MANUAL DE ESTUDIO · SISTEMAS Y CONTROL EN MECATRÓNICA

Control óptimo, robusto, no lineal y adaptativo

Capítulo 05 · En el video: **minuto 42:06** · Duración 10:36

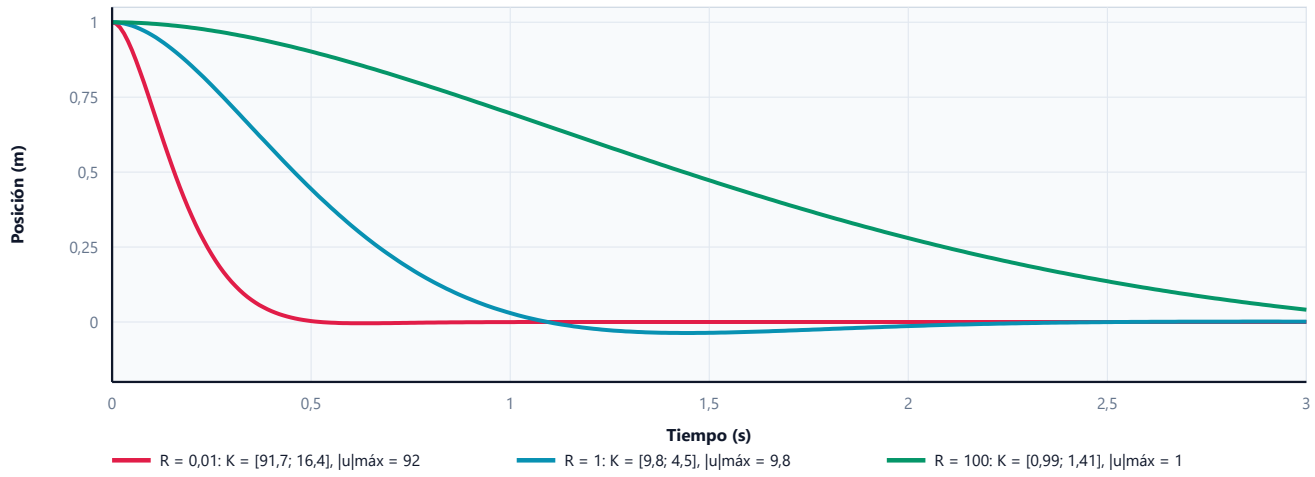
Objetivo. Conocer las técnicas de control moderno y elegir la adecuada según el problema.

Ideas clave

LQR

Minimiza $J = \int (x^T Q x + u^T R u) dt$: Q pesa el error y R el esfuerzo. La ley $u = -Kx$ sale de la ecuación de Riccati. Con todos los estados medidos garantiza un margen de fase de al menos 60°.

LQR de un doble integrador: efecto del peso R del esfuerzo cursosmecatronica.com



Q = diag(100, 1). La ganancia sale de la ecuación de Riccati, no de prueba y error: el diseñador solo decide cuánto pesa el error frente al esfuerzo.

Compromiso entre rapidez y esfuerzo en el control óptimo.

LQG y H∞

Si no se miden todos los estados, se combinan LQR y Kalman (principio de separación). La norma H2 mide la energía media de la respuesta; H∞, el peor caso de ganancia: el control H∞ asegura robustez frente a incertidumbres acotadas.

