# G53FUZ
# Fuzzy Sets and Systems

**TSK Inference and Fuzzy Control**

Bob John
*ASAP Research Group*

# TSK Inference Overview

## Overview

- Overview of TSK inference
  - zeroth-order inference models
- Comparison of TSK and Mamdani inference
  - performance (output) comparison
  - efficiency (speed) comparison
- Higher-order TSK models
- Fuzzy logic control
  - PID control
  - overview of fuzzy methodology
  - motor control examples

## Background

- Takagi, Sugeno and Kang introduced the second principal method of fuzzy reasoning in 1985
  - "Fuzzy Identification of Systems and its Applications To Modeling And Control", *IEEE Transactions on Systems Man and Cybernetics*, 15(1), 116-132, 1985
- It is quite similar to Mamdani inference, but avoids the need for defuzzification
- Also introduced in the context of fuzzy control
  - based on control principles rather than logic
- It is also known as TSK inference

## Zeroth-Order

- Rules in a zeroth-order Sugeno are of the form

  IF $x$ is $A_1$ AND/OR $y$ is $B_1$ ... THEN $z$ is $k_1$

  IF $x$ is $A_2$ AND/OR $y$ is $B_2$ ... THEN $z$ is $k_2$

  ...

  IF $x$ is $A_n$ AND/OR $y$ is $B_n$ ... THEN $z$ is $k_n$

  - $A$, $B$, ... are fuzzy sets in the antecedent
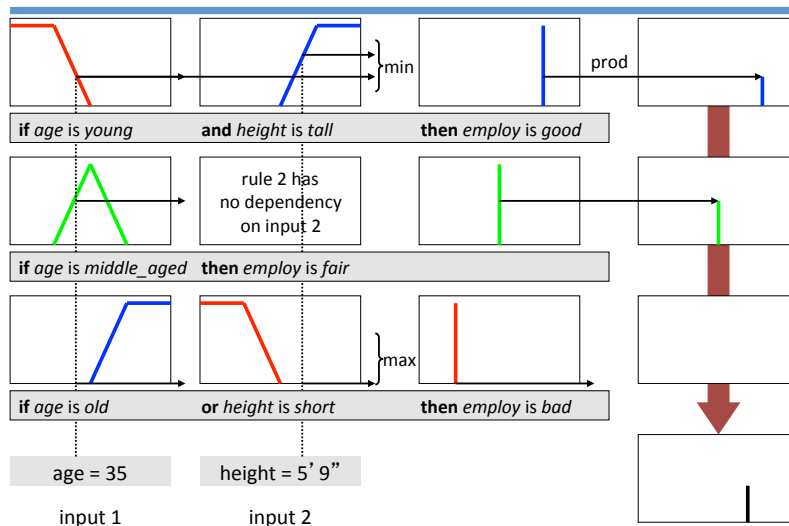  - each $k_i$ for $i = 1 ... n$ is a constant in the consequent

## Zeroth-Order

- The antecedents are evaluated as per Mamdani to find the firing strength (*truth*) of each rule
  - $w_i$
- The weighted average of the $k_i$ associated with each rule gives the overall output

$$f = \frac{\sum_{i=1}^{n} w_i k_i}{\sum_{i=1}^{n} w_i}$$

  - other 'defuzzifications' are possible, but rare

## TSK Example



if *age* is *young*  and *height* is *tall*  then *employ* is *good*  {min  {prod

rule 2 has no dependency on input 2

if *age* is *middle_aged*  then *employ* is *fair*

if *age* is *old*  or *height* is *short*  then *employ* is *bad*  {max

age = 35  height = 5' 9"

input 1  input 2

## Comparison of Methods

# Example: Variables

- Age
  - *young* = 1/0 + 1/10 + .75/20 +.5/30 + .25/40
  - *middle_aged* = 0/30 + .5/40 + 1/50 +.5/60 + 0/70
  - *old* = .25/60 + .5/70 + .75/80 + 1/90 + 1/100
- Height
  - *short* = 1/1.4 + .75/1.5 + .5/1.6 + .25/1.7 + 0/1.8
  - *tall* = .25/1.6 + .5/1.7 + .75/1.8 + 1/1.9 + 1/2.0
- Employ
  - *bad* = 2
  - *fair* = 5
  - *good* = 8



# Example: Rules

- Three rules
  - IF *Age* is *young* AND *Height* is *tall* THEN *Employ* is *good*
  - IF *Age* is *middle_aged* THEN *Employ* is *fair*
  - IF *Age* is *old* OR *Height* is *short* THEN *Employ* is *bad*
- Inputs
  - Age = 40 (years)
  - Height = 1.8 (metres)
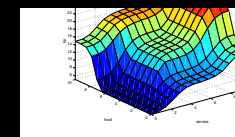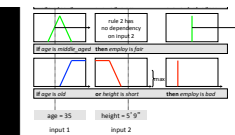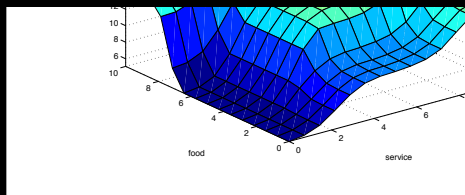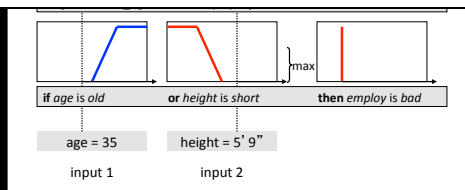
# Rule 1

