

```
classdef SIMC_Aux < handle
    methods (Static)
        function PID=S_IMC(procddata,tauc,CoefFiltro)
            arguments
                procdata %struct que contiene tau1, tau2, theta (retardo),
                kp (ganancia)
                tauc %constante de tiempo deseada de bucle cerrado
                CoefFiltro = 15; % filtro ruido 15 veces más rápido que
                taud por defecto
            end
            if(~isfield(procddata,'tau2'))
                procddata.tau2 = 0;
            end
            if(isfield(procddata,'tau1') && (procddata.tau1<procddata.tau2))
                warning("S_IMC: tau1 debe ser mayor de tau2, las
intercambio.")
                tmp = procddata.tau2;
                procddata.tau1 = procddata.tau2; procddata.tau2 = tmp;
            end
            if(tauc<procddata.theta)
                warning("Parámetro tau_c demasiado pequeño, no
recomendable en S-IMC pero sigo adelante.")
            end
            s=tf('s');
            if(isfield(procddata,'tau1')) % no es integrador puro, tau1
proporcionado por usuario
                tauI = min(procddata.tau1, 4*(tauc+procddata.theta));
                if(tauI<procddata.tau1-1e-3)
                    disp('S-IMC: evitamos cancelación IMC polo dominante
cerca de origen')
                end
                Kproporcional= 1/procddata.kp*procddata.tau1/(tauc
+procddata.theta);
            else%ya es integrador el proceso.
                tauI = 4*(tauc+procddata.theta);
                Kproporcional = 1/procddata.kp*1/(tauc+procddata.theta);
            end
            Integral = (s+1/tauI)/s; % 1+1/(tauI*s)
            if(procddata.tau2>1e-6)
                taud = procddata.tau2;
                Derivada = (taud*s+1)/(taud/CoefFiltro*s+1);
            else
                Derivada = 1;taud = NaN;
            end
            PID = Kproporcional*Integral*Derivada; %forma "serie"
            fprintf('Diseñado PID/S_IMC con K_c=%d, tau_I=%d, tau_d=%d
\n',Kproporcional,tauI,taud)
        end

        function SimulaRefYEntrada(Proceso, Regulador, IdealAnteRef)
            arguments
```

```

        Proceso
        Regulador
        IdealAnteRef = tf(1) %si no se pone...
    end
    BCref = minreal(feedback(Proceso*Regulador, 1), 1e-6, false);
    BC_u_ref = minreal(feedback(Regulador, Proceso), 1e-6,
false);
    BCdu = minreal(feedback(Proceso, Regulador), 1e-6, false);
    BC_u_du = -BCref;
    subplot(2,2,1)
    step(BCref, IdealAnteRef), grid on
    title("Resp. ante referencia")
    subplot(2,2,2)
    step(BCdu), grid on
    title("Resp. ante pert. entrada")
    subplot(2,2,3)
    step(BC_u_ref), grid on
    title("Acc. control ante referencia")
    subplot(2,2,4)
    step(BC_u_du), grid on
    title("Acc. control ante pert. entrada")
end

function Regulador=SimulaIMC(Proceso, Q)
    Regulador = minreal(feedback(Q, Proceso, +1), 1e-6, false);
    SIMC_Aux.SimulaRefYEntrada(Proceso, Regulador);
end
end %methods
end %classdef

ans =
    SIMC_Aux with no properties.

```

Published with MATLAB® R2019b