No linealidades

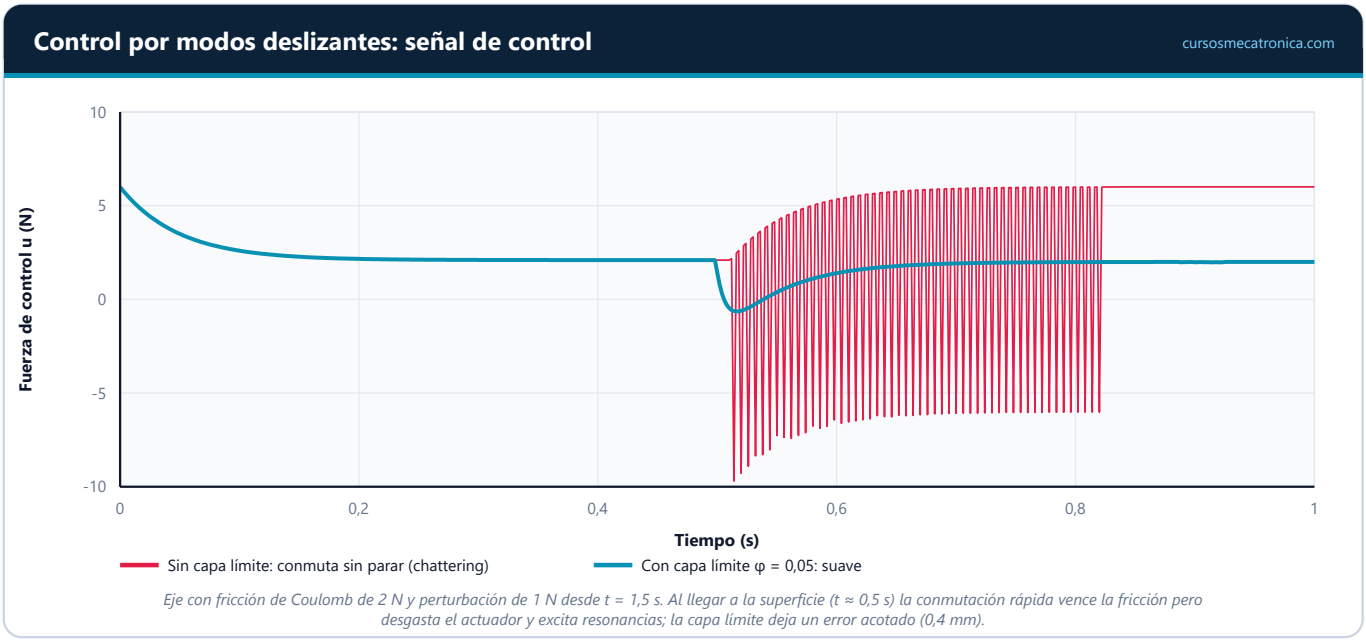
Fricción de Stribeck y stick-slip, zonas muertas, saturación, backlash e histéresis limitan al control lineal: producen ciclos límite, errores estacionarios y windup.

Diseño no lineal

Lyapunov: una función de energía $V > 0$ que siempre decrece garantiza estabilidad. Backstepping la construye paso a paso. La linealización exacta (par calculado en robótica) cancela las no linealidades conocidas del modelo.

Modos deslizantes

Se fuerza al error a una superficie $s = \dot{e} + \lambda e = 0$ con una acción conmutada: es robusto ante perturbaciones acotadas, pero conmuta rápido (chattering); una capa límite lo suaviza a cambio de un pequeño error.



Chattering y capa límite.

Control adaptativo

Ajusta sus parámetros en línea: el auto-sintonizador estima la planta con mínimos cuadrados recursivos (RLS); el MRAC hace que la planta siga a un modelo de referencia con una ley de adaptación derivada de Lyapunov.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Costo LQR	$J = \Sigma\ (x^T Q x + u^T R u)$	Discreto.
Riccati discreta	$P = Q + \Phi^T P \Phi - \Phi^T P \Gamma (R + \Gamma^T P \Gamma)^{-1} \Gamma^T P \Phi$	Se itera hasta converger.
Ganancia LQR	$K = (R + \Gamma^T P \Gamma)^{-1} \Gamma^T P \Phi,\ u = -Kx$	Realimentación de estados.
Superficie deslizante	$s = \dot{e} + \lambda e$	Error → 0 con constante 1/λ sobre s = 0.
Capa límite	$u = u_{eq} + k \cdot \mathrm{sat}(s/\varphi)$	Reduce el chattering.
Adaptación MRAC	$\theta = -\gamma \cdot e \cdot r$	e = y − y_m.

