmf | gbellmf | mf2mf | psigmf | sigmf | smf | trapmf | trimf | zmf

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

plotfis

Plot Fuzzy Inference System

Syntax

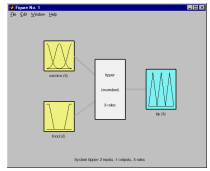
```
plotfis(fismat)
```

Description

This function displays a high level diagram of an FIS, `fismat`. Inputs and their membership functions appear to the left of the FIS structural characteristics, while outputs and their membership functions appear on the right.

Examples

```
a = readfis('tipper');
plotfis(a)
```



See Also

`evalmf` | `plotmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

plotmf

Plot all membership functions for given variable

Syntax

```
plotmf(fismat,varType,varIndex)
```

Description

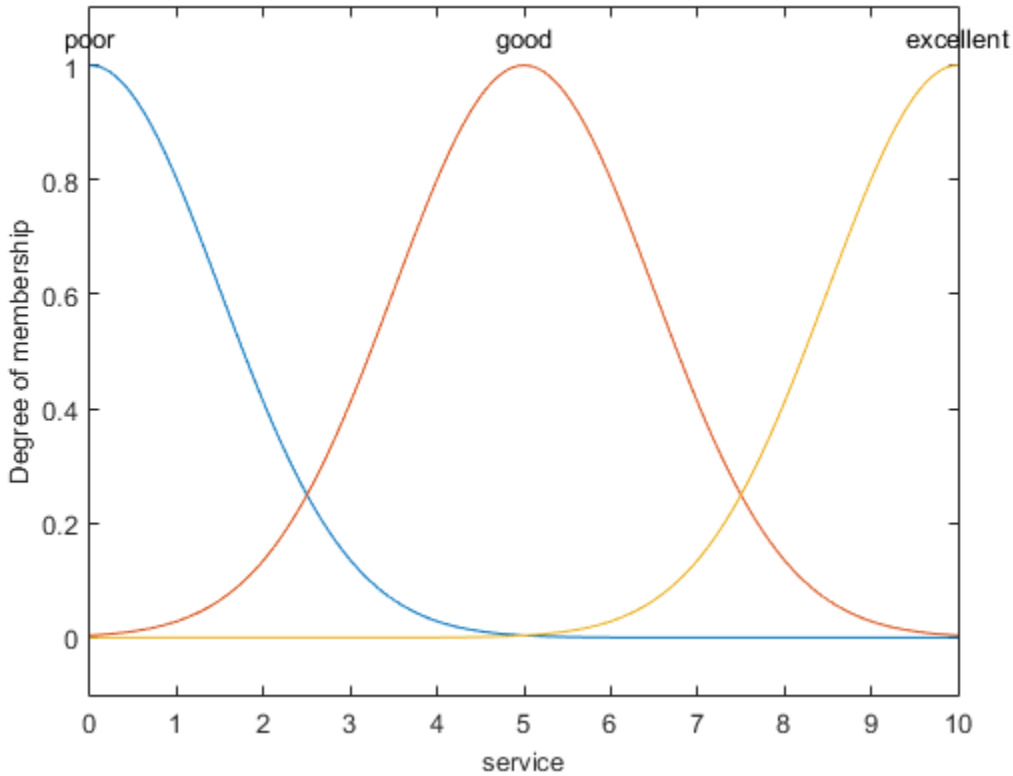
This function plots all of the membership functions in the FIS called `fismat` associated with a given variable whose type and index are respectively given by *`varType`* (must be `'input'` or `'output'`), and `varIndex`. This function can also be used with the MATLAB function, `subplot`.

Examples

Plot All Membership Functions for FIS Variable

```
a = readfis('tipper');
plotmf(a,'input',1);
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`evalmf` | `plotfis`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

probor

Probabilistic OR

Syntax

```
y = probor(x)
```

Description

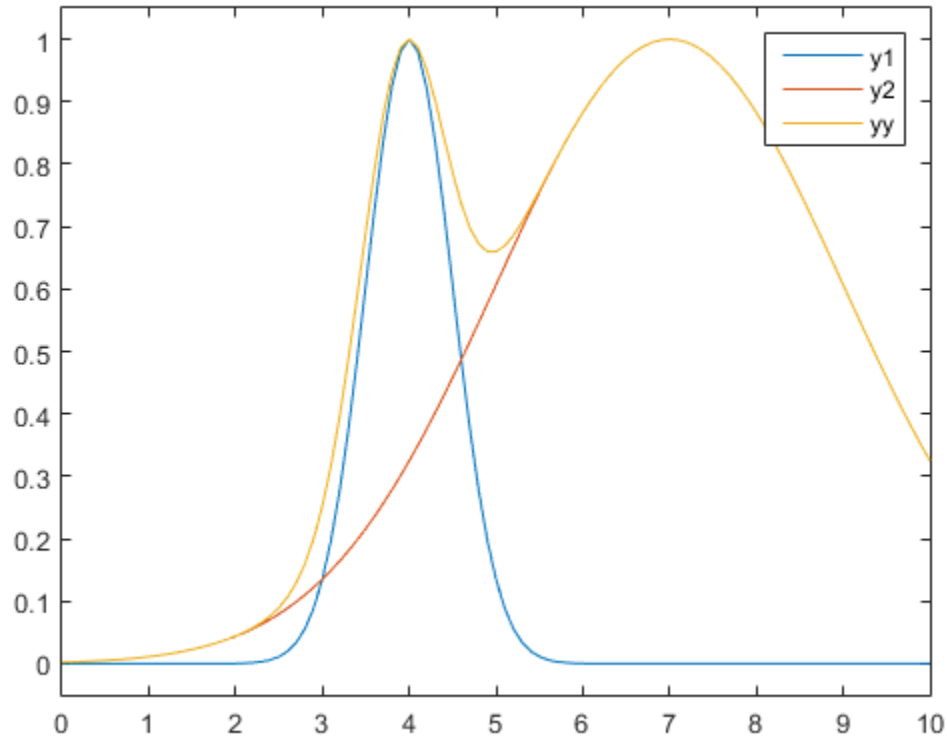
`y = probor(x)` returns the probabilistic OR (also known as the *algebraic sum*) of the columns of `x`. if x has two rows such that `x = [a; b]`, then `y = a + b - ab`. If `x` has only one row, then `y = x`.

Examples

Probabilistic OR

```
x = 0:0.1:10;
y1 = gaussmf(x, [0.5 4]);
y2 = gaussmf(x, [2 7]);
yy = probor([y1; y2]);
plot(x,[y1; y2; yy])
legend('y1','y2','yy');
ylim([-0.05 1.05])
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

psigmf

Product of two sigmoidal membership functions

Syntax

```
y = psigmf(x,[a1 c1 a2 c2])
```

Description

The sigmoid curve plotted for the vector `x` depends on two parameters a and c as given by

$$f(x;a,c) = \frac{1}{1+e^{-a(x-c)}}$$

`psigmf` is simply the product of two such curves plotted for the values of the vector `x`
$f_1(x;\, a_1,\, c_1) \times f_2(x;\, a_2,\, c_2)$

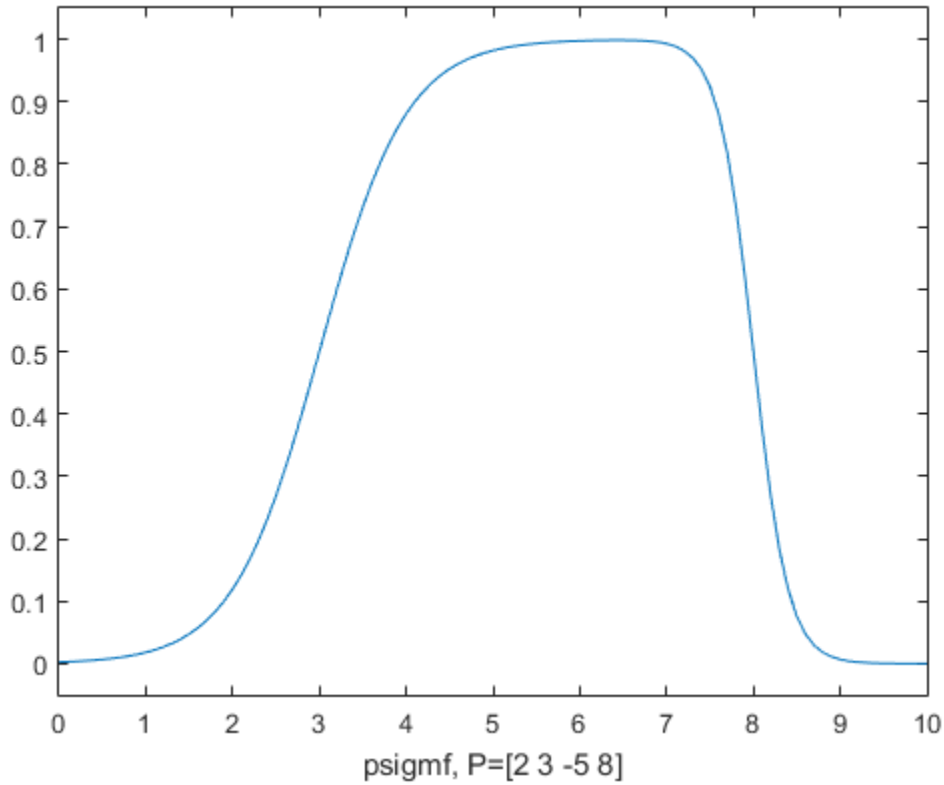
The parameters are listed in the order $[a_1\ c_1\ a_2\ c_2]$.

