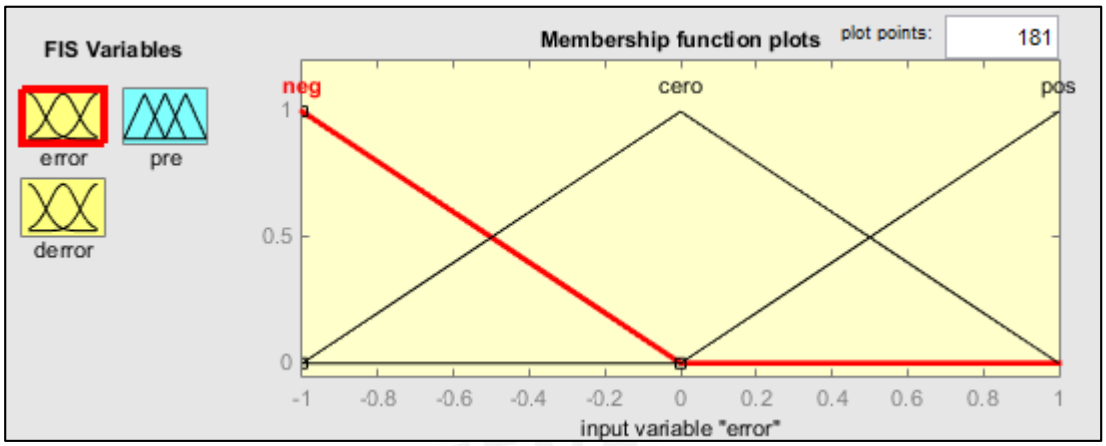
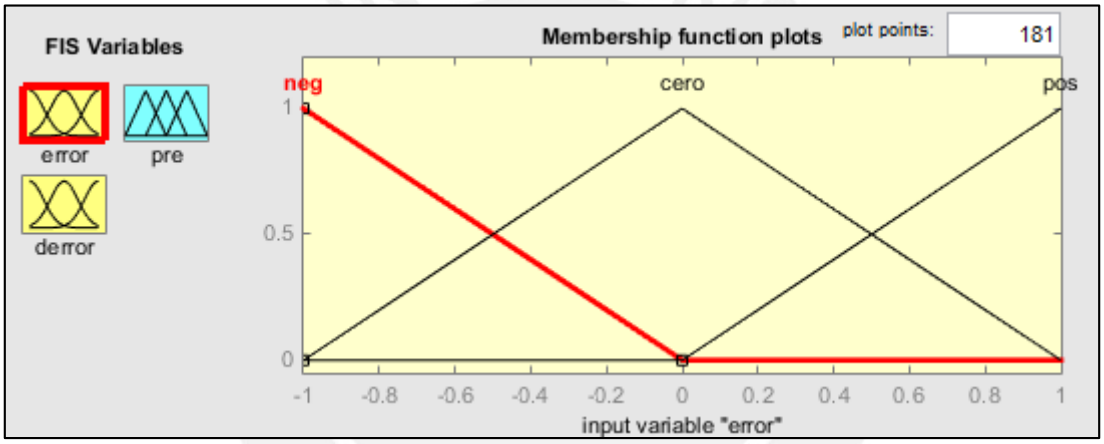
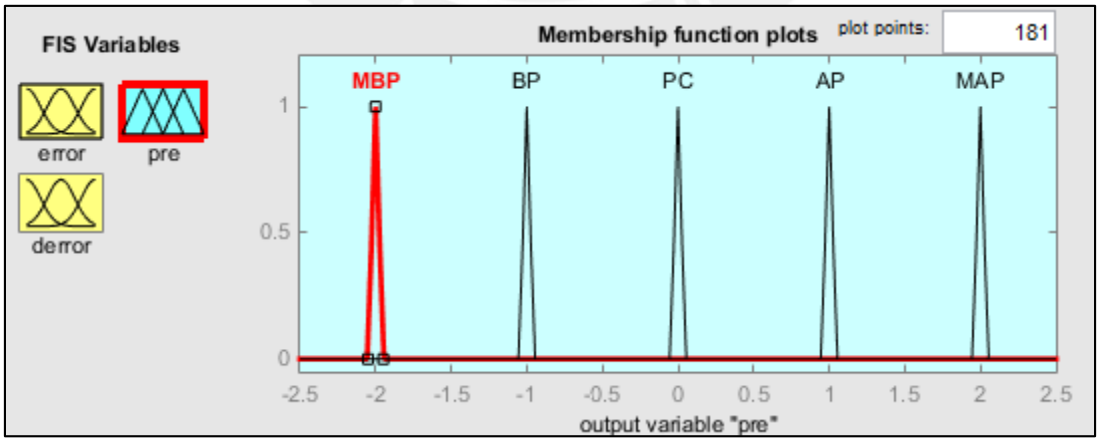
ANEXO 1: Funciones de pertenencia del controlador difuso de presión



