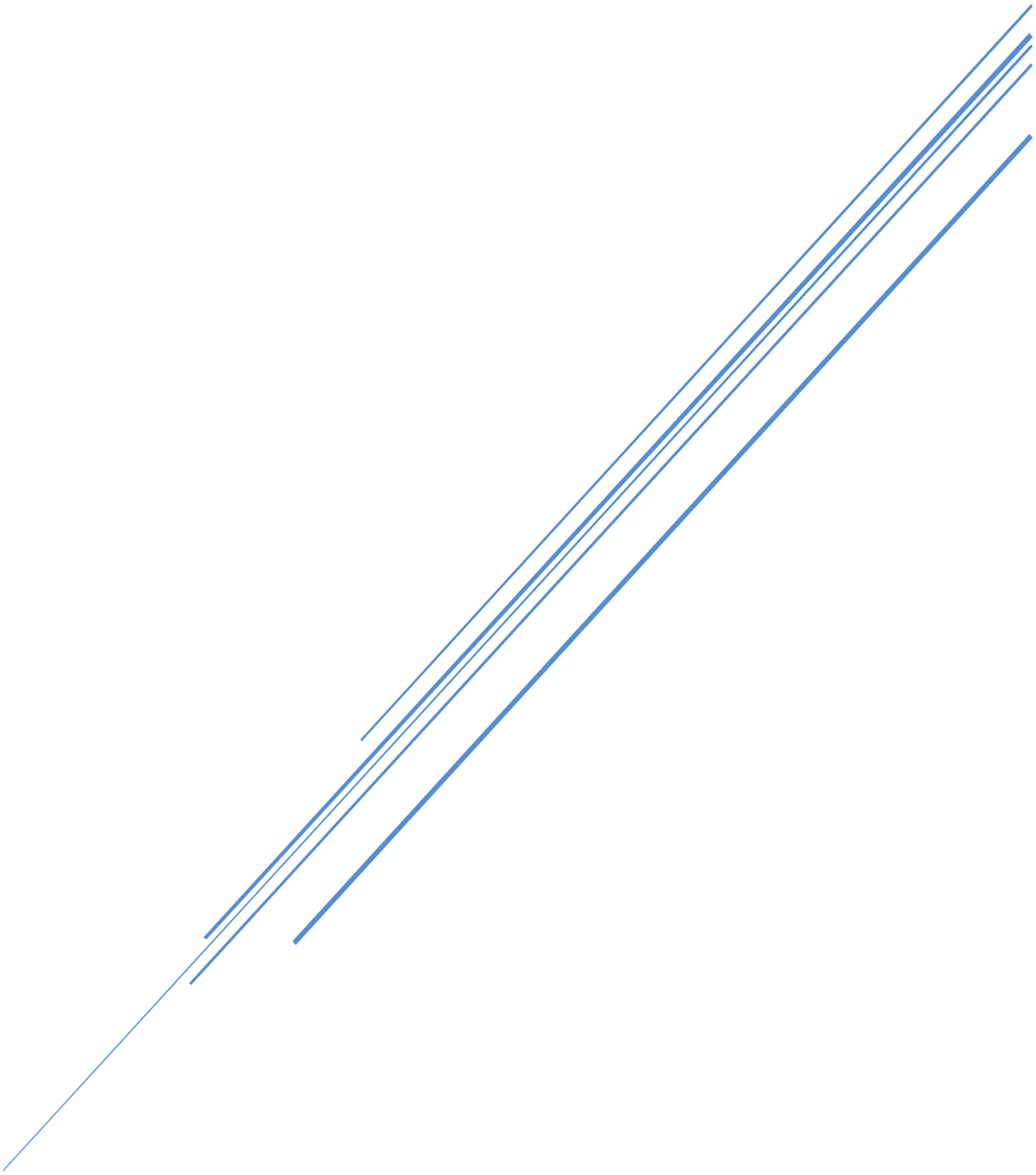
ENTREGA 1ª PARTE DEL PROYECTO

Javier Gómez Jiménez



3º GIERM

Control por computador

OBTENCIÓN DE MODELOS

El primer paso del proyecto ha sido obtener un modelo continuo ante ante escalón. Se ha utilizado como punto de equilibrio 44.78 cm, correspondiente a una entrada de 4 l/min. Se han aplicado escalones en torno al punto de equilibrio de valor 2l/min. El modelo obtenido es el siguiente:

$$G(s) = \frac{K}{(Tau1s + 1)(Tau2s + 1)} = \frac{28.01}{98.43\ s^2 + \ 29.5\ s\ + \ 1}$$

Con K = 28.01 cm*min/l, Tau1 = 25.665 s, Tau2 = 3.8350 s.

Como se ve, el modelo es de orden dos sobreamortiguado (*Se podría haber utilizado uno de primer orden, pero ha sido valorada la bondad de ambos modelos y hemos escogido éste*).