- Antecedent 1
  - *Age* is *young*: $\mu_{young}(40) = 0.25$
- Antecedent 2
  - *Height* is *tall*: $\mu_{tall}(1.8) = 0.75$
- Rule strength = $\text{Ante}_1$ AND $\text{Ante}_2$
  - $\mu_1 = \min(0.25, 0.75) = 0.25$
- Consequent
  - *Employ* is *good*
    - $R_1 = 0.25 * 8 = 2$

# Rule 2

- Antecedent 1
  - *Age* is *middle_aged*: $\mu_{middle\_aged}(40) = 0.5$
- Antecedent 2
  - *BLANK*
- Rule strength = $\text{Ante}_1$
  - $\mu_2 = 0.5$
- Consequent
  - *Employ* is *fair*
    - $R_2 = 0.5 * 5 = 2.5$

# Rule 3

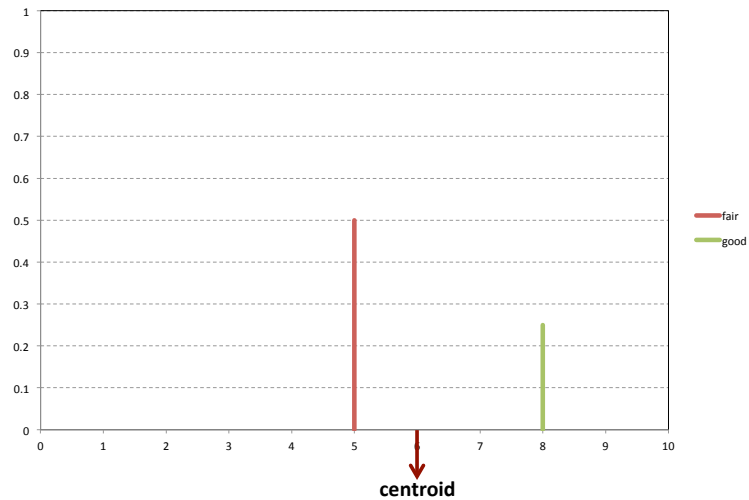- Antecedent 1
  - *Age* is *old*: $\mu_{old}(40) = 0$
- Antecedent 2
  - *Height* is *short*: $\mu_{short}(1.8) = 0$
- Rule strength = $\text{Ante}_1$ OR $\text{Ante}_2$
  - $\mu_3 = \max(0, 0) = 0$
- Consequent
  - *Employ* is *bad*
    - $R_3 = 0$

# Rule Combination

- The three rule results
  - $R_1$: 2
  - $R_2$: 2.5
  - $R_3$: 0
- Rule combination
  - $(R_1 + R_2 + R_3) / (\mu_1 + \mu_2 + \mu_3)$
  - $(2 + 2.5 + 0) / (.25 + .5 + 0)$
  - $4.5 / 0.75$
  - 6
- Same answer as Mamdani with centroid defuzzification!

# Visually



centroid

# Tipper Example

- Recall the standard tipping example
  - If *Service* is *poor* or *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*
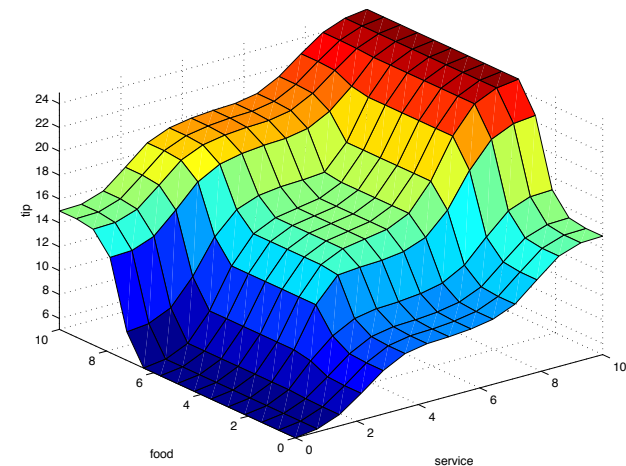- Output sets
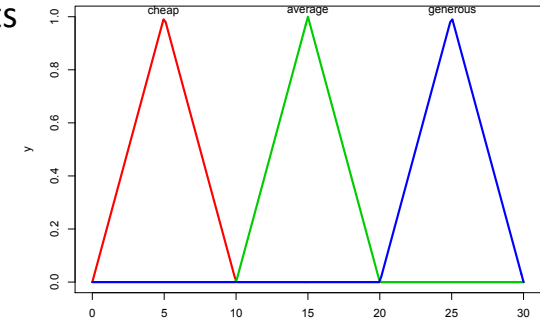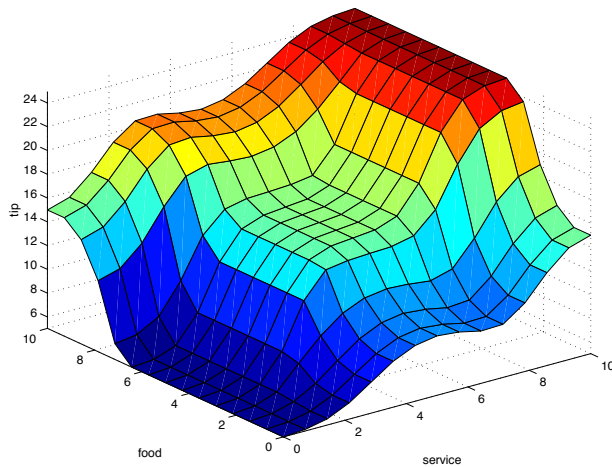


