Control clásico y control borroso

Robótica - Grado en Ingeniería de Computadores

Departamento de Sistemas Informáticos

E.T.S.I. de Sistemas Informáticos - Universidad Politécnica de Madrid

22 de octubre de 2023

La teoría del control (I)

Se ocupa del **control de sistemas dinámicos** en procesos de todo tipo.

- Se considera campo interdisciplinario de la ingeniería y de las matemáticas.

¿Cómo **llevar sistemas a estados deseados** en función de sus entradas ...

- ... **minimizando** el **tiempo** de ajuste, rebasamiento y error estacionario?
- ... **garantizando** un nivel de **estabilidad** de control?
- ... **persiguiendo** el grado de **optimalidad**?

Por cierto, las entradas también reciben nombre de **referencia**.

La teoría del control (y II)

Dentro de la teoría de control existen también otros dos aspectos de estudio:

- **Controlabilidad**: Alterar un sistema usando solo manipulaciones admisibles
- **Observabilidad**: Medida de lo bien que se infieren los estados internos de un sistema a partir del conocimiento de sus salidas externas

Existen dos grandes divisiones en la teoría de control, a saber:

- **Clásica**: Diseño de sistemas de una única entrada y una única salida[1]
- **Moderna**: Diseño de sistemas con múltiples entradas y salidas

[1] Excepto cuando se analiza el impacto de perturbaciones, donde sí se utiliza una segunda entrada.

Función de transferencia

Función que **modela la salida** de un sistema **para cada entrada posible**[2].

El caso más sencillo ofrece una entrada para una salida:

- La gráfica generada se denomina **curva de transferencia**
- Muy común en áreas como tratamiento de señal o teoría de la comunicación

Se suele utilizar sólo en sistemas lineales invariantes en el tiempo (LTI):

- La mayoría de sistemas tienen características de entrada/salida no lineales
- Suelen comportarse linealmente dentro de sus parámetros "normales"

Modelización **teórica**, por lo que no tiene por qué replicar exactamente todos los detalles del sistema modelado.

Ingeniería automática

Puede definirse como la **aplicación práctica de la Teoría del control**

Sus objetivos fundamentales son:

1. **Modelado** de sistemas dinámicos en términos de entradas y salidas
2. **Diseño** de controladores para regular el comportamiento de dichos sistemas
3. **Implementación** de controladores empleando la tecnología disponible

Se suele considerar subcampo de la Ingeniería eléctrica:

- Pero sólo porque muchos controladores son eléctricos
- En realidad no tiene por qué, también existen controladores mecánicos
- Incluso hay sistemas *software* controlados por controladores Software

Sistemas de control (controladores)

