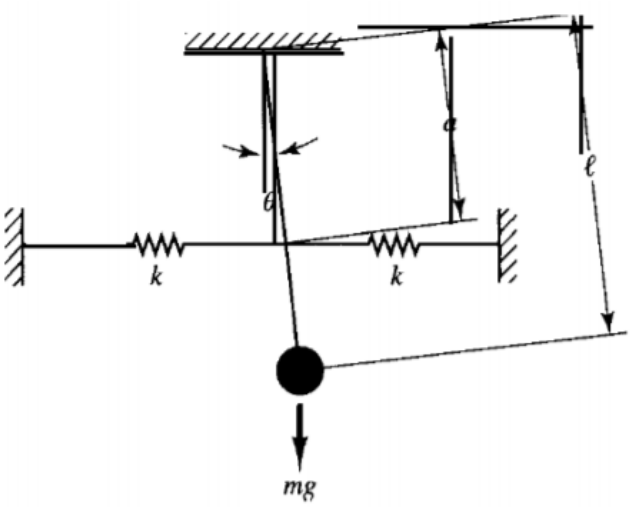
Julian Cortes – Santiago Acebes
1803147 - 1803124

Taller 2 Control

1.



De manera inicial tenemos que el sistema se modelara de tal forma que exista una entrada de torque en la bola, con esto se tienen las siguientes ecuaciones:

$$\dot{\omega} = \frac{1}{ml^2}(\tau_a - 2ka^2 sin\theta cos\theta - lmgsin\theta)$$

$$\dot{\theta} = \omega$$

Con

$$x_e = [\omega \quad \theta]$$

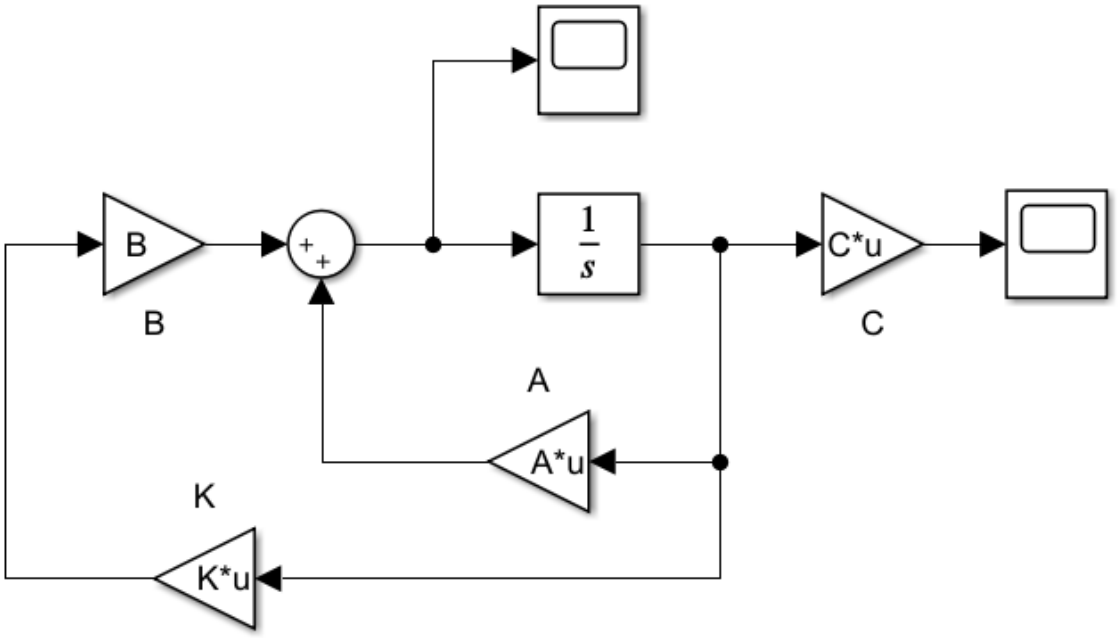
Con esta información se halla la función de trasferencia teniendo en cuenta los siguientes valores:

```
m=1;
g=9.81;
l=0.1;
a=l/2;
k=0.1;
```

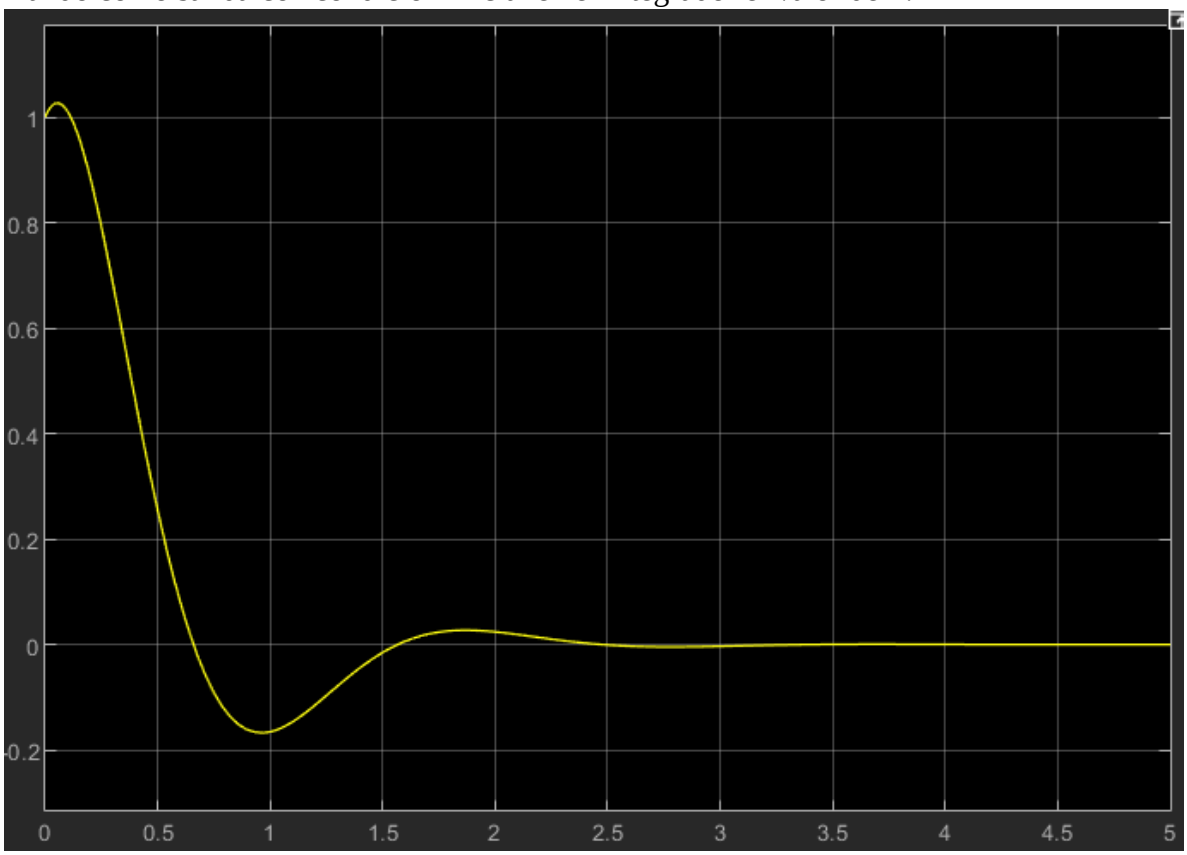
Con lo que la función de transferencia es:

```
      100
-----------
s^2 + 49.03
```

Por lo que haciendo la retroalimentación por Ackermann usando acker() tenemos:



Dando como salida con condición inicial en el integrador el valor de 1:



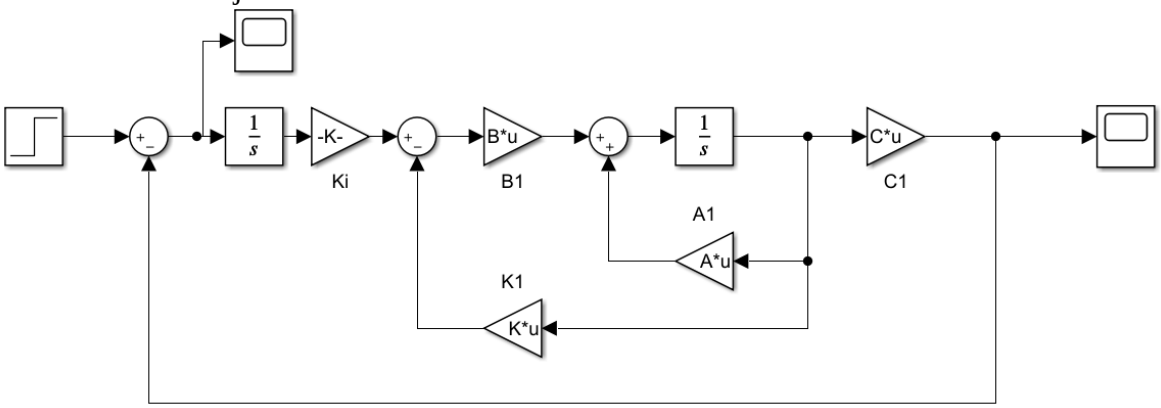
Por lo que vemos que el sistema en 2s se esta estabilizando, se pretende usar parámetros de diseño para el controlador de tsd=1s y ζ=1.

Usando Ackermann

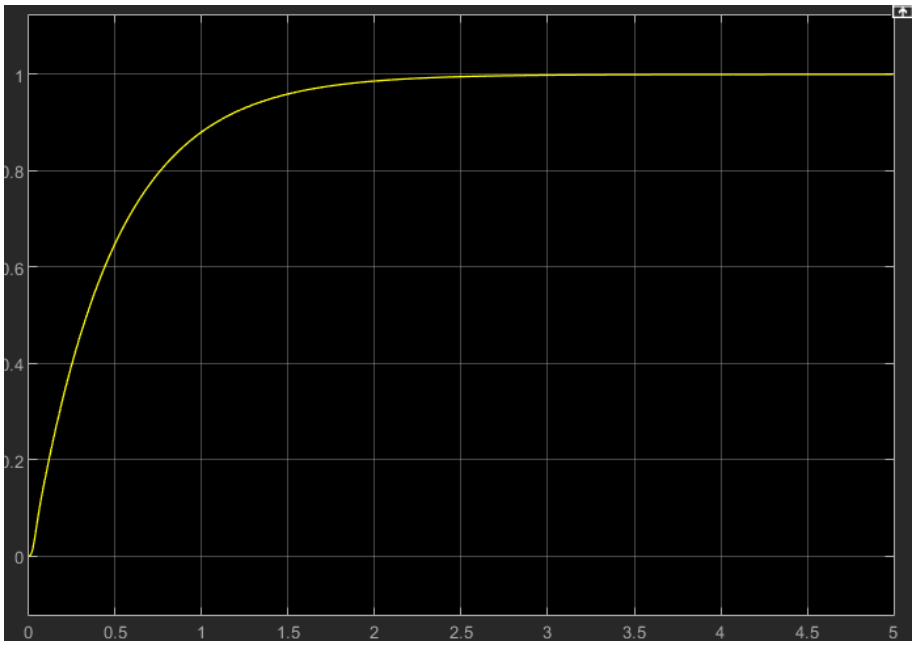
```
wn=4/(1*1)
polos=roots([1 4 16])
K=acker(A,B,polos)
%PARA ESS=0
polosEss=conv(polos,[1 5*4])
Aempaq=[A(1,1),A(1,2),0;A(2,1),A(2,2),0;-C(1,1),
Bempaq=[B(1,1);B(2,1);0]
Cempaq=[C(1,1),C(1,2),0]
K1=acker(Aempaq,Bempaq,polosEss)
```

Tenemos que el sistema arroja las constantes del controlador para Ess=0
K1=[0.84 63.66 -134.4]

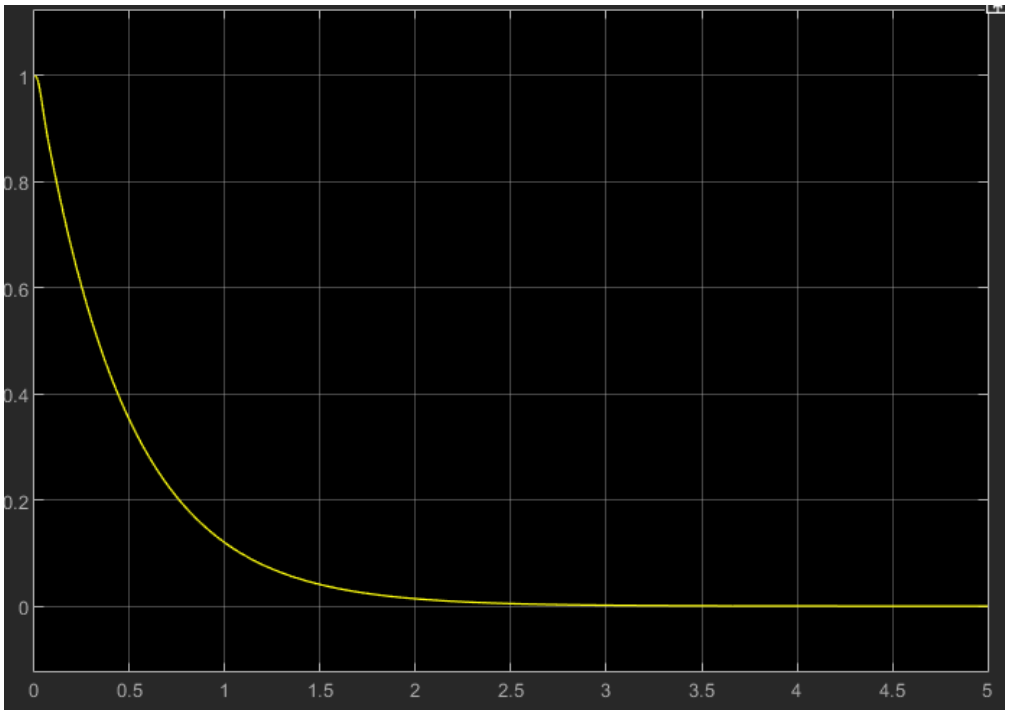
Donde en este vector esta los primeros términos k1 y k2 y el ultimo es -ki
Haciendo el montaje en Simulink tenemos:



Donde si vemos la salida:



Y el error:



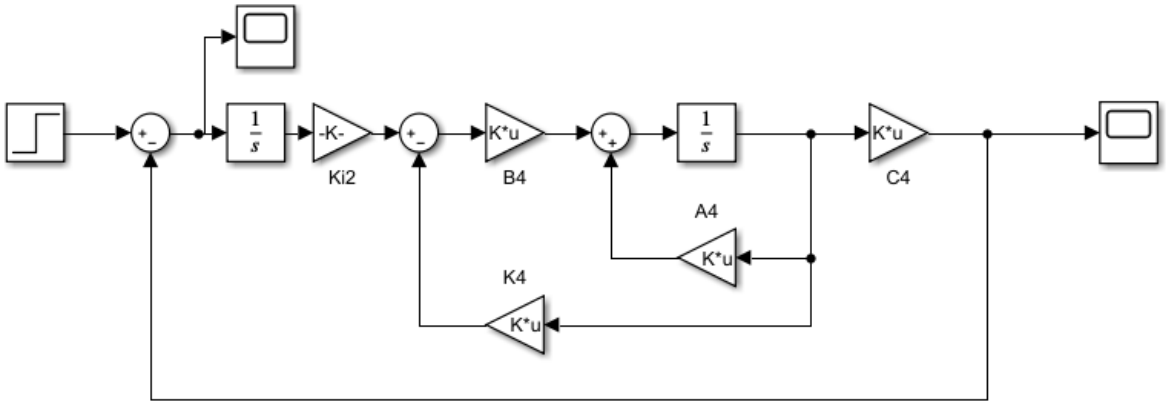
Si colocamos un tiempo muerto de 1s, tenemos las siguientes consideraciones:

- La planta ya no será al cuadrado, será una función de orden 4
- Nuestras matrices A, B, C, D empaquetadas no serán ahora de orden 3, si no de orden 5

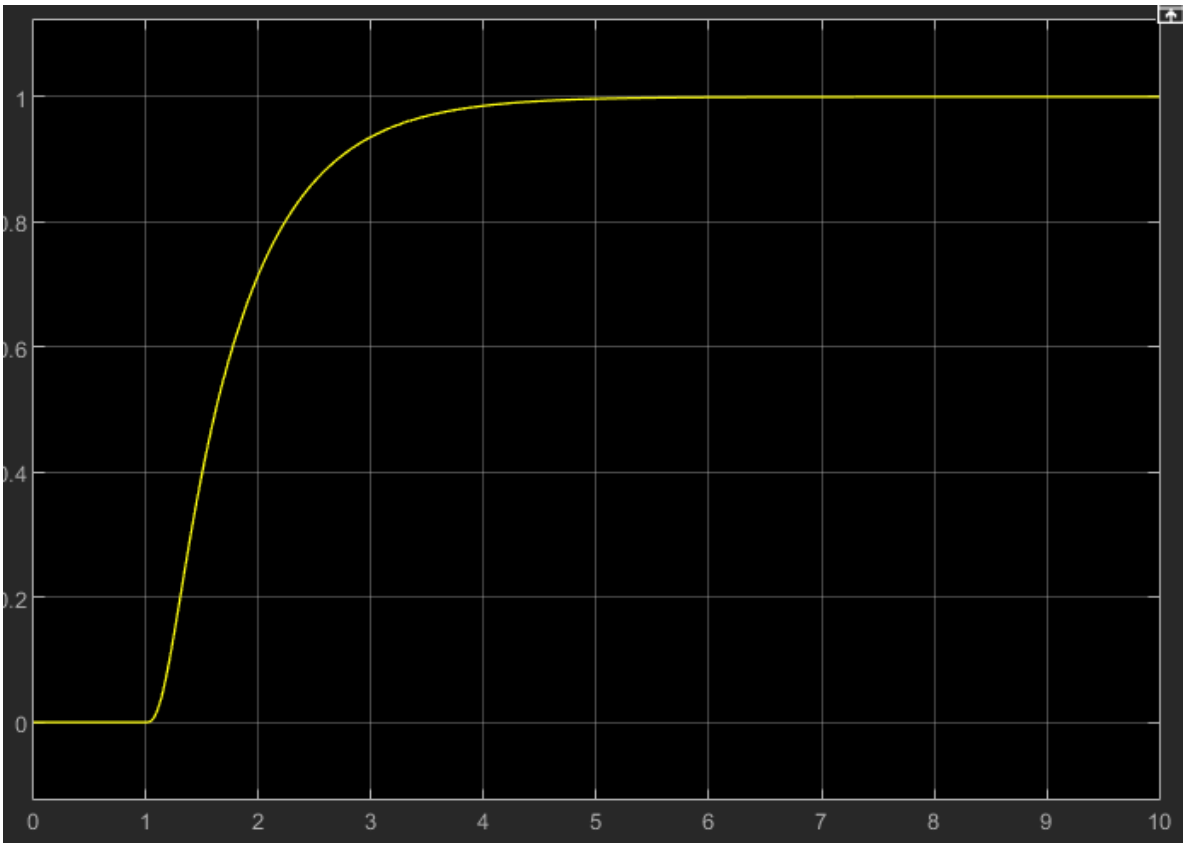
Realizando el mismo procedimiento que se tenía antes:

```
tm=1
polosEssT=conv(polosEss,[1 10*wn*zeta])
polosEssT=conv(polosEssT,[1 10*wn*zeta])
ftdelay=tf([1],[(tm^2)/2 tm 1])
ft2=ft*ftdelay
[A1,B1,C1,D1]=tf2ss([0.5,5.747],[0.245,4.251,14.7,24.64,
Aempaq1=[A1(1,1),A1(1,2),A1(1,3),A1(1,4),0;A1(2,1),A1(2,
Bempaq1=[B1(1,1);B1(2,1);B1(3,1);B1(4,1);0]
KDelay=acker(Aempaq1,Bempaq1,polosEssT)
```

Se hace el montaje:



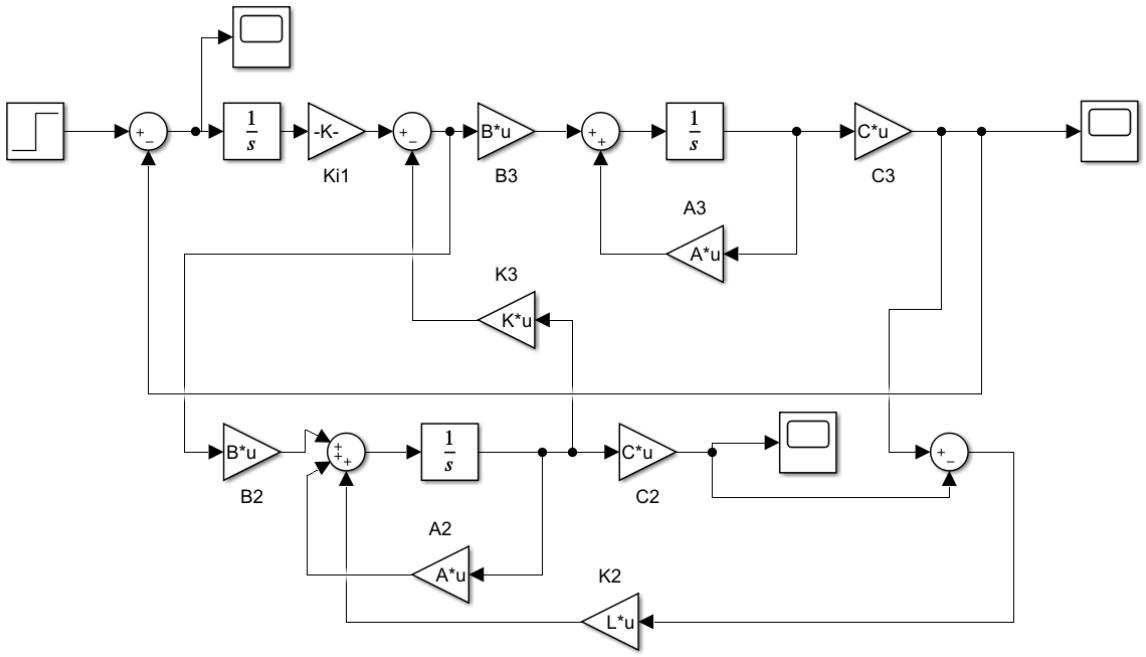
Con la salida:



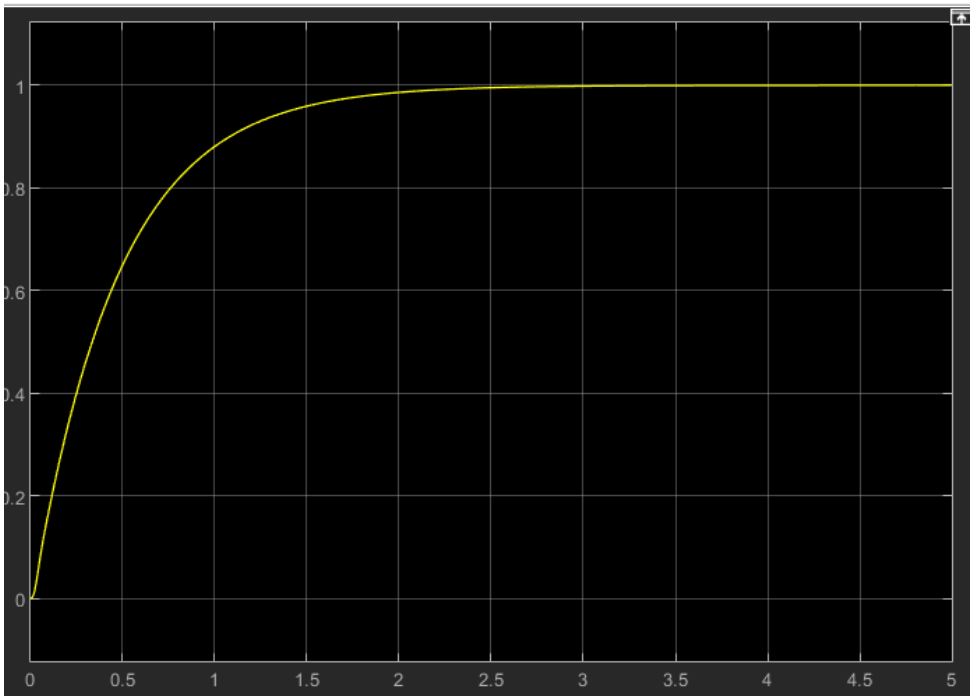
Finalmente se pide realizar un observador de estados para un sensor de la posición, con esto se hace el respectico calculo de L para el observador:

```
obs=[C;C*A]
det(obs)
pdo=conv([1 2*1*40 1600],[1 10*40])
phia= A^3+pdo(2)*A^2+pdo(3)*A+pdo(4)*eye(2)
L=phia*(inv(obs))*[0;1]
```

Se hace el montaje en Simulink:

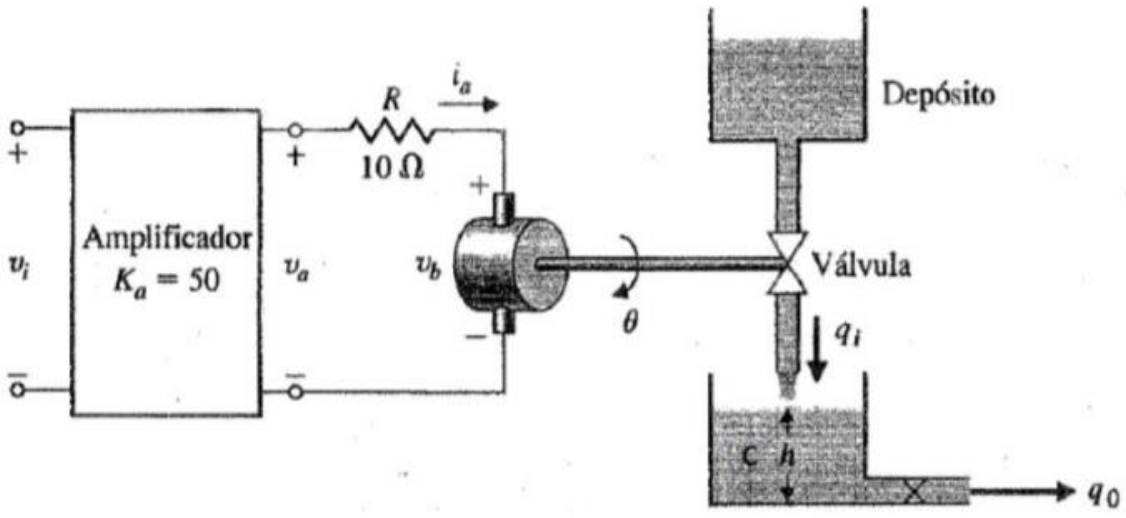


Si miramos la salida del observador tenemos:



Que corresponde a la posición angular.

2.



Realizando los respectivos modelos de los sistemas eléctricos e hidráulicos con flujo turbulento tenemos:

$$\dot{\theta} = \frac{1}{K_\omega}(50vin - 10ia)$$

$$\dot{h} = \frac{1}{C}\left(80\theta - k\sqrt{h}\right)$$

Con una ecuación de salida:

$$y = q_0 = k\sqrt{h}$$

Con estados dados de la forma:

$$x_e = [\,\theta \;\; h\,]$$

Se emplea un punto de operación de h=1m, con eso se realiza el calculo de las matrices A, B, C, D y la matriz K para una retroalimentación de estados sin ess=0 para escalón, además se diseñará para ts=1s y coeficiente amortiguamiento de 1.

```
ia=1;
kw=0.1;
C=1;
k=50;
```

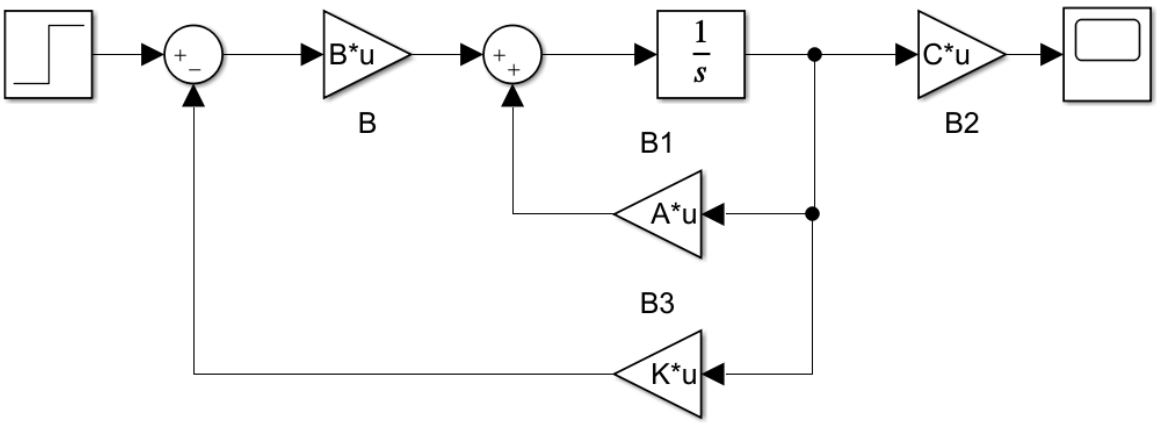
Se usan las constantes mostradas anteriormente

```
%%%%%%%% P.op
h=1;
%%%%%%%%
A=double(eval(A))
B=double(eval(B))
C=double(eval(C))
D=double(eval(D))
[num,den]=ss2tf(A,B,C,D)
ft=tf(num,den)
step(ft)
```