Examples

Product of Two Sigmoidal Membership Functions

```
x = 0:0.1:10;
y = psigmf(x,[2 3 -5 8]);
plot(x,y)
xlabel('psigmf, P=[2 3 -5 8]')
ylim([-0.05 1.05])
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`dsigmf` | `gaussmf` | `gauss2mf` | `gbellmf` | `evalmf` | `mf2mf` | `pimf` | `sigmf` | `smf` | `trapmf` | `trimf` | `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

readfis

Load Fuzzy Inference System from file

Syntax

```
fismat = readfis('filename')
```

```
fismat = readfis
```

Description

`fismat = readfis('filename')` reads a fuzzy inference system from a file and imports the resulting file into the workspace. The file must be named `filename.fis`.

`fismat = readfis` opens a dialog box for retrieving files to select the fuzzy inference system file.

Examples

```
fismat = readfis('tipper');
getfis(fismat)
```

returns

```
 Name = tipper
 Type = mamdani
 NumInputs = 2
 InLabels =
  service
  food
 NumOutputs = 1
 OutLabels =
  tip
 NumRules = 3
 AndMethod = min
 OrMethod = max
 ImpMethod = min
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
 AggMethod = max
 DefuzzMethod = centroid
ans =
tipper
```

See Also

writefis

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

rmmf

Remove membership function from Fuzzy Inference System

Syntax

```
fis = rmmf(fis,'varType',varIndex,'mf',mfIndex)
```

Description

`fis = rmmf(fis,varType,varIndex,'mf',mfIndex)` removes the membership function, `mfIndex`, of variable type `varType`, of index `varIndex`, from the fuzzy inference system associated with the workspace FIS structure, `fis`:

- The string *`varType`* must be `'input'` or `'output'`.
- `varIndex` is an integer for the index of the variable. This index represents the order in which the variables are listed.
- The argument `'mf'` is a string representing the membership function.
- `mfIndex` is an integer for the index of the membership function. This index represents the order in which the membership functions are listed.

Examples

```
a = newfis('mysys');
a = addvar(a,'input','temperature',[0 100]);
a = addmf(a,'input',1,'cold','trimf',[0 30 60]);
getfis(a,'input',1)
```

returns

```
        Name = temperature
        NumMFs = 1
        MFLabels =
                cold
        Range = [0 100]
b = rmmf(a,'input',1,'mf',1);
getfis(b,'input',1)
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

returns

```
Name = temperature
NumMFs = 0
MFLabels =
Range = [0 100]
```

See Also

addmf | addrule | addvar | plotmf | rmvar

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

rmvar

Remove variables from Fuzzy Inference System

Syntax

```
fis2 = rmvar(fis,'varType',varIndex)
[fis2,errorStr] = rmvar(fis,'varType',varIndex)
```

Description

`fis2 = rmvar(fis,'varType',varIndex)` removes the variable '*varType*', of index `varIndex`, from the fuzzy inference system associated with the workspace FIS structure, `fis`:

- The string *varType* must be `'input'` or `'output'`.
- `varIndex` is an integer for the index of the variable. This index represents the order in which the variables are listed.

`[fis2,errorStr] = rmvar(fis,'varType',varIndex)` returns any error messages to the string, `errorStr`.

This command automatically alters the rule list to keep its size consistent with the current number of variables. You must delete from the FIS any rule that contains a variable you want to remove, before removing it. You cannot remove a fuzzy variable currently in use in the rule list.

Examples

```
a = newfis('mysys');
a = addvar(a,'input','temperature',[0 100]);
getfis(a)
```

returns

```
 Name = mysys
      Type      = mamdani
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
      NumInputs = 1
      InLabels  =
            temperature
      NumOutputs = 0
      OutLabels =
      NumRules = 0
      AndMethod = min
      OrMethod = max
      ImpMethod = min
      AggMethod = max
      DefuzzMethod = centroid
ans =
mysys
b = rmvar(a,'input',1);
getfis(b)
```

returns

```
        Name = mysys
        Type = mamdani
        NumInputs = 0
        InLabels =
        NumOutputs = 0
        OutLabels =
        NumRules = 0
      AndMethod = min
      OrMethod = max
      ImpMethod = min
      AggMethod = max
      DefuzzMethod = centroid
ans =
mysys
```

See Also

addmf | addrule | addvar | rmmf

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

ruleedit

Open Rule Editor

Syntax

```
ruleedit('a')
```

```
ruleedit(a)
```

Description

The Rule Editor, when invoked using `ruleedit('a')`, is used to modify the rules of an FIS structure stored in a file, `a.fis`. It can also be used to inspect the rules being used by a fuzzy inference system.

To use this editor to create rules, you must first define all of the input and output variables you want to use with the FIS editor. You can create the rules using the listbox and check box choices for input and output variables, connections, and weights. Refer to "The Rule Editor" on page 2-46 for more information about how to use `ruleedit`.

The syntax `ruleedit(a)` is used when you want to operate on a workspace variable for an FIS structure called `a`.

Menu Items

On the Rule Editor, there is a menu bar that allows you to open related GUI tools, open and save systems, and so on. The **File** menu for the Rule Editor is the same as the one found on the FIS Editor. Refer to `fuzzyLogicDesigner` for more information:

- Use the following **Edit** menu item:

  Undo to undo the most recent change.

  FIS properties to invoke the FIS Editor.

  Membership functions to invoke the Membership Function Editor.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- Use the following **View** menu items:

  Rules to invoke the Rule Viewer.

  Surface to invoke the Surface Viewer.
- Use the **Options** menu items:

  Language to select the language: **English**, **Deutsch**, and **Francais**

  Format to select the format:

  Verbose uses the words "if," "then," "AND," "OR," and so on to create actual sentences.

  Symbolic substitutes some symbols for the words used in the verbose mode. For example, “if *A* AND *B* then *C*” becomes “*A* & *B* => *C*.”

  Indexed mirrors how the rule is stored in the FIS structure.

See Also

`addrule` | `fuzzyLogicDesigner` | `parsrule` | `ruleview` | `showrule` | `surfview` | `mfedit`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

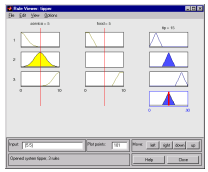
ruleview

Open Rule Viewer

Syntax

```
ruleview(a)
ruleview('a')
```

Description



`ruleview(a)` opens the Rule Viewer for the fuzzy inference system, `a`. It is used to view the entire implication process from beginning to end. You can move around the line indices that correspond to the inputs and then watch the system readjust and compute the new output. Refer to "The Rule Viewer" on page 2-49 for more information about how to use `ruleview`.

`ruleview('a')` depicts the fuzzy inference diagram for the fuzzy inference system stored in file `a.fis`.

Menu Items

On the Rule Viewer, there is a menu bar that allows you to open related GUI tools, open and save systems, and so on. The **File** menu for the Rule Viewer is the same as the one found on the FIS Editor. Refer to `fuzzyLogicDesigner` for more information.

- Use the **Edit** menu items:

  Undo to undo the most recent action

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

FIS properties to invoke the FIS Editor

Membership functions to invoke the Membership Function Editor

Rules to invoke the Rule Editor

- Use the **View** menu item:

  Surface to invoke the Surface Viewer

- Use the **Options** menu item:

  Format to set the format in which the rule appears: **Verbose, Symbolic,** or **Indexed**.

If you click on the rule numbers on the left side of the fuzzy inference diagram, the rule associated with that number appears in the Status Bar at the bottom of the Rule Viewer.

