MANUAL COMPLETO DEL MINICURSO · CURSOSMECATRONICA.COM

Sistemas y Control en Mecatrónica

Transformada Z, espacio de estados, lugar de las raíces, Nyquist, Kalman, DSP, LQR, control no lineal, adaptativo e inteligente.

Cómo usar este manual. Cada capítulo corresponde a un tramo del video (se indica el minuto). Repasa las ideas clave, rehace el ejemplo y responde la autoevaluación. Los programas en Python solo usan la biblioteca estándar (traen sus propias funciones de matrices); los de Arduino muestran la implementación con periodo de muestreo fijo y el archivo .st, la de un PLC.

Contenido

N.º	Capítulo	Video
00	**Introducción al curso** Qué cubre el curso y cómo usar el material.	min 0:00
01	**Modelado, señales, Laplace, transformada Z y espacio de estados** Rol del control, muestreo y ZOH, propiedades de sistemas, G(s), G(z), discretización y espacio de estados.	min 1:01
02	**Análisis en espacio de estados y respuesta dinámica** e^(At), formas canónicas, controlabilidad, observabilidad, Lyapunov, especificaciones y error estacionario.	min 11:23
03	**Lugar de las raíces y métodos de respuesta en frecuencia** Root locus continuo y discreto, adelanto y atraso, Bode, Nyquist, márgenes, S y T.	min 21:19
04	**Filtro de Kalman, observadores y procesamiento digital** Luenberger, Kalman, Riccati, fusión, EKF/UKF, aliasing, FIR/IIR, FFT y DSP embebido.	min 31:38
05	**Control óptimo, robusto, no lineal y adaptativo** H2, LQR, LQG, H∞, no linealidades, Lyapunov, backstepping, linealización exacta, SMC y MRAC.	min 42:06
06	**Control inteligente, eje electrohidráulico y optimización** Lógica difusa, redes neuronales, ANFIS, modelo y control de un eje hidráulico, co-diseño y Pareto.	min 52:42
07	**Resumen y cierre** Síntesis del curso.	min 1:02:59

Curso gratuito en video: **cursosmecatronica.com/cursos/sistemas-y-control-en-mecatronica/**

MANUAL DE ESTUDIO · SISTEMAS Y CONTROL EN MECATRÓNICA

Introducción al curso

Introducción · En el video: minuto 0:00 · Duración 01:01

Objetivo. Conocer el recorrido del curso, desde el modelo hasta el controlador implementado.

Ideas clave

El control en mecatrónica

El controlador no se agrega al final: se diseña junto con la mecánica y la electrónica, porque compensa flexibilidades, fricción y perturbaciones que la mecánica sola no puede eliminar.



Estructura del curso.

Cómo estudiar

Cada capítulo sigue un video. Los programas en Python calculan y simulan todo con la biblioteca estándar (incluidas las matrices); los de Arduino y el PLC muestran cómo se implementa en tiempo real. Conviene tener a mano el curso de Modelado de Sistemas.

Autoevaluación

1. ¿Por qué se habla de co-diseño?
2. ¿Qué herramienta del curso estima estados que no se miden?

Respuestas. 1. Porque mecánica, electrónica y control se diseñan juntos y cada uno afecta a los otros. **2.** El filtro de Kalman (y los observadores).

MANUAL DE ESTUDIO · SISTEMAS Y CONTROL EN MECATRÓNICA

Modelado, señales, Laplace, transformada Z y espacio de estados

Capítulo 01 · En el video: **minuto 1:01** · Duración 10:22

Objetivo. Representar un sistema mecatrónico en tiempo continuo y discreto, por función de transferencia y en espacio de estados.

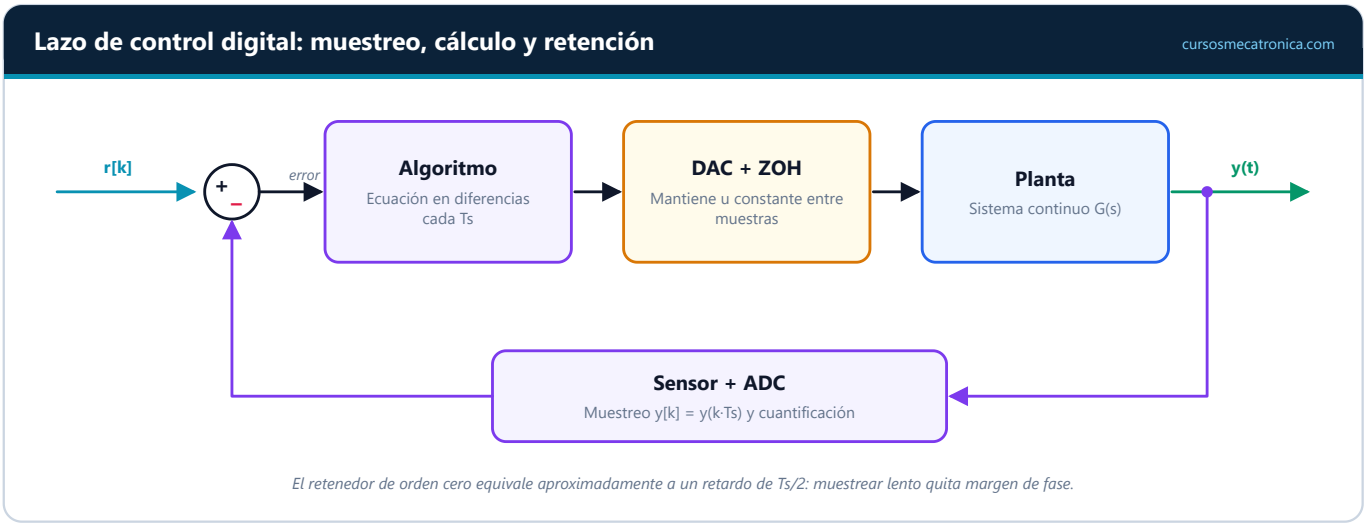
Ideas clave

Rol del control y del modelo

El control aporta precisión, ancho de banda y estabilidad; el modelo, orientado al control, captura solo la dinámica relevante y se valida con mediciones (ciclo en V, gemelos digitales).

