Identificación y Control Adaptativo
Trabajo Práctico 3

Truls Carlson

4 de febrero de 2025

.UBAfiuba

FACULTAD DE INGENIERÍA

Índice

1. Enunciado del Problema **1**

2. Ejercicios **2**

2.1. 1) PID con acciones antiwindup y bumpless 2

2.1.1. Bumpless 2

2.2. 2) Dos métodos de auto ajuste del PID 3

2.2.1. Ziegler-Nichols 3

2.2.2. Ajuste iterativo en lazo cerrado (IFT) 4

2.3. Resultados combinados 7

Referencias **9**

1. Enunciado del Problema

Continuamos con el tanque dado del Trabajo Práctico 1, incluyendo las dinámicas del sensor y de la válvula añadidos en el Trabajo Práctico 2. Ahora vamos a implementar un controlador PID digital con acciones antiwindup y bumpless. Además, vamos a desarrollar dos métodos de auto ajuste de este PID.

2. Ejercicios

2.1. 1) PID con acciones antiwindup y bumpless

Podemos controlar la válvula especificando su porcentaje de apertura (k), que varía entre 0 y 1. Para protegerse contra el *windup* de integral y poder cambiar entre control manual y control de PID, necesitamos implementar acciones de *antiwindup* y *bumpless*. El bloque *PID Controller* ya tiene una rutina de *antiwindup*. Se establecieron los límites de saturación en 0 y 1, y con el método de *clamping*.

2.1.1. Bumpless

Para implementar la acción *bumpless*, utilicé un conmutador (*switch*) para alternar entre el control manual y el control PID en un momento deseado. El control manual se interpreta como un operador que ajusta la apertura o cierre de la válvula. Inicialicé la planta con una altura de $h_0 = 0{,}8$ y una referencia de $r(0) = 0{,}2$. Al principio, la planta estaba controlada manualmente; posteriormente, se cambió al control PID. Es crucial que el controlador PID cuente con un mecanismo de *tracking* que le permita conocer la apertura actual de la válvula. Utilicé [1] para inspirarme. En la figura 1 se puede ver cómo el *bumpless* está integrado en la planta, y en la figura 2 se puede ver su implementación.

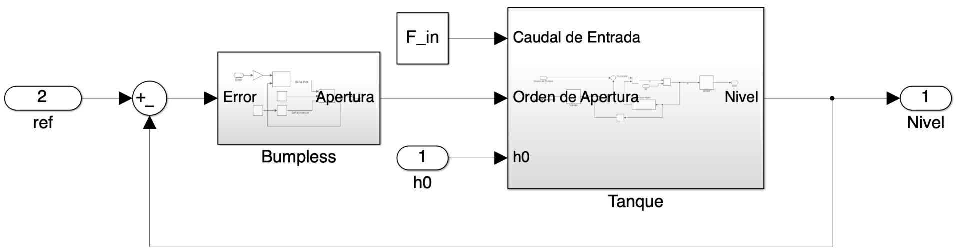

Figura 1: Vista general de la planta

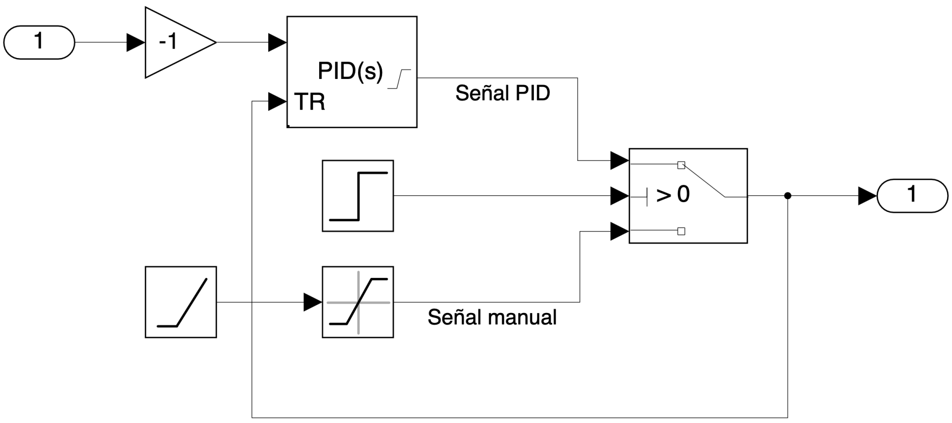

Figura 2: Implementación de bumpless

2.2. 2) Dos métodos de auto ajuste del PID

Como métodos de auto ajuste del PID, utilicé el método de Ziegler-Nichols y el de IFT, como se describe en [2].