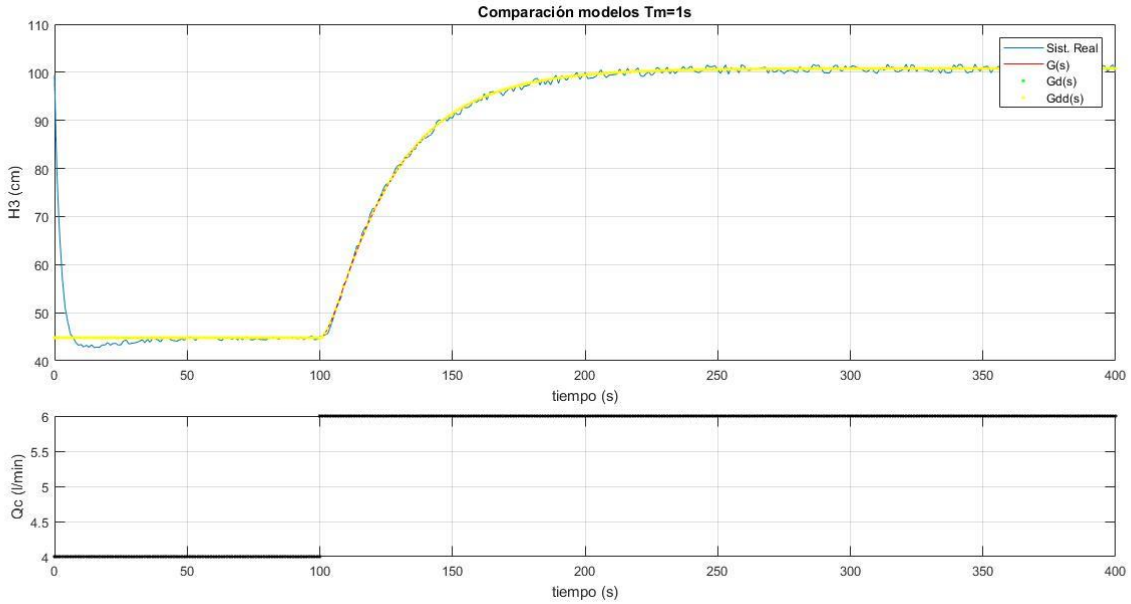
A continuación hemos obtenido dos modelos discretos del sistema: primero utilizando la función de Matlab c2d, y después directamente en discreto, suponiendo desconocido el Tm, utilizando las tablas de las transformadas. El resultado obtenido es el mismo mediante los ods métodos, para un Tm=1s:

$$G(z) = \frac{0.129\ z\ +\ 0.1167}{z^2 -\ 1.732\ z\ +\ 0.741}$$

Por último hemos comparado los cuatro sistemas en una misma gráfica.

GRÁFICA COMPARATIVA

Se ha utilizado un tiempo de muestreo Tm = 1s. No se aprecian bien la distinción entre modelos dado que los tres tienen una bondad alta y coinciden.



OBTENCIÓN DE CONTROLADOR DIRECTAMENTE EN DISCRETO

En primer lugar se ha obtenido un controlador por cancelación de dinámica, a través de la G(z) hallada anteriormente con la orden c2d. La función de transferencia es la siguiente:

$$C(z) = \frac{0.0339\,z^2 - 0.05872\,z + 0.02512}{0.129\,z^3 - 0.1042\,z^2 - 0.108\,z + 0.08316}$$

Antes de implementar este controlador, debido a su fragilidad, se ha descompuesto en fracciones simples:

$$C(z) = \frac{0.00121}{z - 1} + \frac{-0.4315}{z + 0.905} + 0.2628$$

El código realizado ha sido el siguiente, mediante ecuaciones en diferencia:

```
/* CÓDIGO DEL CONTROLADOR */

tiempo = ssGetT(S);  /* Tiempo de la simulación */

uk = 0;

rk = Yeq;

if (tiempo>test+30)

{

rk=Yeq+20;

ek=rk-Yk;

uk = ek1*0.00121+uk1-0.4315*ek1-0.9*uk1+ek; /* DESCOMPOSICIÓN EN GRACCIONES SIMPLES Y DESPUÉS IMPLEMENTACIÓN EN PARALELO */

};

POutAux = rk;
```

OBTENCIÓN DE CONTROLADOR DIRECTAMENTE EN DISCRETO

```
/* Actualización de variables */
uk3=uk2; uk2=uk1; uk1=uk;
ek3=ek2; ek2=ek1; ek1=ek;
/* Cálculo de la señal de la salida real */
Ureal=Ueq+uk;
/* Saturación de la salida real (como medida de seguridad) */
if (Ureal > Umax ) Ureal=Umax;
else if (Ureal < Umin ) Ureal=Umin;
```

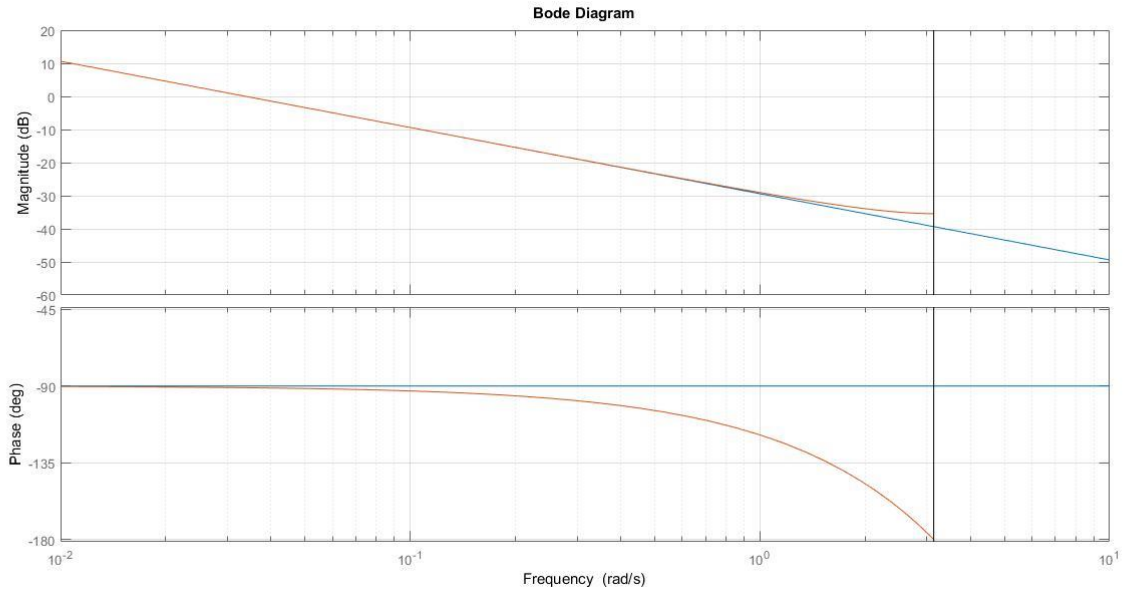
Antes de ejecutarlo se ha observado el bode de la Gba de bucle abierto para ver márgenes de estabilidad. Al tiempo de subida que estamos controlando (Ts = 88.5s), la Gba es fiable, respecto al controlador continuo. Después se ha hecho otra gráfica sobre la localización de los polos en el plano Z. Se ha observado que tiene un polo real negativo; el controlador tendrá 'chatering'.