Señales muestreadas

El ADC toma $x[k] = x(k \cdot T_s)$ y cuantifica con paso q; el DAC con retenedor de orden cero (ZOH) mantiene la salida constante entre muestras. El ruido de cuantificación es $q/\sqrt{12}$ rms.



Partes de un sistema de control muestreado.

Propiedades

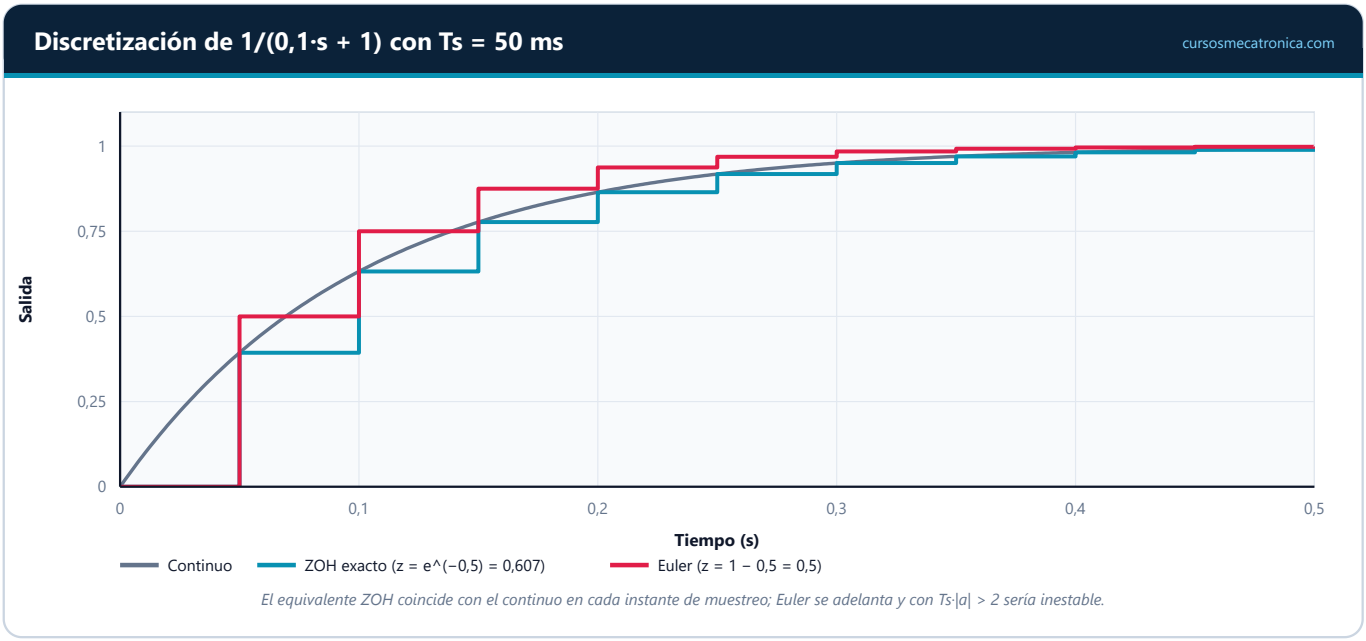
Linealidad, invarianza en el tiempo, causalidad y estabilidad BIBO (toda entrada acotada da salida acotada). La salida es la convolución de la entrada con la respuesta al impulso.

Laplace y transformada Z

G(s) describe sistemas continuos; G(z), con el retardo z^{-1}, describe ecuaciones en diferencias. Un sistema discreto es estable si sus polos están dentro del círculo unitario.

Discretización

Exacta con ZOH ($z = e^{sT_s}$ para los polos), Euler (simple, puede inestabilizar) o Tustin $s = (2/T_s)\cdot(z-1)/(z+1)$, que conserva la estabilidad; con prewarping se respeta una frecuencia crítica.



Equivalentes discretos de un primer orden.

Espacio de estados

$\dot{x} = Ax + Bu$, $y = Cx + Du$; en discreto $x[k+1] = \Phi x[k] + \Gamma u[k]$ con $\Phi = e^{AT_s}$. Representa naturalmente sistemas con varias entradas y salidas, y $G(s) = C(sI - A)^{-1}B + D$.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Muestreo	$x[k] = x(k \cdot T_s)$	$f_s = 1/T_s$.
Ruido de cuantificación	$\sigma_q = q/\sqrt{12}$, $q = \text{rango}/2^n$	0,23 mV en un ADC de 12 bits y 3,3 V.
Mapeo de polos	$z = e^{s \cdot T_s}$	Semiplano izquierdo → interior del círculo.
Tustin	$s \approx (2/T_s) \cdot (z-1)/(z+1)$	Bilineal.
Estado discreto	$\Phi = e^{A \cdot T_s}$, $\Gamma = \int_0^{T_s} e^{A\tau} d\tau \cdot B$	Equivalente ZOH exacto.
Función de transferencia	$G(s) = C(sI - A)^{-1}B + D$	Del estado a la entrada-salida.

Ejemplo resuelto: equivalente discreto de un primer orden

1. $G(s) = 1/(0{,}1s + 1)$, polo en -10; $T_s = 50$ ms.
2. ZOH: $z = e^{-10 \cdot 0{,}05} = 0{,}607$; $y[k+1] = 0{,}607 \cdot y[k] + 0{,}393 \cdot u[k]$.
3. Euler: $z = 1 - 10 \cdot 0{,}05 = 0{,}5$. Tustin: $z = (1 - 0{,}25)/(1 + 0{,}25) = 0{,}6$.
4. Con $T_s = 0{,}25$ s Euler daría $z = -1{,}5$: inestable aunque la planta sea estable.

discretizacion.py calcula Φ y Γ de un motor con posición y velocidad por la exponencial de matrices, y compara ZOH, Euler y Tustin.