# TSK Equivalent

- Recall the standard tipping example
  - If *Service* is *poor* or *Food* is *rancid* then *Tip* is *cheap* (5)
  - If *Service* is *good* then *Tip* is *average* (15)
  - If *Service* is *excellent* or *Food* is *delicious* then *Tip* is *generous* (25)
- Output sets
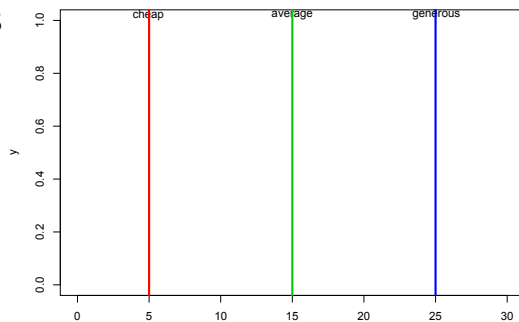


# Tipper Surface

## TSK Surface



## Efficiency Comparison

|  | **Mamdani** | **Sugeno** |
|---|---|---|
| Antecedents | antecedent evaluation is the same | |
| Rule Evaluation | implication operates across consequent set<br>e.g. 101 min operations across universe (per rule) | implication is single multiplication (perhaps three for 1st order) |
| Rule Combination | aggregation operates across each rule result set<br>e.g. 101 x #rules max operations | single summation |
| Defuzzification | numerator: multiplication and summation, e.g. 101 times<br>denominator: summation, e.g. 101 times | summation per rule<br>single division |

## Higher-Order Inference

## First-Order

- A first-order Sugeno system has rules of form

    IF $x$ is $A_1$ AND $y$ is $B_1$ THEN $z = p_1 x + q_1 y + r_1$
    IF $x$ is $A_2$ AND $y$ is $B_2$ THEN $z = p_2 x + q_2 y + r_2$

    …
    IF $x$ is $A_n$ AND $y$ is $B_n$ THEN $z = p_n x + q_n y + r_n$

    where $p_i$, $q_i$ and $r_i$ are constants

- Visualise such a first-order system by imagining each rule as defining a 'moving singleton'
    - these singletons move linearly in the output space and are then combined to form the final output
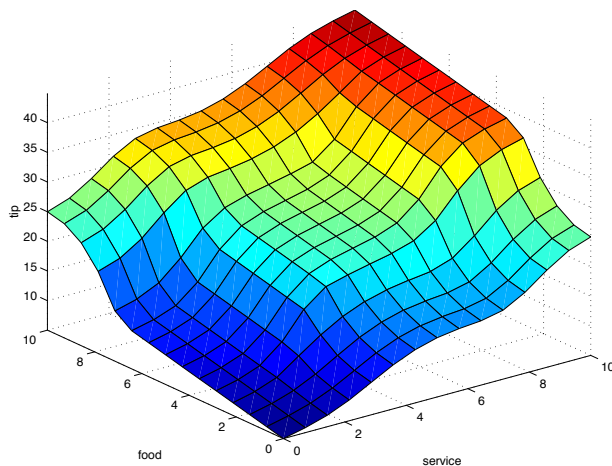
# First Order Tipper