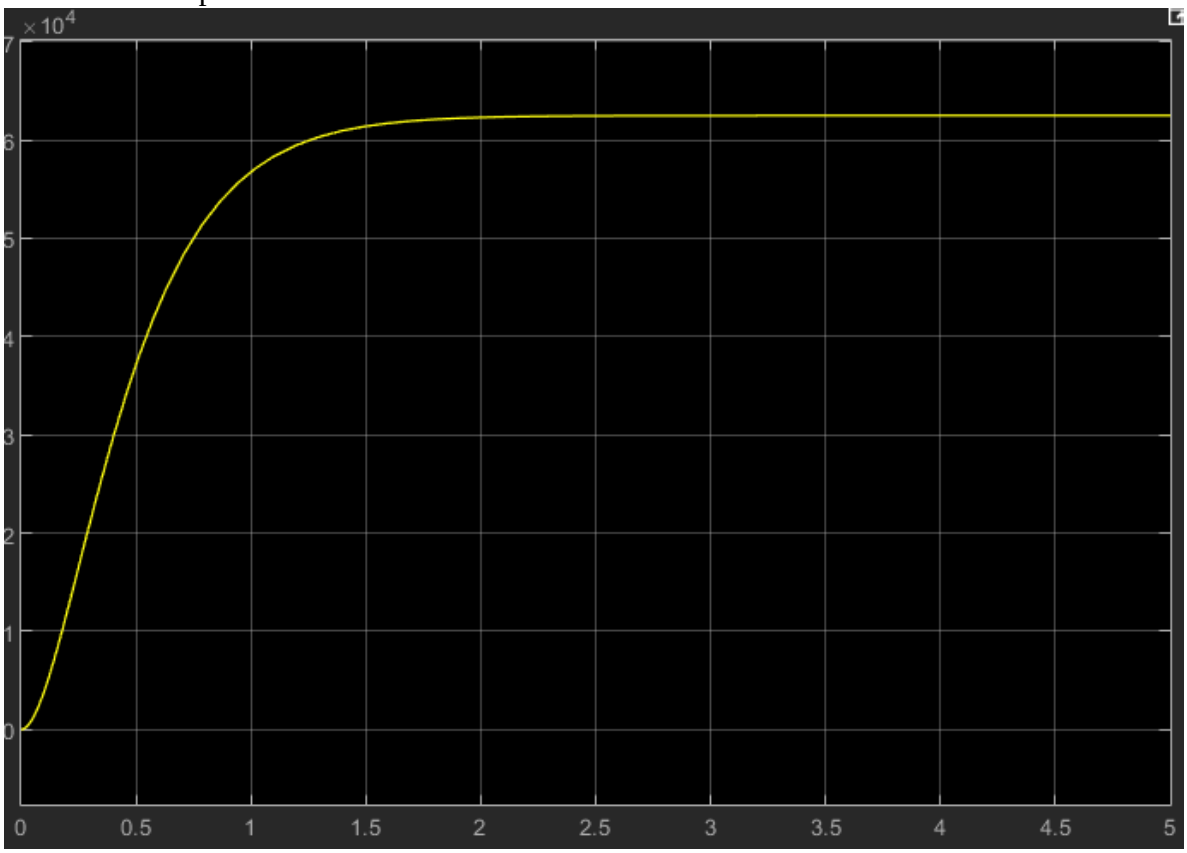
La función de transferencia del sistema vemos que es:

```
     1e06
----------
s^2 + 25 s
```

Realizando el esquema en Simulink de la retroalimentación con una perturbación tenemos:



Tenemos la respuesta:

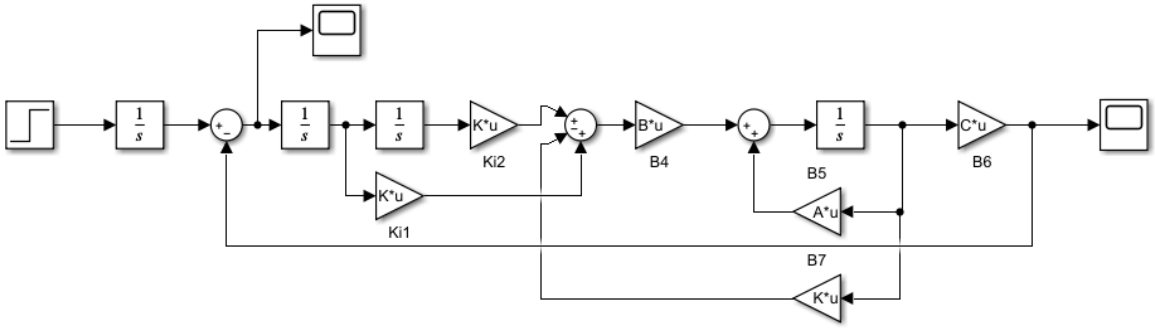


Vemos como el error no es cero, pero si hacemos el diseño del controlador para ess=0 para rampa

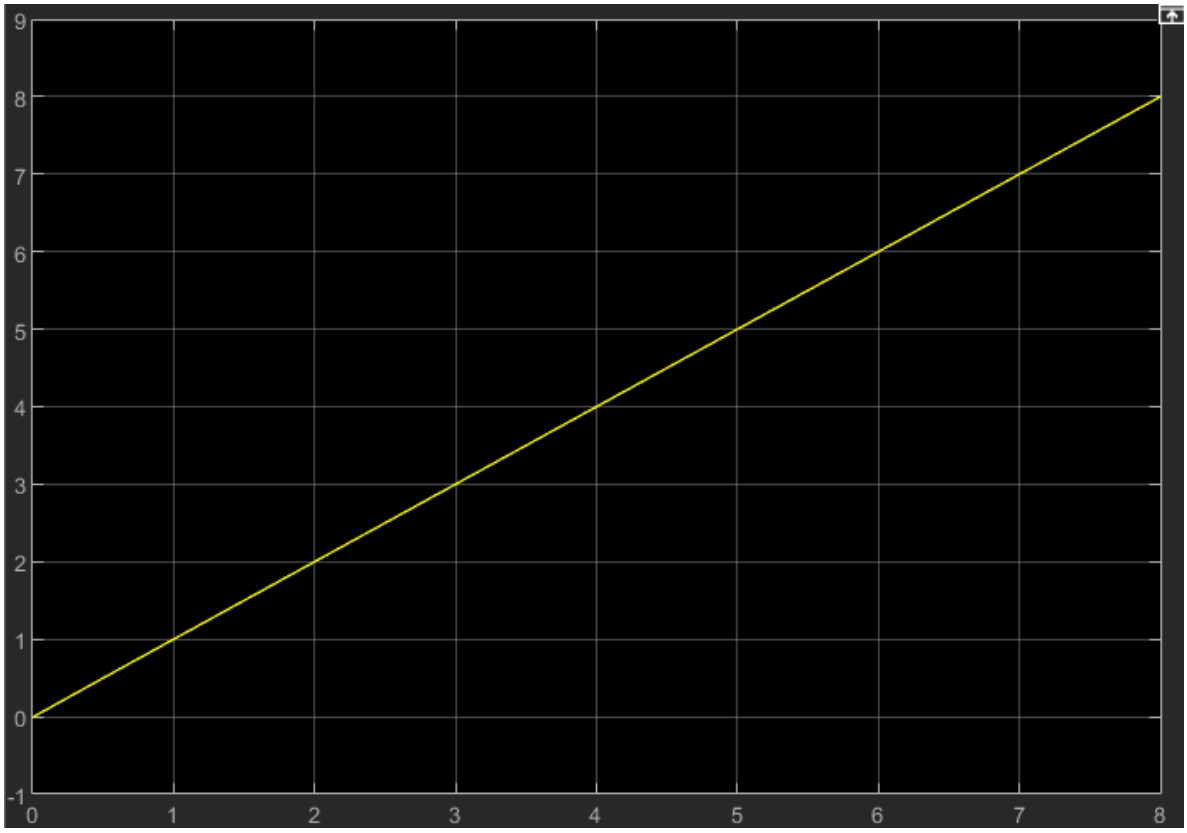
Tenemos:

```
polosEss=conv(polos,[1 5*wn*zta])
polosEss=conv(polosEss,[1 5*wn*zta])
Aempaq=[A(1,1),A(1,2),0,0;A(2,1),A(2,2),0,0;0,0,0,1;-C(1
Bempaq=[B(1,1);B(2,1);0;0]
Cempaq=[C(1,1),C(1,2),0,0]
K1=acker(Aempaq,Bempaq,polosEss)
```

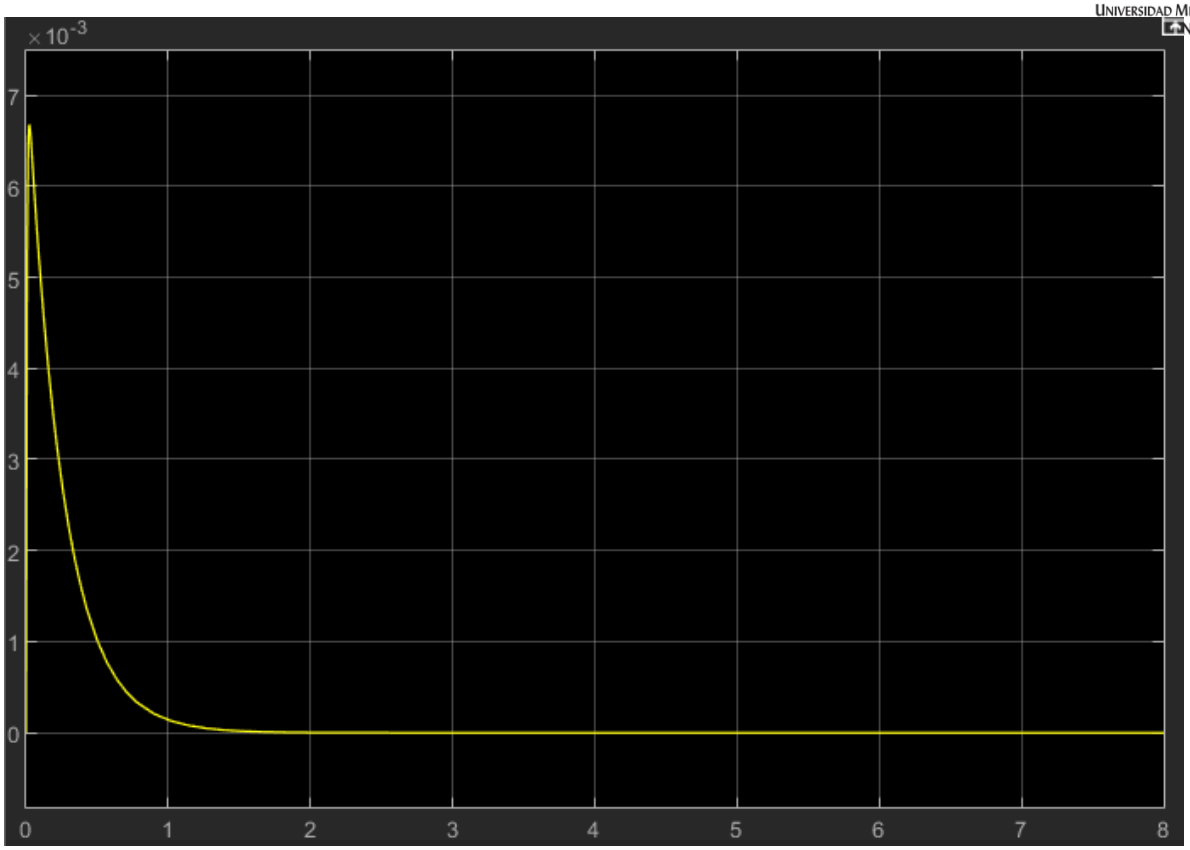
De esta forma se hace el esquema en Simulink:



Dando la salida:



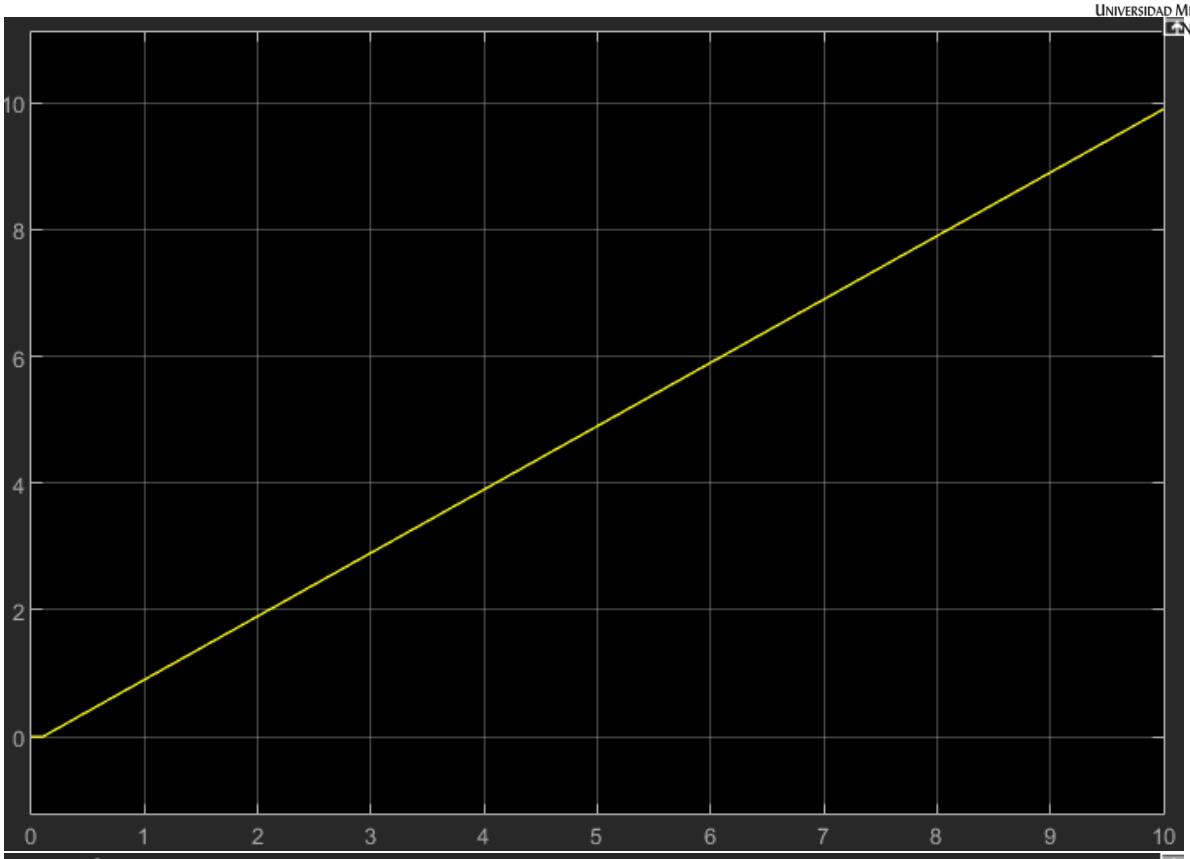
Con un error=0.