Ejemplo resuelto: LQR de un eje con doble integrador

1. Modelo discreto con T_s = 10 ms: $\Phi = [[1, 0{,}01], [0, 1]]$, $\Gamma = [0{,}00005; 0{,}01]$.
2. Q = diag(100, 1). Con R = 1: $K = [9{,}77; 4{,}53]$, polos 0,977 ± 0,021j.
3. Desde 1 m de error llega a 2 cm en 1,87 s con fuerza máxima 9,8.
4. Con R = 0,01 la fuerza máxima sube a 92 y tarda 0,53 s; con R = 100 la fuerza es 1 y tarda más de 3 s.

lqr_lqg.py itera la ecuación de Riccati y simula los tres casos; no_lineal_adaptativo.py compara PD, modos deslizantes y MRAC.

Qué técnica usar

Situación	Técnica
Modelo lineal y todos los estados medidos	LQR
Pocos sensores y ruido	LQG (LQR + Kalman)
Incertidumbre de modelo acotada	H∞ o modos deslizantes
Robot con dinámica conocida	Par calculado (linealización exacta)
Parámetros que cambian (carga variable)	Adaptativo: STR o MRAC
Fricción, saturación	Compensación de fricción, anti-windup, SMC

Código del capítulo

Los archivos completos vienen en el paquete ZIP del capítulo (carpeta `codigo/`).

lqr_lqg.py — LQR discreto por iteración de Riccati y efecto del peso del esfuerzo. *(extracto: primeras 60 de 111 líneas)*

```
"""
lqr_lqg.py
Regulador lineal cuadrático (LQR) discreto para un eje con doble integrador
(masa empujada por un motor lineal) y su combinación con Kalman (LQG)
(capítulo 05).

  x = [posición, velocidad],  Phi = [[1, Ts], [0, 1]],  Gamma = [[Ts²/2], [Ts]]
  Costo J = Σ (x^T·Q·x + R·u²)
La ganancia K se obtiene iterando la ecuación de Riccati discreta hasta que
converge. Se comparan tres valores de R: más caro el actuador, más suave el
control y más lenta la respuesta.

Uso:  python lqr_lqg.py
"""
import cmath
import math

def mat(a, b):
    """Producto de matrices (listas de listas)."""
    return [[sum(a[i][k] * b[k][j] for k in range(len(b))) for j in range(len(b[0]))] for i in range(len(a))]


def suma(a, b, s=1.0):
    return [[x + s * y for x, y in zip(fa, fb)] for fa, fb in zip(a, b)]


def esc(a, k):
    return [[k * x for x in f] for f in a]


def tr(a):
    return [list(f) for f in zip(*a)]


def eye(n):
    return [[1.0 if i == j else 0.0 for j in range(n)] for i in range(n)]


def inv(a):
    n = len(a)
    m = [f[:] + e for f, e in zip(a, eye(n))]
    for c in range(n):
```

```
        p = max(range(c, n), key=lambda f: abs(m[f][c]))
        m[c], m[p] = m[p], m[c]
        piv = m[c][c]
        m[c] = [x / piv for x in m[c]]
        for f in range(n):
            if f != c:
                k = m[f][c]
                m[f] = [x - k * y for x, y in zip(m[f], m[c])]
    return [f[n:] for f in m]


def expm(a, t=1.0):
    """e^(A·t) por escalado y elevación al cuadrado con serie de Taylor."""
    n, s = len(a), 0
    norma = max(sum(abs(x) for x in f) for f in a) * abs(t)
    while norma > 0.5:
        norma, s = norma / 2, s + 1
    at = esc(a, t / 2 ** s)
```

no_lineal_adaptativo.py — PD frente a modos deslizantes con fricción y MRAC con ganancia desconocida.

Errores frecuentes

- Elegir Q y R sin normalizar las unidades de los estados.
- Aplicar LQR cuando el actuador satura constantemente.
- Usar SMC sin capa límite en un actuador real.
- Adaptar sin excitación suficiente: los parámetros derivan.

Autoevaluación

1. ¿Qué pasa con la respuesta LQR si se aumenta R?
2. ¿Qué establece el principio de separación?
3. ¿Qué es el chattering?
4. ¿Qué hace un MRAC?

Respuestas. 1. Se usa menos esfuerzo y la respuesta es más lenta. **2.** Que el regulador y el estimador se pueden diseñar por separado. **3.** La conmutación muy rápida de la acción de control en modos deslizantes. **4.** Ajusta sus parámetros para que la planta siga a un modelo de referencia.