2.2.1. Ziegler-Nichols

Para implementar el procedimiento como se describe en [2], desarrollé el siguiente código. Con este, se determina la ganancia crítica K_c y el período de oscilación T_c. Estos valores se utilizan posteriormente para calcular los parámetros del controlador PID. Los calculé según la tabla en [3].

```
K_list = 1:0.2:40;
Data = [];
T = [];
sustained_oscillation = false;
Kc = 0; % Ganancia critica
Tc = 0; % Periodo de la oscilacion

for i = 1:size(K_list, 2)
    K = K_list(i);
    out = sim("practico3_ziegler.slx");
    Data{i} = out.simout.Data; % Salida
    T{i} = out.simout.Time; % Tiempo

    % Encontrar los picos y sus tiempos correspondientes
    [peaks, locs] = findpeaks(Data{i}, T{i});
    rate_list = [];

    % Calcular la tasa de cambio entre los picos y verificar si
     la oscilacion es sostenida
    rate_list = diff(peaks) ./ diff(locs);

```

```
    threshold = -5e-05;
    avg_rate = mean(rate_list);

    if avg_rate >= threshold
        sustained_oscillation = true;
        Kc = K;
        Tc = mean(diff(locs(2:end)));
        break
    end
end

% Calcular los parametros
if sustained_oscillation
    Kp = 0.6*Kc;
    Ki = 1.2*Kc/Tc;
    Kd = 0.075*Kc*Tc;
end
```

Listing 1: Código para encontrar los parámetros con el método de Ziegler-Nichols.

En la figura 3 se pueden ver los resultados con diferentes alturas iniciales y referencias.

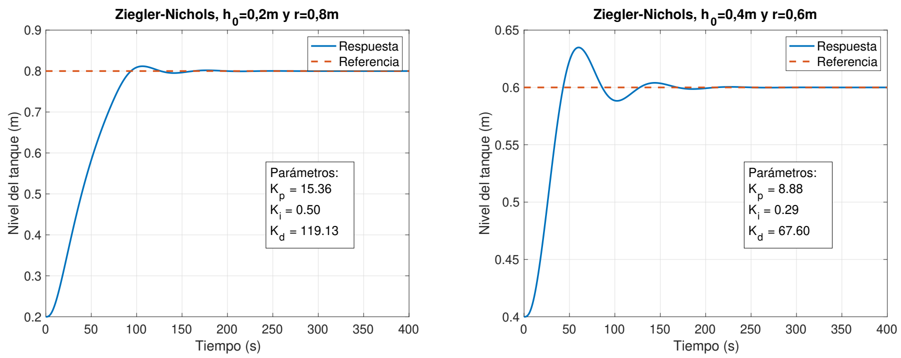

Figura 3: Resultado

2.2.2. Ajuste iterativo en lazo cerrado (IFT)

El objetivo es minimizar una función de costo que penaliza tanto las desviaciones respecto a la referencia como el uso de la señal de control. El parámetro λ determina cuánto se penaliza la acción de control en relación con la desviación de la referencia, mientras que α define la penalización asociada a los errores de seguimiento. La función de costo se expresa como:

$$J(\theta) = \frac{1}{2N} E\left[\alpha \sum_{k=1}^{N} (y_k - y_k^d)^2 + \lambda \sum_{k=1}^{N} (u_k)^2\right] \quad \text{donde } \theta = \begin{bmatrix} K_p \\ K_i \\ K_d \end{bmatrix} \tag{1}$$

Para minimizar esta función, calculamos su derivada e igualamos a cero,

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N} E \left[\alpha \sum_{k=1}^{N} (y_k - y_k^d) \frac{\partial y}{\partial \theta} + \lambda \sum_{k=1}^{N} u_k \frac{\partial u}{\partial \theta} \right] \tag{2}$$

las ecuaciones 1 y 2 son inspiradas de [2].