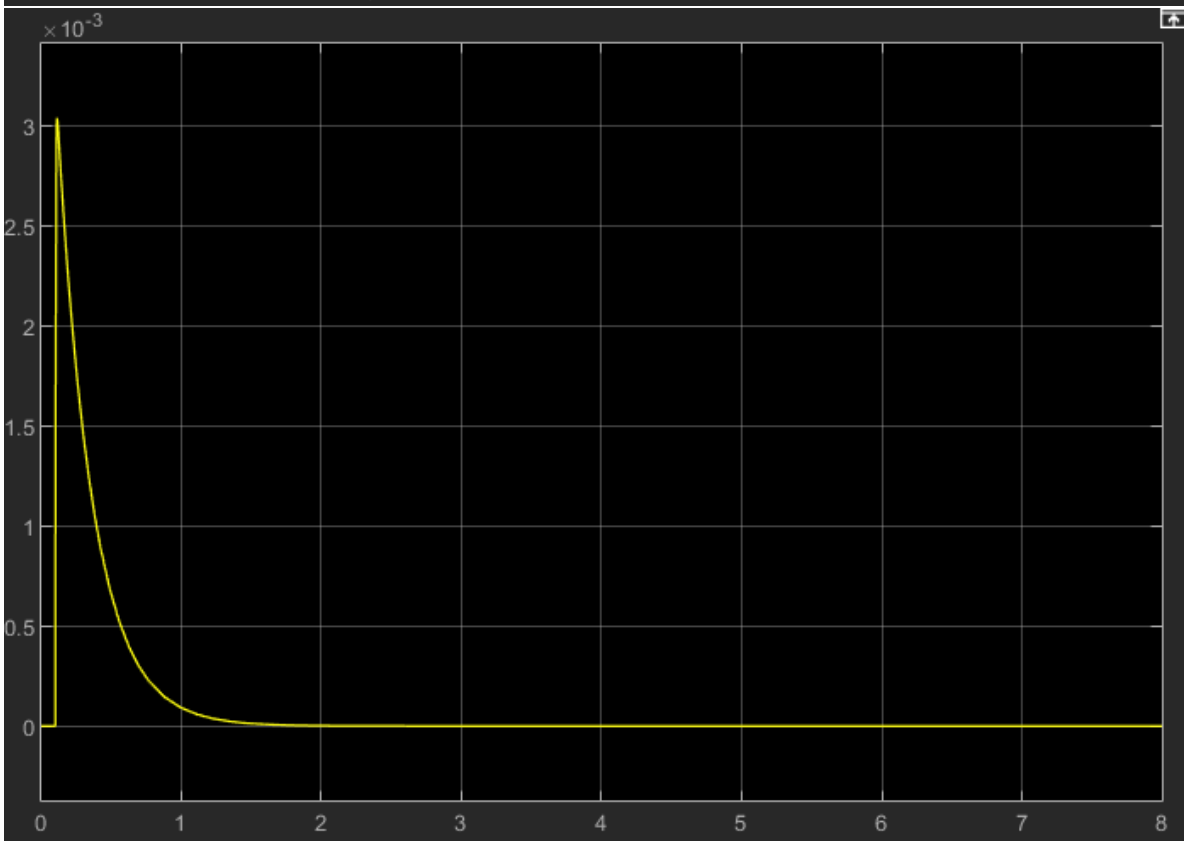


Si ahora se pone un DELAY de 0.1s tenemos:

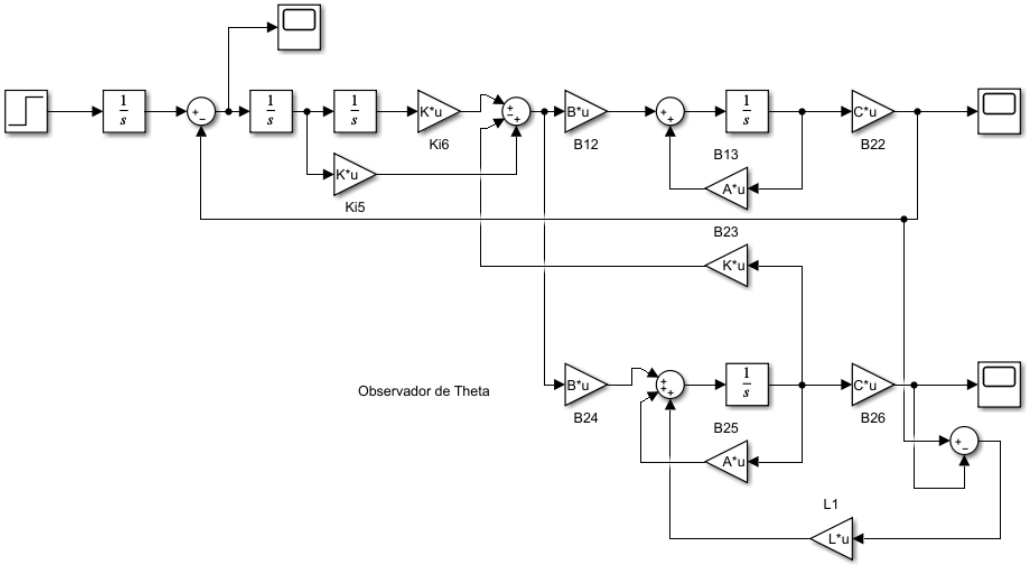
```
%%%% Para Transport delay 0.1s
tm=0.1
polosEssT=conv(polosEss,[1 5*wn*zta])
polosEssT=conv(polosEssT,[1 5*wn*zta])
ftdelay=tf([1],[(tm^2)/2 tm 1])
ft2=ft*ftdelay
[A1,B1,C1,D1]=tf2ss(1e6,[0.005,0.225,3.5,25,0])
Aempaq1=[A1(1,1),A1(1,2),A1(1,3),A1(1,4),0,0;A1(2,1),A1
Bempaq1=[B1(1,1);B1(2,1);B1(3,1);B1(4,1);0;0]
KDelay=acker(Aempaq1,Bempaq1,polosEssT)
```

Eso nos da una matriz de 6x6 en las empaquetadas, por lo que usando el principio de multiplicación de bloques, multiplicamos la función de transferencia con la del tiempo muerto, luego se realiza el calculo del controlador respectivo, con una salida y un error:





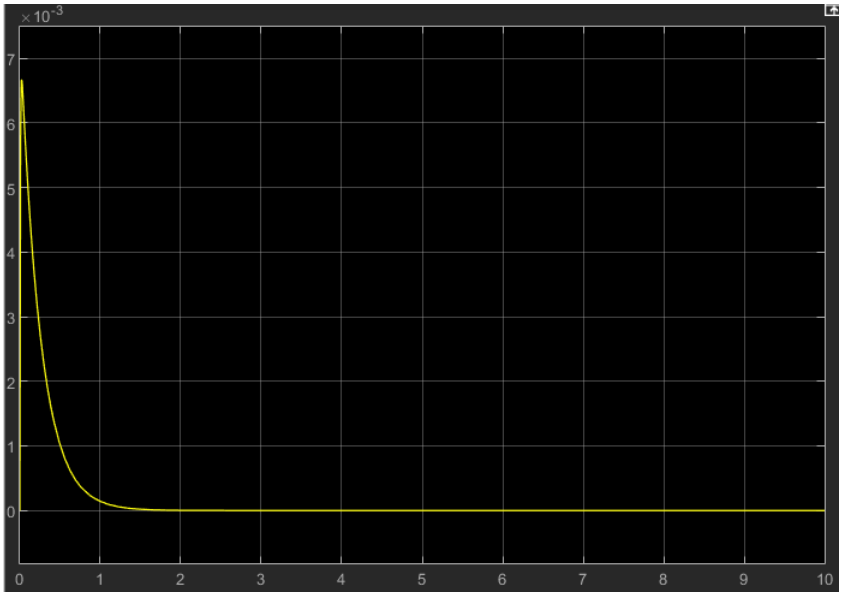
Si montamos un observador para la posición del motor tenemos lo siguiente:



Diseñando el observador la matriz L se da de:

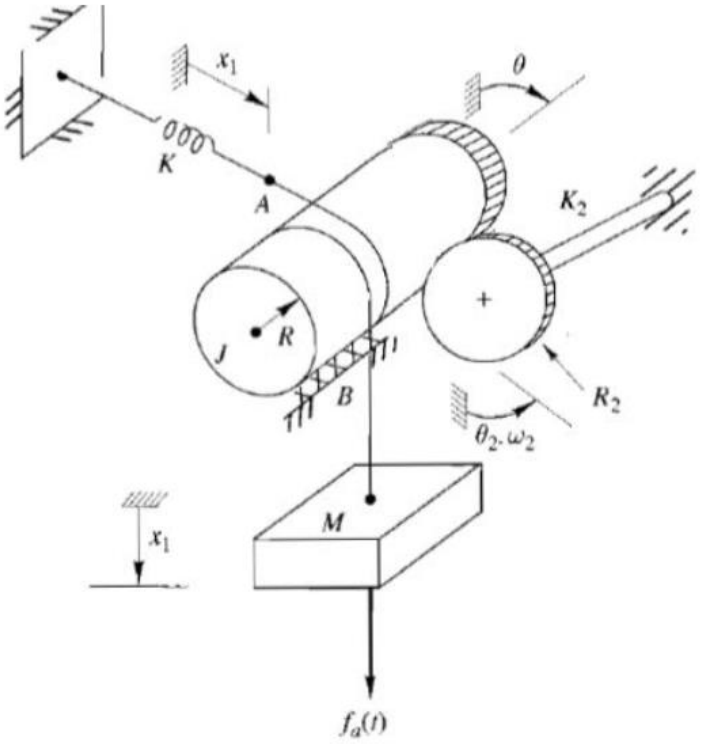
```
tsa=ts/10;
wna=4/(tsa*zta)
obs=[C;C*A]
det(obs)
pdo=conv([1 2*tsa*zta wna^2],[1 10*wna*zta])
phia= A^3+pdo(2)*A^2+pdo(3)*A+pdo(4)*eye(2)
L=phia*(inv(obs))*[1;0]
```

Dando como resultado error:



Por lo que no afecta al sistema el observador.

3.



Para este caso analizamos 3 elementos principales, los cuales son el rodillo central, el piñón y la masa que cuelga del rodillo, de esto se sabe que las ecuaciones que resultan son únicamente la ecuación del rodillo y la ecuación de remplazo para dejar únicamente una máxima derivada en la ecuación.

$$\dot{\omega} = \frac{1}{J + mR^2}\left(R(f_a + mg) - B\omega - KR^2\theta - \frac{R^2}{{R_2}^2}K_2\theta\right)$$

$$\dot{\theta} = \omega$$

Con una ecuación de salida θ_2 la cual es:

$$y = \theta_2 = \frac{R}{R_2}\theta$$

Planteando la función de transferencia correspondiente:

$$\frac{r^2 r_2}{m r^2 r_2^2 s^2 + k r^2 r_2^2 + k_2 r^2 + J r_2^2 s^2 + b r_2^2 s}$$

Asignando valores:

```
J=0.01
r=0.1
r2=r/2
k=1
k2=0.5
b=0.1
g=9.81
m=1
```

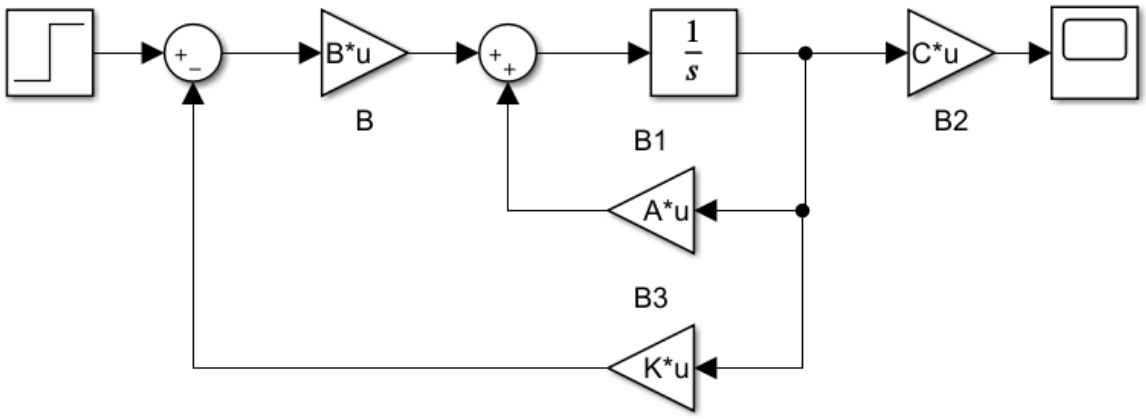
Tenemos la función de transferencia:

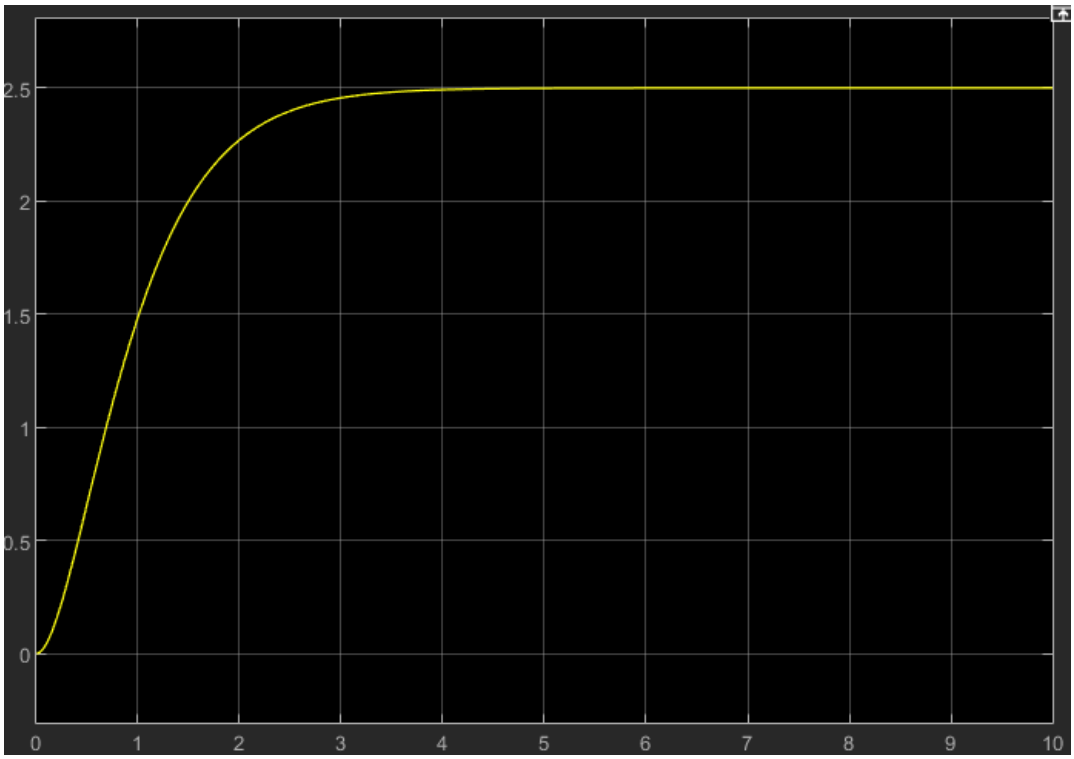
```
        10
-----------------
s^2 + 5 s + 100.5
```

Si se hace una retroalimentación únicamente usando Ackermann:

```
wn=4/(ts*zeta)
polos=roots([1 2*zeta*wn wn^2])
K=acker(A,B,polos)
```

Tenemos:

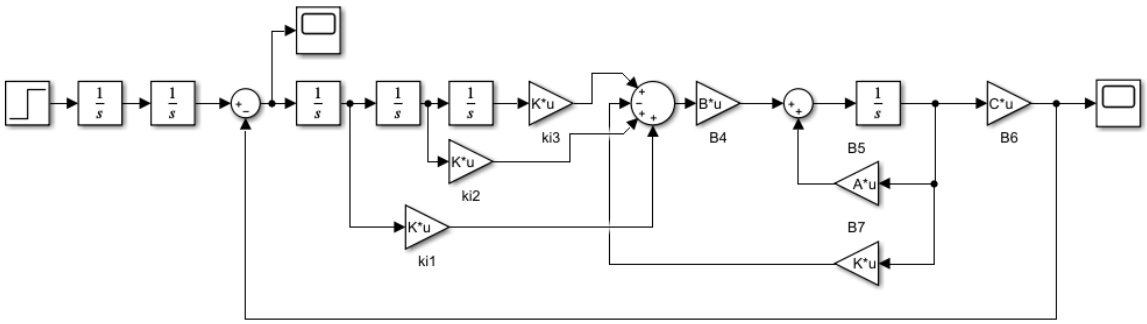




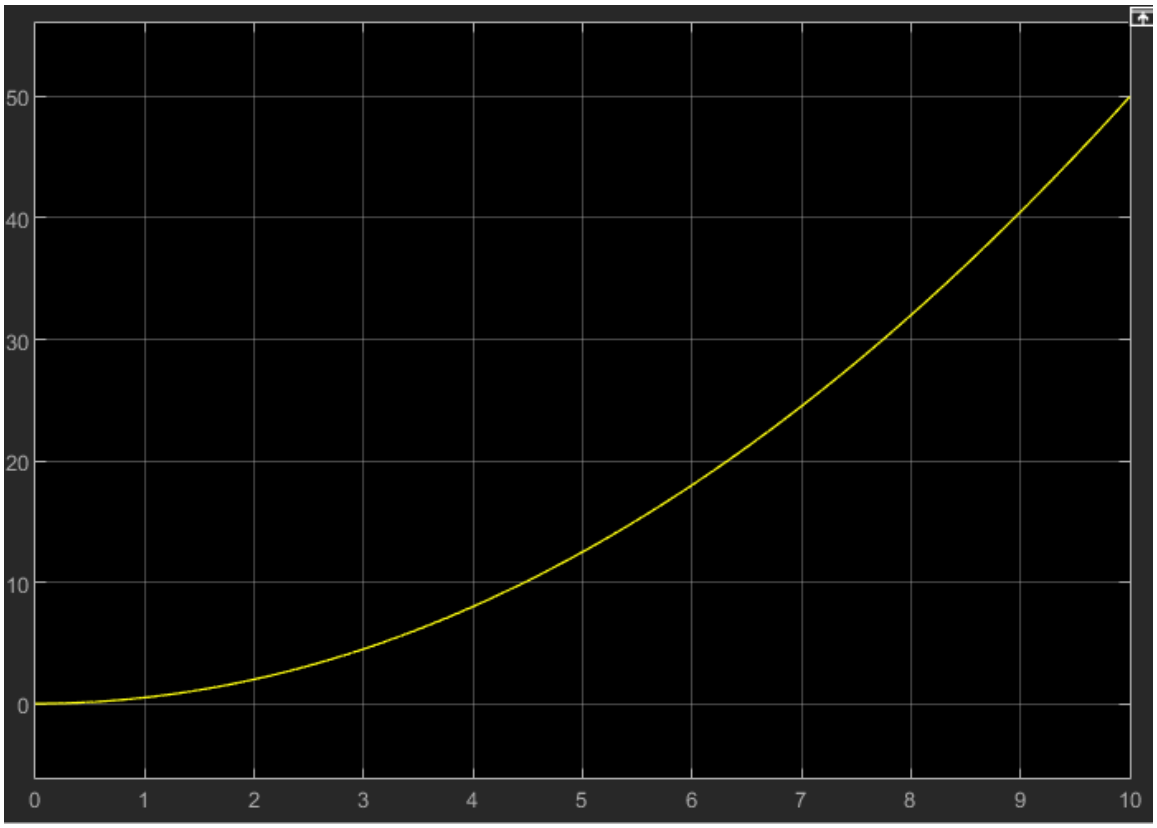
Donde se usó un ts=2 y un coeficiente de 1.
Haciendo el control para Ess=0 para parabola tenemos lo siguiente:

```
polosEss=conv(polos,[1 5*4])
polosEss=conv(polosEss,[1 5*4])
polosEss=conv(polosEss,[1 5*4])
Aempaq=[A(1,1),A(1,2),0,0,0;A(2,1),A(2,2),0,0,0
Bempaq=[B(1,1);B(2,1);0;0;0]
K1=acker(Aempaq,Bempaq,polosEss)
```

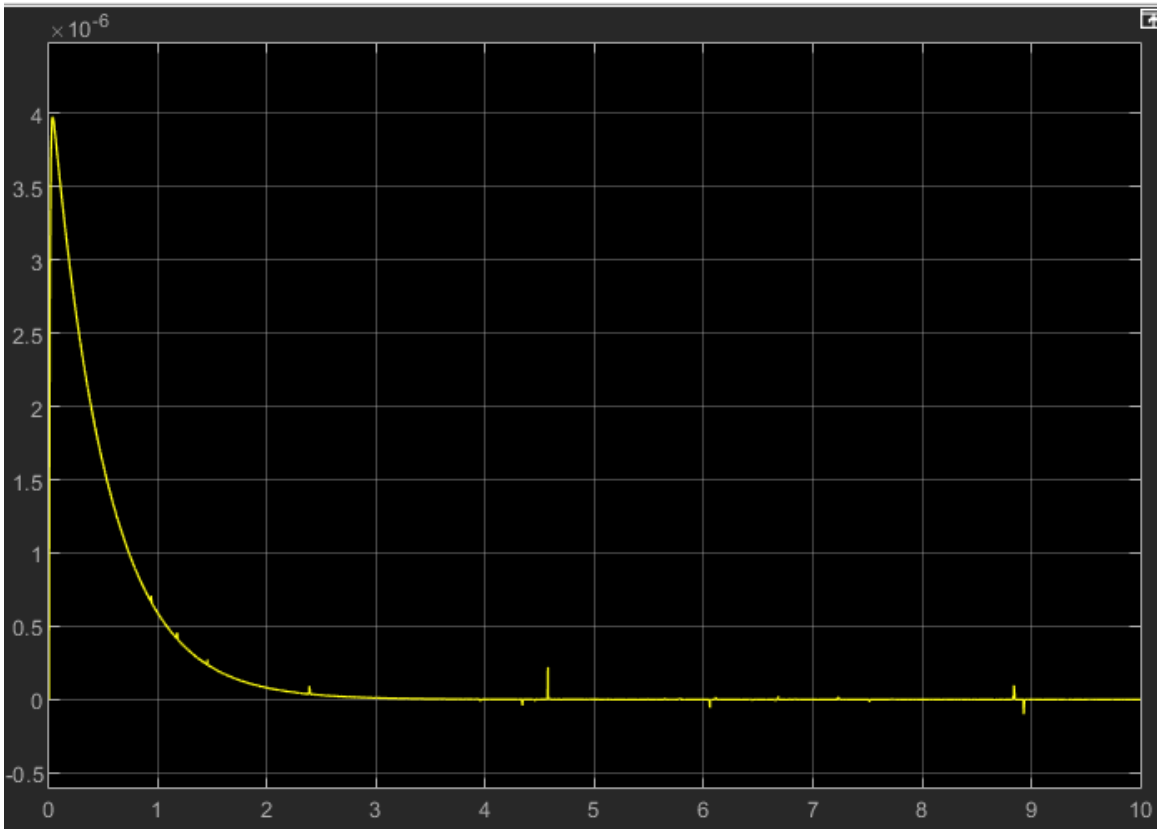
Por lo que:



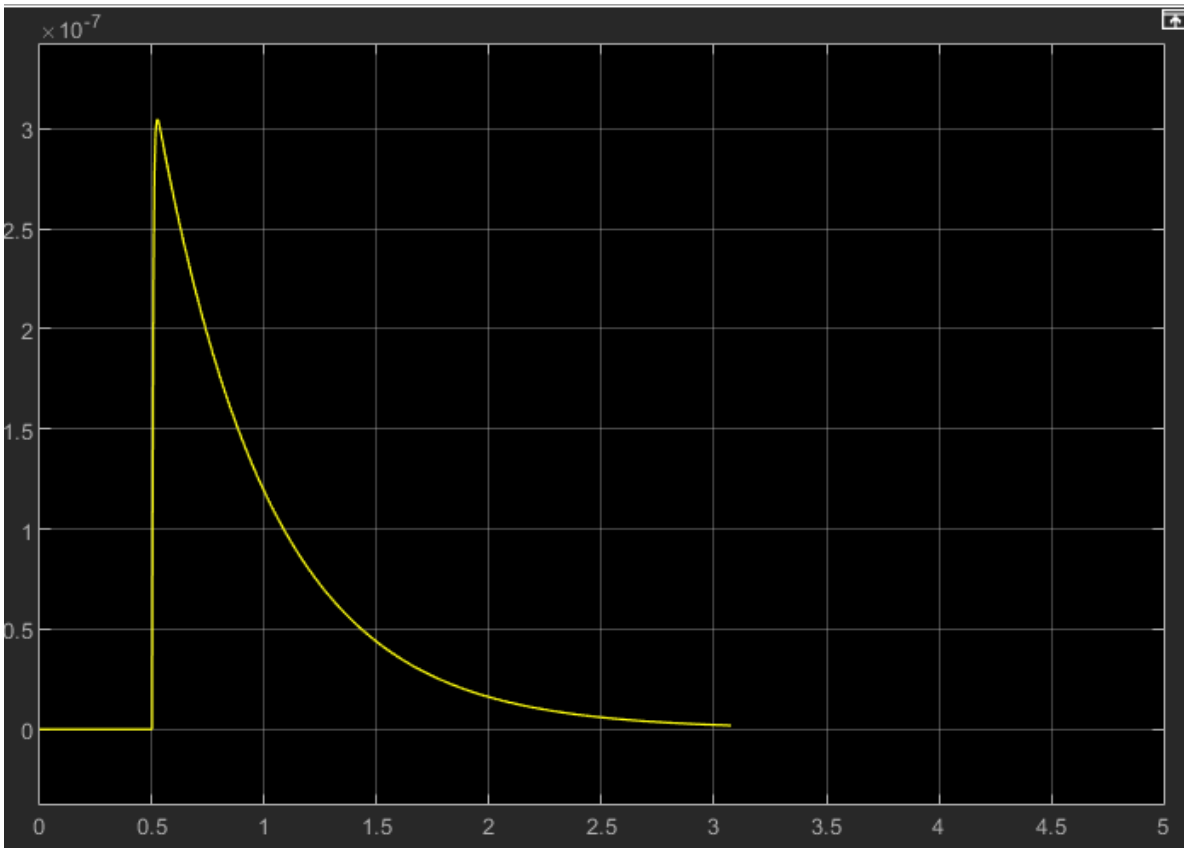
Con una salida:



Y un error:



Si colocamos un DELAY de 0.5s tenemos la respuesta y un error:



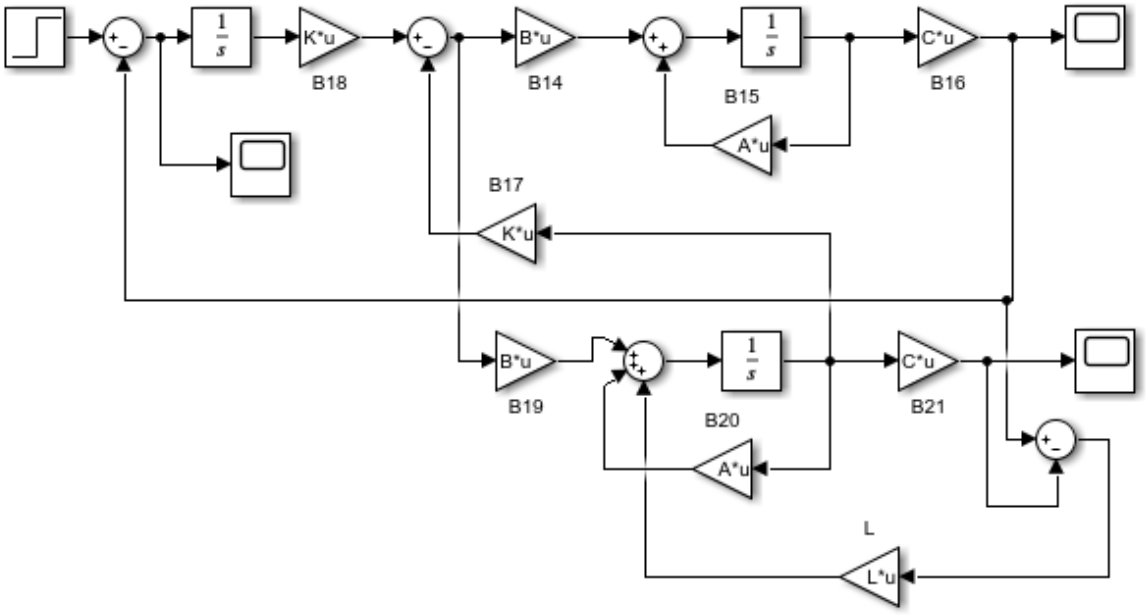
Vemos que el error se va a cero, la simulación no acaba por tiempo, ya que la capacidad de computo se limita por las matrices de 7x7 que están dentro. Esta respuesta de halla multiplicando la función de transferencia del delay con la funcion de la planta. Como vemos a continuación:

```
%%%%%%%% tm=0.5s
tm=0.5
polosEssT=conv(polosEss,[1 10*wn*zeta])
polosEssT=conv(polosEssT,[1 10*wn*zeta])
ftdelay=tf([1],[(tm^2)/2 tm 1])
ft2=ft*ftdelay
[A1,B1,C1,D1]=tf2ss(10,[0.125,1.125,16.06,55.25,100.5])
Aempaq1=[A1(1,1),A1(1,2),A1(1,3),A1(1,4),0,0,0;A1(2,1),A1
Bempaq1=[B1(1,1);B1(2,1);B1(3,1);B1(4,1);0;0;0]
KDelay=acker(Aempaq1,Bempaq1,polosEssT)
```

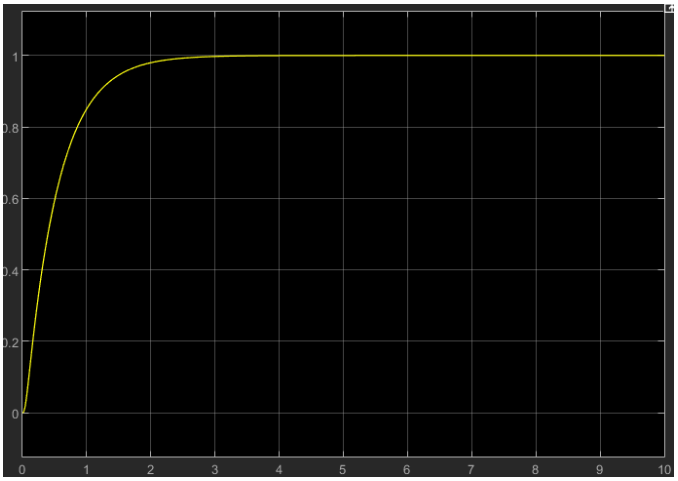
Si se tiene un sensor que mide la posición de la masa que cuelga del rodillo, se diseña un observador de estados, haciendo uso de los siguiente:

```
tsa=ts/10;
wna=4/(tsa*zeta)
obs=[C;C*A]
det(obs)
pdo=conv([1 2*tsa*zeta wna^2],[1 10*wna*zeta])
phia= A^3+pdo(2)*A^2+pdo(3)*A+pdo(4)*eye(2)
L=phia*(inv(obs))*[1/r;0]
```

La salida está en 1/r debido a que nuestros estados son θ y ω por lo que si dividimos el ángulo del rodillo en el radio del rodillo tenemos la posición de la masa.
Lo que nos da:

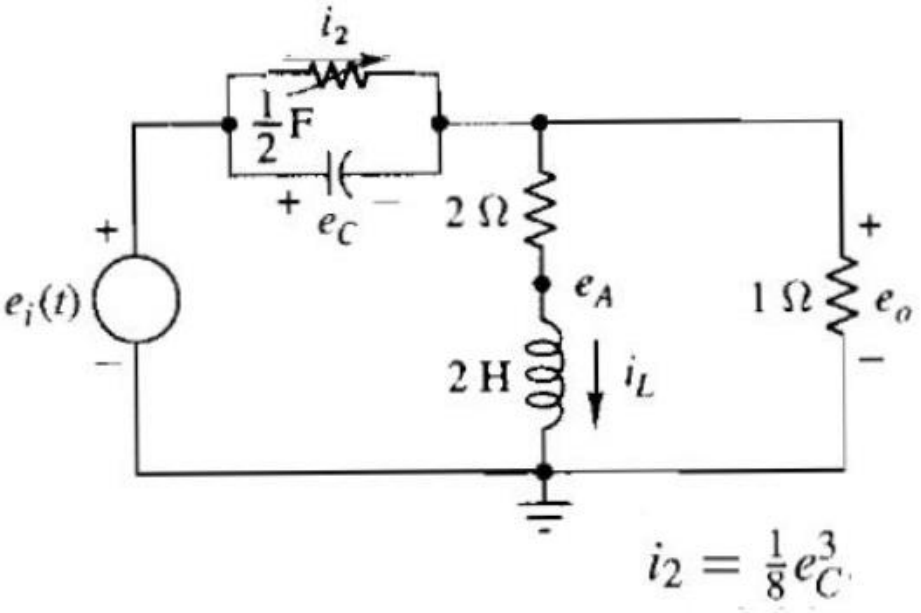


Mirando la salida del observador:



Este observador se diseñó para escalón pero vemos como perfectamente observa sin problemas, por lo que funcionaria para parábola.