Regulan el **comportamiento** de otros sistemas mediante bucles de control

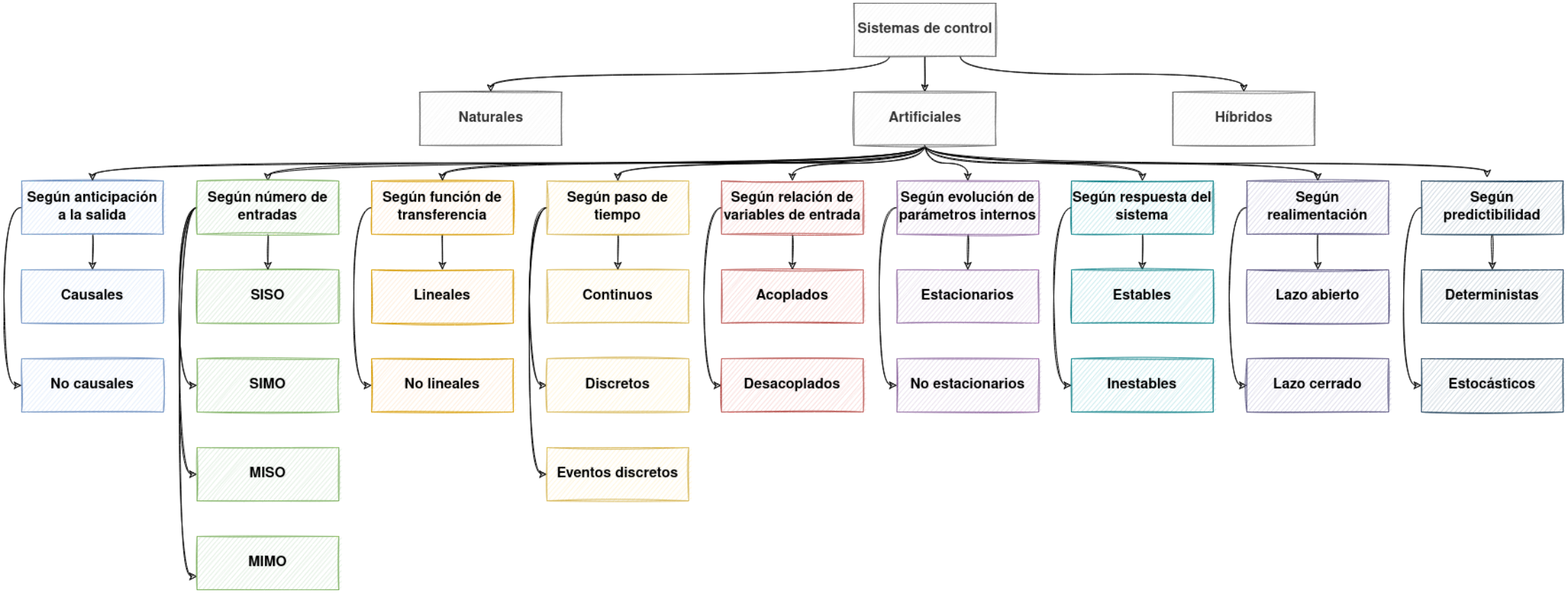


Fig.1 - Clasificación de sistemas de control

Sistema de control automático: Diseñado para funcionar sin intervención.

Error y rebasamiento en un controlador

Error: Diferencia entre estado actual y estado deseado de un sistema

Rebasamiento: Magnitud o dirección cuando el estado supera el *set point*

Ambos son dos tipos de divergencias. Pueden ofrecer diferente información:

- **Existe/no existe** error: La menor cantidad de información
- **Dirección**: Hacia dónde hay que ir para minimizar el error
- **Magnitud**: La distancia al estado objetivo

Controlar un sistema es mejor cuando conocemos dirección y magnitud del error

Clasificación según anticipación a la salida

Punto de vista respecto la relación entre salida y los valores actual y pasados

Causales: La salida es consecuencia del valor actual y pasado de la entrada

- Son con los que trabajaremos normalmente porque modelan sistemas reales

No causales: No es posible determinar la salida en función de la entrada

- No existen físicamente, son representaciones abstractas

Estos controladores se diseñan de tal manera que la salida depende de valores futuros de la entrada.

Clasificación según número de entradas y salidas

Clasificación sencilla en función de si hay una o muchas entradas o salidas:

- **SISO** (Single input, single output)
- **SIMO** (Single input, multipe output)
- **MISO** (Multiple input, single output)
- **MIMO** (Multiple input, multiple output)

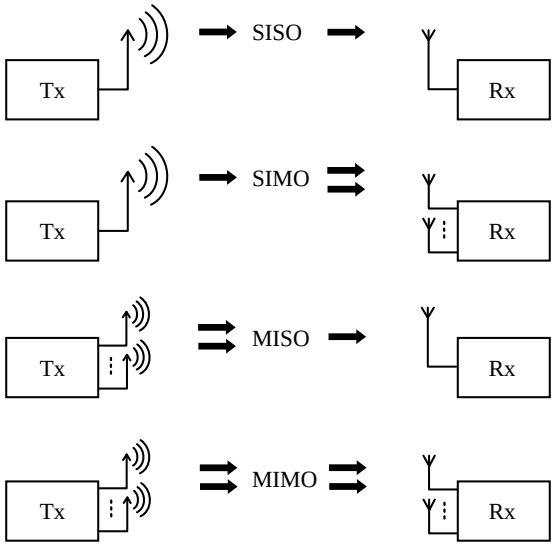
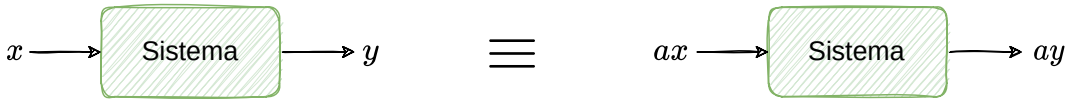


Fig.2 - Clasificación según el número de entradas y salidas

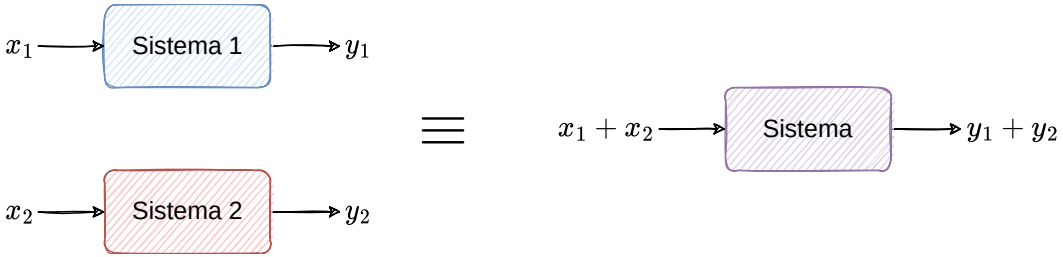
Clasificación según función de transferencia

Un sistema es lineal si su función característica cumple los principios de:

Homogeneidad



Superposición



Por tanto el controlador se denominará:

- **Lineal**: Si cumple ambos principios de superposición y homogeneidad
- **No lineal**: Si no cumple al menos uno de ellos.