See Also

`fuzzyLogicDesigner` | `mfedit` | `ruleedit` | `surfview`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

setfis

Set fuzzy system properties

Syntax

```
a = setfis(a,'fispropname','newfisprop')
```

```
a = setfis(a,'vartype',varindex,'varpropname',newvarprop)
```

```
a = setfis(a,'vartype',varindex,'mf',mfindex,...,'mfpropname',newmfprop)
```

Description

The command `setfis` can be called with three, five, or seven input arguments, depending on whether you want to set a property of the entire FIS structure, for a particular variable belonging to that FIS structure, or for a particular membership function belonging to one of those variables. The arguments are:

- `a` — a variable name of an FIS from the workspace
- '*vartype*' — a string indicating the variable type: `input` or `output`
- `varindex` — the index of the input or output variable
- `'mf'` — a required string for the fourth argument of a seven-argument call for `setfis`, indicating this variable is a membership function
- `mfindex` — the index of the membership function belonging to the chosen variable
- `'fispropname'` — a string indicating the property of the FIS field you want to set: `name`, `type`, `andmethod`, `ormethod`, `impmethod`, `aggmethod`, `defuzzmethod`
- `'newfisprop'` — a string describing the name of the FIS property or method you want to set
- `'varpropname'` — a string indicating the name of the variable field you want to set: `name` or `range`
- `newvarprop` — a string describing the name of the variable you want to set (for *name*), or an array describing the range of that variable (for `range`)
- `'mfpropname'` — a string indicating the name of the membership function field you want to set: `name`, `type`, or `params`.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- 'newmfprop' — a string describing the name or type of the membership function field want to set (for name or type), or an array describing the range of the parameters (for params)

Examples

setfis is called with three arguments:

```
a = readfis('tipper');
a2 = setfis(a, 'name', 'eating');
getfis(a2, 'name');
```

which results in

```
out =
eating
```

If it is used with five arguments, setfis updates two variable properties.

```
a2 = setfis(a,'input',1,'name','help');
getfis(a2,'input',1,'name')
ans =
 help
```

If used with seven arguments, setfis updates any of several membership function properties.

```
a2 = setfis(a,'input',1,'mf',2,'name','wretched');
getfis(a2,'input',1,'mf',2,'name')
ans =
 wretched
```

See Also

getfis

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

showfis

Display annotated Fuzzy Inference System

Syntax

```
showfis(fismat)
```

Description

`showfis(fismat)` prints a version of the MATLAB workspace variable FIS, `fismat`, allowing you to see the significance and contents of each field of the structure.

Examples

```
a = readfis('tipper');
showfis(a)
```

Returns:

```
1.  Name             tipper
2.  Type             mamdani
3.  Inputs/Outputs   [2 1]
4.  NumInputMFs      [3 2]
5.  NumOutputMFs     3
6.  NumRules         3
7.  AndMethod        min
8.  OrMethod         max
9.  ImpMethod        min
10. AggMethod        max
11. DefuzzMethod     centroid
12. InLabels         service
13.                  food
14. OutLabels        tip
15. InRange          [0 10]
16.                  [0 10]
17. OutRange         [0 30]
18. InMFLabels       poor
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
19.                 good
20.                 excellent
21.                 rancid
22.                 delicious
23. OutMFLabels     cheap
24.                 average
25.                 generous
26. InMFTypes       gaussmf
27.                 gaussmf
28.                 gaussmf
29.                 trapmf
30.                 trapmf
31. OutMFTypes      trimf
32.                 trimf
33.                 trimf
34. InMFParams      [1.5 0 0 0]
35.                 [1.5 5 0 0]
36.                 [1.5 10 0 0]
37.                 [0 0 1 3]
38.                 [7 9 10 10]
39. OutMFParams     [0 5 10 0]
40.                 [10 15 20 0]
41.                 [20 25 30 0]
42. Rule Antecedent [1 1]
43.                 [2 0]
44.                 [3 2]
42. Rule Consequent 1
43.                 2
44.                 3
42. Rule Weight     1
43.                 1
44.                 1
42. Rule Connection 2
43.                 1
44.                 2
```

See Also

getfis

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

showrule

Display Fuzzy Inference System rules

Syntax

```
showrule(fis)
showrule(fis,indexList)
showrule(fis,indexList,format)
showrule(fis,indexList,'verbose',lang)
```

Description

This command is used to display the rules associated with a given system. It can be invoked with one to four arguments. The first argument, `fis`, is required. This argument is the MATLAB workspace variable name for an FIS structure. The second (optional) argument `indexList` is the vector of rules you want to display. The third (optional) argument is the string representing the format in which the rules are returned. `showrule` can return the rule in any of three different formats: `'verbose'` (the default mode, for which English is the default language), `'symbolic'`, and `'indexed'`, for membership function index referencing.

When used with four arguments, the third argument must be `'verbose'`, and `showrule(fis,indexList,'verbose',lang)` displays the rules in the language given by `lang`, which must be either `'english'`, `'francais'`, or `'deutsch'`.

Examples

Display Rules for a Fuzzy Inference System

```
a = readfis('tipper');
showrule(a,1)
```

```
ans =
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
1. If (service is poor) or (food is rancid) then (tip is cheap) (1)
```

```
showrule(a,2)
```

```
ans =

2. If (service is good) then (tip is average) (1)
```

```
showrule(a,[3 1],'symbolic')
```

```
ans =

3. (service==excellent) | (food==delicious) => (tip=generous) (1)
1. (service==poor) | (food==rancid) => (tip=cheap) (1)
```

```
showrule(a,1:3,'indexed')
```

```
ans =

1 1, 1 (1) : 2
2 0, 2 (1) : 1
3 2, 3 (1) : 2
```

See Also

addrule | parsrule | ruleedit

Introduced before R2006a

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

sigmf

Sigmoidal membership function

Syntax

```
y = sigmf(x,[a c])
```

Description

The sigmoidal function, `sigmf(x,[a c])`, as given in the following equation by $f(x,a,c)$ is a mapping on a vector `x`, and depends on two parameters a and c.

$$f(x,a,c)=\frac{1}{1+e^{-a(x-c)}}$$

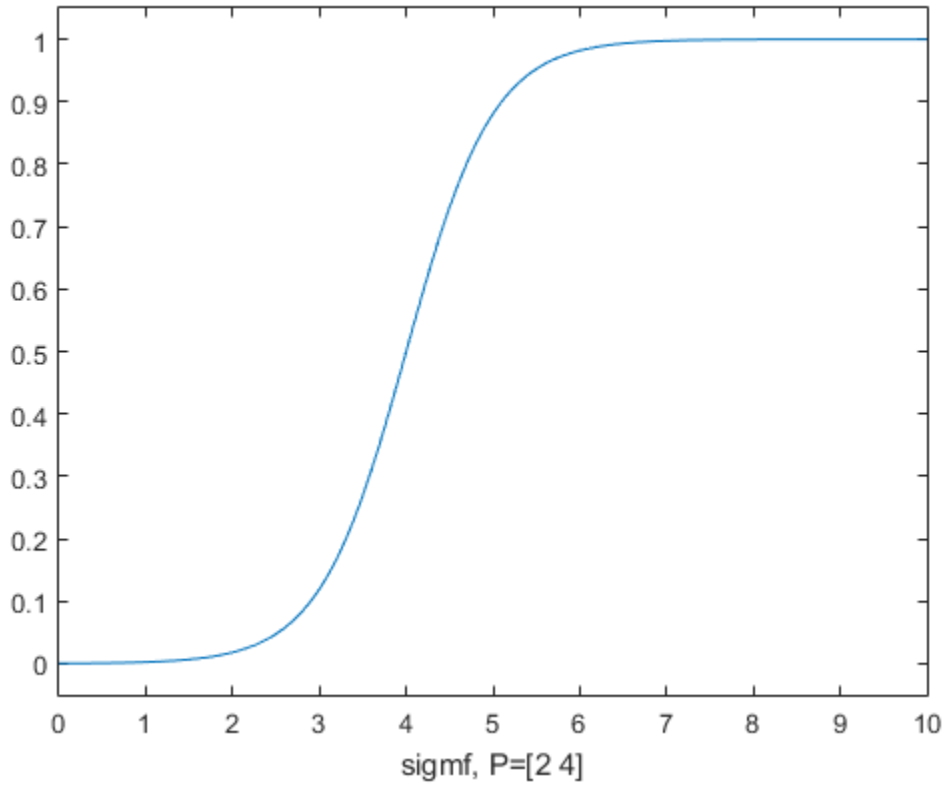
Depending on the sign of the parameter a, the sigmoidal membership function is inherently open to the right or to the left, and thus is appropriate for representing concepts such as “very large” or “very negative.” More conventional-looking membership functions can be built by taking either the product or difference of two different sigmoidal membership functions. For more information see`dsigmf` and `psigmf`.

Examples

Sigmoidal Membership Function

```
x = 0:0.1:10;
y = sigmf(x,[2 4]);
plot(x,y)
xlabel('sigmf, P=[2 4]')
ylim([-0.05 1.05])
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`dsigmf` | `gaussmf` | `gauss2mf` | `gbellmf` | `evalmf` | `mf2mf` | `pimf` | `psigmf` | `sigmf` | `smf` | `trapmf` | `trimf` | `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

smf

S-shaped membership function

Syntax