Funciones de membresía del error.

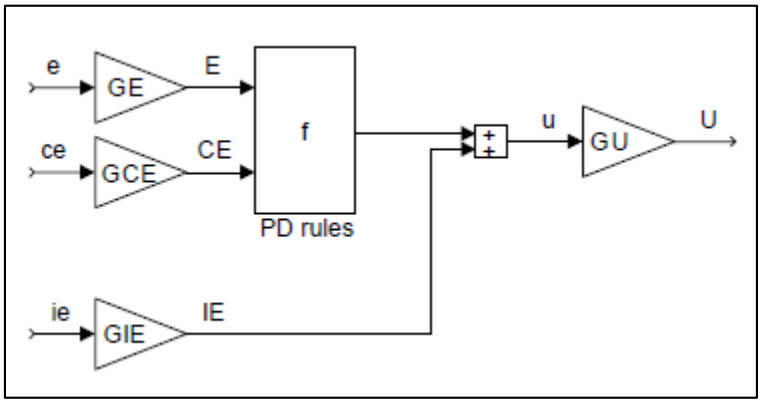


Funciones de membresía de la derivada del error.



Funciones de membresía de la salida del control difuso.

ANEXO 2: Traslado de ganancias PID lineal y PID difuso no lineal



Estructura general PD + I difuso.

Relación entre ganancias PID y PID difuso.

Controlador	Kp	1/Ti	Kd
FP	GE*GU	-	-
Finc	GCE*GCU	GE/GCE	-
FPD	GE*GU	-	GCE/GE
FPD+I	GE*GU	GIE/GE	GCE/GE

ANEXO 3: Texto estructurado en PLC ControlLogix5000 para planta desalinizadora con el algoritmo de control propuesto:

Rutina_planta:

```
all:= -0.1477; al2:= 0.0172;
a21:= -5.7289; a22:= 0.4252; (*contantes para G11 *)
b11:= 0.0018; b21:= 0.0285;
d11 := 0.7340; d12 := 0.0862;
d21 := -0.1436; d22 := 0.9924; (*contantes para G22 *)
e11 := -3.0494; e21 := -9.2460;
g11 := 0.7011; g12 := 0.0846;
g21 := -0.3970; g22 := 0.9790; (*contantes para G21 *)
h11 := -0.0816; h21 := -0.2201;

flujo := a11*flujo_ant + a12*f1_ant + b11*c;
f1 := a21*flujo_ant + a22*f1_ant + b21*c;
flow := flujo + 1.25;

cond := d11*cond_ant + d12*c1_ant + e11*f;
c1 := d21*cond_ant + d22*c1_ant + e21*f;
conductivity := cond + pert + 442;

pert := g11*pert_ant + g12*pert1_ant + h11*c;
pert1 := g21*pert_ant + g22*pert1_ant + h21*c;

flujo_ant := flujo;
f1_ant := f1;
cond_ant := cond;
c1_ant := c1;
pert_ant := pert;
pert1_ant := pert1;
```