- First-order Sugeno with $p=q=1$
  - If *Service* is *poor* or *Food* is *rancid* then *Tip = service + food + 5*
  - If *Service* is *good* then *Tip = service + food + 15*
  - If *Service* is *excellent* or *Food* is *delicious* then *Tip = service + food + 25*
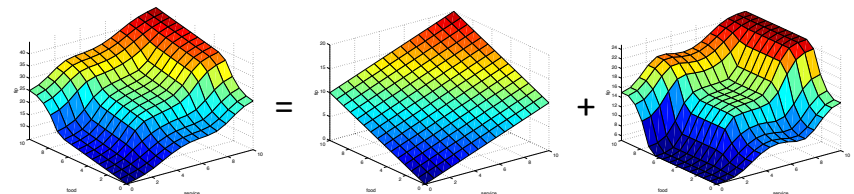
- Output sets
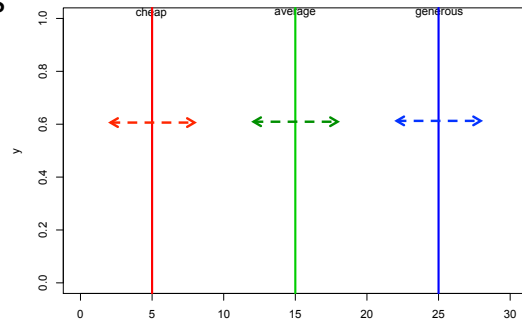


# TSK Surface
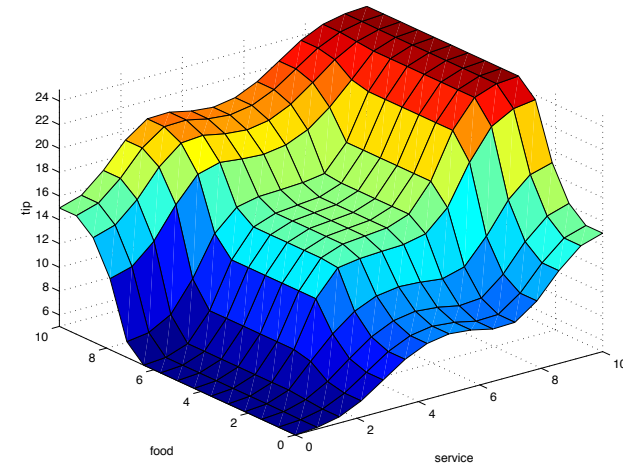


# First-Order TSK Surface



# Conceptualising the Output

- Roughly the same as a combination of the plane *tip= service + food* with the $0^{th}$-order system



- Different values of $p$ and $q$ adjust the incline of the plane on each axis

# Higher Orders

- Higher orders are possible
  - featuring $x^2$, $y^2$ and $xy$, etc., in the outputs

- These are extremely rare in practice
  - theoretical, rather than practical
  - not implemented in MATLAB (or R Fuzzy)

# Mamdani v. TSK

- Sugeno advantages
  - it's computationally efficient
  - it works well with linear techniques (e.g. PID control)
  - it works with optimisation & adaptive techniques
  - it has guaranteed continuity of the output surface
  - it's well-suited to mathematical analysis
  - it does not require defuzzification
- Mamdani advantages
  - it's simple and intuitive
  - it has widespread acceptance
  - it's well-suited to human input

# Fuzzy Logic Control (FLC)

# DC Motor

Input: voltage $v_{app}(t)$
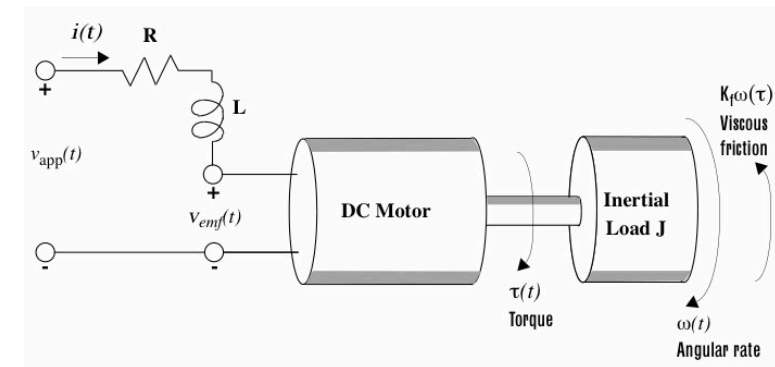

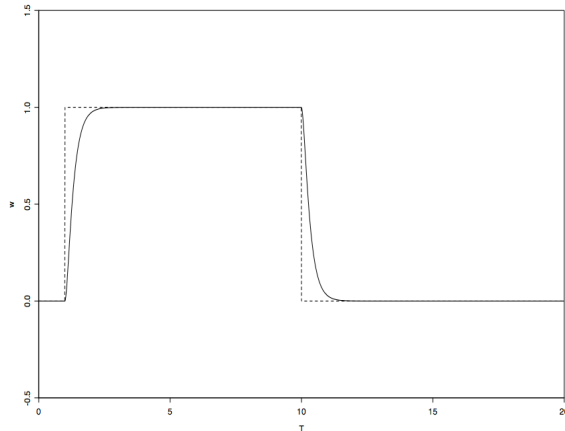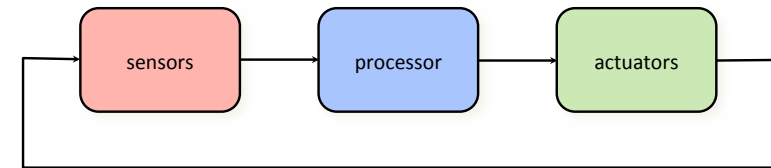
Output: angular velocity $\omega(t)$