Clasificación según paso del tiempo

Otro punto de vista: ¿cómo se modela el paso del tiempo en un sistema?:

- De **tiempo continuo**: El tiempo evoluciona de manera continua
- De **tiempo discreto**: El tiempo evoluciona de manera discreta
- De **eventos discretos**: La tiempo evoluciona cuando ocurren ciertos eventos

Clasificación según relación entre las variables de entrada

Cuando hablamos de varios controladores, estos se pueden clasificar como:

- **Acoplados**: Si las variables de ambos están relacionadas entre sí
- **Desacoplados**: Si no lo están

Clasificación según evolución de parámetros internos

Los controladores mantienen parámetros que modulan su respuesta

Así diferenciamos dos tipos de controladores:

- **Estacionarios**: Los parámetros no varían durante su funcionamiento
- **No estacionarios**: Los parámetros pueden variar a lo largo del tiempo

Clasificación según respuesta del sistema

La salida de un sistema petenece a un dominio, por lo que podemos clasificarlos:

- **Estables**: Para toda entrada acotada la respuesta es acotada
- **Inestables**: Al menos una entrada acotada produce una salida no acotada

Clasificación según realimentación

Realimentación: Relación secuencial de causas y efectos entre variables.

- O de otro modo, cuando una o más variables de salida se pasan a la entrada
- También se la conoce como **retroalimentación** o ***feedback***
- Concepto muy antiguo, aunque fue formalizado por Norbert Wiener en 1948

Dependiendo de la acción correctiva que tome el sistema:

- Si es apoyar la salida: Realimentación positiva o "efecto bola de nieve"
- En caso contrario: Realimentación negativa o regulación autocompensatoria

Dos tipos, de **lazo cerrado** y de **lazo abierto**

Control de lazo cerrado

Cuando se usa la realimentación para minimizar el error de la salida

- El controlador usa el *feedback* para conocer en cada momento la salida real

El *feedback* provee al controlador de un comportamiento correctivo:

1. El controlador monitoriza una variable de salida (PV, de *Process Variable*)
2. La compara con la referencia, consigna o punto de ajuste (SP, de *set point*)
3. $SP - PV$ da lugar a la **señal de error**, que es la salida a minimizar

Ejemplos de estos sistemas de control:

- Convergencia fonética de un humano
- Control de crucero de un vehículo

Control de lazo abierto

Aquellos controladores que no tienen en cuenta su influencia en el entorno

Ejemplos de estos sistemas de control:

- Tostadora (las hay que comprueban el color de la rebanada)
- Secadora estándar (las hay que comprueban la humedad del tambor de secado)

Controladores de lazo abierto vs. lazo cerrado

Lazo abierto

Ventajas

- Sencillos, de fácil mantenimiento

Inconvenientes

- Requieren calibración inicial
- Sensibles a perturbaciones
- Mejor en modelos simples