Dado que la derivada exacta puede ser difícil de calcular, se puede aproximar mediante diferencias finitas,

$$\frac{\partial J}{\partial \theta_k^i} \approx \frac{J(\theta + \Delta\theta_k^i) - J(\theta_k)}{\Delta\theta_k^i} \tag{3}$$

donde θ^i representa los parámetros del PID y k indica la iteración del algoritmo. En cada iteración, se toma un paso en la dirección que minimiza la función de costo, como el algoritmo de descenso del gradiente 4. El tamaño del paso depende también de la tasa de aprendizaje γ.

$$\theta_{k+1} = \theta_k - \gamma \frac{\partial J}{\partial \theta_k} \tag{4}$$

Implementé el algoritmo a través de Matlab, cómo se puede ver en el siguiente listing 2.

```
theta_IFT = [Kp, Ki, Kd];

gamma = 1; % Tasa de aprendizaje
alpha = 100; % Penalizacion, referencia
lambda = 0.1; % Penalizacion, control

num_iteraciones = 150;

for iter = 1:num_iteraciones

    % Simular el modelo
    out = sim('practico3_IFT.slx');
    y = out.nivout.Data;
    u = out.ctrout.Data;

    % Funcion de costos
    y_d = h_list(4) * ones(size(y)); % Referencia
    J = (alpha*sum((y - y_d).^2) + lambda*sum(u.^2)) /
    (2*length(y));

    % Aproximacion del gradiente
    gradient = zeros(1, 3);
    for j = 1:3
        theta_perturbado = theta_IFT;
        delta = 0.1 * abs(theta_IFT(j)); % La delta depende del
     tamano del parametro
        theta_perturbado(j) = theta_perturbado(j) + delta;

        % Simular el modelo con parametros perturbado
```

```
        Kp = theta_perturbado(1);
        Ki = theta_perturbado(2);
        Kd = theta_perturbado(3);
        out_perturbado = sim('practico3_IFT.slx');

        % Funcion de costos perturbado
        y_perturbado = out_perturbado.nivout.Data;
        u_perturbado = out_perturbado.ctrout.Data;
        J_perturbado = (alpha*sum((y_perturbado-y_d).^2) +
    lambda*sum(u_perturbado.^2))/(2*length(y_perturbado));

        % Calcular el gradiente
        gradient(j) = (J_perturbado - J) / delta;
    end

    % Actualizar los parametros
    theta_IFT = theta_IFT - gamma * gradient;
    Kp = theta_IFT(1);
    Ki = theta_IFT(2);
    Kd = theta_IFT(3);

end
```

Listing 2: Implementación de IFT

Para iniciar el algoritmo, los parámetros se establecieron según la figura 3, permitiendo así encontrar un mínimo local cercano. Experimenté con diferentes valores de la tasa de aprendizaje y las penalizaciones. Como se puede ver en la figura 4, el método de IFT claramente mejora los parámetros. La respuesta de la planta no rebasa tanto, y alcanza la referencia más rápida. Los parámetros que obtuvimos a través del algoritmo de IFT se muestran en la figura.

El método de IFT ofrece un ajuste de PID mejor al método de Ziegler-Nichols porque optimiza directamente una función de costo basada en la respuesta real del sistema. Mientras que Ziegler-Nichols ajusta los parámetros PID según un punto de oscilación crítico, IFT mejora el desempeño general reduciendo el sobrepaso y el tiempo de establecimiento. Además, al adaptarse iterativamente a los datos experimentales, IFT es más robusto frente a las no linealidades del tanque cónico y a la incertidumbre del modelo.