Métodos de discretización

Método	Sustitución	Ventaja	Cuidado
ZOH exacto	$\Phi = e^{AT_s}$	Coincide en los instantes de muestreo	Requiere calcular la exponencial
Euler hacia adelante	$s = (z - 1)/T_s$	Muy simple	Puede volver inestable un sistema estable
Tustin (bilineal)	$s = (2/T_s)(z - 1)/(z + 1)$	Conserva la estabilidad	Distorsiona las frecuencias altas (usar prewarping)

Código del capítulo

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

discretizacion.py — equivalente ZOH por exponencial de matrices, Euler, Tustin, mapeo de polos y cuantificación. *(extracto: primeras 60 de 108 líneas)*

```
"""
discretizacion.py
De un modelo continuo a uno digital (capítulo 01).

Planta: motor con carga, G(s) = 1 / (s·(0,1·s + 1)) (posición), en espacio de
estados con x = [posición, velocidad]:
    A = [[0, 1], [0, -10]],  B = [[0], [10]],  C = [[1, 0]]

  1. Equivalente discreto exacto con retenedor de orden cero (ZOH):
       Phi = e^(A·Ts),  Gamma = ∫ e^(A·t) dt · B   (se obtiene con una matriz aumentada)
  2. Comparación de la respuesta al escalón de velocidad con Euler y con Tustin.
  3. Polos: z = e^(s·Ts) y estabilidad dentro del círculo unitario.
  4. Ruido de cuantificación de un ADC de 12 bits: q/raíz(12).

Uso:  python discretizacion.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:
```

controlador_pi_discreto.ino — PI discretizado con Tustin, periodo fijo con micros() y anti-windup.

Errores frecuentes

- Ejecutar el controlador con delay() y un periodo que varía.
- Discretizar con Euler y un periodo grande.
- Olvidar que el ZOH y el cálculo agregan retardo al lazo.
- Confundir la estabilidad en s (semiplano izquierdo) con la de z (círculo unitario).

Autoevaluación

1. ¿Dónde deben estar los polos de un sistema discreto estable?
2. ¿Qué hace el retenedor de orden cero?
3. ¿Qué método de discretización conserva siempre la estabilidad?
4. ¿Cómo se obtiene G(s) desde el espacio de estados?

Respuestas. 1. Dentro del círculo unitario. **2.** Mantiene constante la salida del DAC entre dos muestras. **3.** Tustin (bilineal) y el ZOH exacto. **4.** $G(s) = C(sI - A)^{-1}B + D$.

MANUAL DE ESTUDIO · SISTEMAS Y CONTROL EN MECATRÓNICA

Análisis en espacio de estados y respuesta dinámica

Capítulo 02 · En el video: minuto 11:23 · Duración 09:56

Objetivo. Analizar la estructura, la estabilidad y el desempeño de un sistema a partir de su modelo de estados.

Ideas clave

Solución y modos

$x(t) = e^{At}x(0) + \int e^{A(t-\tau)}Bu(\tau)d\tau$: respuesta libre más forzada. Los autovalores de A son los polos y cada uno, con su autovector, define un modo natural.

Formas canónicas

Un cambio de variables $x = Pz$ no cambia los autovalores ni la función de transferencia. Las formas controlable, observable y modal facilitan el diseño y la interpretación.

Controlabilidad y observabilidad

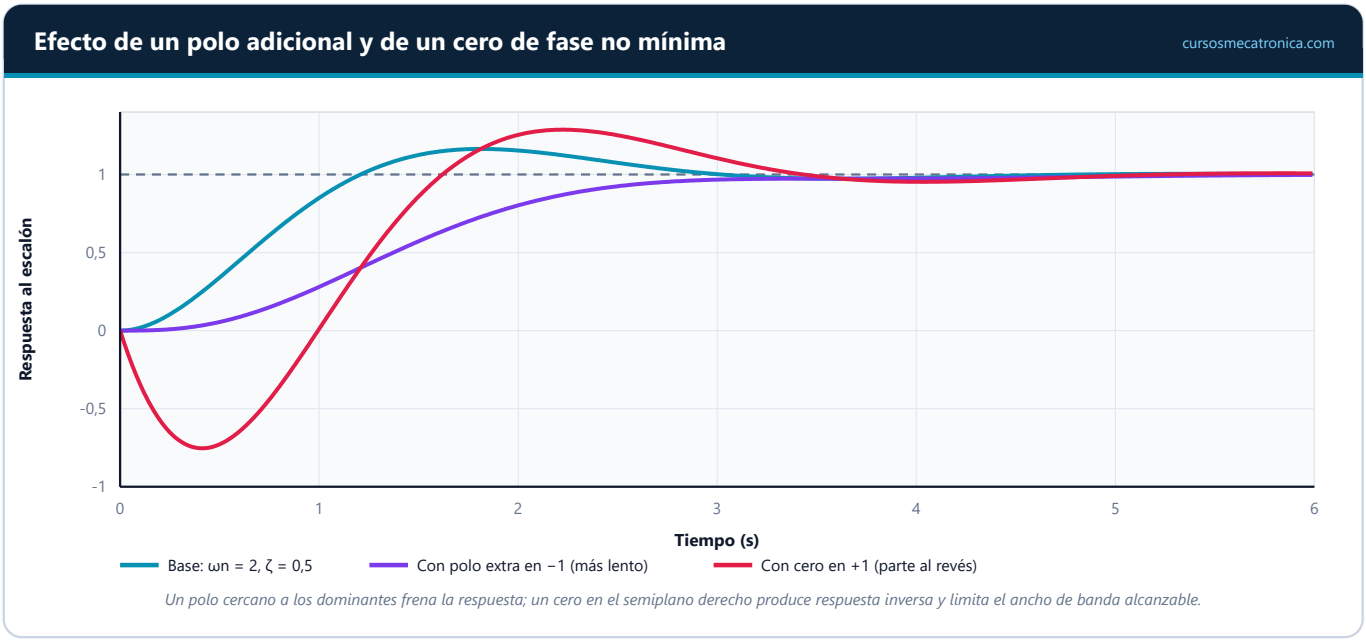
Controlable si la matriz $[B\ AB\ \dots\ A^{n-1}B]$ tiene rango completo: el actuador puede llevar el estado a cualquier valor. Observable si $[C; CA; \dots]$ tiene rango completo: los sensores permiten reconstruir todo el estado. Indican dónde poner actuadores y sensores.

