

Apoyo en el desarrollo del Proyecto de Grado MIACON

MÓDULO 2: Identificación y control de una planta de primer orden

- **Hecho por:** Diego Andrés García Díaz.
- **Código:** 2195533.
- **Fecha:** 26/09/2025.
- **Asignatura:** Control II (Adelanto de Nota).

Módulo para un Sistema Térmico (Primer Orden)

$$G_P(s) = \frac{K_p}{T_{p1} \cdot s + 1} \cdot e^{-T_d \cdot s};$$

$$G_P(s) = \text{Actuador} + \text{Planta} + \text{Sensor};$$

$$K = K_p = \text{Ganancia};$$

$$K = \frac{\Delta Y \text{ (salida)}}{\Delta U \text{ (entrada)}};$$

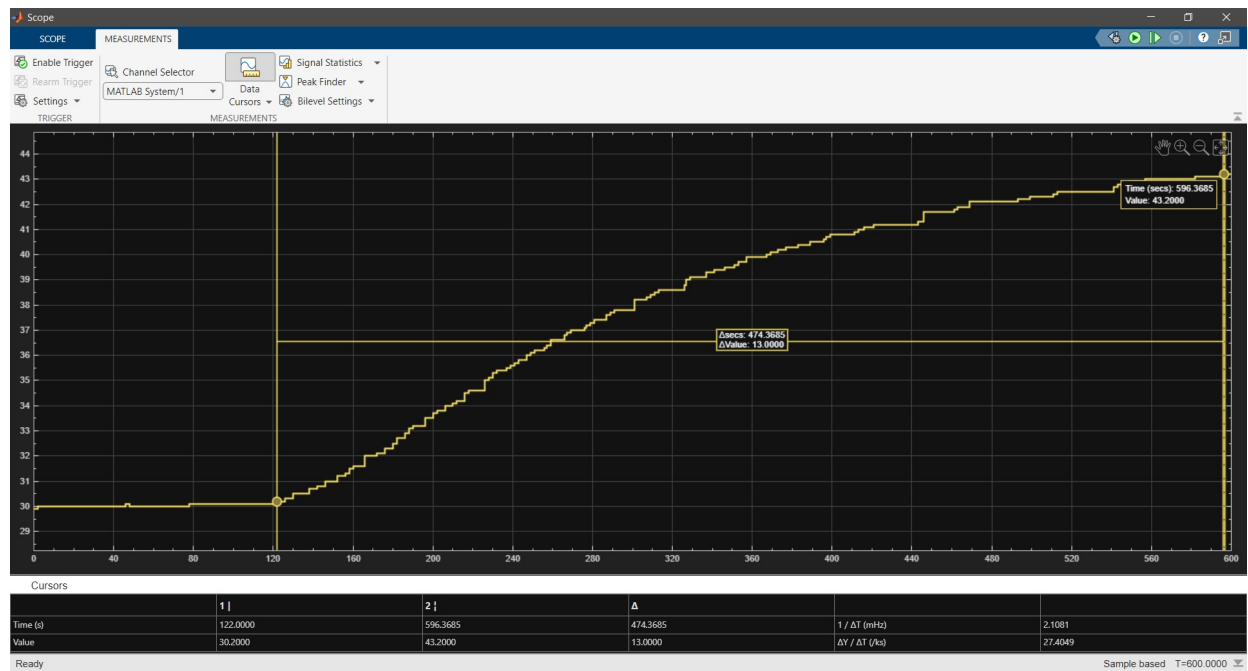
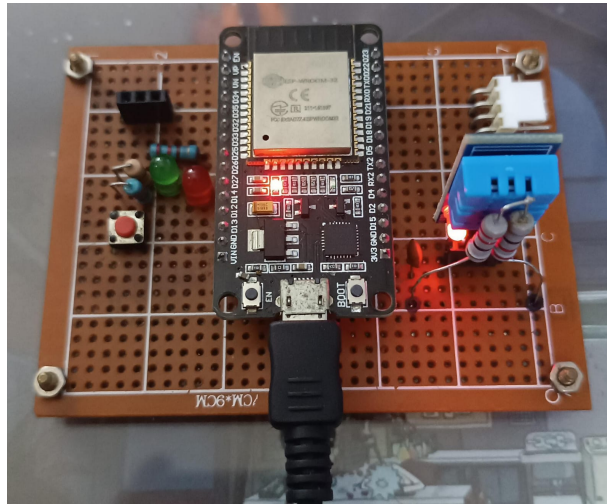
$$T = T_{p1} = \text{Constante de tiempo};$$