Lazo cerrado

Ventajas

- Control de sistemas inestables
- Robustez frente perturbaciones

Inconvenientes

- Mayor coste (más sensores)
- Son más complejos de modelar

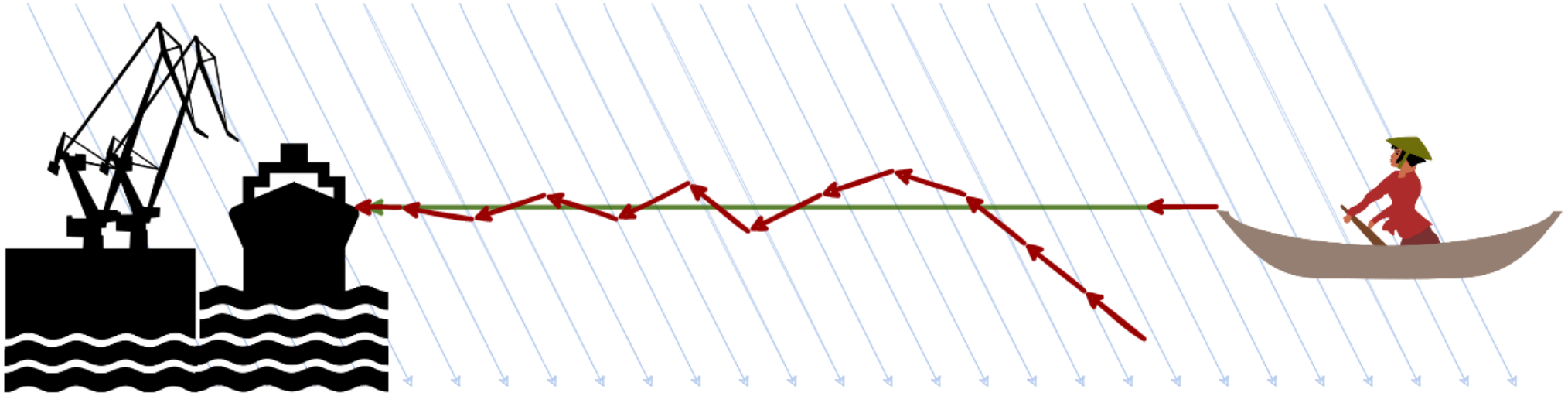
Clasificación según predictibilidad

En función de lo predecible de la respuesta de un sistema, lo podemos clasificar como:

- **Determinista**: Si su comportamiento es extremadamente predecible
- **Estocástico**: Si es imposible predecir su comportamiento futuro

Cibernética

Etimología



La palabra timonel (en inglés *steersman*) viene el griego antiguo *kybernetes*:

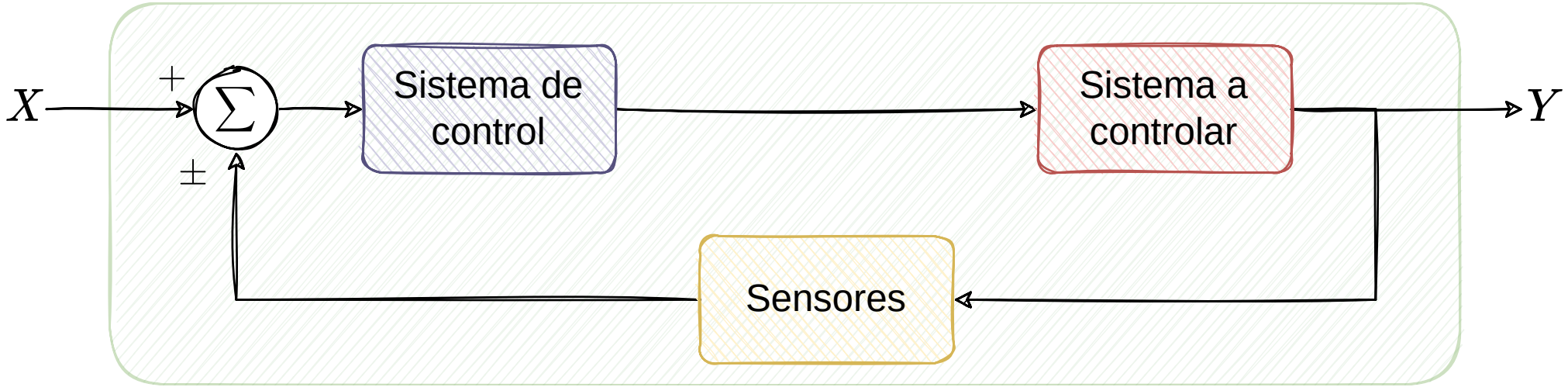
- Los romanos la usaron para su *gubernare* (no eran muy buenos navegando)
- Norbert Wiener tomó la palabra griega y le añadió el sufijo *-ics*[3]

Podemos definir la **cibernética** como el **arte de** gobernar o **controlar**

[3] Disculpas por anticipado a todo estudiante de griego clásico. En realidad remplazó el sufijo *-tes* (actor, agente) por *-ike* (disciplina, práctica, actividad), pasando de *κυβερνήτης* (kybernetes) a *κυβερνητική* (kybernetike).

Sistema de control genérico

Desde el punto de vista de la cibernética, un sistema tiene la siguiente forma:



Generalmente son bucles de control con realimentación

- Realimentación positiva o negativa (de ahí el ± en la generación del input)
- Puede haber sistemas de lazo abierto, pero no suelen ser de interés aquí

Componentes más importantes de la cibernética

Realimentación: Mejora el rendimiento dinámico del sistema

- Es un principio muy general que abarca tecnología, astronomía, biología, ...

Información: Flujos de datos que rodean un sistema

Modelo: Basada en que existe isomorfismo[5] entre diferentes sistemas

Ley de Ashby[6]: *«cuanto mayor es la variedad de acciones, mayor es la variedad de perturbaciones a controlar»*.

