

13.5. Fuzzy Logic Control.

Incremental

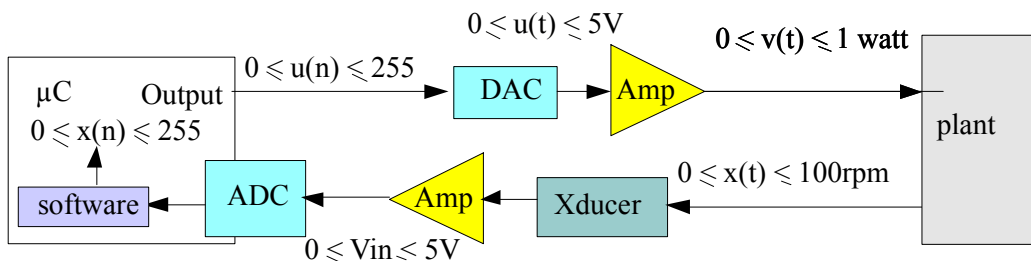
- Simple
- Stable
- Slow

PID

plant can be described using linear differential equations

Fuzzy logic

- execute faster
- good when there is expert knowledge
- maps intuition into rules
- abstractive approach



Our system has:

- two control inputs
 - X^* the desired motor speed 0 to 255
 - X' the current estimated motor speed 0 to 255
- one control output
 - U the digital value that we write to the D/A

The approach to fuzzy design can be summarized as

- The **Physical plant** has *real state variables*
 X speed
- The **Data Acquisition System** creates the *estimated state variables*;
 X' measured speed
- The **Preprocessor** may calculate relevant parameters called *crisp inputs*;
 $E = X^* - X'$ the error in motor speed

```
short lSpeedError; // 16-bit, -256 to 256, 0 means no
short SpeedError; // 0 to 255, 128 means no error
```

```
          D = X'(n) - X'(n-1)    the change in motor speed
short lAcceleration; // 16-bit, -256 to 256, 0 means no
short Acceleration; // 0 to 255, 128 means no
```

- **Fuzzification** will convert crisp inputs into *input fuzzy membership sets*;

- VerySlow True if the motor is spinning way too slow
- Slow True if the motor is spinning too slow
- OK True if the motor is spinning at the proper speed
- Fast True if the motor is spinning too fast
- VeryFast True if the motor is spinning way too fast
- Up True if the motor speed is getting larger
- Constant True if the motor speed is remaining the same
- Down True if the motor speed is getting smaller.

- The **fuzzy rules** calculate *output fuzzy membership sets*;

These 7 rules can be illustrated in a table form.

		D		
	E	<i>Down</i>	<i>Constant</i>	<i>Up</i>
<i>Slow</i>		<i>Increase</i>	<i>Increase</i>	
<i>OK</i>		<i>Increase</i>	<i>Same</i>	<i>Decrease</i>
<i>Fast</i>			<i>Decrease</i>	<i>Decrease</i>

Figure 13.21. Fuzzy logic rules shown in table form.

- **Defuzzification** will convert output sets into *crisp outputs*;
 - bigsub* True if the motor speed should be decreased a lot
 - sub* True if the motor speed should be decreased
 - zero* True if the motor speed should remain the same
 - add* True if the motor speed should be increased
 - bigadd* True if the motor speed should be increased a lot
 - The **Postprocessor** modify crisp outputs into a more convenient format;
 - dU
 - The **Actuator System** affects the Physical plant based on these output.
- $U = U+dU$

13.5.2. Pulse Width Modulated Fuzzy Controller

The objective of this section is to design a *fuzzy logic* microcomputer-based motor controller. The actuator is a pulse-width-modulated digital wave. The power to the motor is controlled by varying the duty cycle of the 200Hz squarewave. A switching power transistor provides current to the motor only when the digital output is high. The diodes protect the electronics from the back EMF (as high as 200 volts) that occurs when the current is switched (large di/dt) to the DC motor coil. The frequency of the squarewave is chosen faster than the motor response, so that the motor responses only to the duty-cycle, and not to the individual high's and low's of the squarewave.