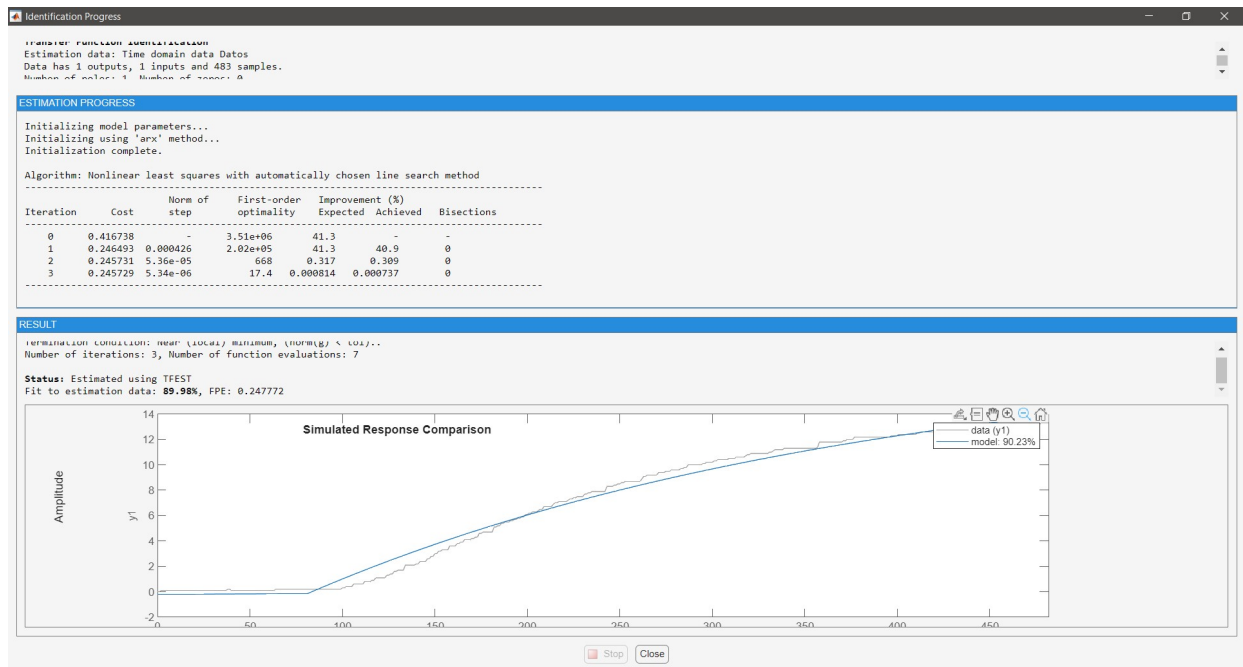
$$T = \frac{3}{2}(t_2 - t_1);$$

$$L = T_d = \text{Tiempo muerto (Delay)} ;$$

$$L = t_2 - T$$

Modelo identificado siguiendo las instrucciones de la página web:

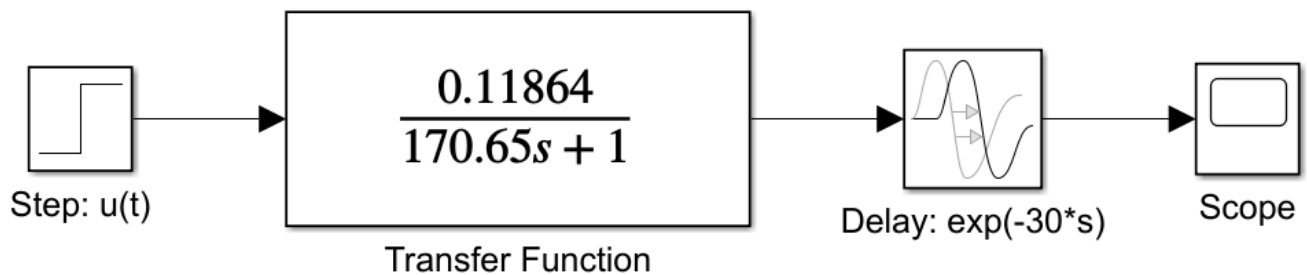


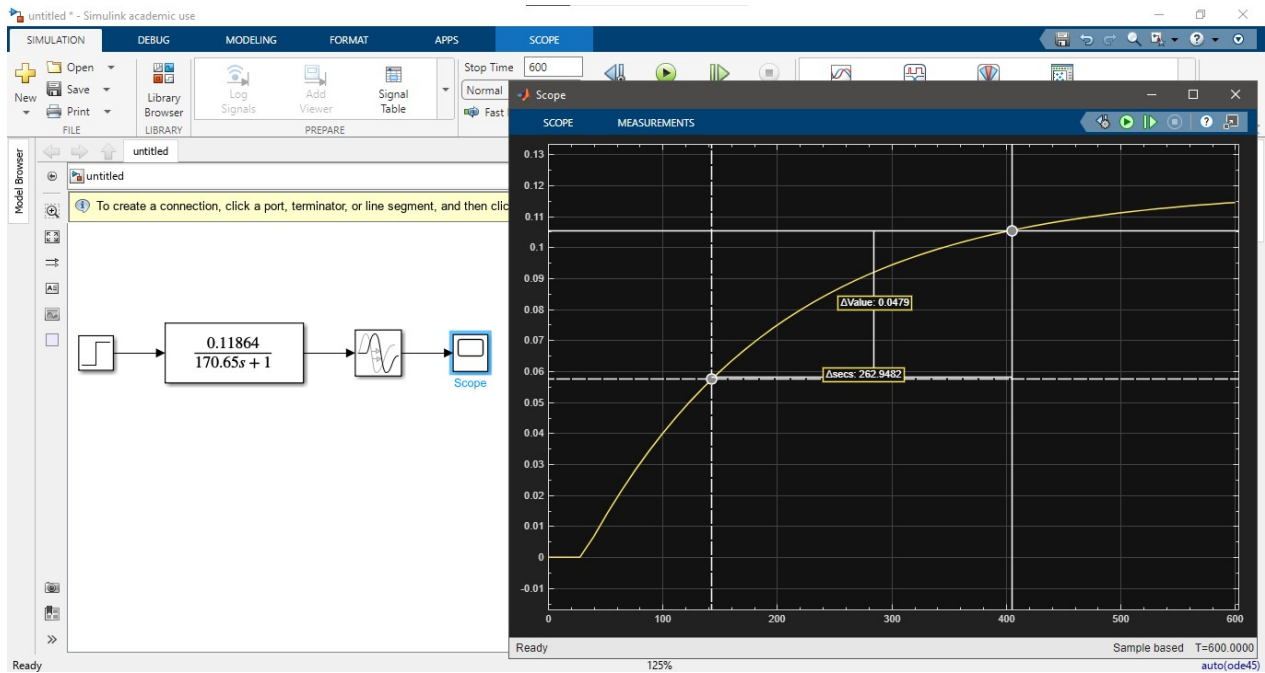


$$P1D \rightarrow G_{P_1}(s) = P(s) = \frac{0.11864}{170.65 \cdot s + 1} \cdot e^{-30 \cdot s}$$