- Sólo podemos controlar cuando sistema y controlador comparten variedad.

4 Fue definida por *Shannon* como la cantidad de incertidumbre eliminada que se describe probabilísticamente.
5 Sistemas mecánicos, electrónicos, etc. se pueden describir mediante las mismas ecuaciones diferenciales.
6 En algunos contextos se conoce como la **Ley de la Variedad Requerida**.

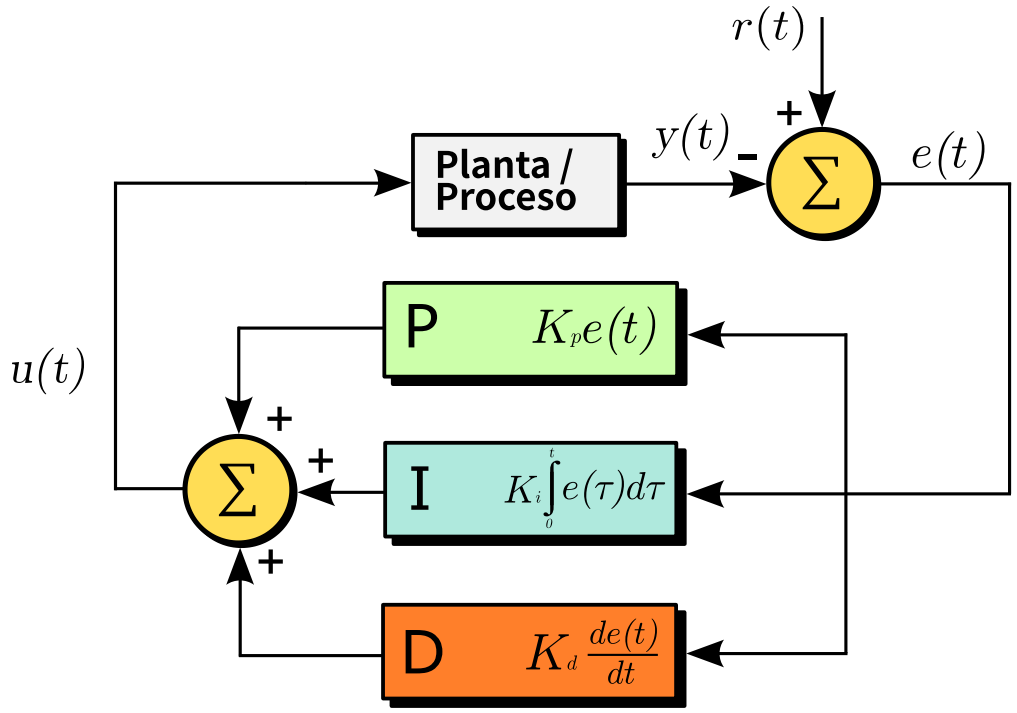
Control clásico

Control clásico con realimentación (principios 1920)

Se basa en la realimentación de la señal de salida al sistema de control

Modifica la señal de entrada para que la señal de salida se aproxime a la señal deseada, basándose en un modelo matemático del sistema

Las componentes P (*Proportional*), I (*Integral*) y D (*Derivative*) se pueden combinar para obtener un control más eficiente



La señal de salida tiene que ser reescalada en función del actuador que se esté utilizando

Control proporcional P

En el algoritmo de control proporcional, la salida del controlador es proporcional a la señal de error, que es la diferencia entre el punto objetivo que se desea y la variable de proceso:

$$u(t) = K_p e(t)$$

donde K_p es la ganancia proporcional y $e(t)$ es la señal de error

El control proporcional es el más simple de los controladores, pero también el más inestable. El control proporcional es adecuado para sistemas que tienen un error pequeño y que no cambian rápidamente

Control integral I

En el algoritmo de control integral, la salida del controlador es proporcional a la integral de la señal de error, que es la suma de las diferencias entre el punto objetivo que se desea y la variable de proceso:

$$u(t) = K_i \int_0^t e(\tau)d\tau$$

donde K_i es la ganancia integral y $e(t)$ es la señal de error

Aumenta la acción en relación no sólo con el error sino también con el **tiempo** durante el cual ha persistido. Así, si la fuerza aplicada no es suficiente para llevar el error a cero, esta fuerza se incrementará a medida que pase el tiempo

Control derivativo D

En el algoritmo de control derivativo, la salida del controlador es proporcional a la derivada de la señal de error, que es la velocidad de cambio de la señal de error:

$$u(t) = K_d \frac{de(t)}{dt}$$

donde K_d es la ganancia derivativa y $e(t)$ es la señal de error.