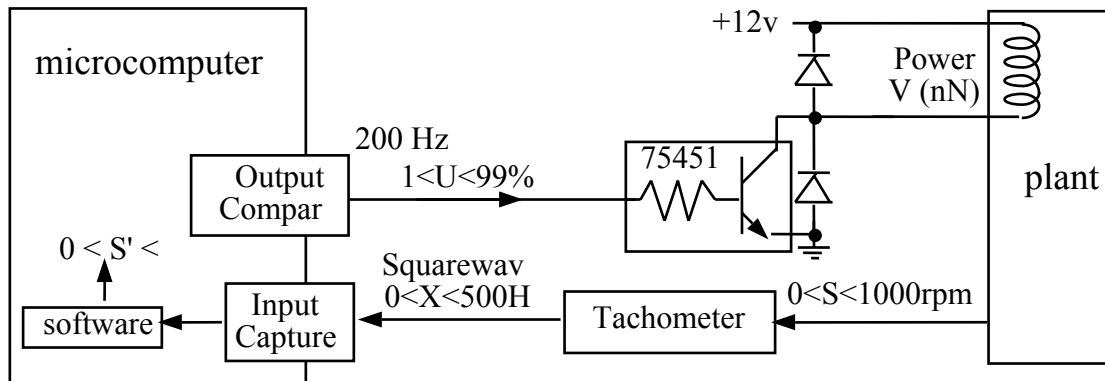


Figure 13.23. Interface of a motor controller with fuzzy logic.

The tachometer output is a frequency (in Hz) that is related to the motor speed (in rpm). The input capture system can be used to measure the period of this signal. Period measurement is faster than frequency measurement, so it is better suited for real time control. The motor speed is a nonlinear function of the applied power. Because of friction and inertia, there is a range of power, below which the motor will not spin. Below 40% duty cycle, the motor has a higher DC gain.

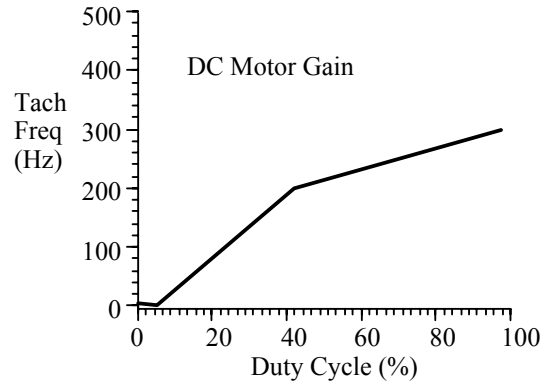


Figure 13.24. Steady state response of the physical plant.

Our system has:

- two control inputs
 - S^* the desired motor speed in rpm
 - S' the current estimated motor speed rpm
- one control output
 - U duty cycle in E clock cycles
- two crisp inputs
 - $E=S^*-S'$ the error in motor speed in rpm
 - $D=dS'/dt = S'(n)-S'(n-1)$ change in speed in Hz/time or rpm/time
- one crisp output
 - ΔU change in duty cycle, $U=U+\Delta U$ in E clock cycles

Next we introduce FUZZY membership sets that define the current state of the crisp inputs and outputs. The calculation of the input membership sets is called *Fuzzification*. For this simple FUZZY controller, we will define 10 membership sets for the crisp inputs:

- ENL* True if the motor is spinning much too slow
- ENS* True if the motor is spinning a little bit too slow
- EZE* True if the motor is spinning at the proper speed
- EPS* True if the motor is spinning a little bit too fast
- EPL* True if the motor is spinning much too fast
- DNL* True if the motor speed is slowing down a lot
- DNS* True if the motor speed is slowing down a little
- DZE* True if the motor speed is remaining the same
- DPS* True if the motor speed is speeding up a little
- DPL* True if the motor speed is speeding up a lot

We will define 5 membership sets for the crisp output:

- ONL* True if the motor speed should be decreased a lot
- ONS* True if the motor speed should be decreased a little
- OZE* True if the motor speed should remain the same
- OPS* True if the motor speed should be increased a little
- OPL* True if the motor speed should be increased a lot