El resultado ha sido un controlador estable, pero con 'chatering', como se había predicho.

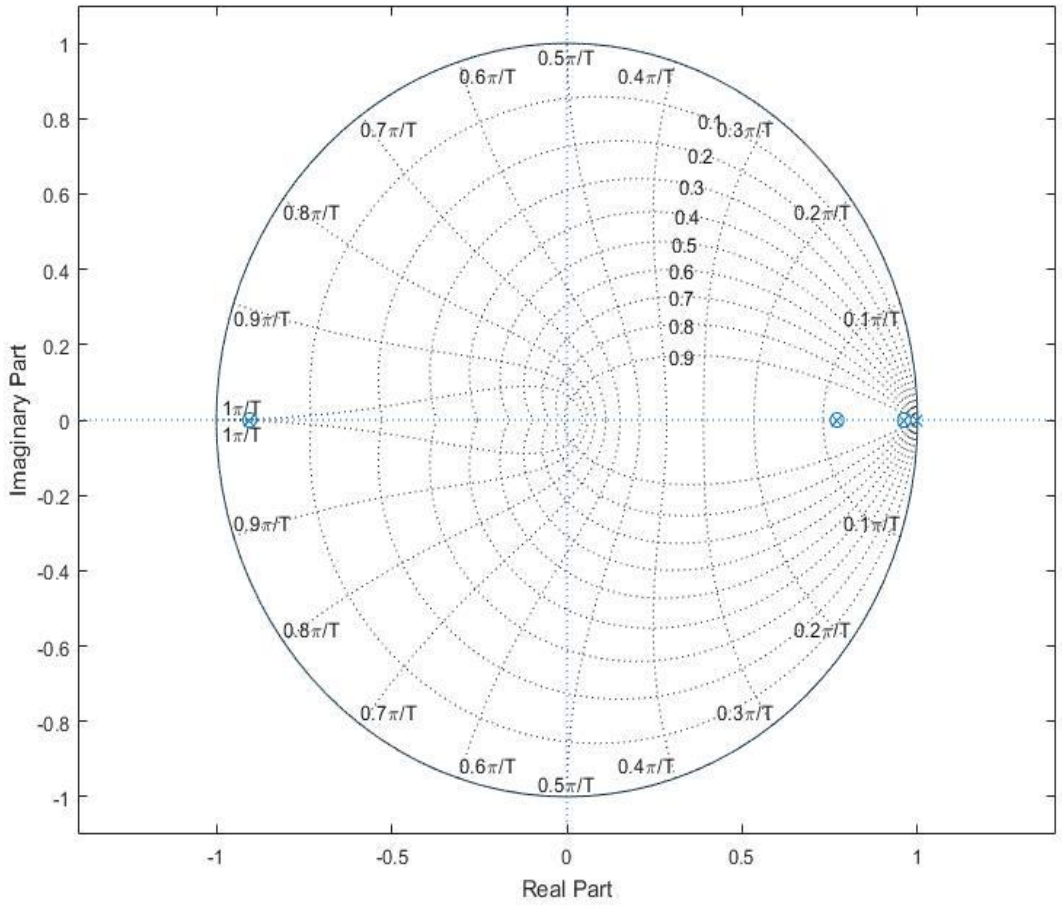
Las gfráficas se muestran en la siguiente hoja.

GRÁFICAS CONTROLADOR

Bode de la Gba (Ts = 88.5s):

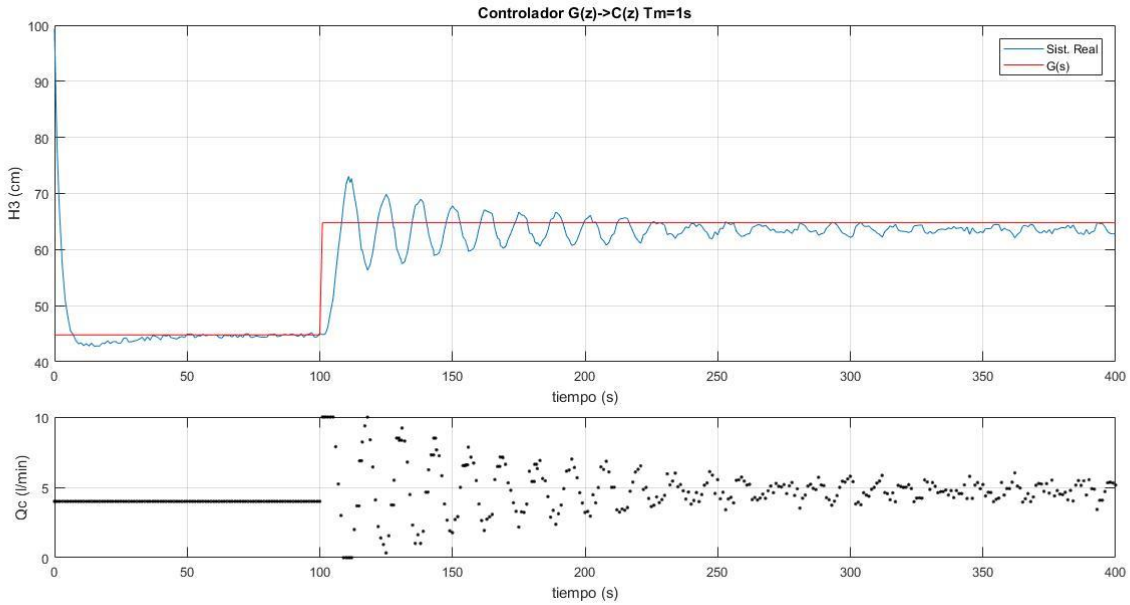


Mapeo de los polos y ceros del controlador:



GRÁFICAS CONTROLADOR

Respuesta temporal del controlador (fíjese que a los 70s se cierra el lazo suave, pero que la señal de control tiene chatering)



OBTENCIÓN DE CONTROLADORES MEDIANTE LA C(S)

Como se ha observado en el bode anterior, la función de transferencia C(s) está dentro de los márgenes de estabilidad. La función de transferencia es la siguiente:

$$C(s) = \frac{3.336\,s^2 + s + 0.0339}{028.01\,s}$$

Se ha equiparado a un PID, con estos parámetros:

$$Kp = 0.0357$$

$$Td = 3.3360$$

$$Ti = 29.4985$$

A continuación, se han hecho tres aproximaciones para su debida implementación. Éstos son los códigos:

```
/* CÓDIGO DEL CONTROLADOR APROXIMACIÓN TUSTIN */


  q0 = Kp*(1+Tm/(2*Ti)+Td/Tm);
  q1 = Kp*(-1+Tm/(2*Ti)-2*Td/Tm);
  q2 = Kp*(Td/Tm);


 tiempo = ssGetT(S);  /* Tiempo de la simulación */
 uk = 0;
 rk = Yeq;


 if (tiempo>test+50)
```

OBTENCIÓN DE CONTROLADORES MEDIANTE LA C(S)

```
 {
    rk=Yeq+20;
 };
 if (tiempo>test)
 {
    ek=rk-Yk;
    uk = uk1+q0*ek+q1*ek1+q2*ek2;
 };

 POutAux = rk;

/* Actualización de variables */
uk3=uk2; uk2=uk1; uk1=uk;
ek3=ek2; ek2=ek1; ek1=ek;
/* Cálculo de la señal de la salida real */
Ureal=Ueq+uk;
/* Saturación de la salida real (como medida de seguridad) */
if (Ureal > Umax ) Ureal=Umax;
else if (Ureal < Umin ) Ureal=Umin;
```

OBTENCIÓN DE CONTROLADORES MEDIANTE LA C(S)

```
/* CÓDIGO DEL CONTROLADOR APROXIMACIÓN EULER 1 */
   q0 = Kp*(1+Td/Tm);
   q1 = Kp*(-1+Tm/Ti-2*Td/Tm);
   q2 = Kp*(Td/Tm);
 tiempo = ssGetT(S);  /* Tiempo de la simulación */
 uk = 0;
 rk = Yeq;
 if (tiempo>test+50)
 {
    rk=Yeq+20;
 };
 if (tiempo>test)
 {
    ek=rk-Yk;
    uk = uk1+q0*ek+q1*ek1+q2*ek2;
 };
 POutAux = rk;
/* Actualización de variables */
uk3=uk2; uk2=uk1; uk1=uk;
ek3=ek2; ek2=ek1; ek1=ek;
/* Cálculo de la señal de la salida real */
Ureal=Ueq+uk;
```

OBTENCIÓN DE CONTROLADORES MEDIANTE LA C(S)

```
/* Saturación de la salida real (como medida de seguridad) */
if (Ureal > Umax ) Ureal=Umax;
else if (Ureal < Umin ) Ureal=Umin;
```

```
/* CÓDIGO DEL CONTROLADOR APROXIMACIÓNEULER 2 */
q0 = Kp*(1+Td/Tm+Tm/Ti);
q1 = Kp*(-1-2*Td/Tm);
q2 = Kp*(Td/Tm);
 tiempo = ssGetT(S);  /* Tiempo de la simulación */
 uk = 0;
 rk = Yeq;
 interr += ek1;
 if (tiempo>test+50)
 {
    rk=Yeq+40;
 };
 if (tiempo>300)
 {
    rk=Yeq+500;
```

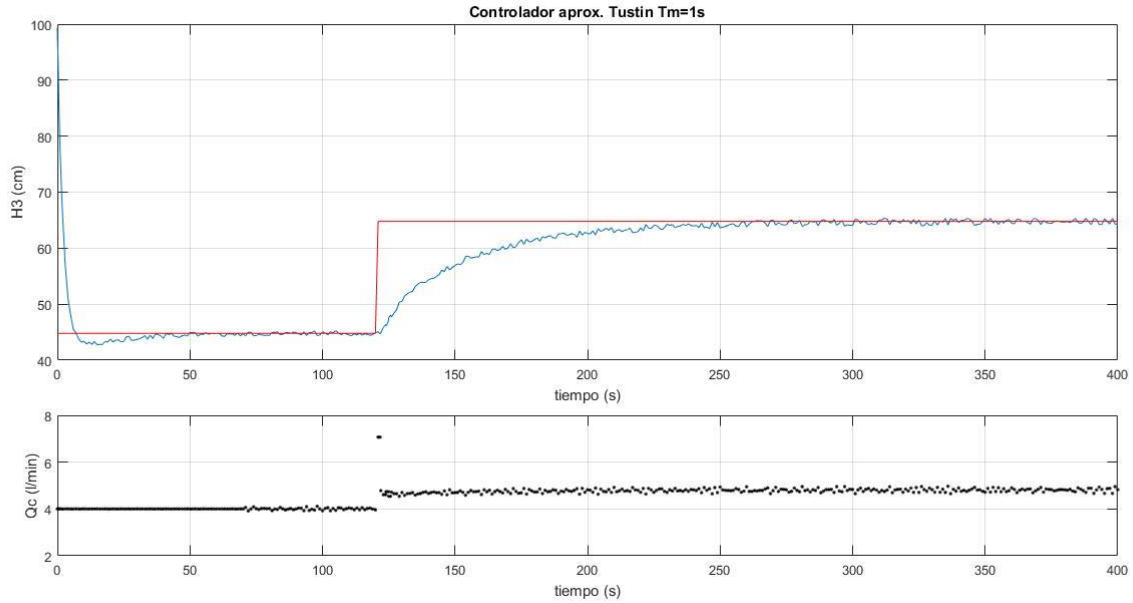
OBTENCIÓN DE CONTROLADORES MEDIANTE LA C(S)

```
 };
  if (tiempo>600)
 {
     rk=Yeq+40;
 };
 if (tiempo>test)
 {
     ek=rk-Yk;
     uk = Kp*(ek+Tm/Ti*interr+Td/Tm*(ek-ek1));
    // uk = uk1+q0*ek+q1*ek1+q2*ek2; // Forma incremental
 };
 POutAux = rk;
/* Actualización de variables */
uk3=uk2; uk2=uk1; uk1=uk;
ek3=ek2; ek2=ek1; ek1=ek;
/* Cálculo de la señal de la salida real */
Ureal=Ueq+uk;
/* Saturación de la salida real (como medida de seguridad) */
if (Ureal > Umax ) Ureal=Umax;
else if (Ureal < Umin ) Ureal=Umin;
```

