G53FUZ
Fuzzy Sets and Systems

TSK Inference and Fuzzy Control

Jon Garibaldi
Intelligent Modelling and Analysis Research Group

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

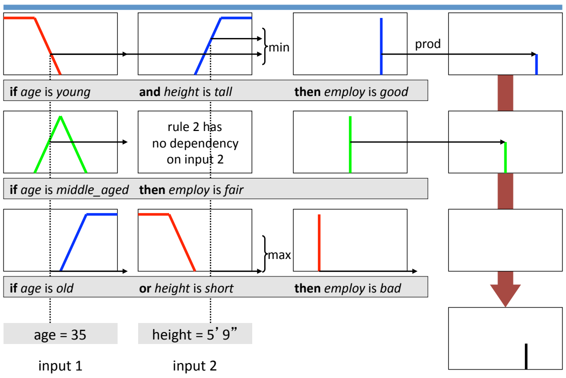
Zeroth-Order

- 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



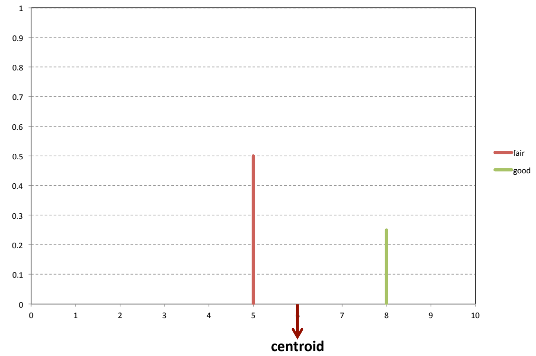
Example: Variables

- 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

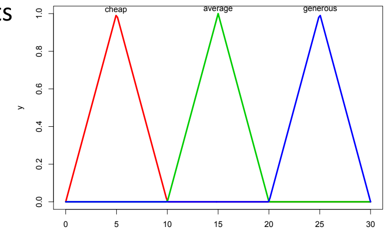
- Three rules
  - IF *Age* is *young* AND *Height* is *tall* THEN *Employ* is *good*
  - IF *Age* is *middle_aged* THEN *Employ* is *fair*
  - IF *Age* is *old* OR *Height* is *short* THEN *Employ* is *bad*
- Inputs
  - Age = 40 (years)
  - Height = 1.8 (metres)

Visually



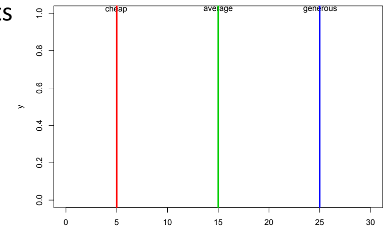
Tipper Example

- 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

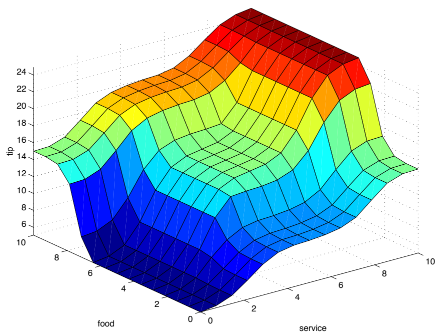


TSK Equivalent

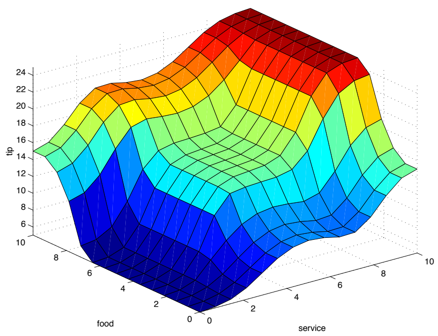
- Recall the standard tipping example
  - If *Service* is *poor* or *Food* is *rancid* then *Tip* is 5
  - If *Service* is *good* then *Tip* is 15
  - If *Service* is *excellent* or *Food* is *delicious* then *Tip* is 25
- Output sets



Tipper Surface



TSK Surface



Efficiency Comparison

	Mamdani	Sugeno
Antecedents	antecedent evaluation is the same	
Rule Evaluation	implication operates across consequent set 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 e.g. 101 x #rules max operations	single summation
Defuzzification	numerator: multiplication and summation, e.g. 101 times denominator: summation, e.g. 101 times	summation per rule single division