Aumenta la acción en relación no sólo con el error sino también con la **velocidad** a la que cambia el error. Así, si la fuerza aplicada no es suficiente para llevar el error a cero, esta fuerza se incrementará a medida que el error cambie de signo.

Control proporcional-integral-derivativo PID

El control PID es un controlador que combina las tres componentes básicas de control: proporcional, integral y derivativa. El control PID es el más utilizado en la industria

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

donde K_p, K_i y K_d son las ganancias proporcional, integral y derivativa, respectivamente

Hay que optimizar los parámetros K_p, K_i y K_d para obtener un control eficiente

Simulador

Pseudocódigo del control PID

```
def PID(Kp, Ki, Kd, dt):
    integral = 0
    last_error = 0
    while True:
        error = setpoint - actual_position
        integral += error * dt
        derivative = (error - last_error) / dt
        output = Kp * error + Ki * integral + Kd * derivative
        last_error = error
        yield output
```

`yield` devuelve el valor de la salida del controlador y se suspende hasta la siguiente iteración. Así se define un generador que se puede utilizar en un bucle `for`.

Control en cascada

El control en cascada es un método de control en el que se utilizan dos o más controladores para controlar un proceso

La salida del controlador externo se utiliza como entrada del controlador interno, y representa el punto de referencia del controlador interno

Se utiliza para controlar procesos que tienen un tiempo de respuesta muy lento, como la temperatura de un horno, una habitación, etc

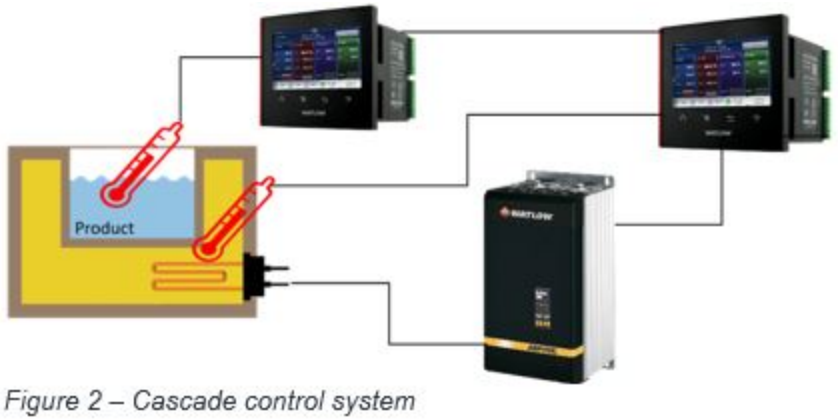


Fig.1 - Sistema de control en cascada. Fuente: Watlow

Ventajas y desventajas del control clásico

Ventajas

1. Fácil de implementar (sólo una simple ecuación)
2. Utiliza pocos recursos
3. Resistente a los desajustes de optimización
4. Fácil de optimizar
5. Buena respuesta a las perturbaciones

Desventajas

1. Bajo rendimiento en procesos con largos tiempos de espera
2. Bajo rendimiento para tratar fuertes no linealidades
3. Dificultad para manejar múltiples variables con fuerte interrelación
4. Dificultad para manejar múltiples restricciones

Control borroso

Recordatorio de lógica borrosa

Se puede considerar una extensión de la teoría clasica de conjuntos:

- En esta teoría, los elementos pertenecen o no a un conjunto
- Función característica: $f(x) = 1$ si $x \in A$ y $f(x) = 0$ si $x \notin A$

Trata información a priori imprecisa en términos de conjuntos borrosos:

- Los elementos pertenecen a un conjunto con un grado de pertenencia
- Función de pertenencia: $f(x) = \mu(x) \in [0, 1]$

Los conjuntos borrosos se agrupan en particiones

- Una partición se define sobre una variable denominada lingüística

Definiciones

Variable lingüística: Variable cuyos valores son términos en lenguaje natural

Partición borrosa: Todos los conjuntos borrosos de una variable lingüística

Función de pertenencia: Determina el grado de pertenencia de un elemento a un conjunto borroso (en tanto por uno)

Ejemplo: La variable lingüística $precio$ *puede tomar los valores* $precio \equiv \{barato, normal, caro\}$*. Estos serán tres conjuntos borrosos, cada uno con las funciones de pertenencia* $\{f_{barato}(x)$, $f_{normal}(x)$ *y* $f_{caro}(x)\}$.

