Fuzzy Logic Control

- Simpler than PID controllers and can be more robust.
- Physical plant has real state variables.
- DAS monitors them and creates estimated state variables.
- Preprocessor calculates crisp inputs.
- Fuzzification converts into input fuzzy membership sets.
- The fuzzy rules calculate output fuzzy membership sets.
- Defuzzification converts them into crisp outputs.
- Postprocessor modifies them into more convenient format.
- Actuator system affects physical plant based on outputs.

Interface of a Motor Controlled with Fuzzy Logic

(See Figure 13.17)

DAC, ADC Fuzzy Controller

- Two control inputs:
  - $S^*$: desired motor speed, rpm
  - $S'$: current estimated motor speed, rpm
- One control output:
  - $N$: digital value that we write to the DAC
- To use 8-bit math, change units to $1000/256 = 3.9$ rpm
  - $T^* = (256 \cdot S^*)/1000$: desired motor speed, 3.9 rpm
  - $T' = (256 \cdot S')/1000$: current motor speed, 3.9 rpm
- Two crisp inputs:
  - $E = T^* - T'$: error in motor speed, 3.9 rpm
  - $D = T'(n) - T'(n-1)$: change in speed, 3.9 rpm
Subtraction with Overflow/Underflow Checking

```c
char Subtract(unsigned char N, unsigned char M){
    /* returns N-M */
    unsigned int N16, M16;
    int Result16;
    N16=N;        /* Promote N, M */
    M16=M;
    Result16=N16-M16;  /* -255Result16+255 */
    if(Result16<-128) Result16 = -128;
    if(Result16>127) Result16 = 127;
    return(Result16);
}
```

Crisp Inputs

```c
unsigned char Ts;  /* Desired Speed */
unsigned char T;   /* Current Speed */
unsigned char Told; /* Previous Speed */
char D;            /* Change in Speed */
char E;            /* Error in Speed */
```

void CrispInput(void){
    E=Subtract(Ts,T);
    D=Subtract(T,Told);
    Told=T;  /* Set up Told for next time */
}

Fuzzy Membership Sets

- Input fuzzy membership sets:
  - Slow - True if motor is spinning too slow
  - OK - True if motor is spinning at proper speed
  - Fast - True if motor is spinning too fast
  - Up - True if motor speed is getting larger
  - Constant - True if motor speed is remaining the same
  - Down - True if motor speed is getting smaller

- Output fuzzy membership sets:
  - Decrease - True if motor speed should be decreased
  - Same - True if motor speed should remain the same
  - Increase - True if motor speed should be increased

Fuzzification

(See Figures 13.18 and 13.19)
Calculation of the Fuzzy Membership Variables

```c
#define TE 20
unsigned char Fast, OK, Slow, Down, Constant, Up;
#define TD 20
unsigned char Increase, Same, Decrease;
#define TN 20

void InputMembership(void)
{
    if(E <= -TE) /* E <= -TE */
        Fast=255; OK=0; Slow=0;
    else if(E < 0) /* -TE < E < 0 */
        Fast=(255*(-E))/TE; OK=255-Fast; Slow=0;
    else if(E < TE) /* 0 < E < TE */
        Fast=0; Slow=(255*E)/TE; OK=255-Slow;
    else /* +TE <= E */
        Fast=0; OK=0; Slow=255;
}
```

Calculation of the Fuzzy Membership Var (cont)

```c
if(D <= -TD) /* D <= -TD */
    Down=255; Constant=0; Up=0;
else if(D < 0) /* -TD < D < 0 */
    Down=(255*(-D))/TD; Constant=255-Down; Up=0;
else if(D < TD) /* 0 < D < TD */
    Down=0; Up=(255*D)/TD; Constant=255-Up;
else /* +TD <= D */
    Down=0; Constant=0; Up=255;
}
```

Fuzzy Logic Rules

(See Figure 13.20)

Calculation of the Output Fuzzy Membership

```c
unsigned char min(unsigned char u1, unsigned char u2)
{
    if(u1>u2) return(u2);
    else return(u1);
}
unsigned char max(unsigned char u1, unsigned char u2)
{
    if(u1<u2) return(u2);
    else return(u1);
}

void OutputMembership(void)
{
    Same=min(OK, Constant);
    Decrease=min(OK, Up);
    Increase=min(OK, Down);
    Decrease=max(Decrease, min(Fast, Constant));
    Decrease=max(Decrease, min(Fast, Up));
    Increase=max(OK, Down);
    Increase=max(Increase, min(Slow, Constant));
    Increase=max(Increase, min(Slow, Down));
}
```
Defuzzification

(See Figure 13.21)

Calculation of the Crisp Output

char dN;

void CrispOutput(void){
    dN=(TN*(Increase-Decrease)) / (Decrease+Same+Increase);
}

Main Program for Fuzzy Logic Control

unsigned int Time;
#define rate 2000

void Initialize(void){
    OPTION=0x80;  /* Turn on A/D */
    PORTB=0;
    N=0;         /* Initial Actuator */
    Told=0;
    Ts=128; }    /* 500 rpm */

#define CCF 0x80

unsigned char Sample(unsigned char channel){
    ADCTL=channel;   /* Start A/D */
    while((ADCTL&CCF)==0); /* Wait for CCF */
    return(ADR1);
}

Main Program for Fuzzy Logic Control (cont)

void Main(void){ int dT;
    Initialize();  /* Turn on A/D init globals */
    Time=TCNT+rate; /* First TCNT value */
    while(1){
        while((dT=Time-TCNT)>0){}; Time=Time+rate; /* Next TCNT value */
        T=Sample(0);       /* Sample A/D and set T */
        CrispInput();  /* Calculate E,D and new Told*/
        InputMembership();/*Fast,OK,Slow,Down,Cons,Up*/
        OutputMembership();/* Inc,Same,Dec */
        CrispOutput();  /* dN */
        N=max(0,min(N+dN,255));
        PORTB=N;        /* Set Actuator */
    }