Ajuste de 90.23% > 80%

Validación en Simulink:





$$G_P(s) = \frac{0.1144}{158.7423 \cdot s + 1} \cdot e^{-31.7485 \cdot s}$$

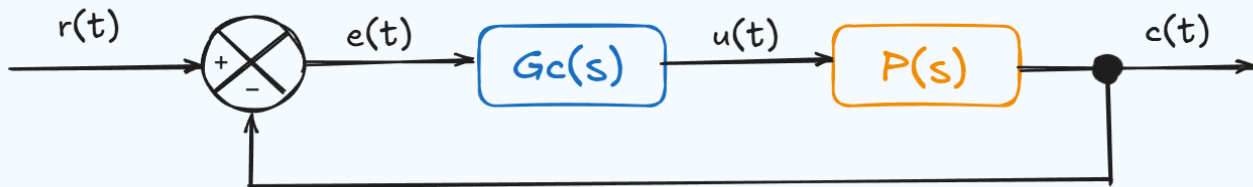
$$t_1 = L + \frac{1}{3} \cdot T = 84.6626 [s]; t_2 = L + T = 190.4908 [s];$$

$$\text{Ganancia} \rightarrow K = \frac{\Delta Y (\text{salida})}{\Delta U (\text{entrada})} = 0.1144;$$

$$\text{Constante de tiempo} \rightarrow T = \frac{3}{2}(t_2 - t_1) = 158.7423 [s];$$

$$\text{Retardo} \rightarrow L = t_2 - T = 31.7485 [s]$$

A continuación se presenta el desarrollo para el respectivo diseño de los controladores P, PI, PD y PID:



El siguiente procedimiento corresponde al cálculo del **controlador proporcional (P)**:

$$G_{LC}(s) = \frac{C(s)}{R(s)} = \frac{G_C(s) \cdot P(s)}{1 + G_C(s) \cdot P(s)} \approx \frac{\omega_n^2}{s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2}$$

$$G_C(s) = \frac{U(s)}{E(s)} = K \frac{(s + z_1) \cdot (s + z_2) \dots (s + z_m)}{(s + p_1) \cdot (s + p_2) \dots (s + p_n)} \rightarrow \text{Función de Transferencia del Controlador Generalizada}$$

$$P(s) = \frac{N(s)}{D(s)} \rightarrow \text{Función de Transferencia del Sistema o Planta}$$

$$G_{LC}(s) = \frac{C(s)}{R(s)} = \frac{K \frac{(s + z_1) \cdot (s + z_2) \dots (s + z_m)}{(s + p_1) \cdot (s + p_2) \dots (s + p_n)} \cdot \frac{N(s)}{D(s)}}{1 + K \frac{(s + z_1) \cdot (s + z_2) \dots (s + z_m)}{(s + p_1) \cdot (s + p_2) \dots (s + p_n)} \cdot \frac{N(s)}{D(s)}}$$

$$G_{LC}(s) = \frac{K \cdot (s + z_1) \cdot (s + z_2) \dots (s + z_m) \cdot N(s)}{(s + p_1) \cdot (s + p_2) \dots (s + p_n) \cdot D(s) + K \cdot (s + z_1) \cdot (s + z_2) \dots (s + z_m) \cdot N(s)}$$

$$\frac{K \cdot (s + z_1) \cdot (s + z_2) \dots (s + z_m) \cdot N(s)}{(s + p_1) \cdot (s + p_2) \dots (s + p_n) \cdot D(s) + K \cdot (s + z_1) \cdot (s + z_2) \dots (s + z_m) \cdot N(s)} \approx \frac{\omega_n^2}{s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2}$$

$$(s + p_1) \cdot (s + p_2) \dots (s + p_n) \cdot D(s) + K \cdot (s + z_1) \cdot (s + z_2) \dots (s + z_m) \cdot N(s) \approx s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2$$

$$(170.65 \cdot s + 1) + K_P \cdot (0.11864 \cdot e^{-30 \cdot s}) = s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2$$

El término $e^{-30 \cdot s}$ no se puede igualar directamente con los coeficientes, ya que la parte izquierda de la igualación, en su forma actual, tiene un "orden infinito" debido a la exponencial. Una solución a este inconveniente es **aproximar el retardo** con una "**aproximación racional**", es decir, una fracción de dos polinomios. La más común y utilizada es la **APROXIMACIÓN DE PADÉ** de primer orden.

Cabe resaltar que si se usa un orden mayor en la aproximación de Padé, también aumenta el orden del polinomio resultante para hacer la respectiva igualación.

$$e^{-\theta \cdot s} \approx \frac{1 - \frac{\theta}{2} \cdot s}{1 + \frac{\theta}{2} \cdot s} \rightarrow e^{-30 \cdot s} = \frac{1 - \frac{30}{2} \cdot s}{1 + \frac{30}{2} \cdot s} = \frac{1 - 15 \cdot s}{1 + 15 \cdot s}$$

Entonces se obtiene lo siguiente:

$$(170.65 \cdot s + 1) + K_P \cdot \left(0.11864 \cdot \left(\frac{1 - 15 \cdot s}{1 + 15 \cdot s} \right) \right) = s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2$$

Operando la parte izquierda se obtiene lo siguiente:

$$s^2 + (0.07253 - 0.00069 \cdot K_P) \cdot s + (0.0003907 + 0.0000463 \cdot K_P) = s^2 + 2 \cdot \zeta \cdot \omega_n \cdot s + \omega_n^2$$

Se hacen las respectivas igualaciones:

$$2 \cdot \zeta \cdot \omega_n = 0.07253 - 0.00069 \cdot K_P$$

$$\zeta \cdot \omega_n = 0.03627 - 0.00035 \cdot K_P$$

Se definen y calculan parámetros para el diseño del controlador y poder despejar la primera igualación:

Parámetro que se asume para el diseño del controlador \rightarrow OV (Overshoot) $\leq 20 \%$

$\zeta \rightarrow$ Factor de Amortiguamiento

$$\zeta \geq \sqrt{\frac{\ln(OV)^2}{\pi^2 + \ln(OV)^2}} \rightarrow \zeta = 0.456$$

$\zeta < 1 \rightarrow$ Subamortiguado

$$t_{s(2\%)} = \frac{4}{\zeta \cdot \omega_n} \rightarrow \text{Tiempo de Establecimiento}$$