Estabilidad de Lyapunov

Interna: todos los autovalores con parte real negativa. Equivale a que exista $P > 0$ que resuelve $A^TP + PA = -Q$. La estabilidad BIBO puede ocultar un modo inestable no controlable o no observable.

Desempeño

Las especificaciones de M_p, t_p y t_s definen regiones del plano s. Un polo adicional cercano frena la respuesta; un cero en el semiplano derecho produce respuesta inversa.



Por qué la aproximación de segundo orden necesita polos dominantes claros.

Error estacionario

Depende del tipo del sistema (integradores en el lazo): tipo 0 deja error ante un escalón, tipo 1 lo elimina y deja error ante una rampa. Principio del modelo interno: para seguir o rechazar una señal sin error, el lazo debe contener su modelo.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Solución	$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$	Libre + forzada.
Controlabilidad	$\text{rango}[B\ AB\ \dots\ A^{n-1}B] = n$	Criterio de Kalman.
Observabilidad	$\text{rango}[C;\ CA;\ \dots;\ CA^{n-1}] = n$	Dual de la controlabilidad.
Lyapunov	$A^TP + PA = -Q,\ P > 0$	Estabilidad interna.
Error ante escalón	$e_{ss} = 1/(1 + K_p),\ K_p = \lim G(s)\ \text{con}\ s \to 0$	Tipo 0.
Error ante rampa	$e_{ss} = 1/K_v,\ K_v = \lim s{\cdot}G(s)$	Tipo 1.

Ejemplo resuelto: masa-resorte-amortiguador con motor lineal

1. m = 1 kg, b = 2 N·s/m, k = 25 N/m: $A = [[0, 1], [-25, -2]]$, $B = [0; 1]$.
2. Autovalores $-1 \pm 4{,}9j$: ω_n = 5 rad/s, ζ = 0,2.
3. $\det[B\ AB] = -1 \neq 0$: controlable. Con un sensor de posición, $\det[C;\ CA] = 1$: observable.
4. Lyapunov con Q = I: $P = [[6{,}54;\ 0{,}02], [0{,}02;\ 0{,}26]]$, definida positiva → estable.
5. Lazo de posición sin integrador: error ante escalón del 96 %; con un integrador, cero.

analisis_espacio_estados.py realiza estos cálculos y verifica que $A^TP + PA = -I$.

Tipo de sistema y error estacionario

Tipo	Escalón	Rampa	Parábola
0	$1/(1 + K_p)$	∞	∞
1	0	$1/K_v$	∞
2	0	0	$1/K_a$

Código del capítulo

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

analisis_espacio_estados.py — autovalores, controlabilidad, observabilidad, ecuación de Lyapunov y error estacionario. *(extracto: primeras 70 de 121 líneas)*

```
"""
analisis_espacio_estados.py
Propiedades de un modelo en espacio de estados (capítulo 02).

Sistema: masa-resorte-amortiguador con motor lineal, x = [posición, velocidad]
    A = [[0, 1], [-k/m, -b/m]],  B = [[0], [1/m]]
  1. Autovalores, wn y zeta.
  2. Controlabilidad: rango de [B, AB]; y qué pasa si la fuerza no actúa sobre la masa.
  3. Observabilidad con un sensor de posición y con uno de velocidad.
  4. Estabilidad de Lyapunov: se resuelve A^T·P + P·A = -I y se verifica P > 0.
  5. Error en estado estacionario y tipo de sistema.

Uso:  python analisis_espacio_estados.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)
    r, term = eye(n), eye(n)
    for k in range(1, 20):
        term = esc(mat(term, at), 1.0 / k)
        r = suma(r, term)
    for _ in range(s):
        r = mat(r, r)
    return r


def det2(a):
```

Errores frecuentes

- Concluir estabilidad interna a partir de la función de transferencia (puede haber cancelaciones).
- Poner un sensor que no permite observar el estado que se quiere controlar.
- Aproximar por segundo orden con polos que no son claramente dominantes.
- Agregar un integrador sin revisar los márgenes de estabilidad.

Autoevaluación

1. ¿Qué indican los autovalores de A?
2. ¿Qué significa que un sistema no sea observable?
3. ¿Qué error ante un escalón deja un lazo tipo 1?
4. ¿Qué efecto tiene un cero en el semiplano derecho?

Respuestas. 1. Los polos del sistema (sus modos naturales). **2.** Que con los sensores disponibles no se puede reconstruir todo el estado. **3.** Cero. **4.** Respuesta inversa al inicio y límite al ancho de banda.

MANUAL DE ESTUDIO · SISTEMAS Y CONTROL EN MECATRÓNICA

Lugar de las raíces y métodos de respuesta en frecuencia

Capítulo 03 · En el video: **minuto 21:19** · Duración 10:18

Objetivo. Diseñar compensadores con el lugar de las raíces y verificar la estabilidad relativa en frecuencia.