```
y = smf(x,[a b])
```

Description

This spline-based curve is a mapping on the vector x, and is named because of its S-shape. The parameters a and b locate the extremes of the sloped portion of the curve, as given by:

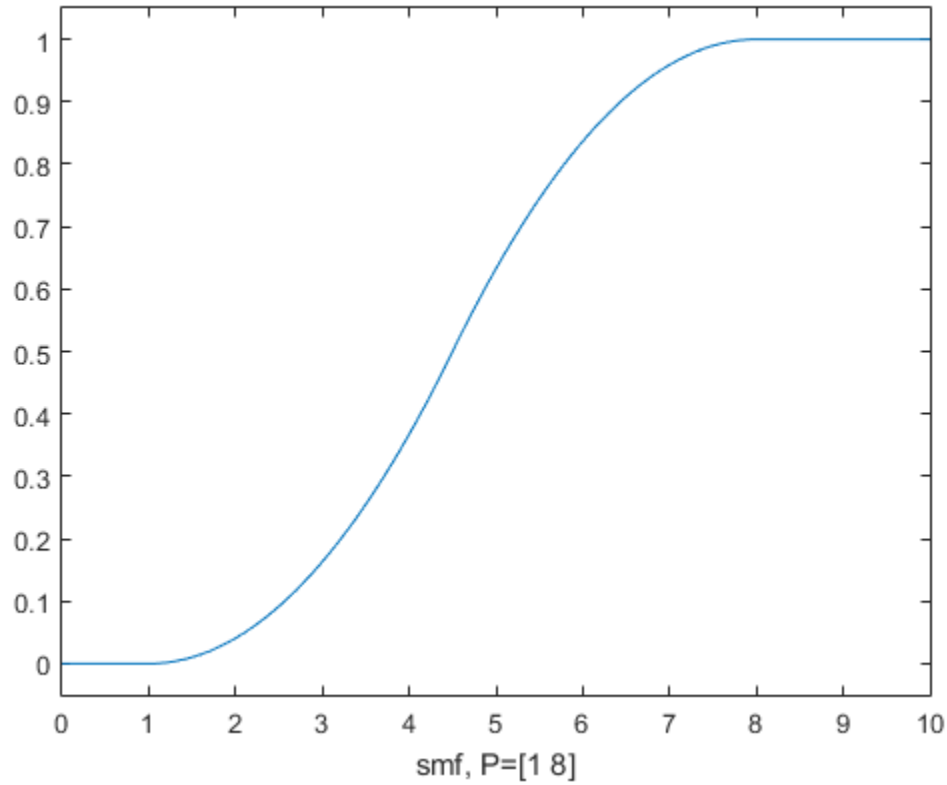
$$f(x;a,b) = \begin{cases} 0, & x \le a \\ 2\left(\frac{x-a}{b-a}\right)^2, & a \le x \le \frac{a+b}{2} \\ 1 - 2\left(\frac{x-b}{b-a}\right)^2, & \frac{a+b}{2} \le x \le b \\ 1, & x \ge b \end{cases}$$

Examples

S-Shaped Membership Function

```
x = 0:0.1:10;
y = smf(x,[1 8]);
plot(x,y)
xlabel('smf, P=[1 8]')
ylim([-0.05 1.05])
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`dsigmf` | `gaussmf` | `gauss2mf` | `gbellmf` | `evalmf` | `mf2mf` | `pimf` | `psigmf` | `sigmf` | `smf` | `trapmf` | `trimf` | `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

subclust

Find cluster centers with subtractive clustering

Syntax

```
[C,S] = subclust(X,radii,xBounds,options)
```

Description

`[C,S] = subclust(X,radii,xBounds,options)` estimates the cluster centers in a set of data by using the subtractive clustering method.

The function returns the cluster centers in the matrix `C`. Each row of `C` contains the position of a cluster center. The returned `S` vector contains the sigma values that specify the range of influence of a cluster center in each of the data dimensions. All cluster centers share the same set of sigma values.

The subtractive clustering method assumes each data point is a potential cluster center and calculates a measure of the likelihood that each data point would define the cluster center, based on the density of surrounding data points. The algorithm does the following:

- Selects the data point with the highest potential to be the first cluster center
- Removes all data points in the vicinity of the first cluster center (as determined by `radii`), in order to determine the next data cluster and its center location
- Iterates on this process until all of the data is within `radii` of a cluster center

The subtractive clustering method is an extension of the mountain clustering method proposed by R. Yager.

The matrix `X` contains the data to be clustered; each row of `X` is a data point. The variable `radii` is a vector of entries between 0 and 1 that specifies a cluster center's range of influence in each of the data dimensions, assuming the data falls within a unit hyperbox. Small `radii` values generally result in finding a few large clusters. The best values for `radii` are usually between 0.2 and 0.5.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

For example, if the data dimension is two (X has two columns), radii=[0.5 0.25] specifies that the range of influence in the first data dimension is half the width of the data space and the range of influence in the second data dimension is one quarter the width of the data space. If radii is a scalar, then the scalar value is applied to all data dimensions, i.e., each cluster center has a spherical neighborhood of influence with the given radius.

xBounds is a 2-by-N matrix that specifies how to map the data in X into a unit hyperbox, where N is the data dimension. This argument is optional if X is already normalized. The first row contains the minimum axis range values and the second row contains the maximum axis range values for scaling the data in each dimension.

For example, xBounds = [-10 -5; 10 5] specifies that data values in the first data dimension are to be scaled from the range [-10 +10] into values in the range [0 1]; data values in the second data dimension are to be scaled from the range [-5 +5] into values in the range [0 1]. If xBounds is an empty matrix or not provided, then xBounds defaults to the minimum and maximum data values found in each data dimension.

The options vector can be used for specifying clustering algorithm parameters to override the default values. These components of the vector options are specified as follows:

- options(1) = quashFactor: This factor is used to multiply the radii values that determine the neighborhood of a cluster center, so as to quash the potential for outlying points to be considered as part of that cluster. (default: 1.25)
- options(2) = acceptRatio: This factor sets the potential, as a fraction of the potential of the first cluster center, above which another data point is accepted as a cluster center. (default: 0.5)
- options(3) = rejectRatio: This factor sets the potential, as a fraction of the potential of the first cluster center, below which a data point is rejected as a cluster center. (default: 0.15)
- options(4) = verbose: If this term is not zero, then progress information is printed as the clustering process proceeds. (default: 0)

Examples

```
[C,S] = subclust(X,0.5)
```

This command sets the minimum number of arguments needed to use this function. A range of influence of 0.5 has been specified for all data dimensions.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
[C,S] = subclust(X,[0.5 0.25 0.3],[],[2.0 0.8 0.7])
```

This command assumes the data dimension is 3 (`X` has 3 columns) and uses a range of influence of 0.5, 0.25, and 0.3 for the first, second, and third data dimension, respectively. The scaling factors for mapping the data into a unit hyperbox are obtained from the minimum and maximum data values. The `squashFactor` is set to 2.0, indicating that you only want to find clusters that are far from each other. The `acceptRatio` is set to 0.8, indicating that only data points that have a very strong potential for being cluster centers are accepted. The `rejectRatio` is set to 0.7, indicating that you want to reject all data points without a strong potential.

References

Chiu, S., "Fuzzy Model Identification Based on Cluster Estimation," *Journal of Intelligent & Fuzzy Systems*, Vol. 2, No. 3, Sept. 1994.

Yager, R. and D. Filev, "Generation of Fuzzy Rules by Mountain Clustering," *Journal of Intelligent & Fuzzy Systems*, Vol. 2, No. 3, pp. 209-219, 1994.

See Also

`genfis2`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

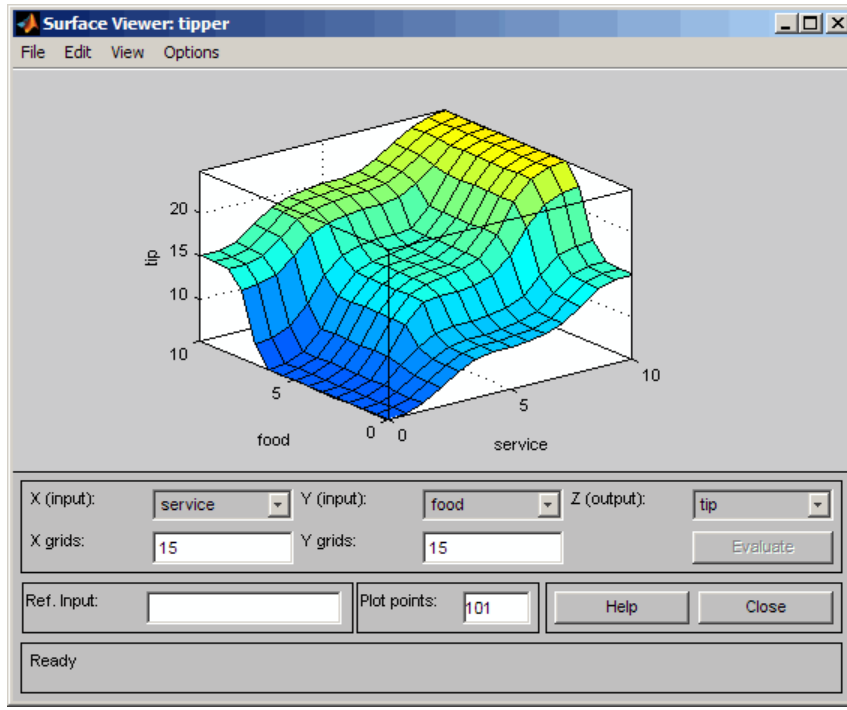
surfview

Open Surface Viewer

Syntax