Ahora, se puede despejar ω_n y obtener una **Ecuación #1**:

$$\zeta \cdot \omega_n = 0.03627 - 0.00035 \cdot K_P$$

$$(0.456) \cdot \omega_n = 0.03627 - 0.00035 \cdot K_P$$

$$\omega_n = 0.03627 - 0.00035 \cdot K_P \rightarrow \text{Ecuación \#1}$$

Finalmente, se hace la última igualación y obtener una **Ecuación #2**:

$$\omega_n^2 = 0.0003907 - 0.0000463 \cdot K_P \rightarrow \text{Ecuación \#2}$$

El siguiente paso es **reemplazar** la **Ecuación #1** en la **Ecuación #2**, agrupar términos semejantes y **despejar** la respectiva constante (K_P) del controlador, haciendo las respectivas operaciones se obtiene lo siguiente:

$$(0.07954 - 0.00077 \cdot K_P)^2 = 0.0003907 + 0.0000463 \cdot K_P$$

$$K_{P_1} = 243.5872$$

$$K_{P_2} = 41.1009$$

Se usa la **Ecuación #2** para obtener ω_n :

$$\omega_{n_1} = 0.10802 \frac{\text{rad}}{\text{s}}$$

$$\omega_{n_2} = 0.04789 \frac{\text{rad}}{\text{s}}$$

Por último, se puede calcular el **Tiempo de Establecimiento**:

$$t_{s(2\%)_1} = 81.20653 \text{ [s]}$$

$$t_{s(2\%)_2} = 183.1683 \text{ [s]}$$

El siguiente paso a realizar corresponde a la validación en SIMULINK, la sintonización del controlador usando la teoría de Ziegler-Nichols y el uso de una herramienta para sintonizar el controlador de forma más "automática" y realizar las respectivas comparaciones. Finalmente, se recalca que este sería el mismo procedimiento a realizar para el diseño de otro controlador de forma analítica.

El siguiente código se usa para el diseño y comparación de los diferentes controladores diseñados:

```
% Hecho por: Diego Andrés García Díaz.
% Código: 2195533.
% Fecha: 24/09/2025.
% Asignatura: Control II (Adelanto de Nota).
%
% -----
% ----- Diseño Controladores para Control de Temperatura -----
% -----
%
% Esta sección borra la ventana de comandos, el espacio de trabajo y cierra
% cualquier figura abierta. También define s como una variable de función de
% transferencia para su uso posterior.
clc; clear; close all;
s = tf('s');

% Definición de parámetros del sistema (Planta) identificada
Kp = 0.11864; % Ganancia del sistema
Tp1 = 170.65; % Tiempo de establecimiento
L = 30; % Tiempo muerto o Delay

% Función de Transferencia del modelo identificado (Actuador + Planta + Sensor)
P_s = ( (Kp * exp(-L*s) ) / ((Tp1*s) + 1) )
```

P_s =

$$\exp(-30s) * \frac{0.1186}{170.7 s + 1}$$

Continuous-time transfer function.
Model Properties

% Función de Transferencia factorizada

P_s_fact = zpk(P_s)

P_s_fact =

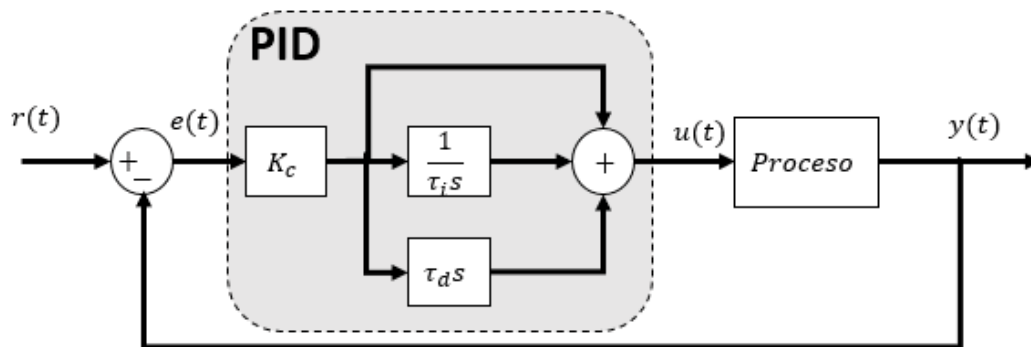
$$\exp(-30*s) * \frac{0.00069522}{(s+0.00586)}$$

Continuous-time zero/pole/gain model.
Model Properties

% Abrir 'sisotool' para el diseño y análisis de los controladores

% sisotool(P_s)

Usando el método de Ziegler-Nichols:



$$G_{C_{PID}}(s) = K_p \left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s \right)$$

$$K_i = \frac{K_p}{T_i} \quad ; \quad K_d = K_p \cdot T_d$$

Controlador	K_p	T_i	T_d
P	$\frac{T}{kL}$	∞	0
PI	$\frac{0.9T}{kL}$	$\frac{L}{0.3}$	0
PID	$\frac{1.2T}{kL}$	$2L$	$0.5L$

Tabla 1. Ecuaciones necesarias para el diseño de controladores usando el método de Ziegler-Nichols.

Controlador Proporcional (P)

$$G_{C_p}(s) = K_p$$

```
clc; clear; close all;
s = tf('s');

Kp = 0.11864; % Ganancia del sistema
Tp1 = 170.65; % Tiempo de establecimiento
L = 30; % Tiempo muerto o Delay

% P_s = ( (Kp * exp(-L*s) ) / ((Tp1*s) + 1) )

% ----- Aproximación de Padé del retardo -----
[num, den] = pade(L, 1); % 1er orden (> 2 si deseas más precisión)
Delay = tf(num, den)
```

Delay =

```
-s + 0.06667
-----
s + 0.06667
```

Continuous-time transfer function.
Model Properties

```
% ----- Planta aproximada -----
P_s = Kp * Delay / (Tp1*s + 1)
```

P_s =

```
-0.1186 s + 0.007909
-----
170.7 s^2 + 12.38 s + 0.06667
```

Continuous-time transfer function.
Model Properties

```
% Parámetros del Criterio de Ziegler-Nichols para controlador P
kp_ZN = (Tp1)/(Kp*L); % Cálculo de la ganancia proporcional del controlador P
Ti_ZN = inf;
Td_ZN = 0;
```

```
% Función de transferencia controlador
Gc_P1 = 10 % Ganancia proporcional del controlador P, aleatoria
```

Gc_P1 =
10

```
Gc_P2 = kp_ZN % Ganancia proporcional del controlador P, método Z-N
```