## Open-Loop Control
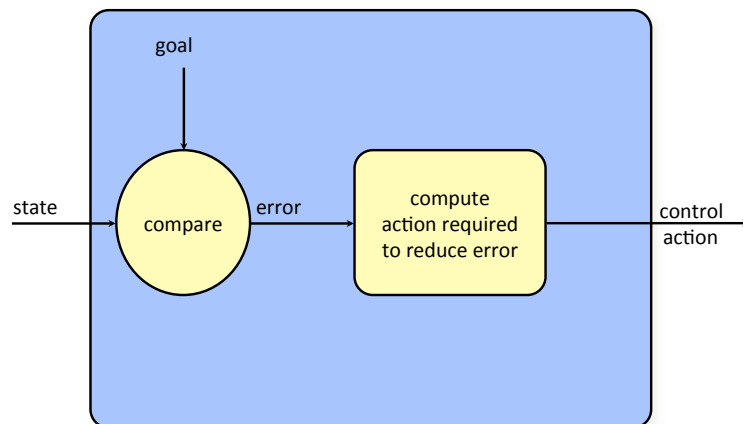
$v_{app}(t)$ = 26.681V



$\omega(t)$ reaches approx. 1 rotation per second

## The Sense-Think-Act Cycle



- Repeat
  - ‣ sense the current state
  - ‣ reduce difference between current state and goal state
- Until
  - ‣ current state = goal state
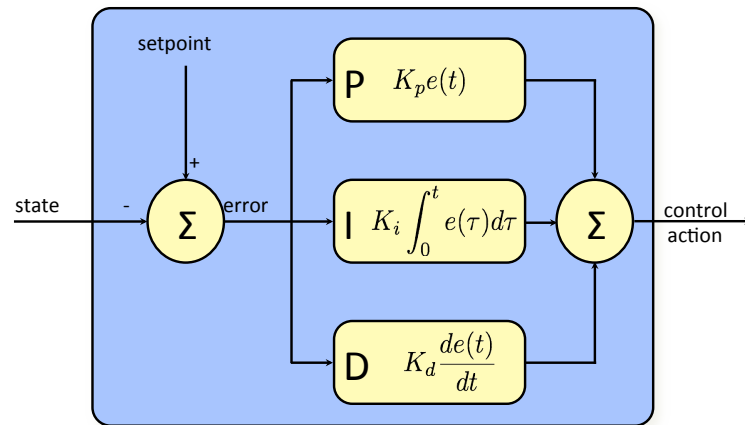
## The Sense-Think-Act Cycle



## PID Control

- Present
  - – the current error (**p**roportional)
- Past
  - – the sum of errors up to present time (**i**ntegral)
- Future
  - – the rate of change of the error (**d**erivative)
- A PID controller combines these three terms in a weighted sum to obtain the control action