Notice in the following rules when the system is operating near the set point (*ENS,EZE,EPS*) with small changes in speed (*DNS, DZE,DPS*) then this fuzzy system is similar to the previous implementation. On the other hand, when the system has large errors and/or large acceleration, then it makes large changes in the output (*OPL,ONL*) in an attempt to reach the desired state faster. The 6811 assembly or C implementation to this controller would be very similar to the previous example. On the other hand, the 6812 has a rich set of fuzzy logic assembly instructions that will be used to solve this problem.

```

org $800
; crisp inputs
speed:  ds 1
acceleration: ds 1
    
```

```
; input membership variables
fuzvar:  ds    0    ; inputs
EPL:    ds    1    ; speed way too fast
EPS:    ds    1    ; speed too fast
EZE:    ds    1    ; speed OK
ENS:    ds    1    ; speed too slow
ENL:    ds    1    ; speed way too slow
DPL:    ds    1    ; speed decreasing a lot
DPS:    ds    1    ; speed decreasing
DZE:    ds    1    ; speed constant
DNS:    ds    1    ; speed increasing
DNL:    ds    1    ; speed increasing a lot

fuzout:  ds    0    ; outputs
OPL:    ds    1    ; add a lot of power to system
OPS:    ds    1    ; add some power to system
OZE:    ds    1    ; leave power as is
ONS:    ds    1    ; subtract some power from system
ONL:    ds    1    ; subtract a lot of power
BREAK:  ds    1    ; apply break?
; input membership variables relative offsets
epl:    equ    0    ; speed way too fast
eps:    equ    1    ; speed too fast
eze:    equ    2    ; speed ok
ens:    equ    3    ; speed too slow
enl:    equ    4    ; speed way too slow
dpl:    equ    5    ; speed decreasing a lot
dps:    equ    6    ; speed decreasing
dze:    equ    7    ; speed constant
dns:    equ    8    ; speed increasing
dnl:    equ    9    ; speed increasing a lot
;output membership variables
opl:    equ    10   ; add alot of power to system
ops:    equ    11   ; add some power to system
oze:    equ    12   ; leave power as is
ons:    equ    13   ; subtract some power from system
onl:    equ    14   ; subtract alot of power from system
break:  equ    15   ; apply break?
;crisp outputs
dpower: ds    1
```

Program 13.18. Global variables for fuzzy controller in 6812 assembly.

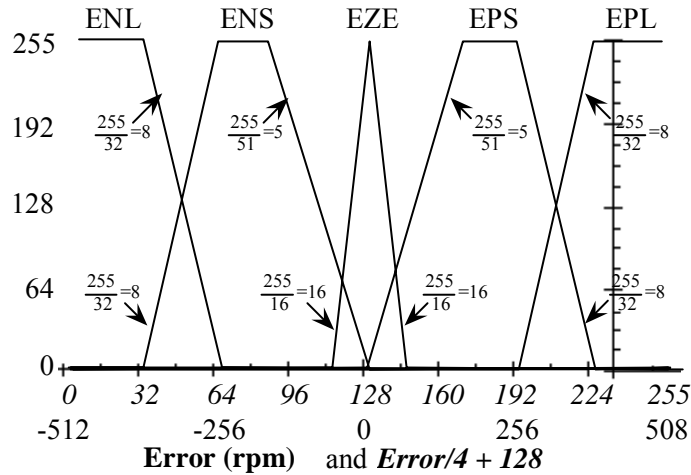


Figure 13.25. Fuzzification of the error input.

```

org $F000
; format is Point1,Point2,Slope1,Slope2
s_tab:  dc.b 192,255,8,0      ;EPL
        dc.b 128,224,5,8     ;EPS
        dc.b 112,144,16,16   ;EZE
        dc.b 32,128,8,5      ;ENS
        dc.b 0,64,0,8        ;ENL
    
```

Program 13.19. Definition of the error input function in 6812 assembly.

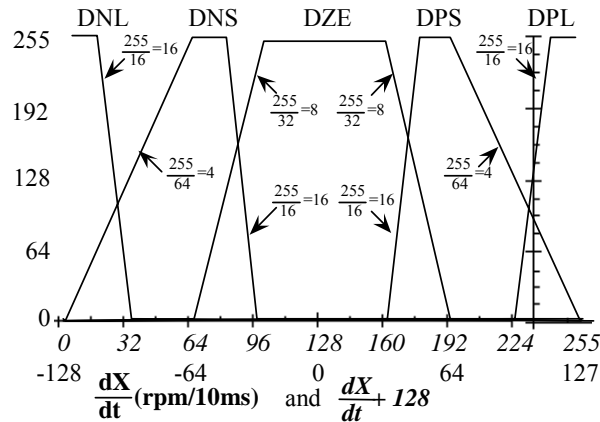


Figure 13.26. Fuzzification of the acceleration input.

```

a_tab:  dc.b 224,255,16,0     ;DPL
        dc.b 160,255,16,4    ;DPS
        dc.b 64,192,8,8      ;DZE
        dc.b 0,96,4,16       ;DNS
        dc.b 0,32,0,16       ;DNL
    
```

Program 13.20. Definition of the acceleration input function in 6812 assembly.