Gc_P2 =
47.9462

```
Gc_P3 = 41.1009 % Ganancia proporcional del controlador P, método analítico
```

```
Gc_P3 =  
41.1009
```

```
Gc_P4 = 68 % Ganancia proporcional del controlador P, aleatoria
```

```
Gc_P4 =  
68
```

```
Gc_Pmax = 104.3 % Ganancia proporcional del controlador P, método "prueba y error"
```

```
Gc_Pmax =  
104.3000
```

```
% Se cierra el lazo
```

```
T_cl1 = feedback((Gc_P1*P_s), 1);  
T_cl2 = feedback((Gc_P2*P_s), 1);  
T_cl3 = feedback((Gc_P3*P_s), 1);  
T_cl4 = feedback((Gc_P4*P_s), 1);  
T_clmax = feedback((Gc_Pmax*P_s), 1);
```

```
% Se grafica la respuesta
```

```
t = 0:1:600;  
isstable(T_cl1)      % Se verifica la estabilidad del sistema
```

```
ans = logical  
1
```

```
isstable(T_cl2)      % Se verifica la estabilidad del sistema
```

```
ans = logical  
1
```

```
isstable(T_cl3)      % Se verifica la estabilidad del sistema
```

```
ans = logical  
1
```

```
isstable(T_cl4)      % Se verifica la estabilidad del sistema
```

```
ans = logical  
1
```

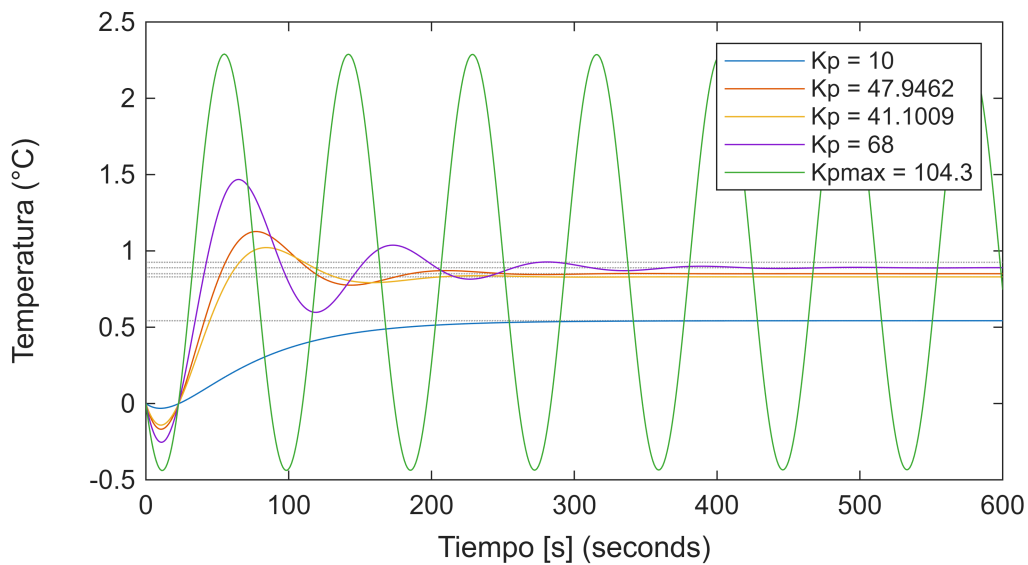
```
isstable(T_clmax)    % Se verifica la estabilidad del sistema
```

```
ans = logical  
1
```

```
figure(1);  
step(T_cl1, T_cl2, T_cl3, T_cl4, T_clmax, t);  
legend('Kp = 10', 'Kp = 47.9462', 'Kp = 41.1009', 'Kp = 68', 'Kpmax = 104.3');  
% grid on;  
title('Respuesta Controlador P');  
xlabel('Tiempo [s]');
```

```
ylabel('Temperatura (°C)');
```

Respuesta Controlador P



Controlador Proporcional-Integral (PI)

$$G_{C_{PI}}(s) = K_p \cdot \left(\frac{s + z_1}{s} \right) \rightarrow G_{C_{PI}}(s) = K_p \cdot \left(\frac{1}{T_i \cdot s} \right)$$

```
clc; clear; close all;
s = tf('s');

Kp = 0.11864; % Ganancia del sistema
Tp1 = 170.65; % Tiempo de establecimiento
L = 30;      % Tiempo muerto o Delay

% ----- Aproximación de Padé del retardo -----
[num, den] = pade(L, 1); % 1er orden (> 2 si desees más precisión)
Delay = tf(num, den)
```

Delay =

```
-s + 0.06667
-----
s + 0.06667
```

Continuous-time transfer function.
Model Properties

```
% ----- Planta aproximada -----
P_s = Kp * Delay / (Tp1*s + 1)
```

```
P_s =

    -0.1186 s + 0.007909
    -----
    170.7 s^2 + 12.38 s + 0.06667

Continuous-time transfer function.
Model Properties
```

```
% Parámetros del Criterio de Ziegler-Nichols para controlador PI
kp_ZN = (0.9*Tp1)/(Kp*L) % Cálculo de la ganancia proporcional del controlador PI
```

```
kp_ZN =
43.1516
```

```
Ti_ZN = L/0.3;
Td_ZN = 0;
```

```
ki_ZN = kp_ZN / Ti_ZN % Cálculo de la ganancia integral del controlador PI
```

```
ki_ZN =
0.4315
```

```
% Función de transferencia controlador
% Gc_PI = kp_ZN * (1 + (1/(Ti_ZN * s))) % Forma alternativa
Gc_PI2 = kp_ZN + (ki_ZN / s)
```

```
Gc_PI2 =

    43.15 s + 0.4315
    -----
           s
```

```
Continuous-time transfer function.
Model Properties
```

```
Gc_PI3 = 41.1009 + (0.88 / s) % Ganancia proporcional del controlador PI
```

```
Gc_PI3 =

    41.1 s + 0.88
    -----
           s
```

```
Continuous-time transfer function.
Model Properties
```

```
% Gc_PI4 = kp_ZN + (ki_ZN / s) % Ganancia proporcional del controlador PI
% Gc_PImax = kp_ZN + (ki_ZN / s) % Ganancia proporcional del controlador PI
```

```
% Se cierra el lazo
T_cl1 = feedback((Gc_PI2*P_s), 1);
T_cl2 = feedback((Gc_PI3*P_s), 1);
% T_cl3 = feedback((Gc_PI4*P_s), 1);
% T_clmax = feedback((Gc_PImax*P_s), 1);
```