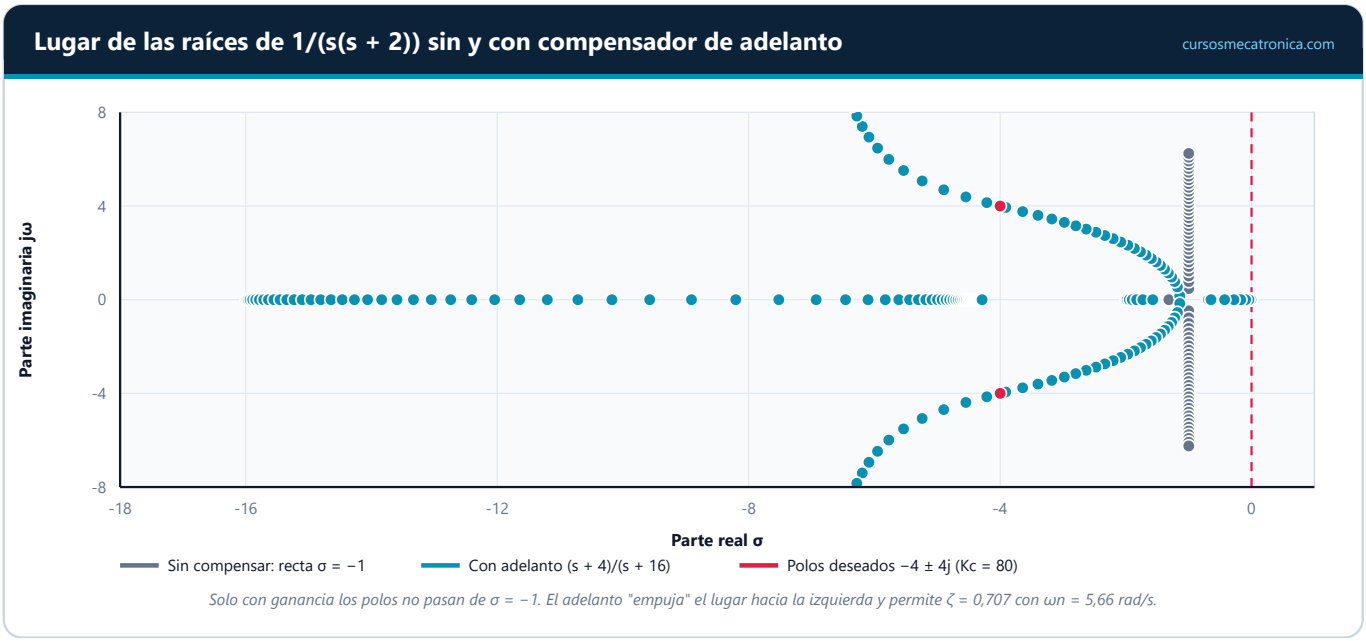
Ideas clave

Lugar de las raíces

Muestra cómo se mueven los polos de lazo cerrado (raíces de $1 + KG(s)H(s) = 0$) al variar K. Un punto pertenece al lugar si cumple la condición de fase (−180°); la de magnitud da el K. Las ramas parten de los polos, terminan en los ceros o en asíntotas, y cruzan el eje imaginario donde lo indica Routh.

Compensadores

El adelanto $(s + z)/(s + p)$ con z < p aporta fase positiva y acerca el lugar a la región deseada (más rápido y amortiguado). El atraso mejora el error estacionario sin mover mucho los polos. El PID combina ambos efectos. En discreto se diseña igual sobre el plano z.



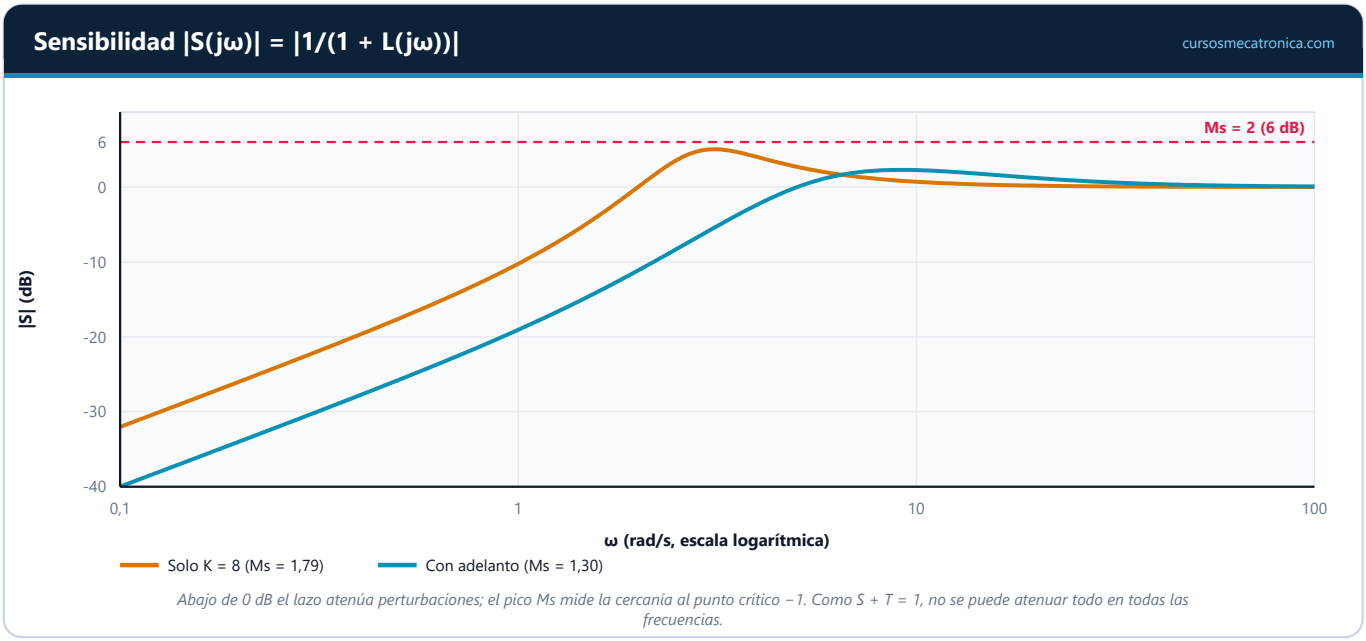
Diseño por lugar de las raíces.

Frecuencia

Bode muestra |G(jω)| y ∠G(jω); Nyquist dibuja L(jω) en el plano complejo y cuenta las vueltas alrededor de −1: $Z = N + P$ polos inestables de lazo cerrado. Sirve también con retardos, que en Bode solo restan fase.

Márgenes y sensibilidad

Margen de ganancia y de fase miden la distancia al punto crítico. La sensibilidad $S = 1/(1 + L)$ indica cuánto se atenúan las perturbaciones y $T = L/(1 + L)$ cuánto ruido de medición pasa; como S + T = 1, el diseño busca S pequeña en baja frecuencia y T pequeña en alta. El pico M_s debe ser menor que 2.



Moldeado del lazo (loop shaping).