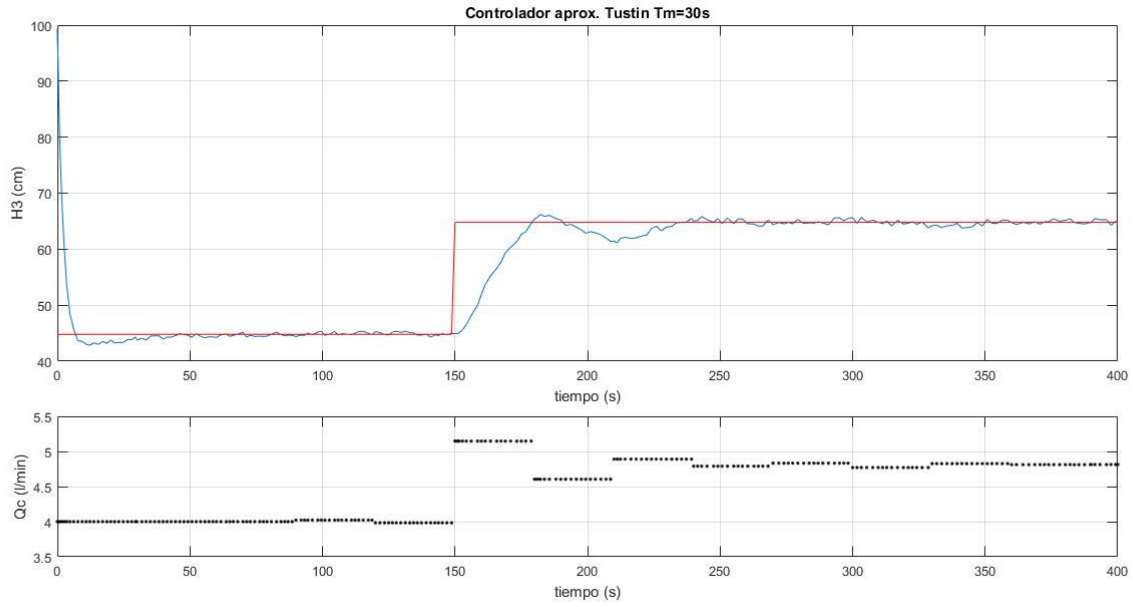
Las gráficas obtenidas son las siguientes:

GRÁFICAS DE LOS CONTROLADORES

Para la aproximación Tustin con Tm = 1s:

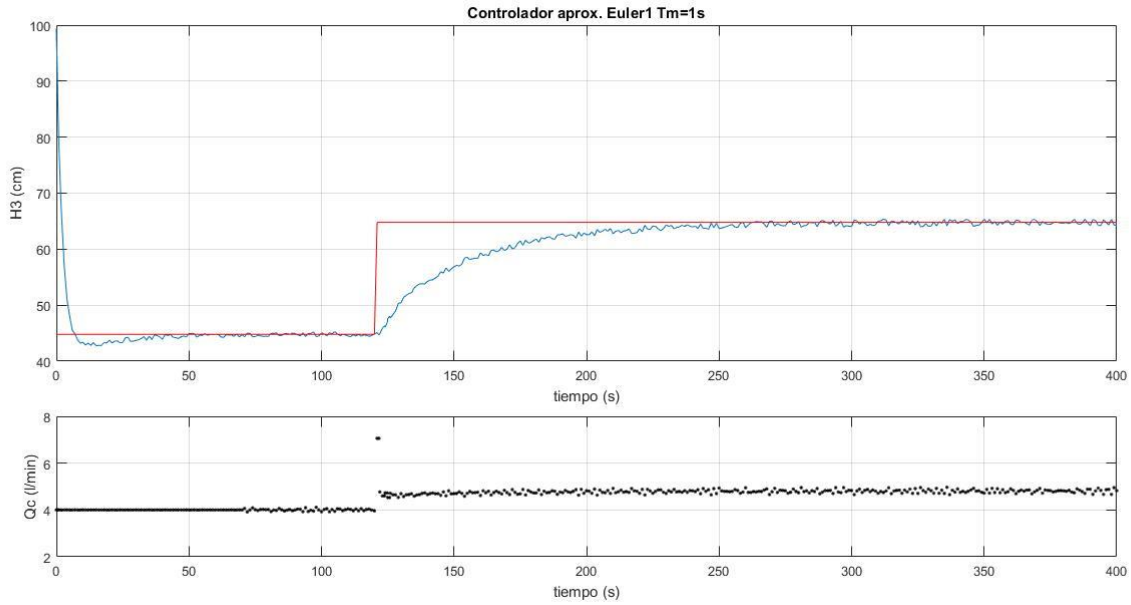


Después se ha cambiado el Tm a 30s, para ver el empeoro del controlador, dado que el Tm al ser mayor, en el bode se refleja con un desplazamiento hacia la izquierda del retraso. El resultado es que perdemos fase. La gráfica es la siguiente:



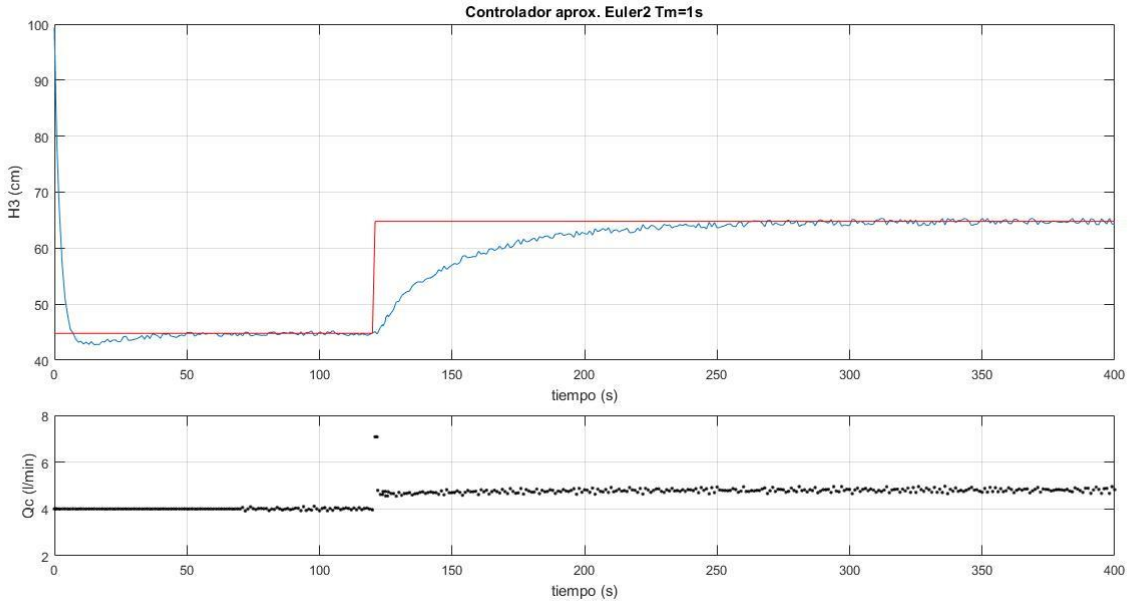
GRÁFICAS DE LOS CONTROLADORES

La gráfica de la aproximación Euler 1 es la siguiente:



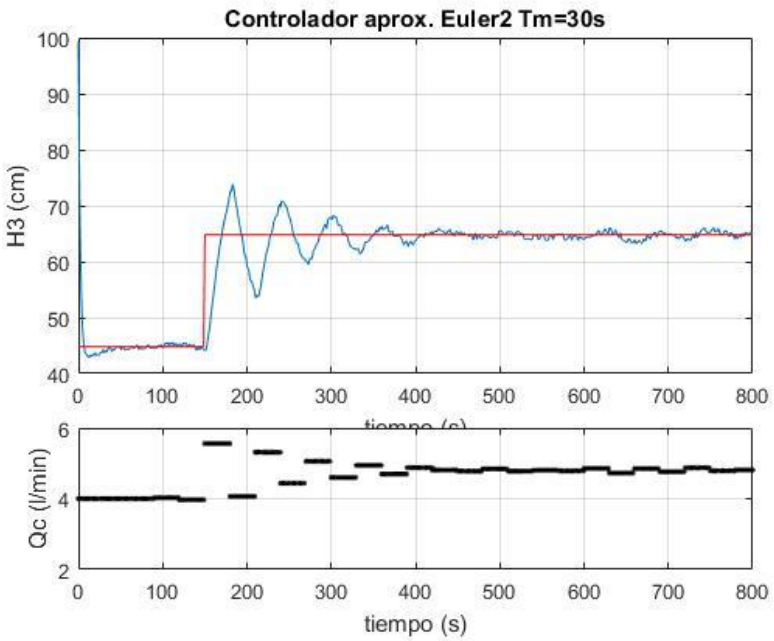
No se ha incorporado otra gráfica cambiando el tiempo de muestreo porque sucede lo mismo.

Por último, las gráficas del Euler 2:



GRÁFICAS DE LOS CONTROLADORES

A éste si se le ha cambiado el tiempo de muestreo porque es el que vamos a utilizar para los mecanismo de AntiWindup:



Como se aprecia, empeora igual que en la aproximación Tustin; por tanto escogemos este controlador (esta aproximación nos garantiza que tenemos todos los polos y ceros en el plano de Z positivo y dentro del círculo unidad) para un Tm=1s.

MECANISMOS DE ANTIWINDUP

Se han implementado tres mecanismos; los códigos son los siguientes:

```
/* CÓDIGO DEL CONTROLADOR EULER 2 CON AW1*/


q0 = Kp*(1+Td/Tm+Tm/Ti);
q1 = Kp*(-1-2*Td/Tm);
q2 = Kp*(Td/Tm);

tiempo = ssGetT(S);  /* Tiempo de la simulación */
uk = 0;
rk = Yeq;

if (tiempo>test+50)
{
rk=Yeq+20;
};
if (tiempo>300)
{
rk=Yeq+500;
};
if (tiempo>600)
{
rk=Yeq+20;
};
```

MECANISMOS DE ANTIWINDUP

```
if (tiempo>test)
{
ek=rk-Yk;
uk = uk1+q0*ek+q1*ek1+q2*ek2;
};

POutAux = rk;
//   ANTIWINDUP

Ureal=Ueq+uk;
if (Ureal > Umax) uk = uk1;
if (Ureal < Umin) uk = uk1;

/* Actualización de variables */
uk3=uk2; uk2=uk1; uk1=uk;
ek3=ek2; ek2=ek1; ek1=ek;
/* Cálculo de la señal de la salida real */
Ureal=Ueq+uk;
/* Saturación de la salida real (como medida de seguridad) */
if (Ureal > Umax ) Ureal=Umax;
else if (Ureal < Umin ) Ureal=Umin;

/* CÓDIGO DEL CONTROLADOR EULER 2 CON AW2*/
```