Rutina_control:

```
//EFP; (*matriz para grado de pertenencia REAL 2x3*)
//AFP; (*matriz para región de pertenencia booleano 2x3*)
KP1:=-0.039; KI1:=-0.036;
Ts:=0.1;
S[0]:= -2; S[1]:= -1; S[2]:= 0; S[3]:= -1; S[4]:= 0; S[5]:= 1; S[6]:= 0; S[7]:= 1; S[8]:= 2;

if Auto then
eflujo := (ref_flujo-1.25) - flujo;
deflujo := 0.01*(eflujo-eflujo_ant)/Ts;
econd := (ref_cond-442) - (cond + pert);

//error membresias
if (eflujo >= -1) AND (eflujo <= 0) then
    AFP0 := 1;
    EFP[0,0]:= 1-(eflujo-(-1))/(0-(-1));
end_if;

if (eflujo >=-1) AND (eflujo <=1) then
    AFP1 := 1;
    if (eflujo < 0) then
    EFP[0,1]:= (eflujo-(-1))/(0-(-1));
    else
    EFP[0,1]:= 1-(eflujo-(0))/(1-(0));
    end_if;
end_if;

// derivada del error membresias
if (deflujo >= -1) AND (deflujo <= 0) then
    AFP3 := 1;
    EFP[1,0]:= 1-(deflujo-(-1))/(0-(-1));
end_if;

if (deflujo >=-1) AND (deflujo <=1) then
    AFP4 := 1;
    if (deflujo < 0) then
    EFP[1,1]:= (deflujo-(-1))/(0-(-1));
    else
    EFP[1,1]:= 1-(deflujo-(0))/(1-(0));
end_if;
end_if;

if (eflujo >= 0) AND (eflujo <= 1) then
    AFP5:= 1;
    EFP[1,2]:= (deflujo-(0))/(1-(0));
end_if;
```

```
//base de reglas (9 reglas)
//inferencia mandani con operador AND de Zadeh
if (AFP0 AND AFP3) then
    if EFP[0,0] > EFP[1,0] then
        temp := EFP[1,0];
    else
        temp := EFP[0,0];
    end_if;
        if WS[0] < temp then
            WS[0]:=temp;
        end_if;
end_if;

if (AFP0 AND AFP4) then
    if EFP[0,0] > EFP[1,1] then
        temp := EFP[1,1];
    else
        temp := EFP[0,0];
    end_if;
        if WS[1] < temp then
            WS[1]:=temp;
        end_if;
end_if;

if (AFP0 AND AFP5) then
    if EFP[0,0] > EFP[1,2] then
        temp := EFP[1,2];
    else
        temp := EFP[0,0];
    end_if;
        if WS[2] < temp then
            WS[2]:=temp;
        end_if;
end_if;

if (AFP1 AND AFP3) then
    if EFP[0,1] > EFP[1,0] then
        temp := EFP[1,0];
    else
        temp := EFP[0,1];
    end_if;
        if WS[3] < temp then
            WS[3]:=temp;
        end_if;
end_if;

if (AFP1 AND AFP4) then
    if EFP[0,1] > EFP[1,1] then
        temp := EFP[1,1];
    else
        temp := EFP[0,1];
    end_if;
        if WS[4] < temp then
            WS[4]:=temp;
        end_if;
end_if;

if (AFP1 AND AFP5) then
    if EFP[0,1] > EFP[1,2] then
        temp := EFP[1,2];
    else
        temp := EFP[0,1];
    end_if;
        if WS[5] < temp then
            WS[5]:=temp;
        end_if;
end_if;
```

```
if (AFP2 AND AFP3) then
    if EFP[0,2] > EFP[1,0] then
        temp := EFP[1,0];
    else
        temp := EFP[0,2];
    end_if;
        if WS[6] < temp then
            WS[6]:=temp;
        end_if;
end_if;

if (AFP2 AND AFP4) then
    if EFP[0,2] > EFP[1,1] then
        temp := EFP[1,1];
    else
        temp := EFP[0,2];
    end_if;
        if WS[7] < temp then
            WS[7]:=temp;
        end_if;
end_if;

if (AFP2 AND AFP5) then
    if EFP[0,2] > EFP[1,2] then
        temp := EFP[1,2];
    else
        temp := EFP[0,2];
    end_if;
        if WS[8] < temp then
            WS[8]:=temp;
        end_if;
end_if;

//defuzzyfucación singletone
FOR n:=0 TO 8 DO
    num := num + WS[n]*S[n];
    den := den + WS[n];
    DxG := num/den;
END_FOR;

int_err := int_err + 100*eflujo*Ts;
c := (DxG+int_err)*30;
f := f_ant + KP1*(econd-econd_ant) + KI1*econd*Ts;
presion := c + 1000;
pH := f + 6.45;
econd_ant := econd;
f_ant := f;
end_if;
```