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Condición de fase	$\angle KG(s)H(s) = \pm 180°(2k+1)$	Define el lugar.
Condición de magnitud	$\lvert KG(s)H(s)\rvert = 1$	Da la ganancia en cada punto.
Centroide de asíntotas	$\sigma_a = (\Sigma \text{ polos} - \Sigma \text{ ceros})/(n-m)$	Ángulos ±180°(2k + 1)/(n − m).
Adelanto	$C(s) = K_c(s+z)/(s+p),\ z < p$	Aporta fase positiva.
Nyquist	$Z = N + P$	Z: polos inestables de lazo cerrado.
Sensibilidad	$S = 1/(1+L),\ T = L/(1+L),\ S+T = 1$	M_s = máx|S|.

Ejemplo resuelto: compensador de adelanto para un servo

1. Planta G = 1/(s(s + 2)); se desean polos en $s_d = -4 \pm 4j$ (ζ = 0,707, ω_n = 5,66 rad/s).
2. Fase de G en s_d: 108,4° → faltan $-180° - 108{,}4° = 71{,}6°$ (módulo 360°).
3. Cero en −4 (justo debajo, aporta 90°); el polo debe restar 18,4°: queda en −16.
4. Magnitud: $K_c = 80$. Polos de lazo cerrado: −4 ± 4j y −10.
5. MF pasa de 38,7° (solo ganancia K = 8) a 55°, y M_s de 1,79 a 1,30.

C(s) = 80·(s + 4)/(s + 16). lugar_raices_y_frecuencia.py repite el diseño y calcula márgenes y sensibilidad.

Compensadores clásicos

Compensador	Efecto principal	Cuándo usarlo
Ganancia	Mueve los polos sobre el lugar	Si el lugar ya pasa por la región deseada
Adelanto	Más fase, más rapidez y amortiguamiento	Respuesta lenta u oscilatoria
Atraso	Más ganancia en baja frecuencia	Error estacionario excesivo
Adelanto-atraso / PID	Ambos efectos	Especificaciones de transitorio y de régimen
Filtro notch	Atenúa una frecuencia	Resonancias mecánicas

Código del capítulo

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

lugar_raices_y_frecuencia.py — lugar de las raíces, diseño de adelanto por la condición de fase, MF y pico de sensibilidad. *(extracto: primeras 70 de 80 líneas)*

```
"""
lugar_raices_y_frecuencia.py
Lugar de las raíces, compensador de adelanto y sensibilidad (capítulo 03).

Planta: G(s) = 1 / (s·(s + 2))   (servo de posición).
  1. Puntos del lugar de las raíces al variar K y asíntotas.
  2. Diseño de un compensador de adelanto C(s) = Kc·(s + z)/(s + p) para
     llevar los polos dominantes a s = −4 ± 4j (zeta = 0,707, wn = 5,66 rad/s)
     por la condición de fase.
  3. Margen de fase y pico de sensibilidad Ms = max|1/(1 + L(jw))|.

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

def evaluar(c, s):
    r = 0
    for a in c:
        r = r * s + a
    return r


def raices(c, iteraciones=500):
    """Raíces de un polinomio (coeficientes de mayor a menor grado), Durand-Kerner."""
    c = [x / c[0] for x in c]
    n = len(c) - 1
    z = [(0.4 + 0.9j) ** k for k in range(n)]
    for _ in range(iteraciones):
        nuevo = []
        for i in range(n):
            den = 1
            for j in range(n):
                if i != j:
                    den *= z[i] - z[j]
            nuevo.append(z[i] - evaluar(c, z[i]) / den)
        z = nuevo
    return sorted(z, key=lambda p: (p.real, p.imag))


def fase_grados(z):
    return math.degrees(cmath.phase(z))
```

```
if __name__ == '__main__':
    print('1) Polos del lazo cerrado s² + 2s + K = 0')
    for K in [0.5, 1, 2, 8, 32]:
        p = raices([1, 2, K])
        z = -p[0].real / abs(p[0]) if abs(p[0].imag) > 1e-9 else 1.0
        print(f'   K = {K:5.1f}: {p[0].real:+.2f}{p[0].imag:+.2f}j, {p[1].real:+.2f}{p[1].imag:+.2f}j   zeta =
{z:.2f}')
    print('   Ruptura en s = −1 (K = 1); para K > 1 los polos suben por la recta σ = −1: solo con ganancia no
se acelera.')

    print('\n2) Compensador de adelanto para s_d = −4 + 4j')
    sd = complex(-4, 4)
    fase_planta = fase_grados(1 / (sd * (sd + 2)))
    falta = -180 - fase_planta
    falta = (falta + 180) % 360 - 180
    print(f'   fase de G(s_d) = {fase_planta:.1f}°, falta aportar {falta:.1f}°')
    z = 4.0                                  # cero bajo el polo deseado
    ang_cero = fase_grados(sd + z)
    ang_polo = ang_cero - falta
    p = -sd.real + sd.imag / math.tan(math.radians(ang_polo))
    Kc = 1 / abs((sd + z) / (sd + p) / (sd * (sd + 2)))
    print(f'   cero en −{z:.0f} (aporta {ang_cero:.1f}°), polo en −{p:.2f}, Kc = {Kc:.2f}')
    num = [Kc, Kc * z]
    den = [1, 2 + p, 2 * p, 0]
    car = [den[0], den[1], den[2] + num[0], den[3] + num[1]]
    polos = raices(car)
    print('   polos de lazo cerrado:', ', '.join(f'{x.real:+.2f}{x.imag:+.2f}j' for x in polos))
```

Errores frecuentes

- Subir la ganancia esperando más rapidez cuando el lugar no lo permite.
- Poner el polo del adelanto demasiado lejos: amplifica el ruido.
- Evaluar márgenes sin considerar el retardo del sistema digital.
- Buscar S pequeña en todas las frecuencias: es imposible.

Autoevaluación

1. ¿Dónde empiezan las ramas del lugar de las raíces?
2. ¿Qué aporta un compensador de adelanto?
3. En Nyquist, ¿qué significa Z = N + P?
4. ¿Qué valor de M_s se recomienda no superar?