```
surfview('a')
```

Description



The Surface Viewer, invoked using `surfview('a')`, is a GUI tool that lets you examine the output surface of an FIS stored in a file, `a.fis`, for any one or two inputs. Because it does not alter the fuzzy system or its associated FIS structure in any way, Surface Viewer is a read-only editor. Using the drop-down menus, you select the two input

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

variables you want assigned to the two input axes (X and Y), as well the output variable you want assigned to the output (or Z) axis.

If you want to create a smoother plot, use the **Plot points** field to specify the number of points on which the membership functions are evaluated in the input or output range. This field defaults to a value of 101.

Click **Evaluate** to perform the calculation and plot the output surface.

By clicking on the plot axes and dragging the mouse, you can manipulate the surface so that you can view it from different angles.

If there are more than two inputs to your system, you must supply the constant values associated with any unspecified inputs in the reference input section.

Refer to "The Surface Viewer" on page 2-51 for more information about how to use `surfview`.

Menu Items

On the Surface Viewer, there is a menu bar that allows you to open related GUI tools, open and save systems, and so on. The Surface Viewer uses the same **File** menu as the one on the FIS Editor. Refer to `fuzzyLogicDesigner` for more information:

- Use the **Edit** menu items:

  Undo to undo the most recent action

  FIS properties to invoke the FIS Editor

  Membership functions to invoke the Membership Function Editor

  Rules... to invoke the Rule Editor
- Use the **View** menu item:

  Rules to invoke the Rule Viewer
- Use the **Options** menu items:

  Plot to choose among eight different kinds of plot styles.

  Color Map to choose among several different color schemes.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Always evaluate to automatically evaluate and plot a new surface every time you make a change that affects the plot, such as changing the number of grid points. This option is selected by default. To clear this option, select it once more.

See Also

`gensurf` | `mfedit` | `ruleedit` | `ruleview`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

trapmf

Trapezoidal-shaped membership function

Syntax

```
y = trapmf(x,[a b c d])
```

Description

The trapezoidal curve is a function of a vector, `x`, and depends on four scalar parameters a, b, c, and d, as given by

$$f(x;a,b,c,d)=\begin{cases} 0, & x\le a \\ \dfrac{x-a}{b-a}, & a\le x\le b \\ 1, & b\le x\le c \\ \dfrac{d-x}{d-c}, & c\le x\le d \\ 0, & d\le x \end{cases}$$

or, more compactly, by

$$f(x;a,b,c,d)=\max\left(\min\left(\frac{x-a}{b-a},1,\frac{d-x}{d-c}\right),o\right)$$

The parameters a and d locate the "feet" of the trapezoid and the parameters b and c locate the "shoulders."

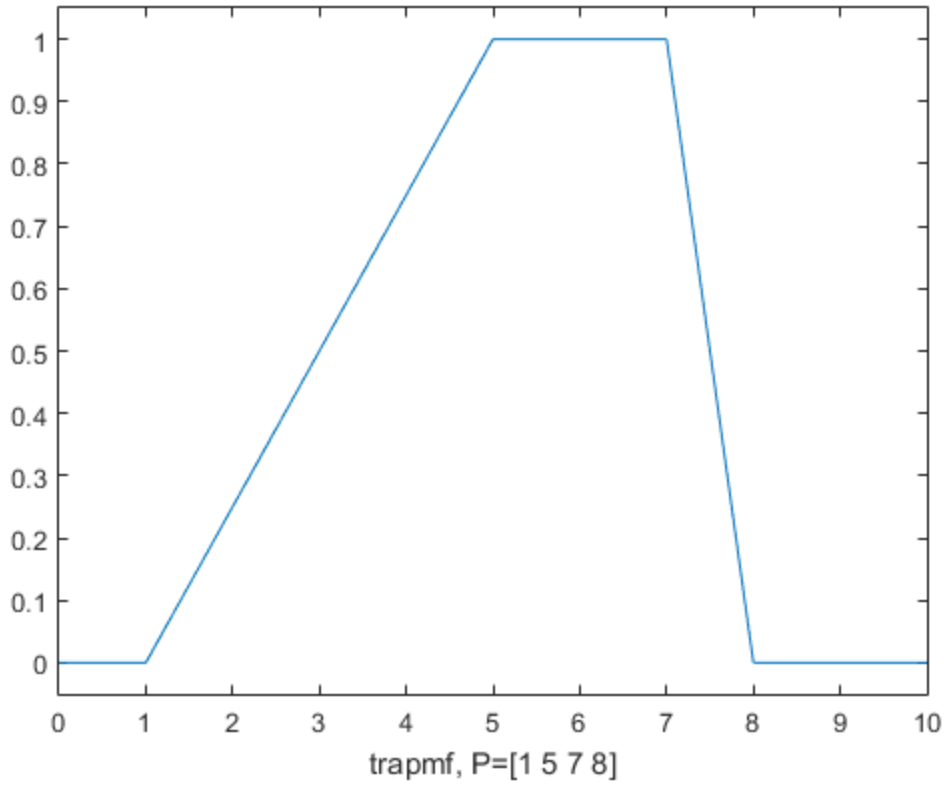
Examples

Trapezoid-Shaped Membership Function

```
x = 0:0.1:10;
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
y = trapmf(x,[1 5 7 8]);
plot(x,y)
xlabel('trapmf, P=[1 5 7 8]')
ylim([-0.05 1.05])
```



See Also

dsigmf | gaussmf | gauss2mf | gbellmf | evalmf | mf2mf | pimf | psigmf | sigmf | smf | trimf | zmf

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

trimf

Triangular-shaped membership function

Syntax

```
y = trimf(x,params)
```

```
y = trimf(x,[a b c])
```

Description

The triangular curve is a function of a vector, `x`, and depends on three scalar parameters a, b, and c, as given by

$$f(x;a,b,c)=\begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases}$$

or, more compactly, by

$$f(x;a,b,c)=\max\left(\min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),o\right)$$

The parameters a and c locate the "feet" of the triangle and the parameter b locates the peak.

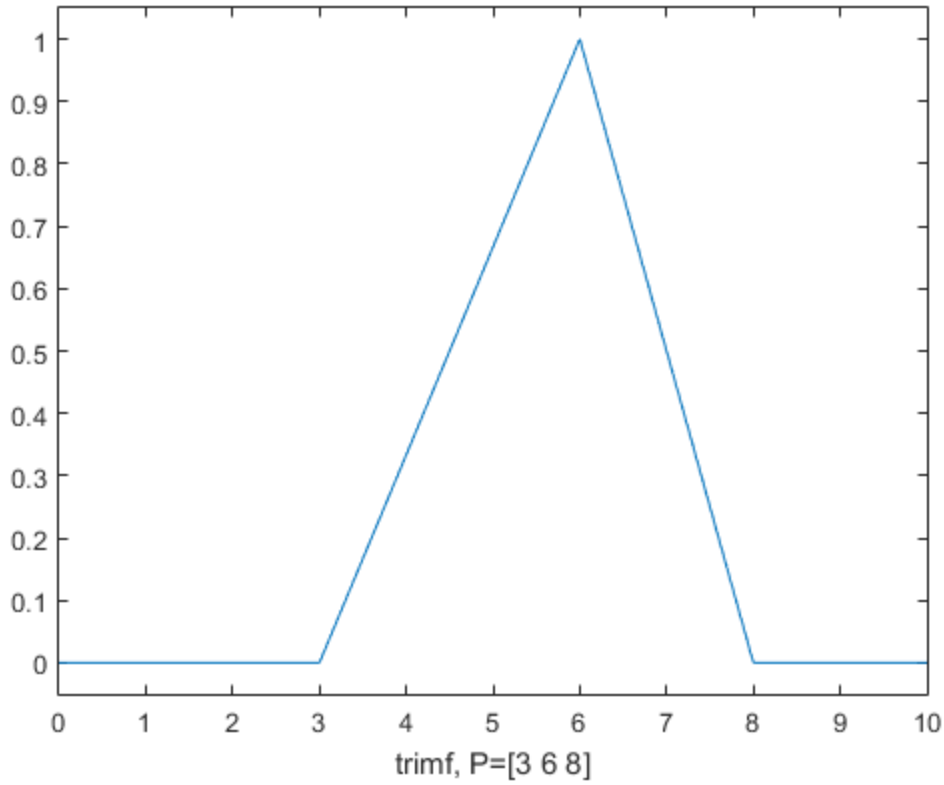
Examples

Triangle-Shaped Membership Function

```
x = 0:0.1:10;
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

```
y = trimf(x,[3 6 8]);
plot(x,y)
xlabel('trimf, P=[3 6 8]')
ylim([-0.05 1.05])
```



See Also

dsigmf | gaussmf | gauss2mf | gbellmf | evalmf | mf2mf | pimf | psigmf | sigmf | smf | trapmf

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

writefis

Save Fuzzy Inference System to file

Syntax

```
writefis(fismat)
writefis(fismat,'filename')
writefis(fismat,'filename','dialog')
```

Description

`writefis` saves a MATLAB workspace FIS structure, `fismat`, as a `.fis` file.

`writefis(fismat)` opens a dialog box to assist with the naming and folder location of the file.

`writefis(fismat,'filename')` writes a `.fis` file corresponding to the FIS structure, `fismat`, to a file called `filename.fis`. No dialog box appears, and the file is saved to the current folder.

`writefis(fismat,'filename','dialog')` opens a dialog box with the default name `filename.fis` supplied.

The extension `.fis` is only added to `filename` if it is not already included in the name.