Operaciones borrosas

Complemento: $f'_{barato}(x) = 1 - f_{barato}(x)$

t-normas (intersección)

- Mínimo: $f_{barato} \cap f_{normal} = min(f_{barato}, f_{normal})$
- Producto algebraico:
  $f_{barato} \cap f_{normal} = f_{barato} \cdot f_{normal}$

t-conormas (unión)

- Máximo: $f_{barato} \cup f_{normal} = max(f_{barato}, f_{normal})$
- Suma algebráica:
  $f_{barato} \cup f_{normal} = f_{barato} + f_{normal} - f_{barato} \cdot f_{normal}$

La **inferencia** ($\rightarrow$) se suele definir como la operación de **intersección**

Reglas borrosas

Son reglas que relacionan varios antecedentes con consecuentes, donde:

- **Antecedentes**: Conjuntos borrosos de entrada
- **Consecuentes**: Conjuntos borrosos de salida

***Si** el precio es barato **Y** la calidad es mala **entonces** la satisfacción es baja.*

Se agrupan en una **base de reglas**, las cuales pueden ser de varios tipos:

- De tipo Mandani: **Si** V_1 es $F_i^{V_1}$ **Y** V_2 es $F_j^{V_2}$ **Y . . . entonces** V_o es $F_k^{V_o}$
- De tipo Sugeno: **Si** V_1 es $F_i^{V_1}$ **Y** V_2 es $F_j^{V_2}$ **Y . . . entonces** $V_o = f(\vec{x})$

Fuzzification y defuzzification

Fuzzification: Convertir valores de entrada concretos en conjuntos borrosos

- Es basicamente aplicar las funciones de pertenencia a los valores de entrada

Defuzzification: Convertir conjuntos borrosos en valores de salida concretos

- Existen muchas técnicas para realizar esta operación
- Las más comunes son el centroide y el centroide simplificado

Centroide

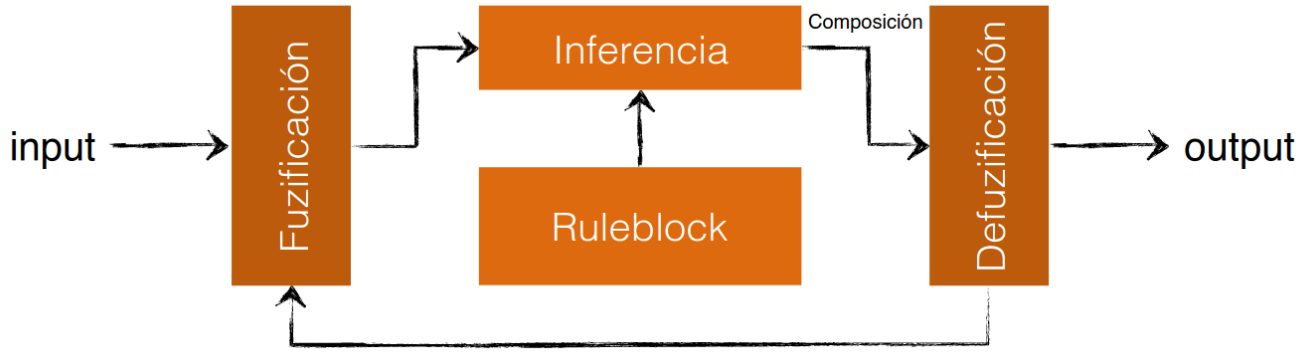
$$y = \frac{\int y \cdot \mu(y) dy}{\int \mu(y) dy}$$

Centroide simplificado

$$y \approx \frac{\sum y \cdot \mu(y)}{\sum \mu(y)}$$

Controlador borroso

Es un sistema de control que se apoya en la lógica borrosa como sigue:



1. Toma la entrada al sistema
2. Pasa los valores a pertenencia a conjuntos borrosos (*fuzzification*)
3. Infiere conjuntos de borrosos de salida haciendo uso de las reglas borrosas
4. Pasa los conjuntos borrosos de salida en valores concretos (*defuzzification*)
5. Aplica la salida al sistema a controlar

Ejemplo: Diseño de un controlador borroso

Para diseñar un controlador borroso, se debe seguir el siguiente proceso:

1. Identificar variables de entrada y de salida
2. Determinar los conjuntos borrosos para cada variable
3. Definir las reglas borrosas que van a regir el comportamiento del controlador
4. (Opcional) Normalización y escalado de entradas y salidas

Implementaremos un controlador para el *problema de las propinas:*

- Problema clasico de control borroso
- ¿Cuánto dar de propina en función de la calidad del servicio y de la comida?
- Usaremos la biblioteca `skfuzzy` para implementar un controlador borroso

Ejemplo: Formulación del problema

Antecedentes (entradas):

- Servicio (de 0 a 10): $malo$, $normal$, $bueno$
- Calidad (de 0 a 10): $mala$, $aceptable$, $buena$