$$CA(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

# PID Block Diagram

setpoint

$+$

state  $-$  $\Sigma$  error

P  $\quad K_p e(t)$

I  $\quad K_i \int_0^t e(\tau) d\tau$

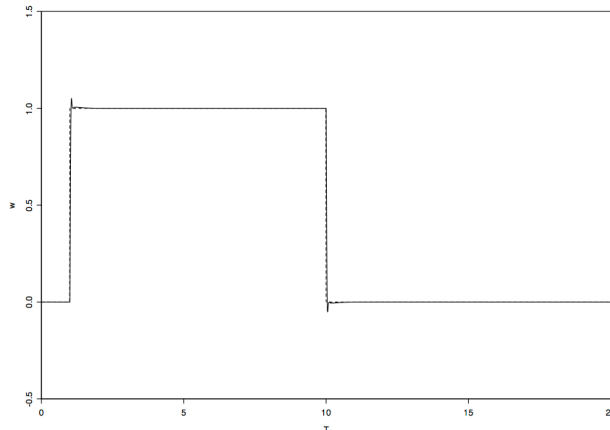D  $\quad K_d \dfrac{de(t)}{dt}$

$\Sigma$  control action

# Appropriate Parameters

- Some applications may not require the use of all three terms
  - appropriate parameters can be set to zero in order to remove unused terms
- PI, PD, P or I controllers
  - PI particularly common since
    - the integral term is required in order to remove steady-state error
    - the derivative term is very sensitive to measurement noise
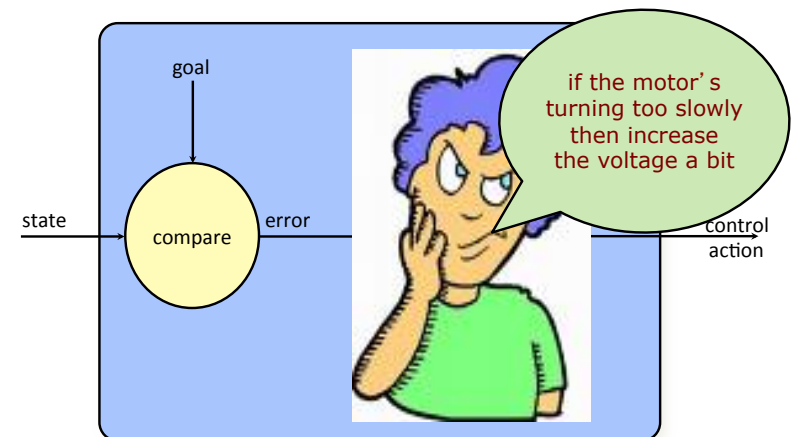
# Tuned PID Control

P = 300, I = 1000, D=20



PID tuned: exceedingly close control

# The Fuzzy Approach

goal

state  compare  error

if the motor's turning too slowly then increase the voltage a bit

control action

# Fuzzy Architecture

- A fuzzy inference system (FIS) takes one or more control parameters as inputs and produces the control action as output
  - inputs
    - error
    - delta error (rate of change of error)
  - output
    - control action
      - in a subtle change to PID control, the output of a fuzzy controller is often the change in control action
- Often referred to as fuzzy logic controller (FLC)

# Fuzzy Control Diagram
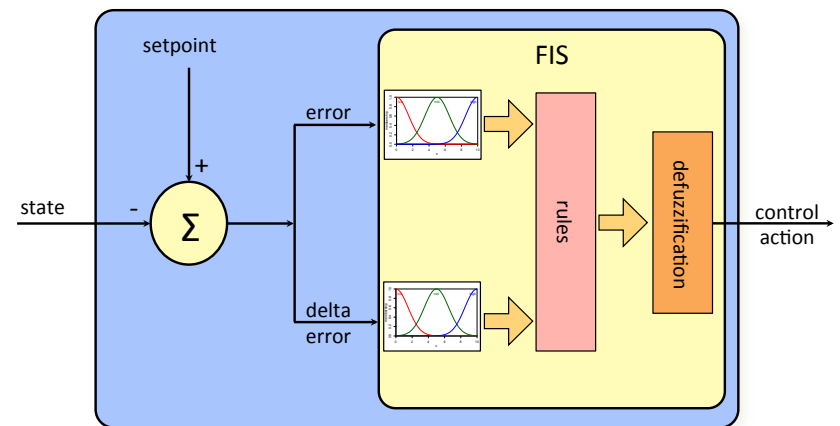


# Four-Rule System

- Each input
  Error
  Delta error
  has two m.f.s
  - neg(-ative) & pos(-itive)

- Output has three m.f.s
  - neg, zero & pos

- Four rules are used
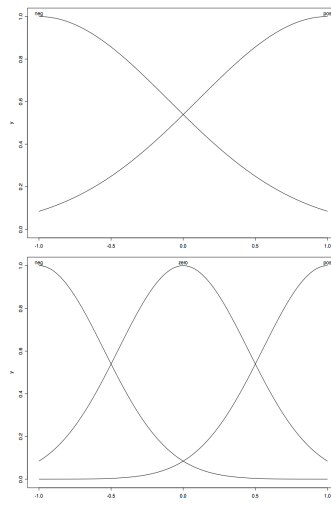


# Four Rules

1. If *Error* is *Neg*  and *Delta_error* is *Neg*  then *Output* is *Neg*
2. If *Error* is *Neg*  and *Delta_error* is *Pos*  then *Output* is *Zero*
3. If *Error* is *Pos*  and *Delta_error* is *Neg*  then *Output* is *Zero*
4. If *Error* is *Pos*  and *Delta_error* is *Pos*  then *Output* is *Pos*

|  |  | Delta_error | |
|---|---|---|---|
|  |  | neg | pos |
| Error | neg | neg | zero |
|  | pos | zero | pos |

# Motor Configuration 1



worse than open-loop control, but completely automatic

# Motor Configuration 2



faster rise time, but overshoot obvious

# Nine-Rule System

- Each input
  Error
  Delta error
  now has three m.f.s
  - neg., zero & pos.
- Output now has five m.f.s
  - neg. big, neg. small
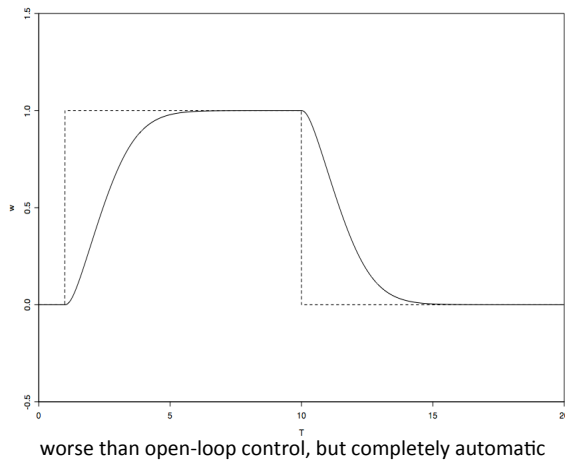  - zero
  - pos. big, pos. small
- Nine rules are needed