Examples

```
a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
writefis(a,'my_file')
```

See Also

`readfis`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

zmf

Z-shaped membership function

Syntax

```
y = zmf(x,[a b])
```

Description

This spline-based function of x is so named because of its Z-shape. The parameters a and b locate the extremes of the sloped portion of the curve as given by.

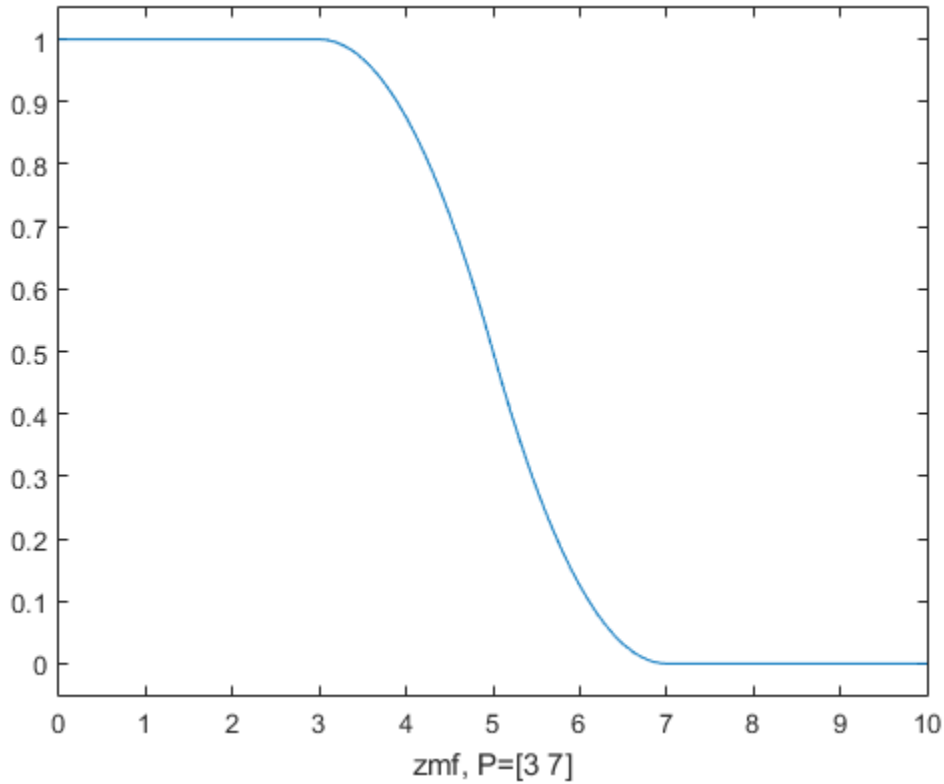
$$f(x;a,b)=\begin{cases} 1, & x \le a \\ 1-2\left(\dfrac{x-a}{b-a}\right)^2, & a \le x \le \dfrac{a+b}{2} \\ 2\left(\dfrac{x-b}{b-a}\right)^2, & \dfrac{a+b}{2} \le x \le b \\ 0, & x \ge b \end{cases}$$

Examples

Z-Shaped Membership Function

```
x = 0:0.1:10;
y = zmf(x,[3 7]);
plot(x,y)
xlabel('zmf, P=[3 7]')
ylim([-0.05 1.05])
```


www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران



See Also

`dsigmf` | `gaussmf` | `gauss2mf` | `gbellmf` | `evalmf` | `mf2mf` | `pimf` | `psigmf` | `sigmf` | `smf` | `trapmf` | `trimf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

6

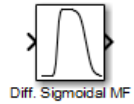
Blocks — Alphabetical List

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Diff. Sigmoidal MF

Difference of two sigmoids membership function in Simulink software

Description



The Diff. Sigmoidal MF block implements a membership function in Simulink based on the difference between two sigmoids. The two sigmoid curves are given by

$$f_k(x) = \frac{1}{1 + \exp(-a_k(x - c_k))}$$

where k=1,2. The parameters $a_1$and a_2 control the slopes of the left and right curves. The parameters c_1 and c_2 control the points of inflection for the left and right curves. The parameters a_1 and a_2 should be positive.

See Also

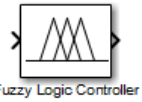
`dsigmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Fuzzy Logic Controller

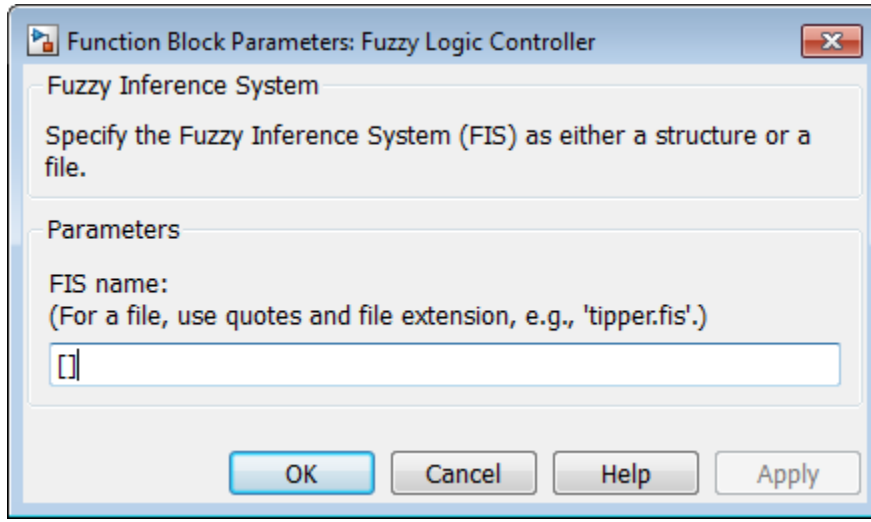
Fuzzy inference system in Simulink software

Description



The Fuzzy Logic Controller block implements a fuzzy inference system (FIS) in Simulink. See "Simulate Fuzzy Inference Systems in Simulink" on page 2-78 for a discussion of how to use this block.

Dialog Box



FIS name

Specify the FIS as one of the following:

- Structure — Specify the name of the FIS structure variable. For example, `fismat`.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

- File — Specify the file name in quotes and include the file extension. For example, `'tipper.fis'`.

Tips

- You can generate embeddable C code for the Fuzzy Logic Controller block using Simulink Coder™ software.

See Also

Fuzzy Logic Controller with Ruleviewer | `newfis` | `readfis`

Related Examples

- "Simulate Fuzzy Inference Systems in Simulink" on page 2-78
- "Build Your Own Fuzzy Simulink Models"

More About

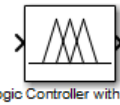
- "Fuzzy Inference Process"

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Fuzzy Logic Controller with Ruleviewer

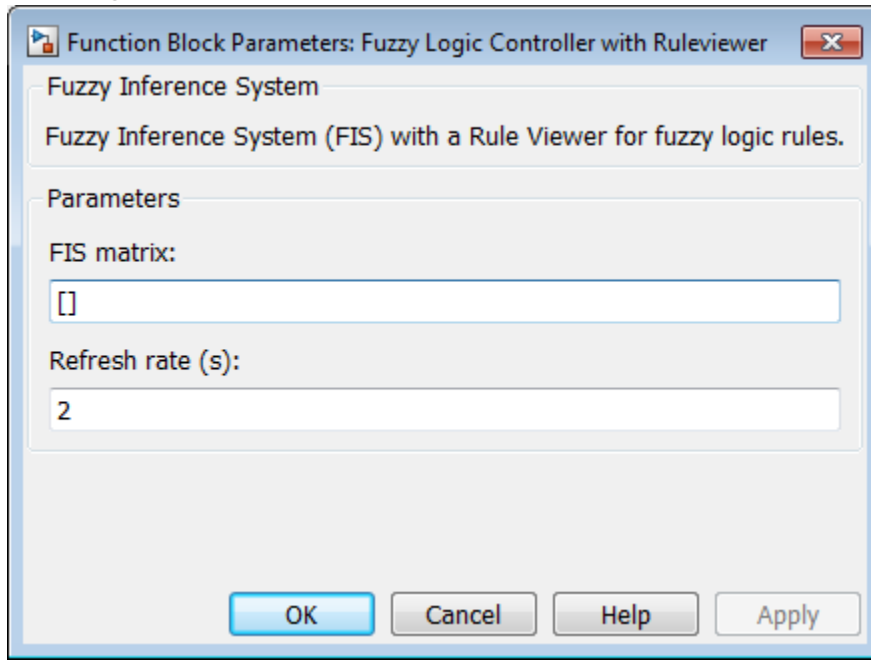
Fuzzy inference system with Ruleviewer in Simulink software

Description



The Fuzzy Logic Controller with Ruleviewer block implements a fuzzy inference system (FIS) with the Rule Viewer in Simulink. See "Simulate Fuzzy Inference Systems in Simulink" on page 2-78 for a discussion of how to use this block.

Dialog Box



FIS matrix

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Specify the name of the FIS structure variable. For example, `fismat`.

Refresh rate (sec)

Specify the refresh rate in seconds.

Tips

- You can generate embeddable C code for the Fuzzy Logic Controller with Ruleviewer block using Simulink Coder software.