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

First-Order

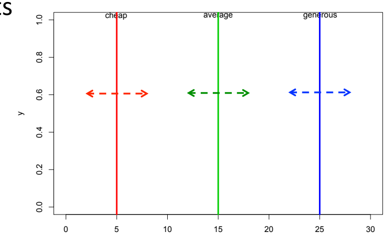
- A first-order Sugeno system has rules of form

  IF x is A_1 AND y is B_1 THEN $z = p_1x + q_1y + r_1$
  IF x is A_2 AND y is B_2 THEN $z = p_2x + q_2y + r_2$
  …
  IF x is A_n AND y is B_n THEN $z = p_nx + q_ny + r_n$

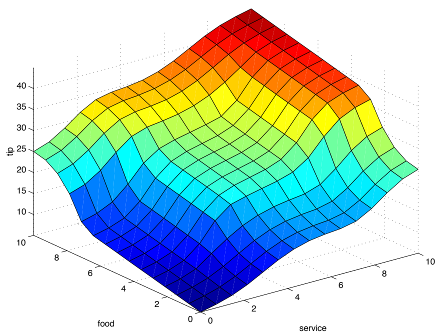
  where p_i, q_i and r_i are constants
- Visualise such a first-order system by imagining each rule as defining a 'moving singleton'
  - these singletons move linearly in the output space and then combined to form the final output

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

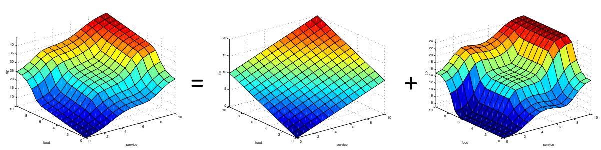


First-Order TSK Surface



Conceptualising the Output

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



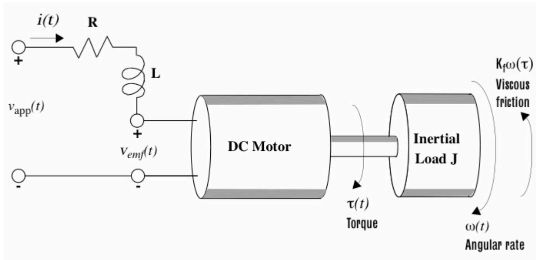
- Different values of *p* and *q* adjust the incline of the plane on each axis

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)

DC Motor

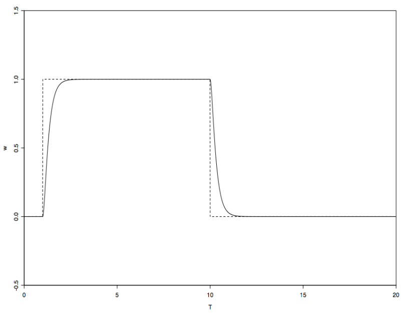
Input: voltage $v_{app}(t)$



Output: angular velocity $\omega(t)$

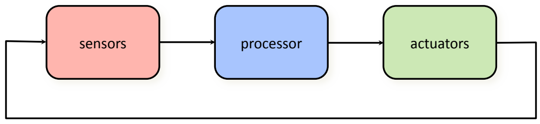
Open-Loop Control

$v_{app}(t)$ = 26.681V



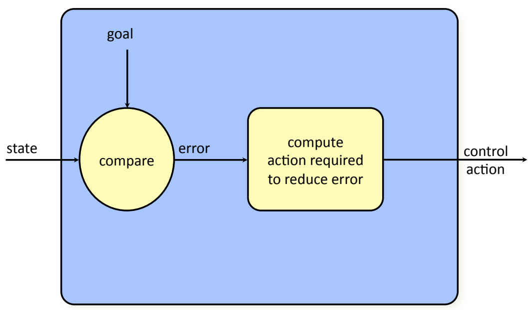
$\omega(t)$ reaches approx. 1 rotation per second