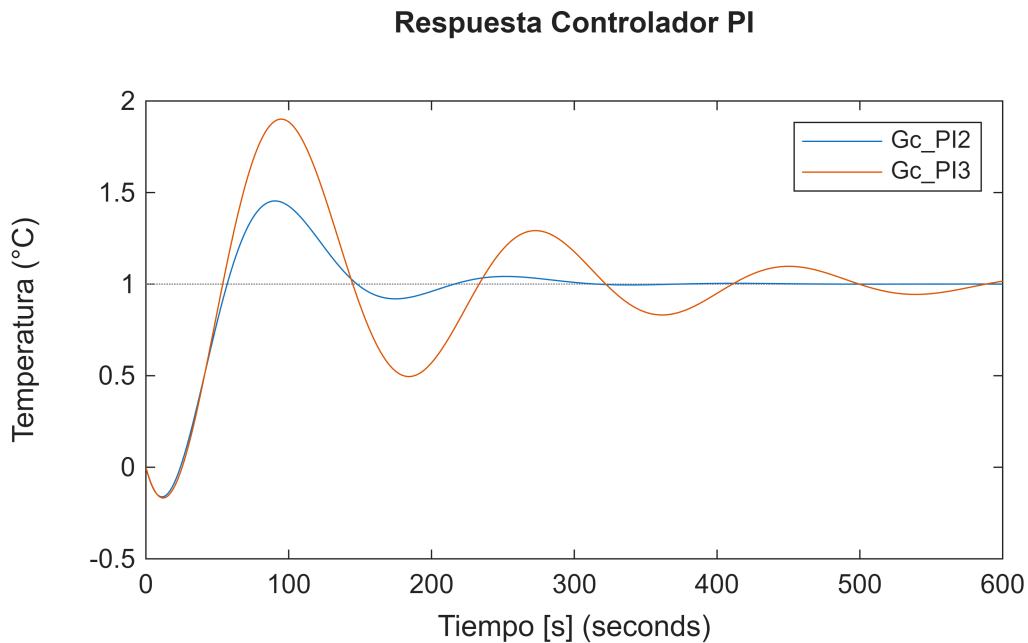
```
% Se grafica la respuesta
t = 0:1:600;
isstable(T_cl1) % Se verifica la estabilidad del sistema
```

```
ans = logical
      1
```

```
isstable(T_cl2) % Se verifica la estabilidad del sistema
```

```
ans = logical
      1
```

```
figure(2);
step(T_cl1, T_cl2, t)
legend('Gc_PI2', 'Gc_PI3');
title('Respuesta Controlador PI');
xlabel('Tiempo [s]');
ylabel('Temperatura (°C)');
```



Controlador Proporcional-Derivativo (PD)

$$G_{C_{PD}}(s) = K_p \cdot (s + z_1) \quad \rightarrow \quad G_{C_{PD}}(s) = K_p \cdot (T_d \cdot s)$$

```
clc; clear; close all;
s = tf('s');

Kp = 0.11864; % Ganancia del sistema
Tp1 = 170.65; % Tiempo de establecimiento
L = 30; % Tiempo muerto o Delay

% ----- Aproximación de Padé del retardo -----
```

```
[num, den] = pade(L, 1); % 1er orden (> 2 si deseas más precisión)
Delay = tf(num, den)
```

```
Delay =
```

```
-s + 0.06667
-----
s + 0.06667
```

```
Continuous-time transfer function.
Model Properties
```

```
% ----- Planta aproximada -----
```

```
P_s = Kp * Delay / (Tp1*s + 1)
```

```
P_s =
```

```
-0.1186 s + 0.007909
-----
170.7 s^2 + 12.38 s + 0.06667
```

```
Continuous-time transfer function.
Model Properties
```

```
% Parámetros del Criterio de Ziegler-Nichols para controlador PD
```

```
kp_ZN = 57.5354
```

```
kp_ZN =
57.5354
```

```
Ti_ZN = 0
```

```
Ti_ZN =
0
```

```
Td_ZN = 863.0310
```

```
Td_ZN =
863.0310
```

```
kd_ZN = kp_ZN * Td_ZN % Cálculo de la ganancia derivativa del controlador PD
```

```
kd_ZN =
4.9655e+04
```

```
% Función de transferencia controlador
```

```
% Gc_PD = kp_ZN * (1 + (Td_ZN * s)) % Forma alternativa
```

```
Gc_PD2 = kp_ZN + (kd_ZN * s)
```

```
Gc_PD2 =
```

```
4.965e04 s + 57.54
```

```
Continuous-time transfer function.
Model Properties
```

```
Gc_PD3 = 57.5354 + (1400 * s)
```

```
Gc_PD3 =
```

```
1400 s + 57.54
```

```
Continuous-time transfer function.  
Model Properties
```

```
Gc_PD4 = 41.1009 + (92.3444 * s)
```

```
Gc_PD4 =
```

```
92.34 s + 41.1
```

```
Continuous-time transfer function.  
Model Properties
```

```
% Gc_PDmax =
```

```
% Se cierra el lazo
```

```
T_cl1 = feedback((Gc_PD2*P_s), 1);
```

```
T_cl2 = feedback((Gc_PD3*P_s), 1);
```

```
T_cl3 = feedback((Gc_PD4*P_s), 1);
```

```
% T_clmax = feedback((Gc_PDmax*P_s), 1);
```

```
% Se grafica la respuesta
```