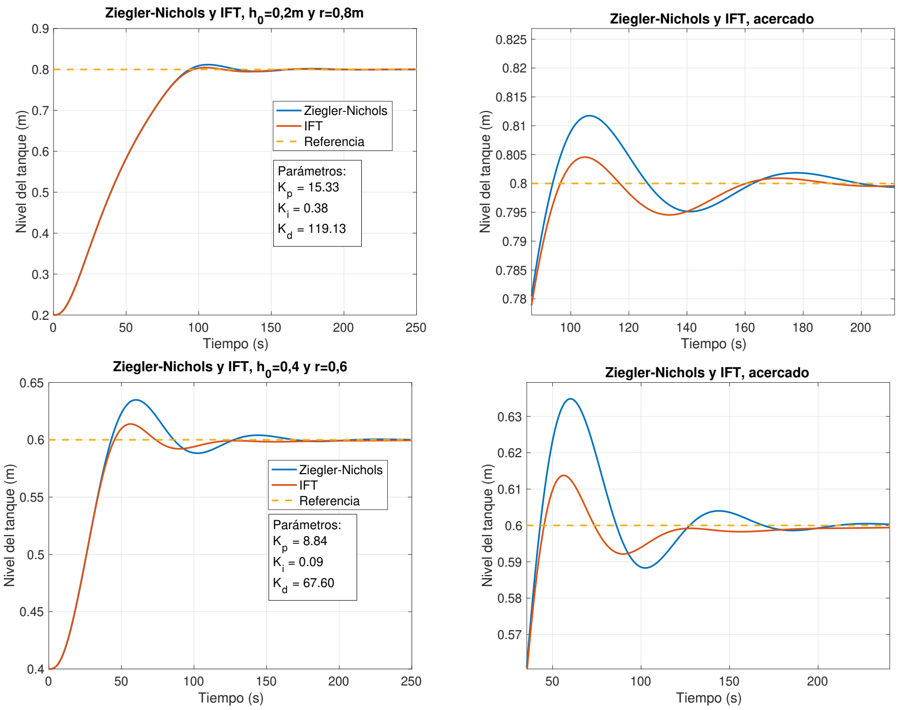

Figura 4: Comparación entre los dos métodos.

2.3. Resultados combinados

Cuando implementé todo el sistema junto, usé los parámetros del PID obtenidos del IFT con $h_0 = 0{,}2$ y $r = 0{,}8$. Se cambia de control manual a PID a $t = 220$s con bumpless. Es decir, el sistema se controla con control manual hasta $t = 220$s y posteriormente con control del PID. En la figura 5 se puede ver la referencia y la respuesta del tanque.

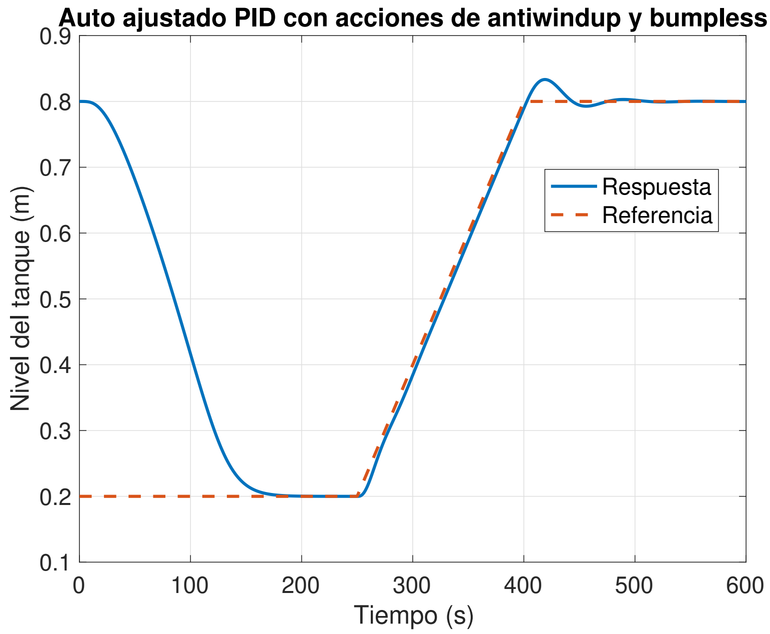

Figura 5: La respuesta del todo el sistema desarrollado.

Referencias

[1] Bumpless control transfer between manual and pid control. https://la.mathworks.com/help/simulink/slref/bumpless-control-transfer-between-manual-and-pid-control.html. Accedido: 19-12-2024.

[2] Universidad de Buenos Aires Facultad de Ingeniería. *Ajuste de PIDs*. 2024.

[3] Ziegler–nichols method. https://en.wikipedia.org/wiki/ZieglerâĂŞNichols_method. Accedido: 02-01-2025.