# Nine Rules

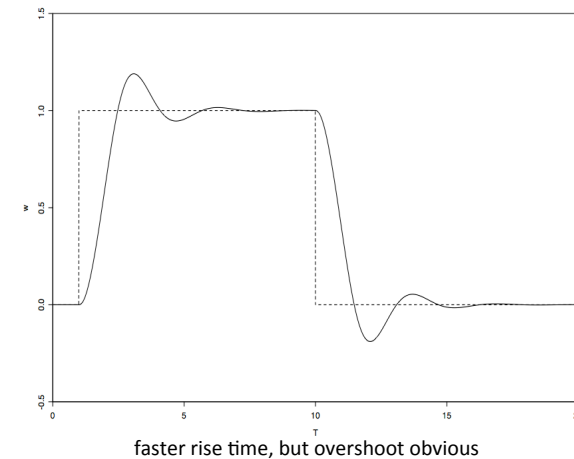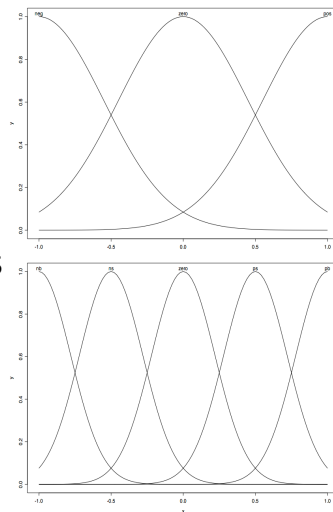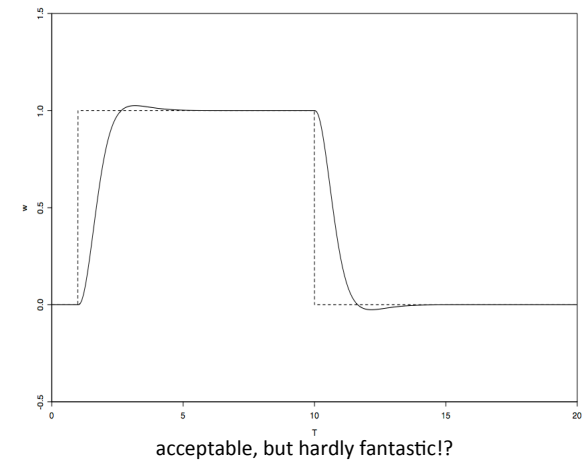1. If *Error* is *Neg*  and *Delta_error* is *Neg*  then *Output* is *NB*
2. If *Error* is *Neg*  and *Delta_error* is *Zero* then *Output* is *NM*
3. If *Error* is *Neg*  and *Delta_error* is *Pos*   then *Output* is *Zero*
4. If *Error* is *Zero* and *Delta_error* is *Neg*  then *Output* is *NM*
5. If *Error* is *Zero* and *Delta_error* is *Zero* then *Output* is *Zero*
6. If *Error* is *Zero* and *Delta_error* is *Pos*   then *Output* is *PM*
7. If *Error* is *Pos*  and *Delta_error* is *Neg*  then *Output* is *Zero*
8. If *Error* is *Pos*   and *Delta_error* is *Zero* then *Output* is *PM*
9. If *Error* is *Pos*   and *Delta_error* is *Pos*   then *Output* is *PB*

# Nine-Rule Table

|  |  | Delta_error | | |
|---|---|---|---|---|
|  |  | Neg | Zero | Pos |
| Error | Neg | NB | NS | Zero |
|  | Zero | NS | Zero | PS |
|  | Pos | Zero | PS | PB |

# Motor Configuration 1



acceptable, but hardly fantastic!?

# Motor Configuration 2



overshoot possibly beyond acceptable?
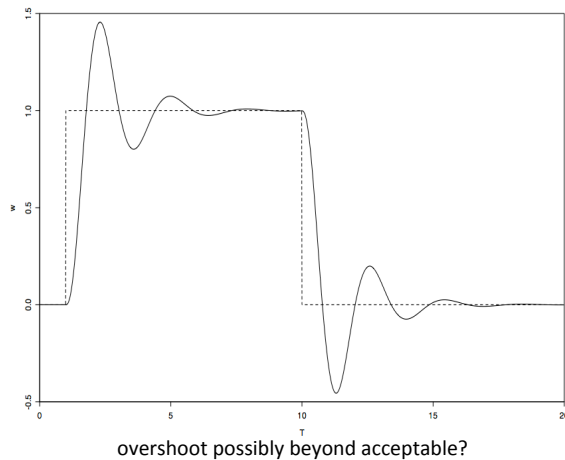
# FLC Control Surfaces
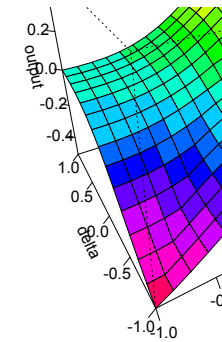