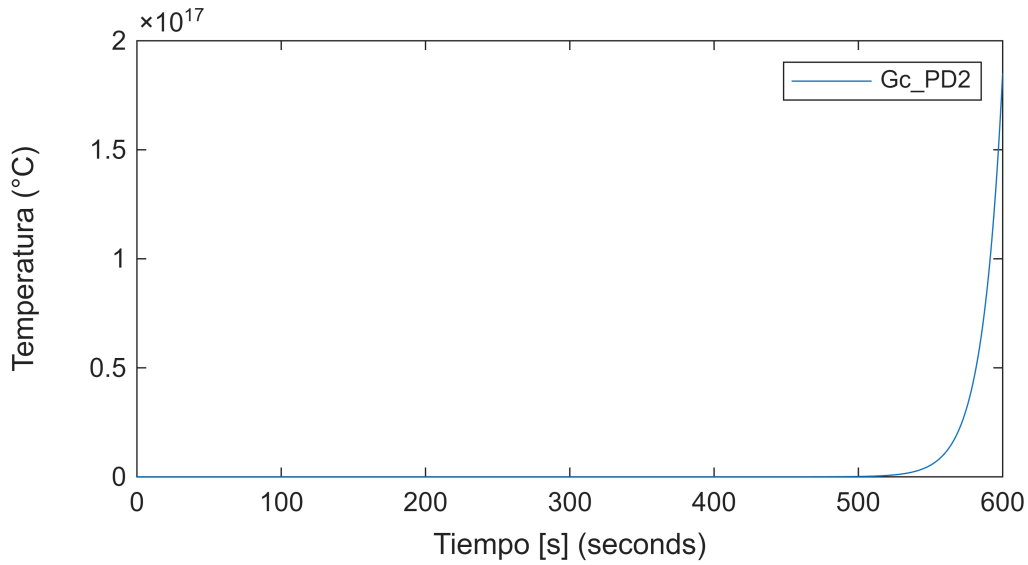
```
t = 0:1:600;
```

```
isstable(T_cl1) % Se verifica la estabilidad del sistema
```

```
ans = logical  
0
```

```
figure(2);  
step(T_cl1, t)  
legend('Gc_PD2');  
title('Respuesta Controlador PD');  
xlabel('Tiempo [s]');  
ylabel('Temperatura (°C)');
```

Respuesta Controlador PD



```
isstable(T_cl2)    % Se verifica la estabilidad del sistema
```

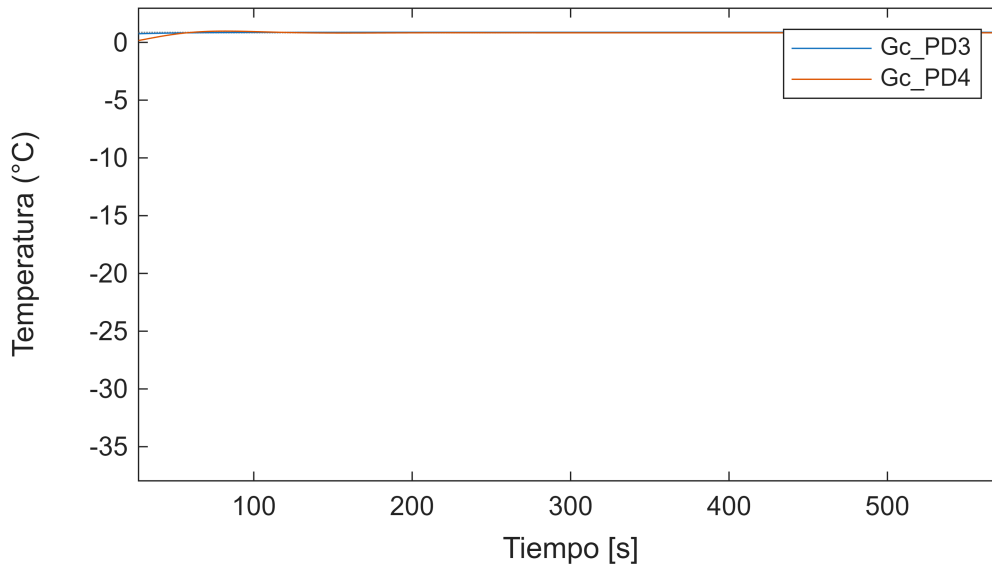
```
ans = logical  
      1
```

```
isstable(T_cl3)    % Se verifica la estabilidad del sistema
```

```
ans = logical  
      1
```

```
figure(3);  
step(T_cl2, T_cl3, t)  
legend('Gc_PD3', 'Gc_PD4');  
title('Respuesta Controlador PD');  
xlabel('Tiempo [s]');  
ylabel('Temperatura (°C)');
```


Respuesta Controlador PD



Controlador Proporcional-Integral-Derivativo (PID)

$$G_{C_{PID}}(s) = K_p \cdot \left(\frac{(s + z_1) \cdot (s + z_2)}{s} \right) \rightarrow G_{C_{PID}}(s) = K_p \cdot \left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s \right)$$

```
clc; clear; close all;
s = tf('s');

Kp = 0.11864; % Ganancia del sistema
Tp1 = 170.65; % Tiempo de establecimiento
L = 30;      % Tiempo muerto o Delay

% ----- Aproximación de Padé del retardo -----
[num, den] = pade(L, 1); % 1er orden (> 2 si desees más precisión)
Delay = tf(num, den)
```

Delay =

```
-s + 0.06667
-----
s + 0.06667
```

Continuous-time transfer function.
Model Properties

```
% ----- Planta aproximada -----
P_s = Kp * Delay / (Tp1*s + 1)
```

P_s =

```
-0.1186 s + 0.007909
```

```
-----
170.7 s^2 + 12.38 s + 0.06667
```

Continuous-time transfer function.
Model Properties

```
% Parámetros del Criterio de Ziegler-Nichols para controlador PID
```

```
kp_ZN = (1.2*Tp1)/(Kp*L)
```

```
kp_ZN =
57.5354
```

```
Ti_ZN = 2*L;
Td_ZN = 0.5*L;
```

```
ki_ZN = kp_ZN / Ti_ZN % Cálculo de la ganancia integral del controlador PID
```

```
ki_ZN =
0.9589
```

```
kd_ZN = kp_ZN * Td_ZN % Cálculo de la ganancia derivativa del controlador PID
```

```
kd_ZN =
863.0310
```

```
% Función de transferencia controlador
```

```
% Gc_PID = kp_ZN * ( 1 + (1/(Ti_ZN * s)) + (Td_ZN * s) ) % Forma alternativa
```

```
Gc_PID2 = kp_ZN + (ki_ZN / s) + (kd_ZN * s)
```

```
Gc_PID2 =
```

```
863 s^2 + 57.54 s + 0.9589
-----
s
```