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

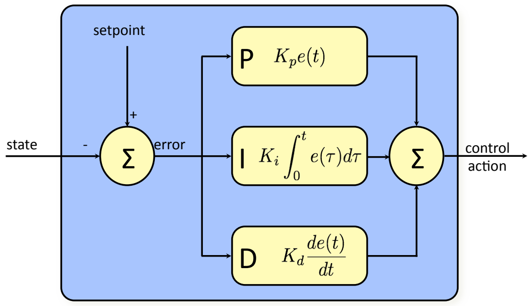


PID Control

- 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

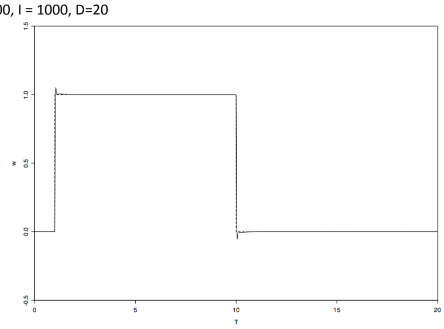


Appropriate Parameters

- 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

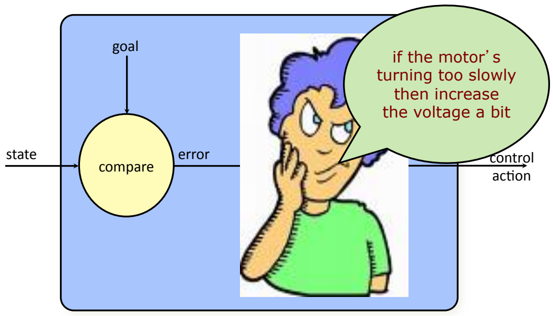
Tuned PID Control

P = 300, I = 1000, D=20



PID tuned: exceedingly close control

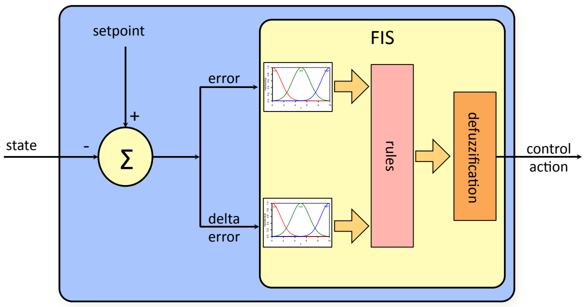
The Fuzzy Approach



Fuzzy Architecture

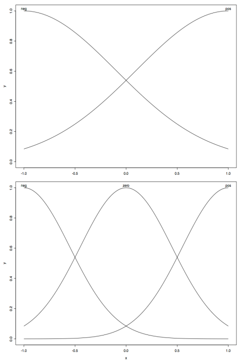
- A fuzzy inference system (FIS) takes one or more control parameters as inputs and produces the control action as output
  - inputs
    - error
    - delta error (rate of change of error)
  - output
    - control action
      - in a subtle change to PID control, the output of a fuzzy controller is often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

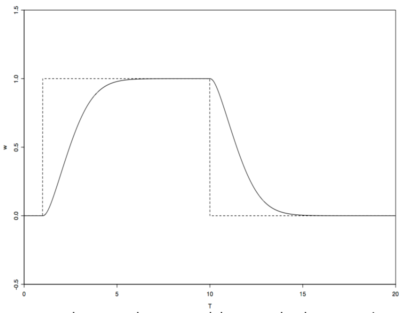


Four Rules

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*

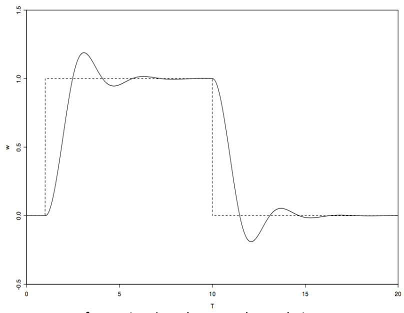
		Delta_error	
		neg	pos
Error	neg	neg	zero
	pos	zero	pos

Motor Configuration 1



worse than open-loop control, but completely automatic

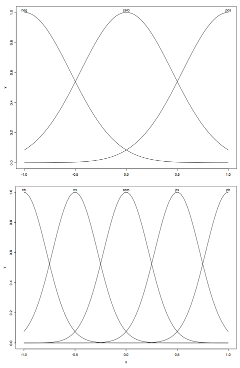
Motor Configuration 2



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



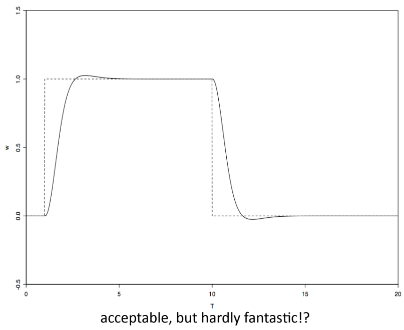
Nine Rules

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

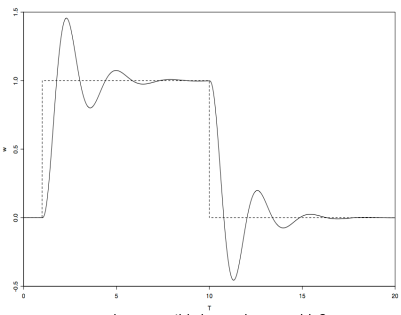
		Delta_error		
		Neg	Zero	Pos
Error	Neg	NB	NS	Zero
	Zero	NS	Zero	PS
	Pos	Zero	PS	PB