4.



Analizando la malla principal y el nodo siguiente al arreglo paralelo de el capacitor y la resistencia variable tenemos:

$$\dot{i_l} = \frac{1}{L}(e_i - ec - R_2 i_l)$$

$$\dot{e_c} = \frac{1}{C}\left(\frac{(e_i - ec)}{R3} + i_l - \frac{1}{8}ec^3\right)$$

Con una salida de corriente del inductor, bajo estos parámetros tenemos lo siguiente:

```
c=1/2
l=2
r2=2
r3=1
e0=5
ec=3.9148
ei=8.9148
```

Dando los puntos de operación.

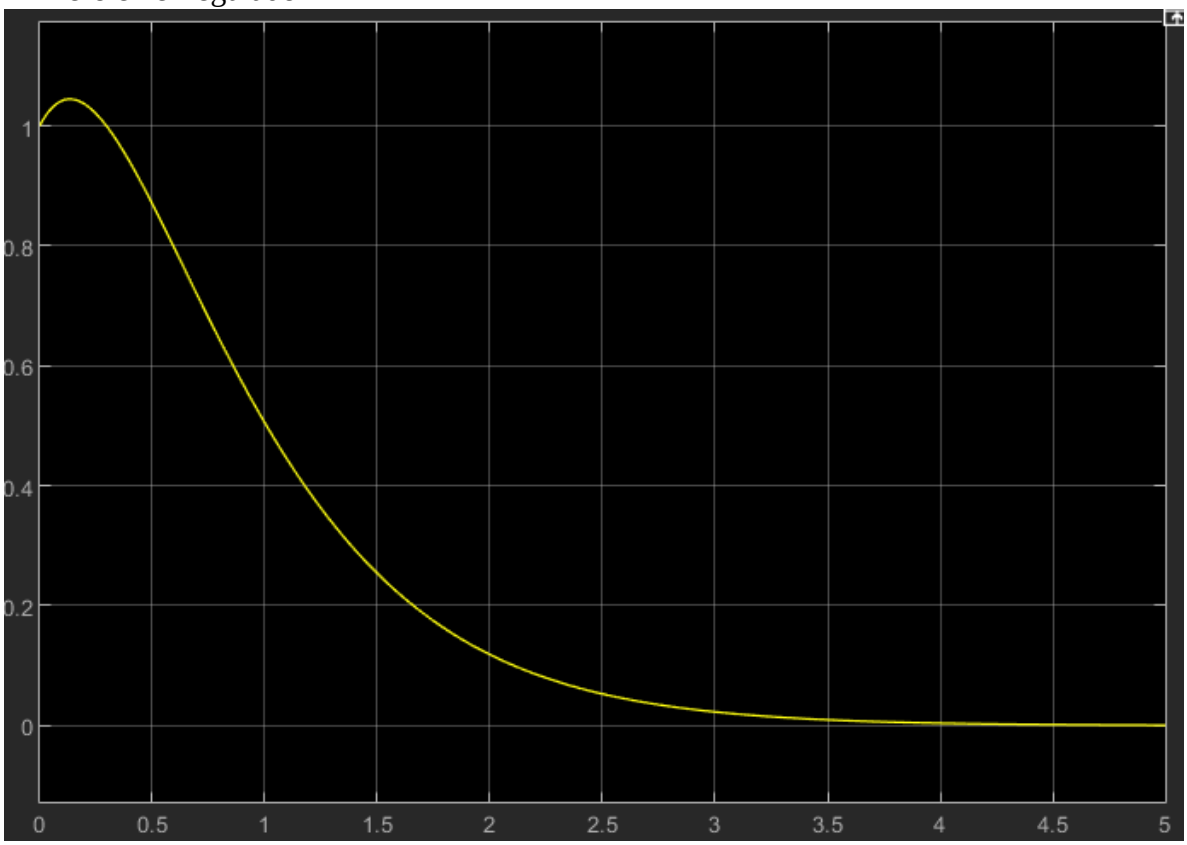
```
      0.5 s + 5.747
---------------------
s^2 + 14.49 s + 14.49
```

Luego haciendo uso de Ackermann tenemos:

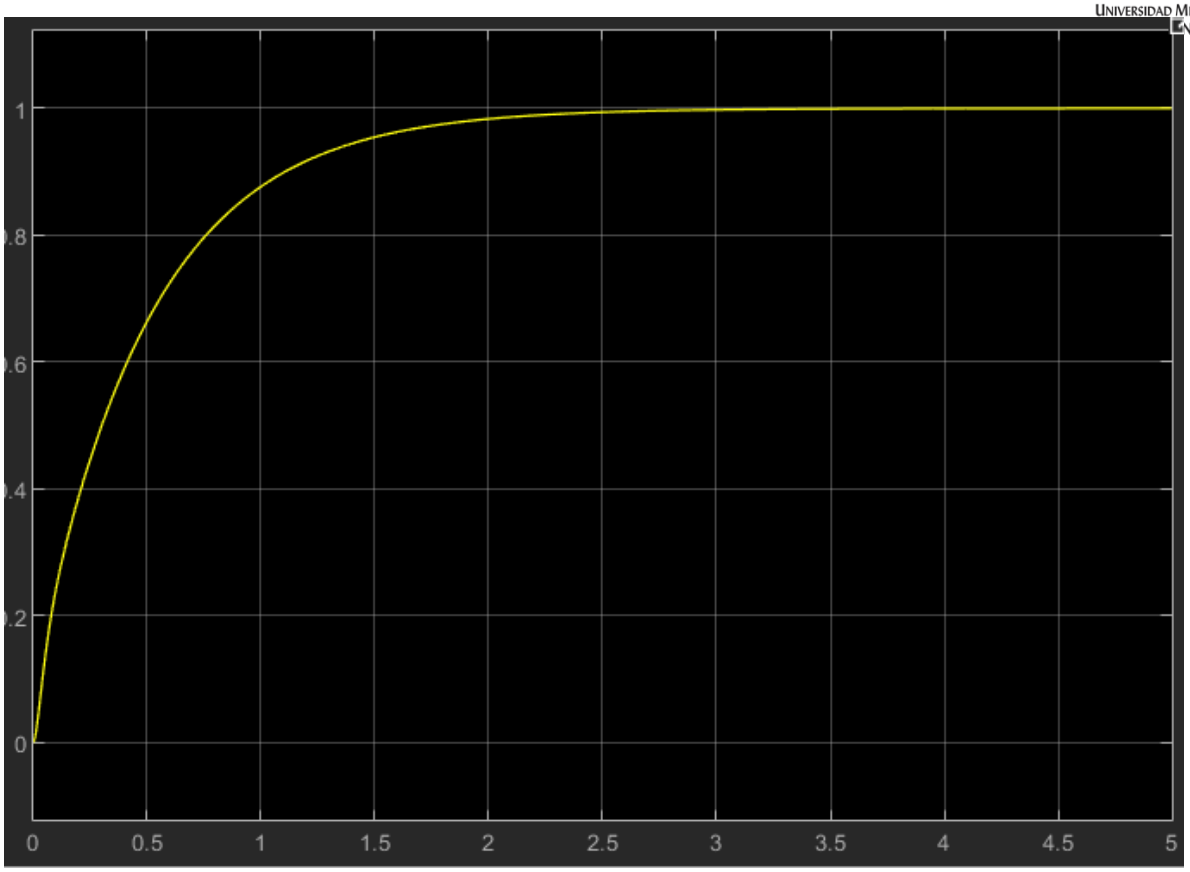
```
wn=4/(ts*zeta)
polos=roots([1 2*zeta*wn wn^2])
K=acker(A,B,polos)
%%%%%%%%%%%%%%%%%%%%%%Ess=0 Escalon
polosEss=conv(polos,[1 10*wn*zeta])
Aempaq=[A(1,1),A(1,2),0;A(2,1),A(2,2),0;-C(1,1),-C(
Bempaq=[B(1,1);B(2,1);0]
K1=acker(Aempaq,Bempaq,polosEss)
```

Lo que nos da:

Primero en el regulador



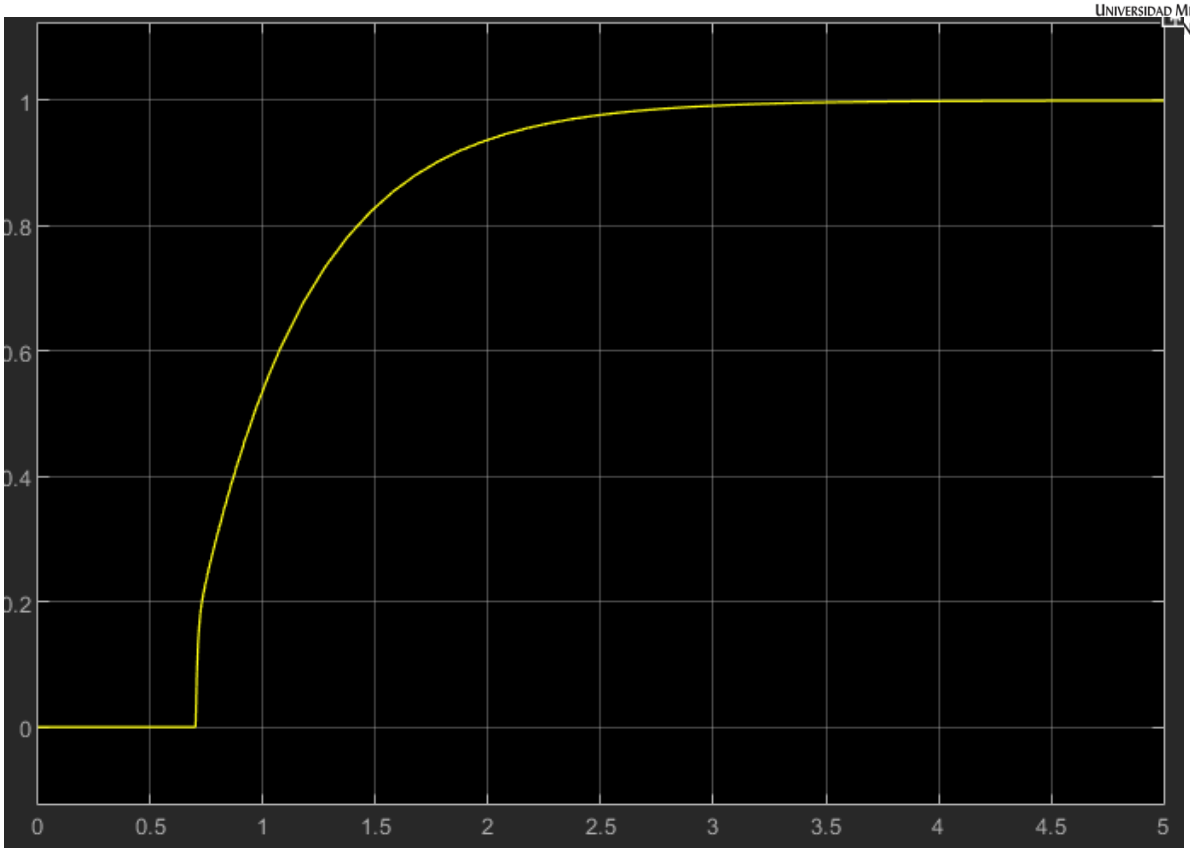
Por lo que vemos que no tiene un ess=0 si hacemos para un coeficiente de 1 y un ts=2, tenemos:



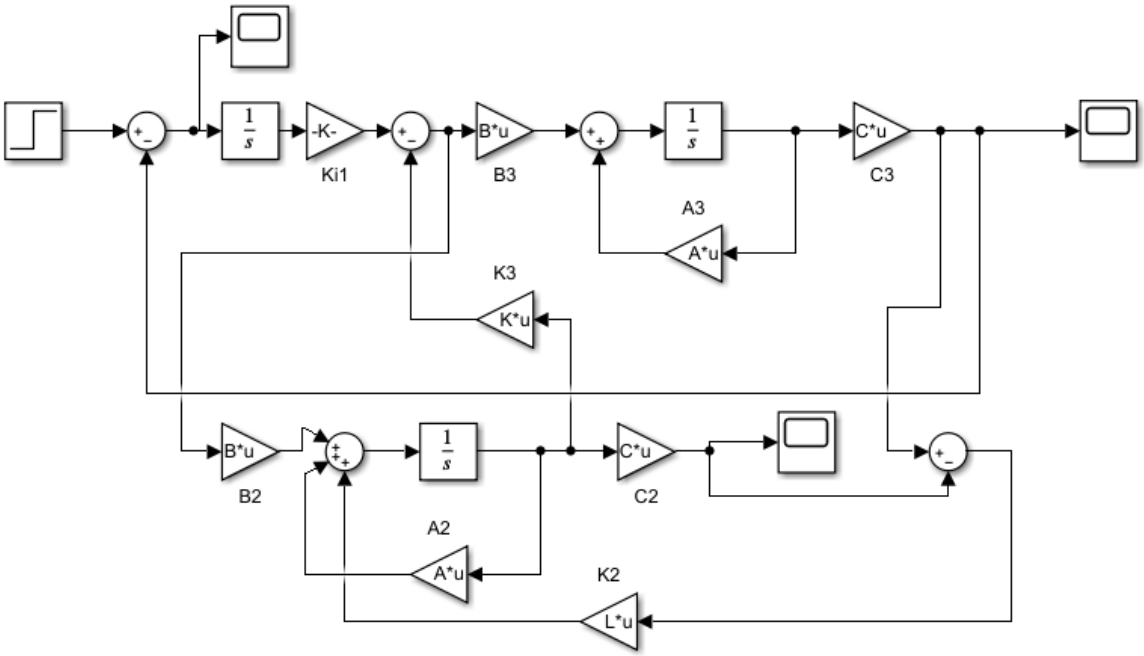
Luego si miramos para un Delay de 0.7s tenemos que hacer lo de antes, la multiplicación de las funciones de transferencia:

```
tm=0.7
polosEssT=conv(polosEss,[1 10*wn*zeta])
polosEssT=conv(polosEssT,[1 10*wn*zeta])
ftdelay=tf([1],[(tm^2)/2 tm 1])
ft2=ft*ftdelay
[A1,B1,C1,D1]=tf2ss(10,[0.125,1.125,16.06,55.25,100.5])
Aempaq1=[A1(1,1),A1(1,2),A1(1,3),A1(1,4),0;A1(2,1),A1(2,
Bempaq1=[B1(1,1);B1(2,1);B1(3,1);B1(4,1);0]
KDelay=acker(Aempaq1,Bempaq1,polosEssT)
```