Consecuentes (salidas):

- Propina (de 0 a 25): $baja$, $media$, $alta$

Reglas:

1. **Si** Servicio $bueno$ o Calidad $buena$ **entonces** Propina $alta$
2. **Si** Servicio $normal$ **entonces** Propina $media$
3. **Si** Servicio $malo$ y Calidad $mala$ **entonces** Propina $baja$

Ejemplo: Implementación de las variables lingüísticas

El primer paso es definir las variables de entrada y salida del controlador

```
import numpy as np
from skfuzzy import control as ctrl

# Antecedentes
servicio = ctrl.Antecedent(np.arange(0, 11, 1), 'servicio')
calidad = ctrl.Antecedent(np.arange(0, 11, 1), 'calidad')
# Consecuente
propina = ctrl.Consequent(np.arange(0, 26, 1), 'propina')
```

Ejemplo: Definición de los conjuntos borrosos

Para cada variable, se definen los conjuntos borrosos que la componen

```
import skfuzzy as fuzz

# Conjuntos borrosos de servicio
servicio['malo'] = fuzz.trimf(servicio.universe, [0, 0, 5])
servicio['normal'] = fuzz.trimf(servicio.universe, [0, 5, 10])
servicio['bueno'] = fuzz.trimf(servicio.universe, [5, 10, 10])
# Conjuntos borrosos de calidad
calidad['mala'] = fuzz.trimf(calidad.universe, [0, 0, 5])
calidad['aceptable'] = fuzz.trimf(calidad.universe, [0, 5, 10])
calidad['buena'] = fuzz.trimf(calidad.universe, [5, 10, 10])
# Conjuntos borrosos de propina
propina['baja'] = fuzz.trimf(propina.universe, [0, 0, 13])
propina['media'] = fuzz.trimf(propina.universe, [0, 13, 25])
propina['alta'] = fuzz.trimf(propina.universe, [13, 25, 25])
```

Se puede usar el método `.automf(n)` para definirlos de forma automática

Ejemplo: Visualización de los conjuntos borrosos

Para visualizar los conjuntos borrosos, se puede usar la función `view()`

```
servicio.view()
calidad.view()
propina.view()
```

Concretamente mostrará la variable lingüística junto con:

- Las **funciones de pertenencia** que caracterizarán a cada conjunto borroso
- El **dominio** de la variable lingüística

Ejemplo: Definición de las reglas

Para definir las reglas, se debe usar la función `ctrl.Rule()`

```
rulebase = [
    ctrl.Rule(servicio['bueno'] | calidad['buena'], propina['alta']),
    ctrl.Rule(servicio['normal'], propina['media']),
    ctrl.Rule(servicio['malo'] & calidad['mala'], propina['baja'])
]
```

Suele ser buena costumbre definir las reglas en una lista

Ejemplo: Definición del controlador

Para definir el controlador, se debe usar la función `ctrl.ControlSystem()`

```
>>> controlador = ctrl.ControlSystem(rulebase)
```

Luego se simula con la función `ctrl.ControlSystemSimulation()`

- Este objeto se encarga de implementar casos concretos sobre un controlador

```
>>> simulacion = ctrl.ControlSystemSimulation(controlador)
```

- El caso concreto se simulará con la función `compute()`

```
>>> simulacion.input['calidad'] = 6.5
>>> simulacion.input['servicio'] = 9.8
>>> simulacion.compute()
>>> print(simulacion.output['propina'])
19.847607361963192
```

Ventajas y desventajas del control borroso

Ventajas

1. Modela sistemas no lineales eficazmente
2. Maneja incertidumbres y sistemas sin modelos matemáticos claros
3. Basado en razonamiento lógico borroso, facilitando el modelado del conocimiento humano (info. cualitativa, sin necesidad de modelo preciso)
4. Flexibilidad para combinarse con otras técnicas de control

Desventajas

1. Subjetividad en la selección de conjuntos borrosos y reglas
2. Reto en determinar la optimización del controlador
3. Potencial alta complejidad computacional
4. Dificultades en el análisis de estabilidad y rendimiento
5. Ausencia de estándares únicos en su diseño

Ejemplos de control PID y borroso

El siguiente notebook contiene un ejemplo de controlador PID y borroso sobre un sistema SISO

Ejercicio: 3.1. Controladores PID y borroso para el mantenimiento de la ventilación en un refugio[2]

2 https://githubtocolab.com/etsisi/Robotica/blob/main/Notebooks/3.1. Controladores PID y borroso para el mantenimiento de la ventilación en un refugio.ipynb

¡GRACIAS!