Respuestas. 1. En los polos de lazo abierto (K = 0). **2.** Fase positiva: más rapidez y amortiguamiento. **3.** Polos inestables de lazo cerrado = vueltas alrededor de −1 + polos inestables de lazo abierto. **4.** 2 (6 dB).

MANUAL DE ESTUDIO · SISTEMAS Y CONTROL EN MECATRÓNICA

Filtro de Kalman, observadores y procesamiento digital

Capítulo 04 · En el video: **minuto 31:38** · Duración 10:29

Objetivo. Estimar estados no medidos y procesar señales de sensores en tiempo real.

Ideas clave

Observador de Luenberger

Copia el modelo y lo corrige con la medición: el error evoluciona como $e[k+1] = (\Phi - LC)e[k]$. Polos rápidos convergen rápido pero amplifican el ruido.

Filtro de Kalman

Elige la ganancia óptima a partir del ruido de proceso Q (cuánto se confía en el modelo) y del ruido de medición R (cuánto se confía en el sensor). Ciclo: predicción ($\hat{x}^- = \Phi\hat{x}$, $P^- = \Phi P\Phi^T + Q$) y actualización ($K = P^-H^T(HP^-H^T + R)^{-1}$, $\hat{x} = \hat{x}^- + K(z - H\hat{x}^-)$).

Velocidad desde un encoder de 400 cuentas/vuelta (Ts = 10 ms), detalle cursosmecatronica.com



La derivada salta entre múltiplos de 1,57 rad/s (una cuenta por periodo). Kalman combina el modelo de velocidad casi constante con las lecturas y reduce el error rms de 0,69 a 0,40 rad/s.

Estimación de estados con el filtro de Kalman.

Aplicaciones

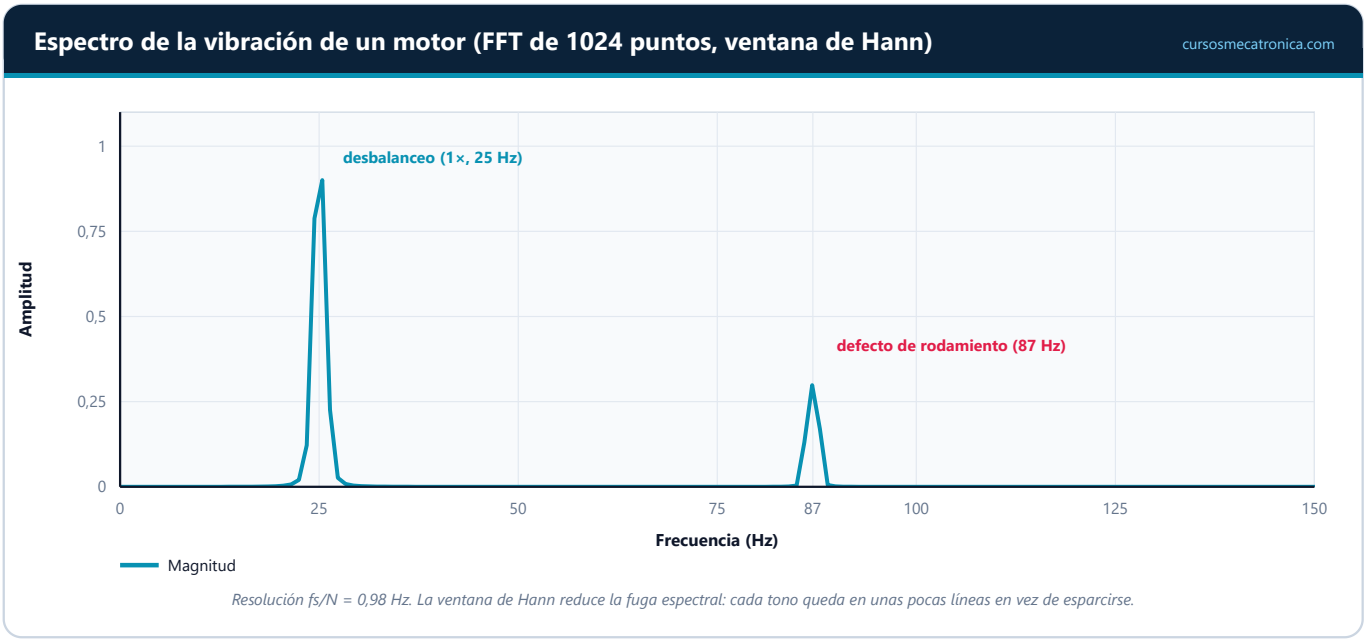
La ganancia converge a un valor fijo (ecuación de Riccati) que puede precalcularse. Se usa para fusionar IMU y encoder, estimar velocidad desde un encoder y, con un estado aumentado, estimar la perturbación de carga. EKF linealiza con jacobianos; UKF propaga puntos sigma y maneja mejor las no linealidades fuertes.

Muestreo y aliasing

Con $f_s > 2f_{máx}$ (Nyquist) la señal se puede reconstruir; si no, las frecuencias altas se pliegan y aparecen como bajas. Por eso se pone un filtro analógico antialias antes del ADC o se sobremuestrea.

Filtros digitales y FFT

FIR: fase lineal exacta pero más retardo. IIR (biquads Butterworth, notch): menos cálculo y retardo, preferidos dentro de los lazos. La FFT calcula el espectro en $N \cdot \log_2 N$ operaciones; la ventana de Hann reduce la fuga espectral. Se usa para diagnosticar rodamientos y motores.



Diagnóstico por análisis espectral.

DSP embebido

Operaciones MAC, aritmética de punto fijo Q15/Q31 con saturación, y DMA en ping-pong para adquirir sin ocupar la CPU. CMSIS-DSP ofrece filtros y FFT optimizados para Cortex-M.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Predicción	$\hat{x}^- = \Phi\hat{x} + \Gamma u,\ P^- = \Phi P \Phi^T + Q$	A priori.
Ganancia de Kalman	$K = P^- H^T (H P^- H^T + R)^{-1}$	Óptima con ruido gaussiano.
Actualización	$\hat{x} = \hat{x}^- + K(z - H\hat{x}^-),\ P = (I - KH)P^-$	A posteriori.
Nyquist-Shannon	$f_s > 2 \cdot f_{máx}$	Si no, aliasing.
Resolución de la FFT	$\Delta f = f_s / N$	0,98 Hz con 1024 puntos a 1 kHz.
Biquad (DF2T)	$y = b_0 x + z_1;\ z_1 = b_1 x - a_1 y + z_2;\ z_2 = b_2 x - a_2 y$	Dos estados por sección.