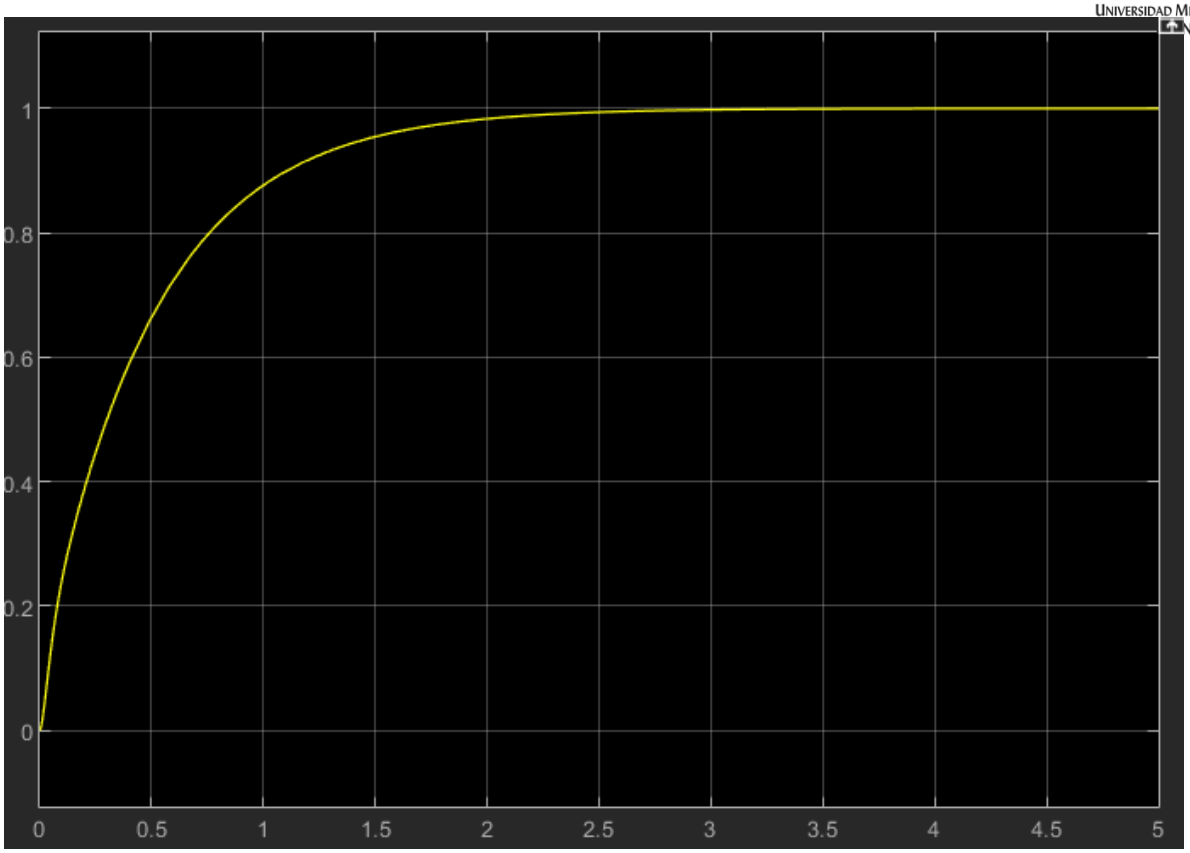
Por lo que vemos en la simulación:



Lo que vemos que cumple los criterios; el transport delay funciona.
Si miramos el observador de estados para la corriente:

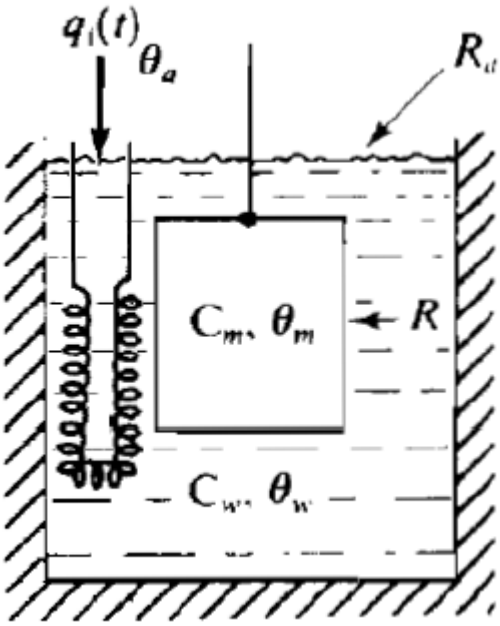


Donde a la salida tenemos:



Por lo que el diseño del observador esta correcto.

5.



Realizando las ecuaciones del modelo matemático del sistema térmico antes mostrado tenemos:

$$\dot{\theta_m} = \frac{1}{C_m}\left(-\frac{1}{R}(\theta_m - \theta_\omega)\right)$$

$$\dot{\theta_\omega} = \frac{1}{C_\omega}\left(q_i(t) + \frac{1}{R_a}(\theta_a - \theta_\omega) + \frac{1}{R}(\theta_m - \theta_\omega)\right)$$

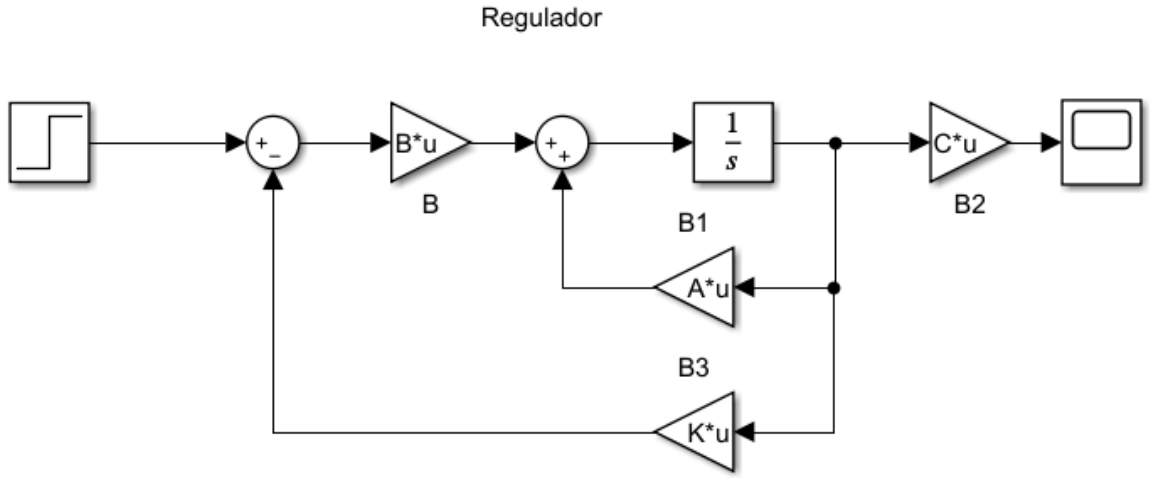
Con salida:

$$y = \theta_m$$

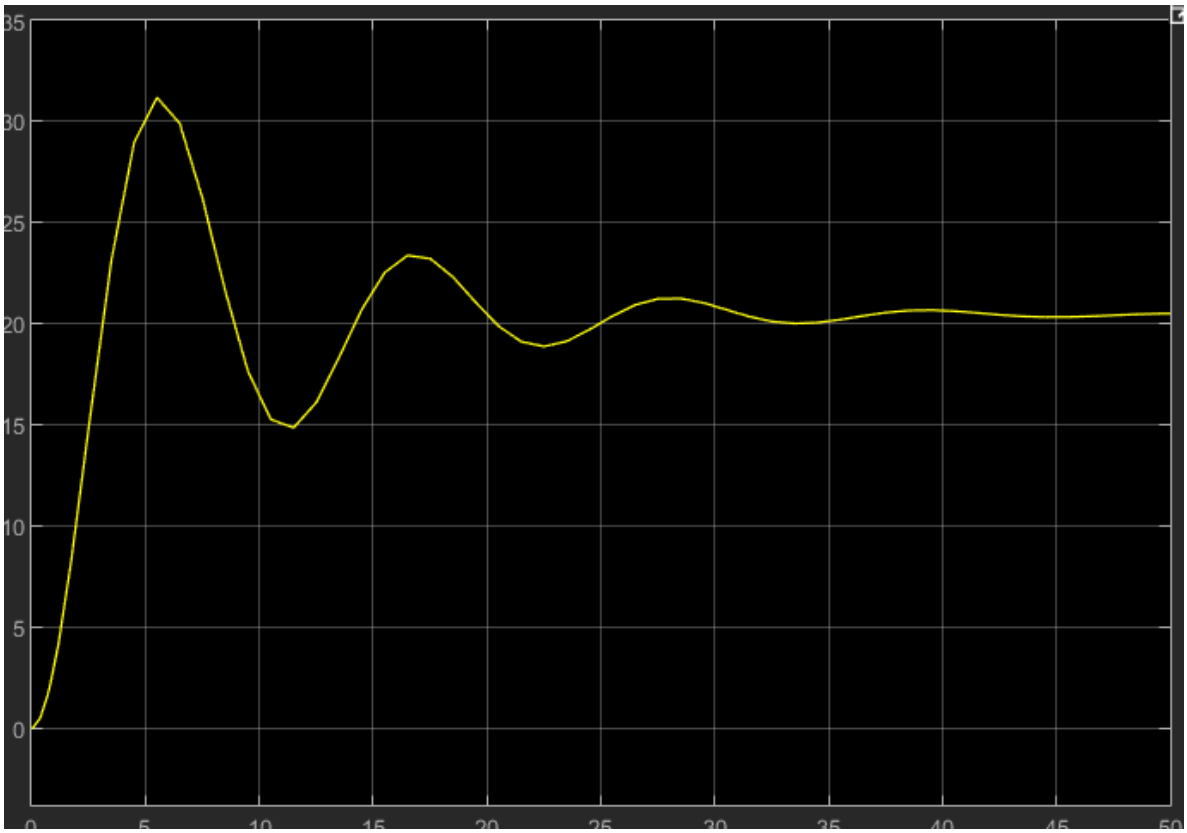
La función de transferencia del sistema:

```
          6.667
--------------------
s^2 + 3.6 s + 0.6667
```

Con esto tenemos el regulador:



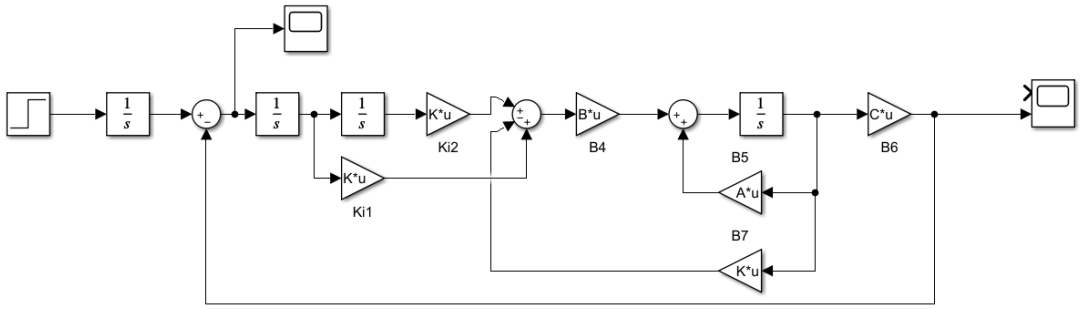
Con salida:



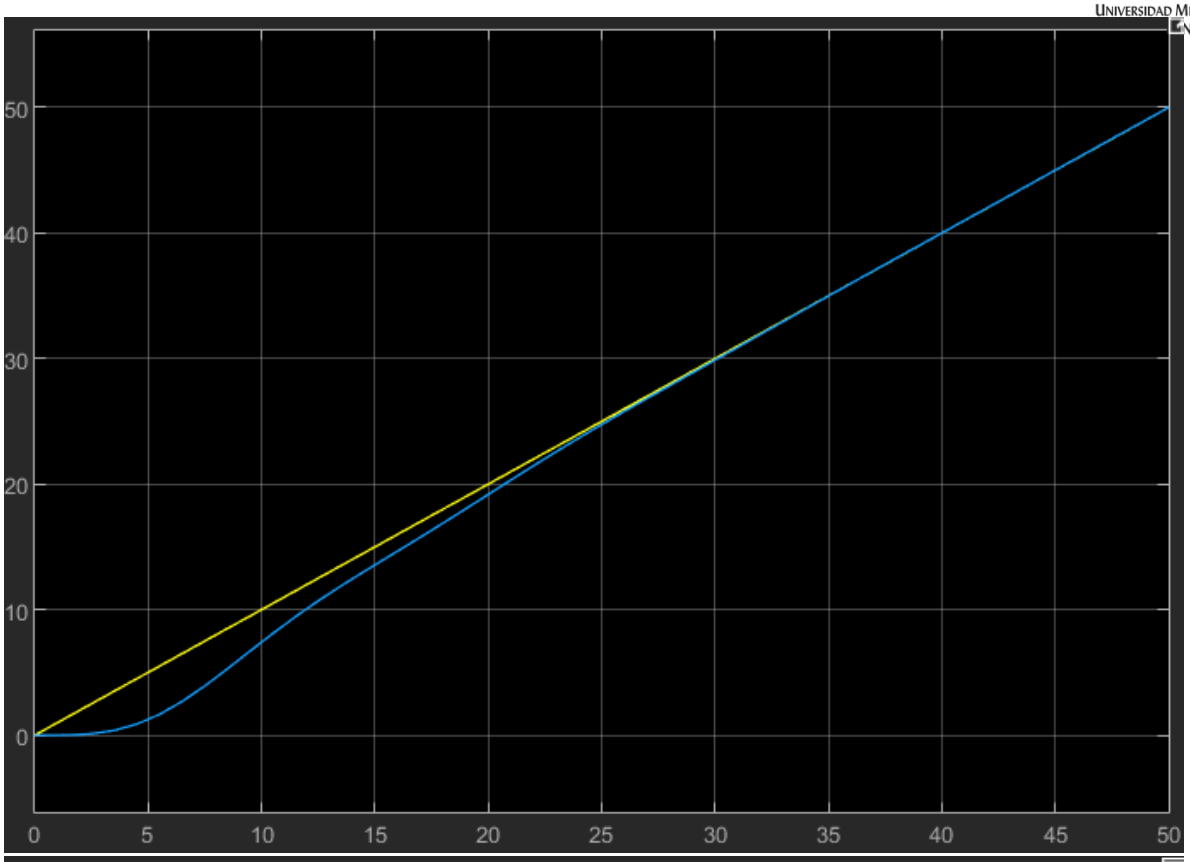
Esto debido a que se dio la operación:

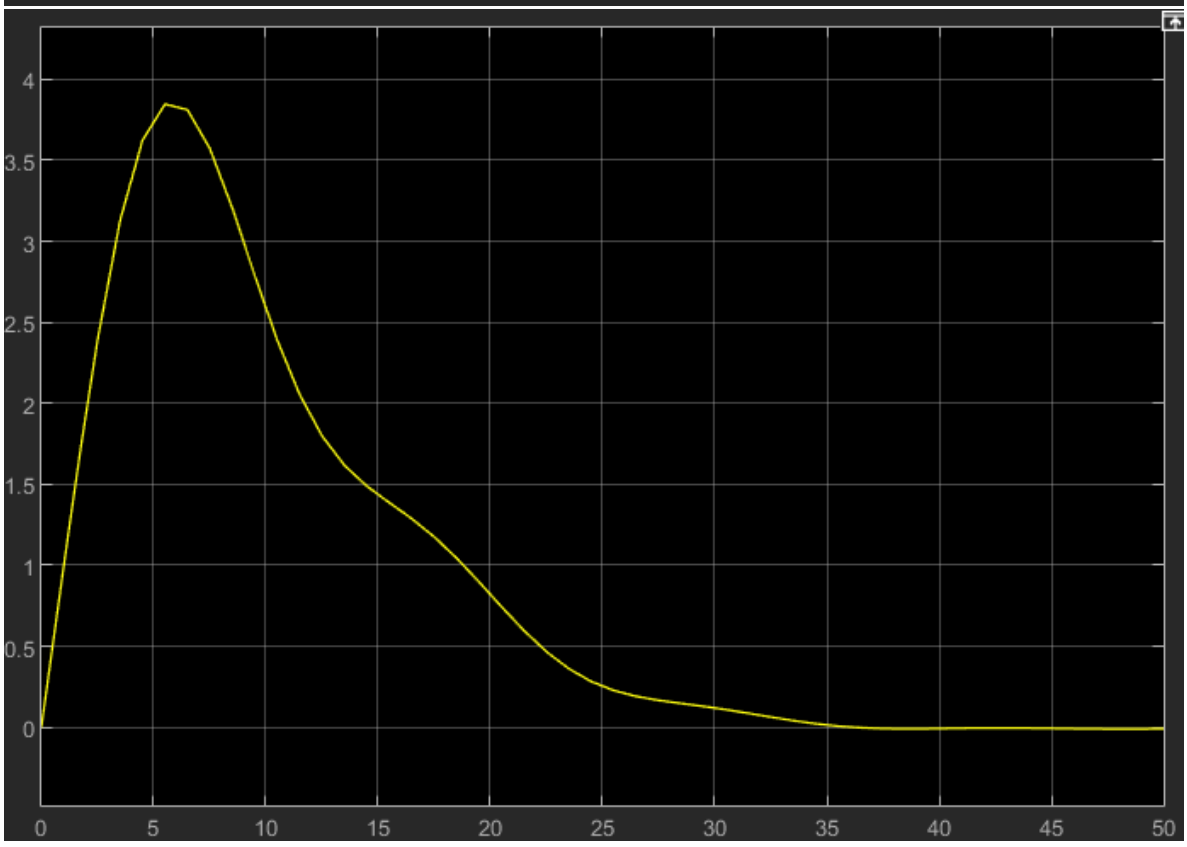
```
ts=35
zeta=0.2
wn=4/(ts*zeta)
polos=roots([1 2*zeta*wn wn^2])
K=acker(A,B,polos)
%PARA ESS=0 RAMPA
polosEss=conv(polos,[1 5*wn*zeta])
polosEss=conv(polosEss,[1 5*wn*zeta])
Aempaq=[A(1,1),A(1,2),0,0;A(2,1),A(2,2),0,0;0,0
Bempaq=[B(1,1);B(2,1);0;0]
K1=acker(Aempaq,Bempaq,polosEss)
```

Con estos valores se hallo en controlador para una entrada de rampa:



Con una salida y un error:

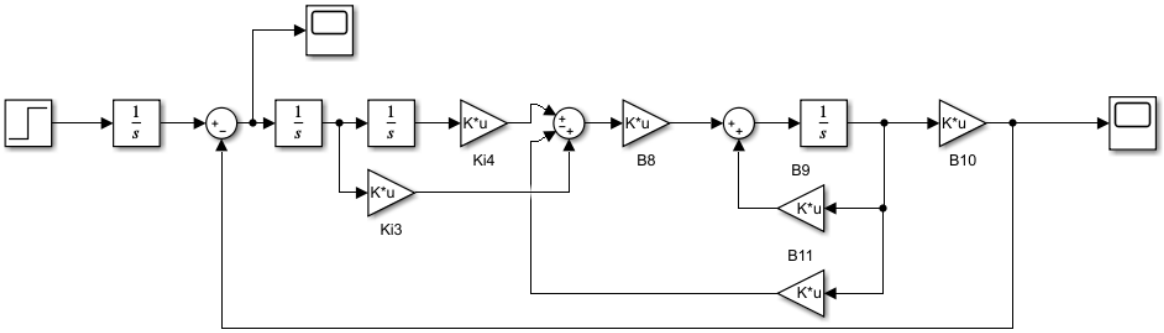

50
40
30
20
10
0
0
5
10
15
20
25
30
35
40
45
50

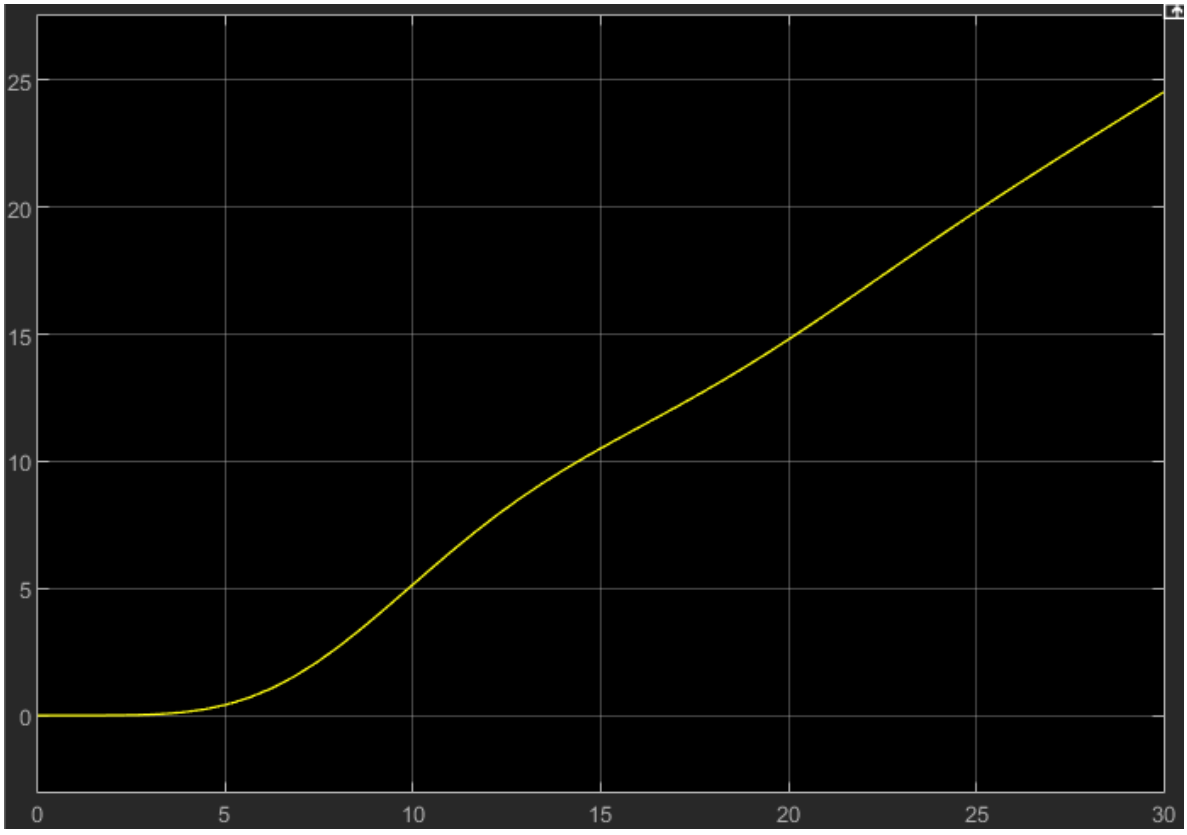

4
3.5
3
2.5
2
1.5
1
0.5
0
0
5
10
15
20
25
30
35
40
45
50

Haciendo un transport delay de 3s, tenemos que hacer el ismo procedimiento que siempre, hallar la F(s) del delay y multiplicarla por la planta, usando:

```
tm=3
polosEssT=conv(polosEss,[1 5*wn*zeta])
polosEssT=conv(polosEssT,[1 5*wn*zeta])
ftdelay=tf([1],[(tm^2)/2 tm 1])
ft2=ft*ftdelay
[A1,B1,C1,D1]=tf2ss(6.667,[4.5,19.2,14.8,5.6,0.6667])
Aempaq1=[A1(1,1),A1(1,2),A1(1,3),A1(1,4),0,0;A1(2,1),A1(2,
Bempaq1=[B1(1,1);B1(2,1);B1(3,1);B1(4,1);0;0]
KDelay=acker(Aempaq1,Bempaq1,polosEssT)
```

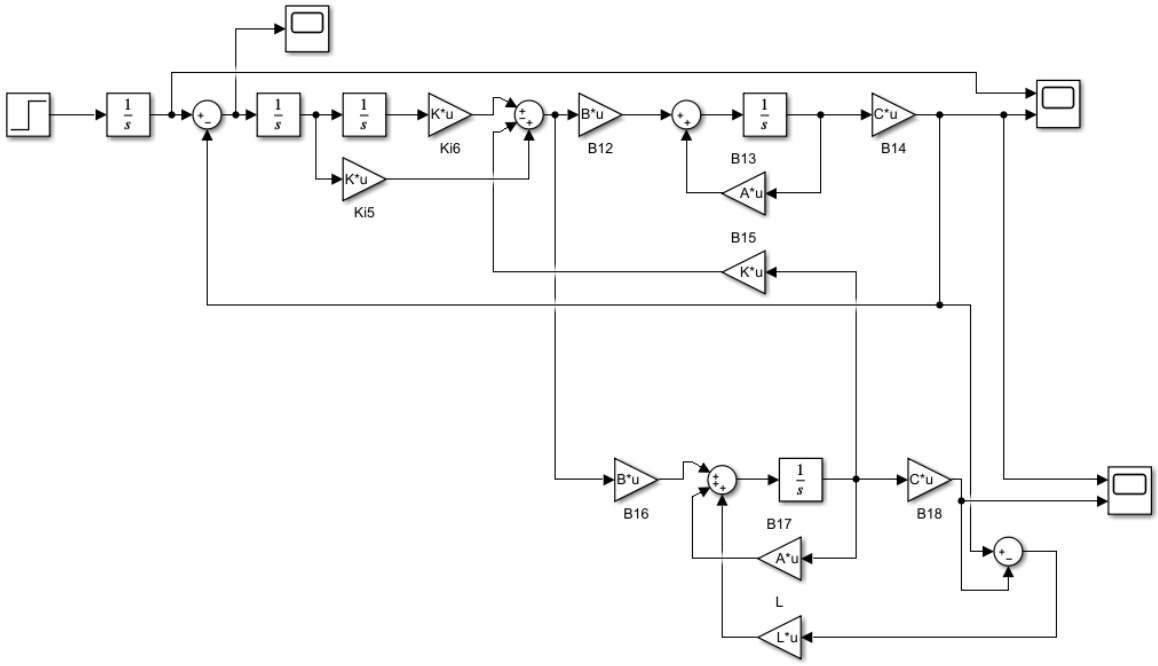
Tenemos matrices de 6x6 lo que nos da a la salida del controlador:



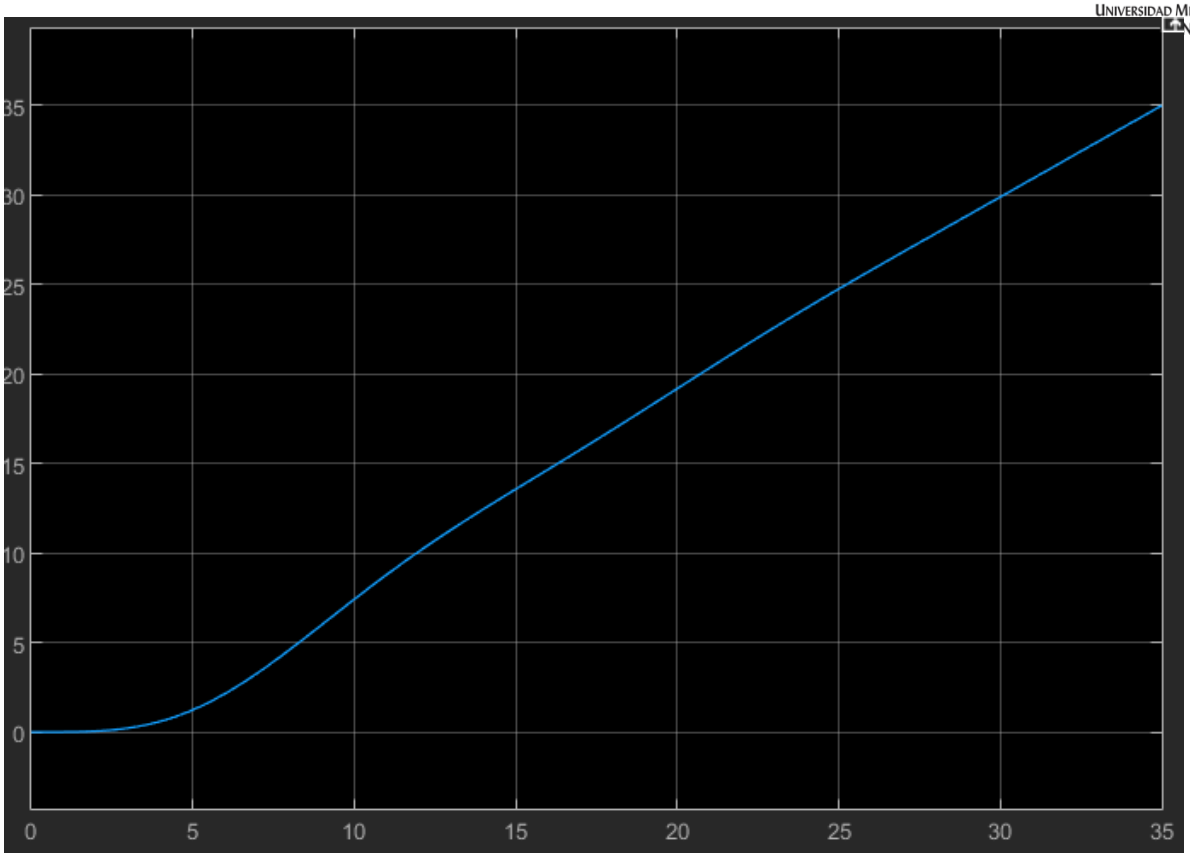


Donde vemos que el Delay de 3s está presente antes de seguir la rampa.
Si se hace un observador de estados para θ_w tenemos:

```
%%Observador
tsa=ts/10;
wna=4/(tsa*zeta)
obs=[C;C*A]
det(obs)
pdo=conv([1 2*tsa*zeta wna^2],[1 10*wna*zeta])
phia= A^3+pdo(2)*A^2+pdo(3)*A+pdo(4)*eye(2)
L=phia*(inv(obs))*[1;0]
```



Con esto si vemos y comparamos las salidas:



No cambia la salida por lo que se dice que el observador de estados esta bien diseñado.