See Also

Fuzzy Logic Controller

Related Examples

- "Simulate Fuzzy Inference Systems in Simulink"
- "Build Your Own Fuzzy Simulink Models"

More About

- "Fuzzy Inference Process"

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Gaussian MF

Gaussian membership function in Simulink software

Description

The Gaussian MF block implements a membership function in Simulink based on a symmetric Gaussian. The Gaussian curve is given by

$$f(x) = \exp\left(\frac{-0.5(x-c)^2}{\sigma^2}\right)$$

where c is the mean and σ is the variance.

See Also

`gaussmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Gaussian2 MF

Combination of two Gaussian membership functions in Simulink software

Description

The Gaussian2 MF block implements a membership function based on a combination of two Gaussian functions. The two Gaussian functions are given by

$$f_k(x) = \exp\left(\frac{-0.5(x - c_k)^2}{\sigma_k^{\,2}}\right)$$

where k=1,2. The parameters c_1 and σ_1 are the mean and variance defining the left-most curve. The parameters c_2 and σ_2 are the mean and variance defining the right-most curve.

See Also

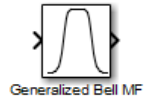
`gauss2mf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Generalized Bell MF

Generalized bell membership function in Simulink software

Description



The Generalized Bell MF block implements a membership function in Simulink based on a generalized bell-shaped curve. The generalized bell-shaped curve is given by

$$f(x) = \frac{1}{1 + \left| \frac{x - c}{a} \right|^{2b}}$$

where the parameters a and b vary the width of the curve and the parameter c locates the center of the curve. The parameter b should be positive.

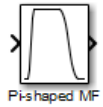
See Also

`gbellmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Pi-shaped MF

Pi-shaped membership function in Simulink software



Description

The Pi-shaped MF block implements a membership function in Simulink based on a spline-based curve, so named because of its Π shape. The parameters a and d locate the left and right base points or "feet" of the curve. The parameters b and c set the left and right top point or "shoulders" of the curve.

See Also

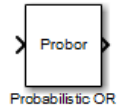
`pimf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Probabilistic OR

Probabilistic OR function in Simulink software

Description



The Probabilistic OR block outputs the probabilistic OR value for the vector signal input, based on

$$y = 1 - \mathrm{prod}(1 - x)$$

See Also

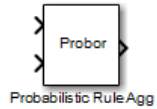
`probor`, Probabilistic Rule Agg

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Probabilistic Rule Agg

Probabilistic OR function, rule aggregation method

Description



The Probabilistic Rule Agg block outputs the element-wise(.*) probabilistic OR value of the two inputs based on

$$y = 1 - \mathrm{prod}\left(1 - [a;b]\right)$$

The two inputs, a and b, are row vectors.

See Also

`probor`, Probabilistic OR

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Prod. Sigmoidal MF

Product of two sigmoid membership functions in Simulink software

Description

The Prod. Sigmoidal MF block implements a membership function based on the product of two sigmoidal curves. The two sigmoidal curves are given by

$$f_k(x) = \frac{1}{1 + \exp\left(-a_k\left(x - c_k\right)\right)}$$

where k=1,2 The parameters a_1 and a_2 control the slopes of the left and right curves. The parameters c_1 and c_2 control the points of inflection for the left and right curves. Parameters a_1 and a_2 should be positive and negative respectively.

See Also

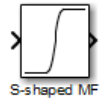
`psigmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

S-shaped MF

S-shaped membership function in Simulink software

Description



The S-shaped MF block implements an S-shaped membership function in Simulink. Going from left to right the function increases from 0 to 1. The parameters a and b locate the left and right extremes of the sloped portion of the curve.

See Also

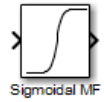
smf

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Sigmoidal MF

Sigmoidal membership function in Simulink software

Description



The Sigmoidal MF block implements a sigmoidal membership function given by

$$f(x) = \frac{1}{1 + \exp(-a(x - c))}$$

When the sign of a is positive the curve increases from left to right. Conversely, when the sign of a is negative the curve decreases from left to right. The parameter c sets the point of inflection of the curve.

See Also

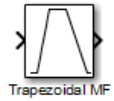
`sigmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Trapezoidal MF

Trapezoidal membership function in Simulink software

Description



The Trapezoidal MF block implements a trapezoidal-shaped membership function. The parameters a and d set the left and right “feet,” or base points, of the trapezoid. The parameters b and c set the “shoulders,” or top of the trapezoid.

See Also

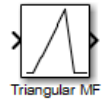
`trapmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Triangular MF

Triangular membership function in Simulink software

Description



The Triangular MF block implements a triangular-shaped membership function. The parameters a and c set the left and right “feet,” or base points, of the triangle. The parameter b sets the location of the triangle peak.

See Also

`trimf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

Z-shaped MF

Z-shaped membership function in Simulink software

Description

The Z-shaped MF block implements a Z-shaped membership function. Going from left to right the function decreases from 1 to 0. The parameters a and b locate the left and right extremes of the sloped portion of the curve.

See Also

`trimf`, `zmf`

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

A

Bibliography

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

[1] Bezdec, J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms,* Plenum Press, New York, 1981.

[2] Chiu, S., “Fuzzy Model Identification Based on Cluster Estimation,” *Journal of Intelligent & Fuzzy Systems*, Vol. 2, No. 3, Spet. 1994.

[3] Dubois, D. and H. Prade, *Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York, 1980.

[4] Jang, J.-S. R., “Fuzzy Modeling Using Generalized Neural Networks and Kalman Filter Algorithm,” *Proc. of the Ninth National Conf. on Artificial Intelligence (AAAI-91)*, pp. 762-767, July 1991.

[5] Jang, J.-S. R., “ANFIS: Adaptive-Network-based Fuzzy Inference Systems,” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, pp. 665-685, May 1993.

[6] Jang, J.-S. R. and N. Gulley, “Gain scheduling based fuzzy controller design,” *Proc. of the International Joint Conference of the North American Fuzzy Information Processing Society Biannual Conference, the Industrial Fuzzy Control and Intelligent Systems Conference, and the NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic*, San Antonio, Texas, Dec. 1994.

[7] Jang, J.-S. R. and C.-T. Sun, “Neuro-fuzzy modeling and control, *Proceedings of the IEEE*, March 1995.

[8] Jang, J.-S. R. and C.-T. Sun, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, 1997.

[9] Kaufmann, A. and M.M. Gupta, *Introduction to Fuzzy Arithmetic*, V.N. Reinhold, 1985.

[10] Lee, C.-C., “Fuzzy logic in control systems: fuzzy logic controller-parts 1 and 2,” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 2, pp 404-435, 1990.

[11] Mamdani, E.H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, Vol. 7, No. 1, pp. 1-13, 1975.

[12] Mamdani, E.H., “Advances in the linguistic synthesis of fuzzy controllers,” *International Journal of Man-Machine Studies*, Vol. 8, pp. 669-678, 1976.

www.MatlabSite.com | متلب سایت

مرجع آموزش برنامه‌نویسی متلب در ایران

[13] Mamdani, E.H., "Applications of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Transactions on Computers*, Vol. 26, No. 12, pp. 1182-1191, 1977.

[14] Schweizer, B. and A. Sklar, "Associative functions and abstract semi-groups," *Publ. Math Debrecen*, 10:69-81, 1963.

[15] Sugeno, M., "Fuzzy measures and fuzzy integrals: a survey," (M.M. Gupta, G. N. Saridis, and B.R. Gaines, editors) *Fuzzy Automata and Decision Processes*, pp. 89-102, North-Holland, NY, 1977.

[16] Sugeno, M., *Industrial applications of fuzzy control*, Elsevier Science Pub. Co., 1985.

[17] Wang, L.-X., *Adaptive fuzzy systems and control: design and stability analysis*, Prentice Hall, 1994.

[18] Widrow, B. and D. Stearns, *Adaptive Signal Processing*, Prentice Hall, 1985.

[19] Yager, R., "On a general class of fuzzy connectives," *Fuzzy Sets and Systems*, 4:235-242, 1980.

[20] Yager, R. and D. Filev, "Generation of Fuzzy Rules by Mountain Clustering," *Journal of Intelligent & Fuzzy Systems*, Vol. 2, No. 3, pp. 209-219, 1994.

[21] Zadeh, L.A., "Fuzzy sets," *Information and Control*, Vol. 8, pp. 338-353, 1965.

[22] Zadeh, L.A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 3, No. 1, pp. 28-44, Jan. 1973.

[23] Zadeh, L.A., "The concept of a linguistic variable and its application to approximate reasoning, Parts 1, 2, and 3," *Information Sciences*, 1975, 8:199-249, 8:301-357, 9:43-80.

[24] Zadeh, L.A., "Fuzzy Logic," *Computer*, Vol. 1, No. 4, pp. 83-93, 1988.