4-rule system

9-rule system

# Mamdani Control

**An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller**

E. H. MAMDANI AND S. ASSILIAN
*Queen Mary College, London University, U.K.*

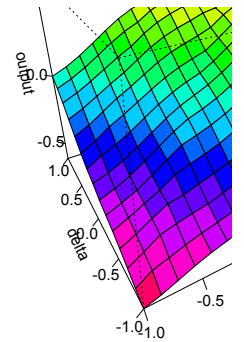This paper describes an experiment on the "linguistic" synthesis of a controller for a model industrial plant (a steam engine). Fuzzy logic is used to convert heuristic control rules stated by a human operator into an automatic control strategy. The experiment was initiated to investigate the possibility of human interaction with a learning controller. However, the control strategy set up linguistically proved to be far better than expected in its own right, and the basic experiment of linguistic control synthesis in a non-learning controller is reported here.

**Introduction**

Many techniques for the synthesis of automatic controllers will be found in the control literature. The standard textbook approaches, however, all involve quantitative, numeric calculations based on mathematical models of the plant and controller. In recent years there have been many studies of self-organizing, or adaptive, controllers in which the control strategy is not synthesized in advance but is generated by optimization algorithms based on the controller's "experience". Such controllers exhibit some characteristics of human learning and in their more general forms are examples of "artificial intelligence".

**The Plant to be Controlled**

The plant for which the controller was implemented comprises a steam engine and boiler combination. The model of the plant used has two inputs: heat input to the boiler and throttle opening at the input of the engine cylinder, and two outputs: the steam pressure in the boiler and the speed of the engine. Simple identification tests on the plant proved that it is highly nonlinear with both magnitude and polarity of the input variables. Therefore, the plant possesses different characteristics at different operating points, so that the direct digital controller implemented for comparison purposes had to be retuned (by trial and error) to give the best performance each time the operating point was altered.

HEATER ALGORITHM
If $PE = NB$
and $CPE = $ not $(NB$ or $NM)$
and $SE = ANY$
and $CSE = ANY$
then $HC = PB$
...

**Results and Conclusions**

The above scheme containing the 24 rules given in the appendix was implemented on the PDP-8 computer and applied to the steam-engine plant. A fixed digital controller was also implemented on the computer and applied to the same plant for the purpose of comparison. With the fixed controller many runs were required to tune the controller for the best performance. This tuning was done by trial and error. Results of many runs with different set points were taken. The quality of control with the fuzzy controller was found to be better each time than the best control obtained by the fixed controller. This is summarized in the figure below.
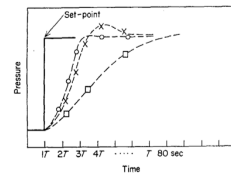
FIG. 2. Fixed controller (DDC algorithm), ×, □; Fuzzy controller, ⊙.

# TSK Control

**Fuzzy Identification of Systems and Its Applications to Modeling and Control**

TOMOHIRO TAKAGI AND MICHIO SUGENO

*Abstract*—A mathematical tool to build a fuzzy model of a system where fuzzy implications and reasoning are used is presented in this paper. The premise of an implication is the description of fuzzy subspace of inputs and its consequence is a linear input–output relation. The method of identification of a system using its input–output data is then shown. Two applications of the method to industrial processes are also discussed: a water cleaning process and a converter in a steel-making process.

*A. Fuzzy modeling of human operator's control actions*

*Water Cleaning Process:* We shall now show an example where an operator's control actions are fuzzily modeled. The control process is a water cleaning process for civil water supply as is illustrated in Fig. 18. In the process, turbid river water first comes into a mixing tank where chemical products called PAC and also chlorine are put and mixed in the water. Then the mixed water flows into a sedimentation tank where the turbid part of water is cohered with the aid of PAC and settled to the bottom.
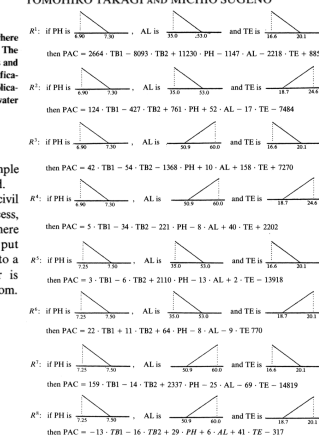
Fig. 15. Input–output relation of the model 3-1.

Fig. 16. Input–output relation of the model 3-2.

# Summary

- Lecture summary
  - TSK inference is a simple and mathematically convenient alternative to Mamdani inference
  - it is more efficient, with similar results
    - hence, it is often found in control applications
  - higher-order models offer more complexity, but are rarely found in practice above first-order
- Next lecture
  - fuzzy modelling and tuning
  - real-world case studies