Motor Configuration 1



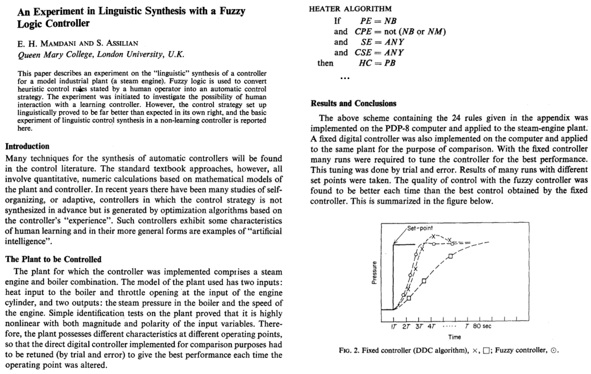
acceptable, but hardly fantastic!?

Motor Configuration 2



overshoot possibly beyond acceptable?

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

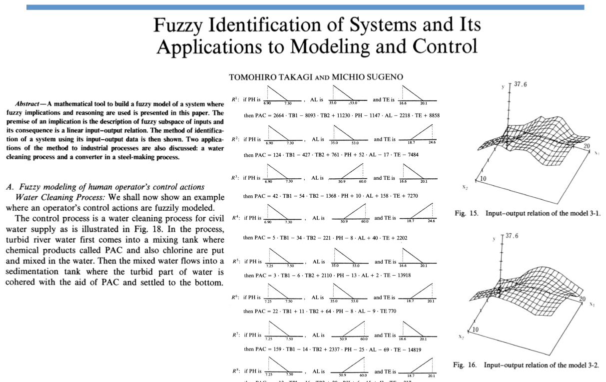
If $PE = NB$
and $CPE =$ not (NB or NM)
and $SE = ANY$
and $CSE = ANY$
then $HC = PB$
...

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.

R^1: if PH is 6.90 7.30, AL is 35.0 53.0 and TE is 16.6 20.1
then PAC = 2664 · TB1 − 8093 · TB2 + 11230 · PH − 1147 · AL − 2218 · TE + 8858

R^2: if PH is 6.90 7.30, AL is 35.0 53.0 and TE is 18.7 24.6
then PAC = 124 · TB1 − 427 · TB2 + 761 · PH + 52 · AL − 17 · TE − 7484

R^3: if PH is 6.90 7.30, AL is 50.9 60.0 and TE is 16.6 20.1
then PAC = 42 · TB1 − 54 · TB2 − 1368 · PH + 10 · AL + 158 · TE + 7270

R^4: if PH is 6.90 7.30, AL is 50.9 60.0 and TE is 18.7 24.6
then PAC = 5 · TB1 − 34 · TB2 − 221 · PH − 8 · AL + 40 · TE + 2202

R^5: if PH is 7.25 7.50, AL is 35.0 53.0 and TE is 16.6 20.1
then PAC = 3 · TB1 − 6 · TB2 + 2110 · PH − 13 · AL + 2 · TE − 13918

R^6: if PH is 7.25 7.50, AL is 35.0 53.0 and TE is 18.7 20.1
then PAC = 22 · TB1 + 11 · TB2 + 64 · PH − 8 · AL − 9 · TE 770

R^7: if PH is 7.25 7.50, AL is 50.9 60.0 and TE is 16.6 20.1
then PAC = 159 · TB1 − 14 · TB2 + 2337 · PH − 25 · AL − 69 · TE − 14819

R^8: if PH is 7.25 7.50, AL is 50.9 60.0 and TE is 18.7 20.1
then PAC = −13 · *TB*1 − 16 · *TB*2 + 29 · *PH* + 6 · *AL* + 41 · *TE* − 317

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
    - so 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