[25] Zadeh, L.A., "Knowledge representation in fuzzy logic," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 1, pp. 89-100, 1989.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

متلب سایت | www.MatlabSite.com
مرجع آموزش برنامه‌نویسی متلب در ایران

Glossary

Adaptive Neuro-Fuzzy Inference System (ANFIS) A technique for automatically tuning Sugeno-type inference systems based on training data.

aggregation The combination of the consequents of each rule in a Mamdani fuzzy inference system in preparation for defuzzification.

antecedent The initial (or "if") part of a fuzzy rule.

consequent The final (or "then") part of a fuzzy rule.

defuzzification The process of transforming a fuzzy output of a fuzzy inference system into a crisp output.

degree of fulfillment See **firing strength**

degree of membership The output of a membership function, this value is always limited to between 0 and 1. Also known as a membership value or membership grade.

firing strength The degree to which the antecedent part of a fuzzy rule is satisfied. The firing strength may be the result of an AND or an OR operation, and it shapes the output function for the rule. Also known as *degree of fulfillment*.

fuzzification The process of generating membership values for a fuzzy variable using membership functions.

fuzzy c-means clustering A data clustering technique wherein each data point belongs to a cluster to a degree specified by a membership grade.

fuzzy inference system (FIS) The overall name for a system that uses fuzzy reasoning to map an input space to an output space.

fuzzy operators AND, OR, and NOT operators. These are also known as *logical connectives*.

fuzzy set A set that can contain elements with only a partial degree of membership.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

fuzzy singleton
A fuzzy set with a membership function that is unity at a one particular point and zero everywhere else.

implication
The process of shaping the fuzzy set in the consequent based on the results of the antecedent in a Mamdani-type FIS.

Mamdani-type inference
A type of fuzzy inference in which the fuzzy sets from the consequent of each rule are combined through the aggregation operator and the resulting fuzzy set is defuzzified to yield the output of the system.

membership function (MF)
A function that specifies the degree to which a given input belongs to a set or is related to a concept.

singleton output function
An output function that is given by a spike at a single number rather than a continuous curve. In the Fuzzy Logic Toolbox software, it is only supported as part of a zero-order Sugeno model.

subtractive clustering
A technique for automatically generating fuzzy inference systems by detecting clusters in input-output training data.

Sugeno-type inference
A type of fuzzy inference in which the consequent of each rule is a linear combination of the inputs. The output is a weighted linear combination of the consequents.

T-conorm
A two-input function that describes a superset of fuzzy union (OR) operators, including maximum, algebraic sum, and any of several parameterized T-conorms Also known as *S-norm*.

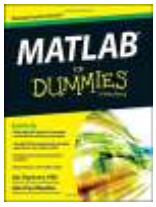
T-norm
A two-input function that describes a superset of fuzzy intersection (AND) operators, including minimum, algebraic product, and any of several parameterized T-norms.

www.MatlabSite.com | متلب سایت
مرجع آموزش برنامه‌نویسی متلب در ایران

معرفی چند منبع در زمینه آموزش برنامه نویسی MATLAB یا متلب

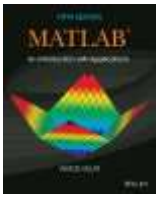
کتاب های به زبان انگلیسی

عنوان: Matlab, Third Edition: A Practical Introduction to Programming and Problem Solving
ترجمه عنوان: متلب: مقدمه ای عملی بر برنامه نویسی و حل مساله، چاپ سوم
مولفین: Stormy Attaway
سال چاپ: 2013
انتشارات: Butterworth-Heinemann
لینک دسترسی: لینک



عنوان: MATLAB For Dummies
ترجمه عنوان: تلب به زبان ساده
مولفین: Jim Sizemore, John Paul Mueller
سال چاپ: ۲۰۱۴
انتشارات: For Dummies
لینک دسترسی: لینک

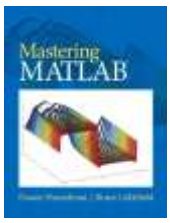
عنوان: Essential MATLAB for Engineers and Scientists
ترجمه عنوان: آنچه باید مهندسین و دانشمندان از متلب بدانند
مولفین: Brian Hahn, Daniel Valentine
سال چاپ: 2013
انتشارات: Academic Press
لینک دسترسی: لینک



عنوان: MATLAB: An Introduction with Applications
ترجمه عنوان: مقدمه ای بر متلب و کاربردهای آن
مولف: Amos Gilat
سال چاپ: 2014
انتشارات: Wiley
لینک دسترسی: لینک

عنوان: MATLAB For Beginners: A Gentle Approach
ترجمه عنوان: متلب برای افراد مبتدی با یک رویکرد تدریجی
مولف: Peter I. Kattan
سال چاپ: 2008
انتشارات: CreateSpace Independent Publishing Platform
لینک دسترسی: لینک

عنوان: MATLAB for Engineers
ترجمه عنوان: متلب برای مهندسین
مولف: Holly Moore
سال چاپ: 2011
انتشارات: Prentice Hall
لینک دسترسی: لینک

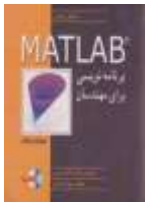


عنوان: Mastering MATLAB
ترجمه عنوان: تسلط بر متلب
مولفین: Duane C. Hanselman, Bruce L. Littlefield
سال چاپ: 2011
انتشارات: Prentice Hall
لینک دسترسی: لینک

کتاب های به زبان فارسی

عنوان: اصول و مبانی متلب برای علوم مهندسی
مولفین: برایان هان، دانیل تی، والنتین
مترجمین: رامین مولانا پور، سارا مولاناپور، نینا اسدی پور
انتشارات: سها دانش
لینک دسترسی: لینک

عنوان: کاربرد MATLAB در علوم مهندسی
مولفین: حیدرعلی شایانفر، حسین شایقی
انتشارات: یاوریان
لینک دسترسی: لینک



عنوان: برنامه نویسی MATLAB برای مهندسان
مولفین: محمود کشاورز مهر، بهزاد عبدی
انتشارات: نوپردازان
لینک دسترسی: لینک

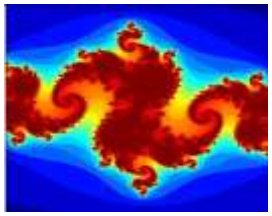
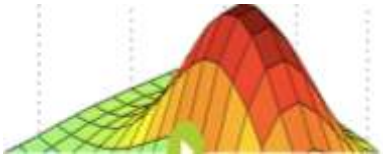
عنوان: آموزش کاربردی مباحث پیشرفته با MATLAB
مولفین: نیما جمشیدی، علی ابویی مهریزی، رسول مولایی
انتشارات: عابد
لینک دسترسی: لینک

عنوان: کاملترین مرجع آموزشی و کاربردی MATLAB
مولفین: علی اکبر علمداری، نسرین علمداری
انتشارات: نگارنده دانش
لینک دسترسی: لینک

عنوان: برنامه نویسی MATLAB برای مهندسین
مولف: استفن چاپمن
مترجم: سعدان زکائی
انتشارات: دانشگاه صنعتی خواجه نصیرالدین طوسی
لینک دسترسی: لینک

عنوان: آموزش گام به گام محاسبات عددی با متلب
مولف: کلیو مولر
مترجم: رسول نصیری
انتشارات: نشر گستر
لینک دسترسی: لینک

منابع آموزشی آنلاین

عنوان: مجموعه فرادرس‌های برنامه‌نویسی متلب **مدرس:** دکتر سید مصطفی کلامی هریس **مدت زمان:** ۹ ساعت و ۳ دقیقه **زبان:** فارسی **ارائه دهنده:** فرادرس **لینک دسترسی:** لینک	
عنوان: مجموعه فرادرس‌های متلب برای علوم و مهندسی **مدرس:** دکتر سید مصطفی کلامی هریس **مدت زمان:** ۱۴ ساعت و ۲۷ دقیقه **زبان:** فارسی **ارائه دهنده:** فرادرس **لینک دسترسی:** لینک	
عنوان: مجموعه فرادرس‌های برنامه نویسی متلب پیشرفته **مدرس:** دکتر سید مصطفی کلامی هریس **مدت زمان:** ۷ ساعت و ۱۲ دقیقه **زبان:** فارسی **ارائه دهنده:** فرادرس **لینک دسترسی:** لینک	
عنوان: Introduction to Programming with MATLAB **ترجمه عنوان:** آشنایی با برنامه‌نویسی متلب **مدرسین:** Akos Ledeczi, Michael Fitzpatrick, Robert Tairas **زبان:** انگلیسی **ارائه دهنده:** Vanderbilt University **لینک دسترسی:** لینک	
عنوان: Introduction to MATLAB **ترجمه عنوان:** مقدمه‌ای بر متلب **مدرس:** Danilo Šćepanović **زبان:** انگلیسی **ارائه دهنده:** MIT OCW **لینک دسترسی:** لینک	
عنوان: Up and Running with MATLAB **ترجمه عنوان:** شروع سریع کار با متلب **مدرس:** Patrick Royal **زبان:** انگلیسی **ارائه دهنده:** lynda.com **لینک دسترسی:** لینک	
عنوان: Modelling and Simulation using MATLAB **ترجمه عنوان:** مدل‌سازی و شبیه‌سازی با استفاده از متلب **مدرسین:** Prof. Dr.-Ing. Georg Fries و دیگران **زبان:** انگلیسی **ارائه دهنده:** iversity.org **لینک دسترسی:** لینک