MECANISMOS DE ANTIWINDUP

```
//     q0 = Kp*(1+Td/Tm+Tm/Ti);
//     q1 = Kp*(-1-2*Td/Tm);
//     q2 = Kp*(Td/Tm);

tiempo = ssGetT(S);  /* Tiempo de la simulación */
uk = 0;
rk = Yeq;

interr += ek1;
if (tiempo>test+50)
{
rk=Yeq+40;
};
if (tiempo>300)
{
rk=Yeq+500;
};
if (tiempo>600)
{
rk=Yeq+40;
};
if (tiempo>test)
```

MECANISMOS DE ANTIWINDUP

```
{
ek=rk-Yk;
uk = Kp*(ek+Tm/Ti*interr+Td/Tm*(ek-ek1));
//       uk = uk1+q0*ek+q1*ek1+q2*ek2;
};

POutAux = rk;

// ANTIWINDUP
Ureal=Ueq+uk;

/* CÓDIGO DEL CONTROLADOR EULER 2 CON AW4*/

//     q0 = Kp*(1+Td/Tm+Tm/Ti);
//     q1 = Kp*(-1-2*Td/Tm);
//     q2 = Kp*(Td/Tm);

tiempo = ssGetT(S);  /* Tiempo de la simulación */
uk = 0;
rk = Yeq;
```

MECANISMOS DE ANTIWINDUP

```
interr += ek1;
interrsat += esat;
if (tiempo>test+50)
{
rk=Yeq+40;
};
if (tiempo>300)
{
rk=Yeq+500;
};
if (tiempo>600)
{
rk=Yeq+40;
};
if (tiempo>test)
{
ek=rk-Yk;
uk = Kp*(ek+Tm/Ti*interr+Td/Tm*(ek-ek1))+Kp/Tt*interrsat;
//      uk = uk1+q0*ek+q1*ek1+q2*ek2;
};

POutAux = rk;
```

MECANISMOS DE ANTIWINDUP

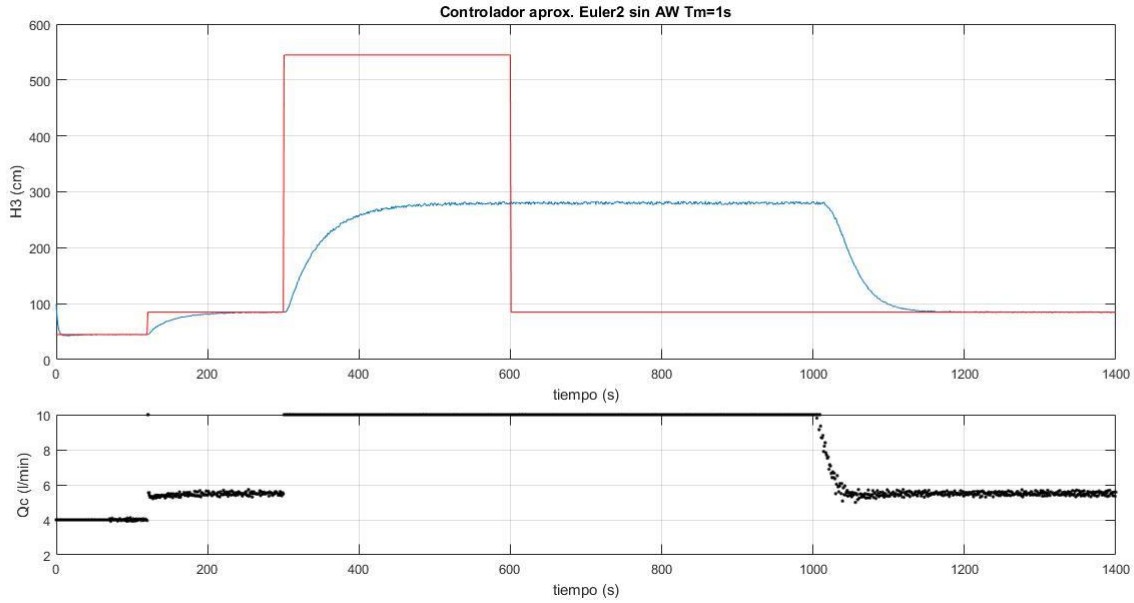
```
/* Actualización de variables */
uk3=uk2; uk2=uk1; uk1=uk;
ek3=ek2; ek2=ek1; ek1=ek;
/* Cálculo de la señal de la salida real */
Ureal=Ueq+uk;
Uasat=Ureal;
/* Saturación de la salida real (como medida de seguridad) */
if (Ureal > Umax ) Ureal=Umax;
else if (Ureal < Umin ) Ureal=Umin;

esat = Ureal-Uasat;
```

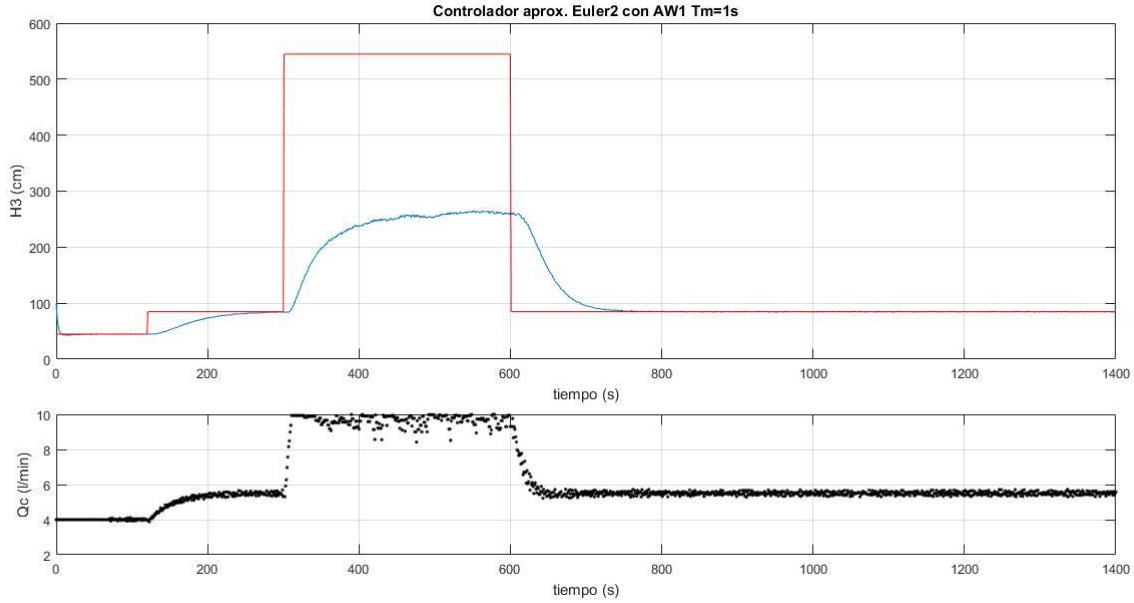
Las gráficas, en orden, son las siguientes:

GRÁFICAS DE ANTIWINDUP

SIN ANTIWINDUP

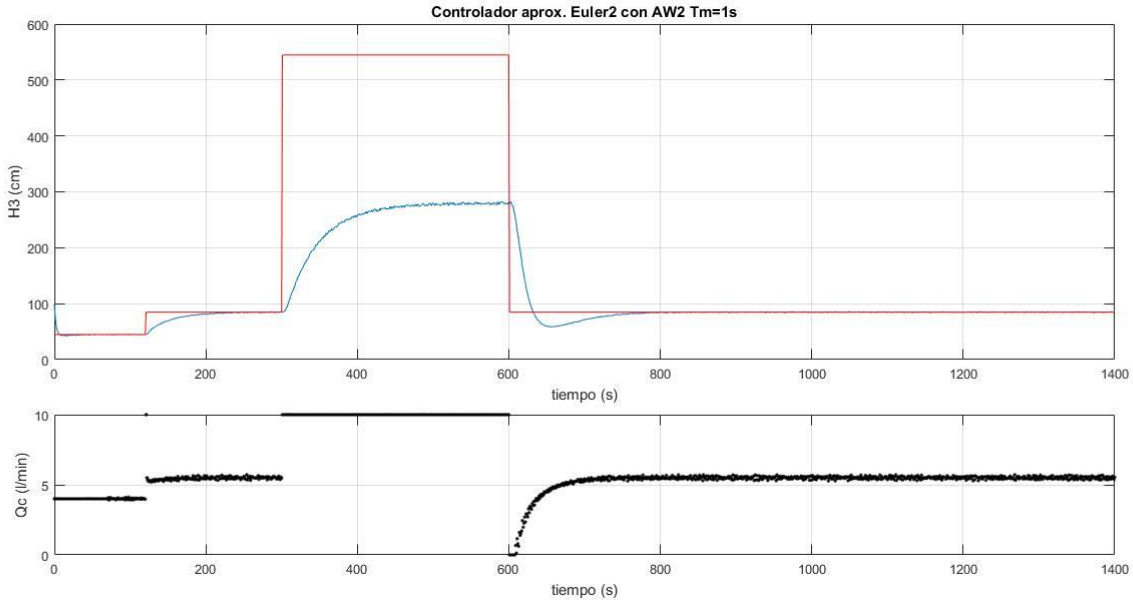


Antiwindup 1:



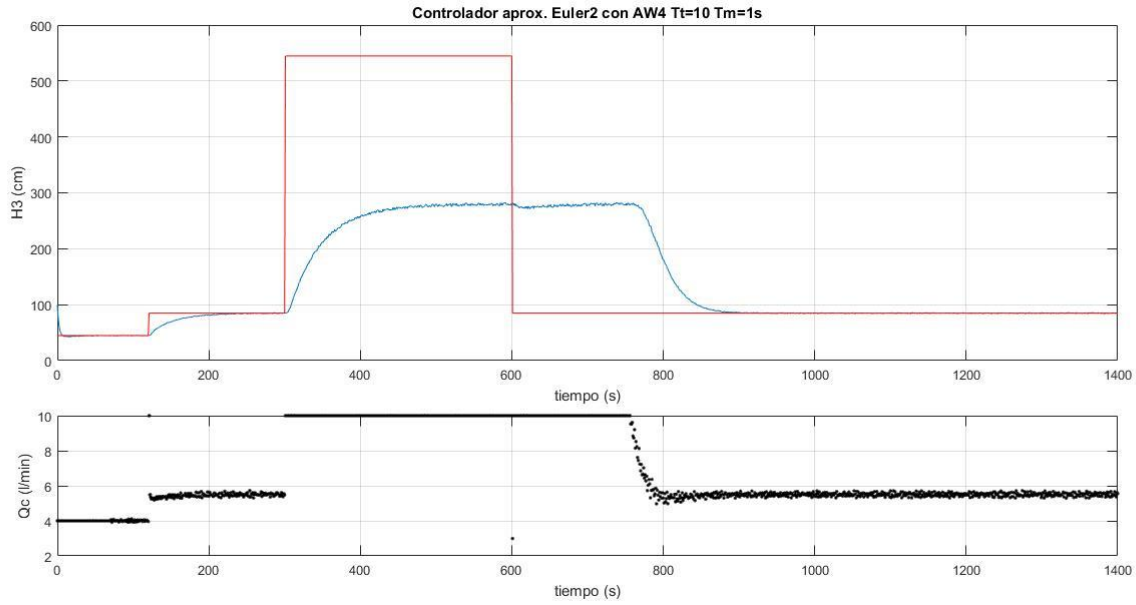
GRÁFICAS DE ANTIWINDUP

Antiwindup 2:



GRÁFICAS DE ANTIWINDUP

Para un Tt=10



Para un Tt = 2;