$D=X(n)-X(n-1)$

$E=X*-X$		<i>DNL</i>	<i>DNS</i>	<i>DZE</i>	<i>DPS</i>	<i>DPL</i>
<i>ENL</i>	<i>OPL</i>	<i>OPL</i>	<i>OPL</i>			
<i>ENS</i>	<i>OPL</i>	<i>OPS</i>	<i>OPS</i>			
<i>EZE</i>	<i>OPL</i>	<i>OPS</i>	<i>OZE</i>	<i>ONS</i>	<i>ONL</i>	
<i>EPS</i>			<i>ONS</i>	<i>ONS</i>	<i>ONL</i>	
<i>EPL</i>			<i>ONL</i>	<i>ONL</i>	<i>ONL</i>	

Figure 13.27. Fuzzy logic rules shown in table form.

rules:

```
dc.b  enl,dnl,$FE,opl,$FE ; if ENL and DNL then OPL
dc.b  ens,dnl,$FE,opl,$FE ; if ENS and DNL then OPL
dc.b  eze,dnl,$FE,opl,$FE ; if EZE and DNL then OPL
dc.b  enl,dns,$FE,opl,$FE ; if ENL and DNS then OPL
dc.b  ens,dns,$FE,ops,$FE ; if ENS and DNS then OPS
dc.b  eze,dns,$FE,ops,$FE ; if EZE and DNS then OPS
dc.b  enl,dze,$FE,opl,$FE ; if ENL and DZE then OPL
dc.b  ens,dze,$FE,ops,$FE ; if ENS and DZE then OPS
dc.b  eze,dze,$FE,oze,$FE ; if EZE and DZE then OZE
dc.b  eps,dze,$FE,ons,$FE ; if EPS and DZE then ONS
dc.b  epl,dze,$FE,onl,break,$FE ; if EPL and DZE then ONL
dc.b  eze,dps,$FE,ons,$FE ; if EZE and DPS then ONS
dc.b  eps,dps,$FE,ons,$FE ; if EPS and DPS then ONS
dc.b  eps,dps,$FE,onl,break,$FE ; if EPL and DPS then ONL
dc.b  eze,dpl,$FE,onl,break,$FE ; if EZE and DPL then ONL
dc.b  eps,dpl,$FE,onl,break,$FE ; if EPS and DPL then ONL
dc.b  epl,dpl,$FE,onl,break,$FE ; if EPL and DPL then ONL
dc.b  $FF
```

Program 13.21. Definition of the fuzzy rules in 6812 assembly.

The defuzzification is shown in the following table.

Output Fuzzy Set	Singleton Value
ONL	-128
ONS	-10
OZE	0
OPS	10
OPL	127

Table 13.2. Defuzzification converts the output memberships into a crisp output.

```
addsingleton: dc.b 255,138,128,118,0
; 128 subtracted, +127,10,0,-10,-128

ritual: sei ;make atomic
bset #$20,TIOS ;OC5
movb #$80,TSCR ;enable, no fast clr
movb #$33,TMSK2 ;lus clk
bset #$20,TMSK1 ;Arm OC5
ldaa #$20 ;clear C5F
staa TFLG1
```

```

        ldd  TCNT          ;current time
        addd #10000       ;first in 10 ms
        std  TC5
        cli                ;enable
        rts
main:   lds  #$0C00
        ldd  #100         ; initial duty cycle 1% is off
        std  dutycycle
        jsr  ritual       ; initialize OC interrupt
        bra  *

```

Program 13.22. Ritual and main program for fuzzy controller in 6812 assembly.

```

timehan: ldaa #$20 ;clear C5F
        staa TFLG1 ;Acknowledge
        ldd  TC5
        addd #10000 ;next in 10 ms
        std  TC5
        jsr  measurespeed ; crisp input speed
;reg A is speed 0 to 255
        ldx  #s_tab
        ldy  #fuzvar
        mem          ; calculate EPL
        mem          ; calculate EPS
        mem          ; calculate EZE
        mem          ; calculate ENS
        mem          ; calculate ENL
        jsr  measureacceleration ; crisp input
;reg A is acceleration 0 to 255
        ldx  #a_tab
        mem          ; calculate DPL
        mem          ; calculate DPS
        mem          ; calculate DZE
        mem          ; calculate DNS
        mem          ; calculate DNL
        ldab #6
cloop:  clr  l,y+ ; clear OPL,OPS,OZE,ONS,ONL,BREAK
        dbne b,cloop
        ldx  #rules
        ldy  #fuzvar
        ldaa #$FF
        rev
        ldy  #fuzout
        ldx  #addsingleton
        ldab #5
        wav
        ediv
        tfr  y,d
        subb #128
        stab dpower
change  ldab dpower
        sex  b,d
        addd dutycycle
; 200 Hz squarewave is 10000 cycles
; correct range is 100 to 9600 cycles
        cpd  #100
        bhs  nolow
low:    ldd  #100 ; underflow
        bra  set
notlow: cpd  #9600
        bls  set

```

```
high:    ldd  #9600 ; overflow
set:     std  dutycycle
        rti
```

Program 13.23. Interrupt handler for fuzzy controller in 6812 assembly.