Continuous-time transfer function.
Model Properties

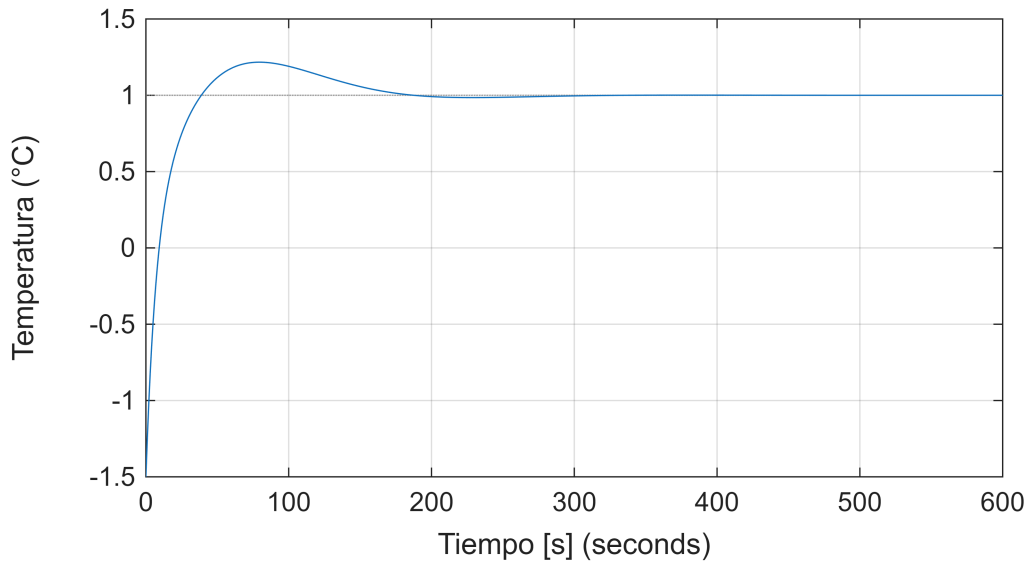
```
% Se cierra el lazo
```

```
T_cl = feedback((Gc_PID2*P_s), 1);
```

```
% Se grafica la respuesta
```

```
t = 0:1:600;
isstable(T_cl); % Se verifica la estabilidad del sistema
figure(3);
step(T_cl, t)
grid on;
title('Respuesta Controlador PID');
xlabel('Tiempo [s]');
ylabel('Temperatura (°C)');
```

Respuesta Controlador PID



Diseño de controladores de forma "automatizada"

pid(Kp,Ki,Kd), sisotool y PID Tuner

```
clc; clear; close all;  
% s = tf('s');  
  
Kp = 0.11864; % Ganancia del sistema  
Tp1 = 170.65; % Tiempo de establecimiento  
L = 30; % Tiempo muerto o Delay  
  
% Se diseñan diferentes tipos de controladores utilizando los parámetros  
% calculados de Ziegler-Nichols. Además, se usa la función 'pid(Kp,Ki,Kd)'.  
  
% Parámetros del Criterio de Ziegler-Nichols para controlador PID  
kp_ZN = (1.2*Tp1)/(Kp*L)
```

```
kp_ZN =  
57.5354
```

```
Ti_ZN = 2*L
```

```
Ti_ZN =  
60
```

```
Td_ZN = 0.5*L
```

```
Td_ZN =  
15
```

```
ki_ZN = kp_ZN / Ti_ZN % Cálculo de la ganancia integral del controlador PID
```

```
ki_ZN =  
0.9589
```

```
kd_ZN = kp_ZN * Td_ZN % Cálculo de la ganancia derivativa del controlador PID
```

```
kd_ZN =  
863.0310
```

```
% Se crean un controlador proporcional (C_p), un controlador PI (C_pi), un  
% controlador PD (C_pd) y un controlador PID (C_pid).
```

```
% Controlador P
```

```
C_p = pid(kp_ZN, 0, 0)
```

```
C_p =
```

```
Kp = 57.5
```

```
P-only controller.  
Model Properties
```

```
% Controlador PI
```

```
C_pi = pid(kp_ZN, ki_ZN, 0)
```

```
C_pi =
```

```
      1  
Kp + Ki * ---  
      s
```

```
with Kp = 57.5, Ki = 0.959
```

```
Continuous-time PI controller in parallel form.  
Model Properties
```

```
% Controlador PD
```

```
C_pd = pid(kp_ZN, 0, kd_ZN)
```

```
C_pd =
```

```
Kp + Kd * s
```

```
with Kp = 57.5, Kd = 863
```

```
Continuous-time PD controller in parallel form.  
Model Properties
```

```
% sirve = isproper(C_pd)
```

```
% Controlador PID
```

```
C_pid = pid(kp_ZN, ki_ZN, kd_ZN)
```

C_pid =

$$K_p + K_i * \frac{1}{s} + K_d * s$$

with Kp = 57.5, Ki = 0.959, Kd = 863

Continuous-time PID controller in parallel form.
Model Properties

% Finalmente, los controladores continuos se convierten a sus formatos digitales
% utilizando un tiempo de muestreo (Ts) especificado.
% Para esta conversión se utiliza la función c2d con el método 'zoh' (retención
% de orden cero), lo que hace que los controladores sean adecuados para su
% implementación en sistemas de control digital.

% Controladores digitales (suponiendo un muestreo de Ts)

Ts = 1; % Tiempo de muestreo

C_p_digital = c2d(C_p, Ts, 'zoh') % P

C_p_digital =

Kp = 57.5

P-only controller.
Model Properties

C_p_digital1 = c2d(C_p, Ts, 'tustin') % P

C_p_digital1 =

Kp = 57.5

P-only controller.
Model Properties

C_pi_digital = c2d(C_pi, Ts, 'zoh') % PI

C_pi_digital =

$$K_p + K_i * \frac{T_s}{z-1}$$

with Kp = 57.5, Ki = 0.959, Ts = 1

Sample time: 1 seconds
Discrete-time PI controller in parallel form.
Model Properties

% C_pd_digital = c2d(C_pd, Ts, 'zoh') % PD
% C_pid_digital = c2d(C_pid, Ts, 'zoh') % PID

```
% Abrir 'sisotool' para el diseño y análisis de los controladores  
% sisotool()
```