Ejemplo resuelto: velocidad desde un encoder de baja resolución

1. Encoder de 400 cuentas/vuelta: resolución 15,7 mrad; T_s = 10 ms.
2. Derivada directa: cada cuenta equivale a $0{,}0157/0{,}01 = 1{,}57$ rad/s: la velocidad salta en escalones.
3. Modelo de velocidad casi constante con Q de aceleración aleatoria y $R = \text{res}^2/12$.
4. La ganancia converge a $K = [0{,}83;\ 64{,}6]$.
5. Error rms de velocidad: 0,69 rad/s por derivada y 0,40 rad/s con Kalman.

kalman_posicion_velocidad.py implementa el filtro con matrices; dsp_filtros_fft.py diseña el biquad y la FFT del capítulo.

Filtros digitales

Tipo	Fase	Costo	Uso típico
FIR	Lineal (sin distorsión)	Muchos coeficientes	Medición, audio, análisis fuera de línea
IIR Butterworth	No lineal, poco retardo	5 coeficientes por biquad	Filtrar sensores dentro del lazo
Notch	Afecta cerca de f_0	1 biquad	Eliminar una resonancia mecánica o 50 Hz
Media móvil	Lineal	Muy bajo	Suavizado simple, rechazo de una frecuencia

Código del capítulo

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

kalman_posicion_velocidad.py — filtro de Kalman de posición y velocidad con un encoder cuantificado. *(extracto: primeras 60 de 118 líneas)*

```
"""
kalman_posicion_velocidad.py
Filtro de Kalman para un eje con encoder de baja resolución (capítulo 04).

Modelo de velocidad casi constante con Ts = 10 ms:
    x = [posición, velocidad],  Phi = [[1, Ts], [0, 1]],  H = [[1, 0]]
    Q = q·[[Ts³/3, Ts²/2], [Ts²/2, Ts]]   (aceleración aleatoria de densidad q)
    R = (resolución²)/12                  (cuantificación del encoder)

Compara la velocidad obtenida por derivada directa del encoder con la del
filtro de Kalman, calcula la ganancia estacionaria iterando la ecuación de
Riccati y la compara con un observador de Luenberger con polos fijos.

Uso:  python kalman_posicion_velocidad.py
"""
import math
import random

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
```

dsp_filtros_fft.py — aliasing, biquad Butterworth, notch y FFT con ventana de Hann.

filtro_biquad.ino — biquad IIR en Arduino a 1 kHz para filtrar un sensor.

Errores frecuentes

- Ajustar Q y R al azar: R se mide con el sensor quieto; Q se ajusta según la dinámica esperada.
- Muestrear sin filtro antialias.
- Usar un FIR largo dentro de un lazo rápido: su retardo quita margen de fase.
- Aplicar la FFT sin ventana a una señal que no completa periodos enteros.

Autoevaluación

1. ¿Qué representa R en un filtro de Kalman?
2. ¿Qué frecuencia mínima de muestreo exige una señal de 200 Hz?
3. ¿Por qué se prefieren los IIR dentro de los lazos?
4. ¿Qué resolución tiene una FFT de 2048 puntos a 10 kHz?

Respuestas. 1. La varianza del ruido de medición. **2.** Más de 400 Hz. **3.** Porque tienen menos retardo y cálculo que un FIR equivalente. **4.** Unos 4,9 Hz.

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

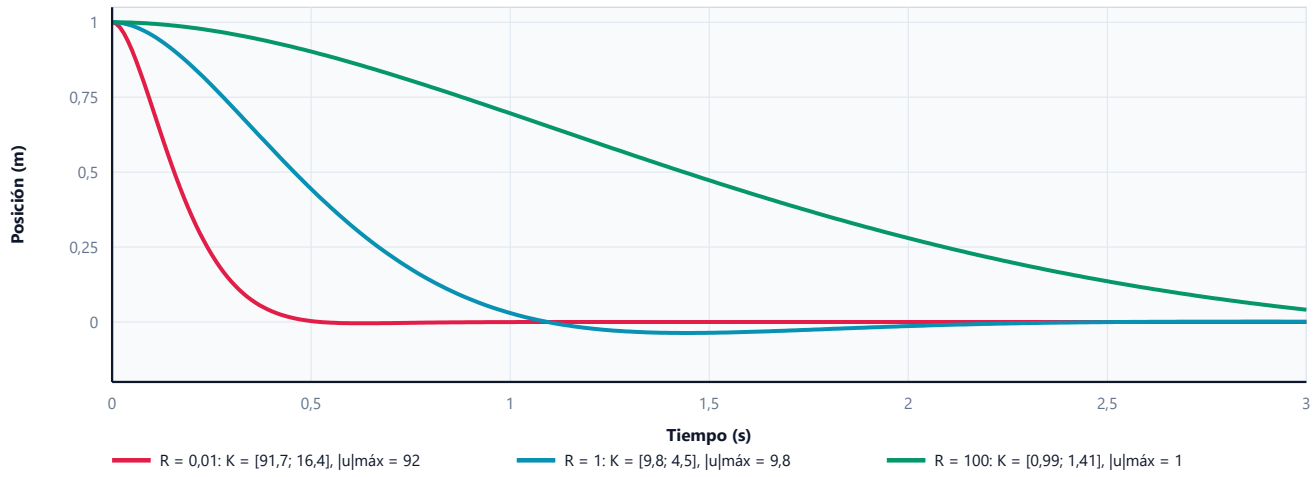
Objetivo. Conocer las técnicas de control moderno y elegir la adecuada según el problema.

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

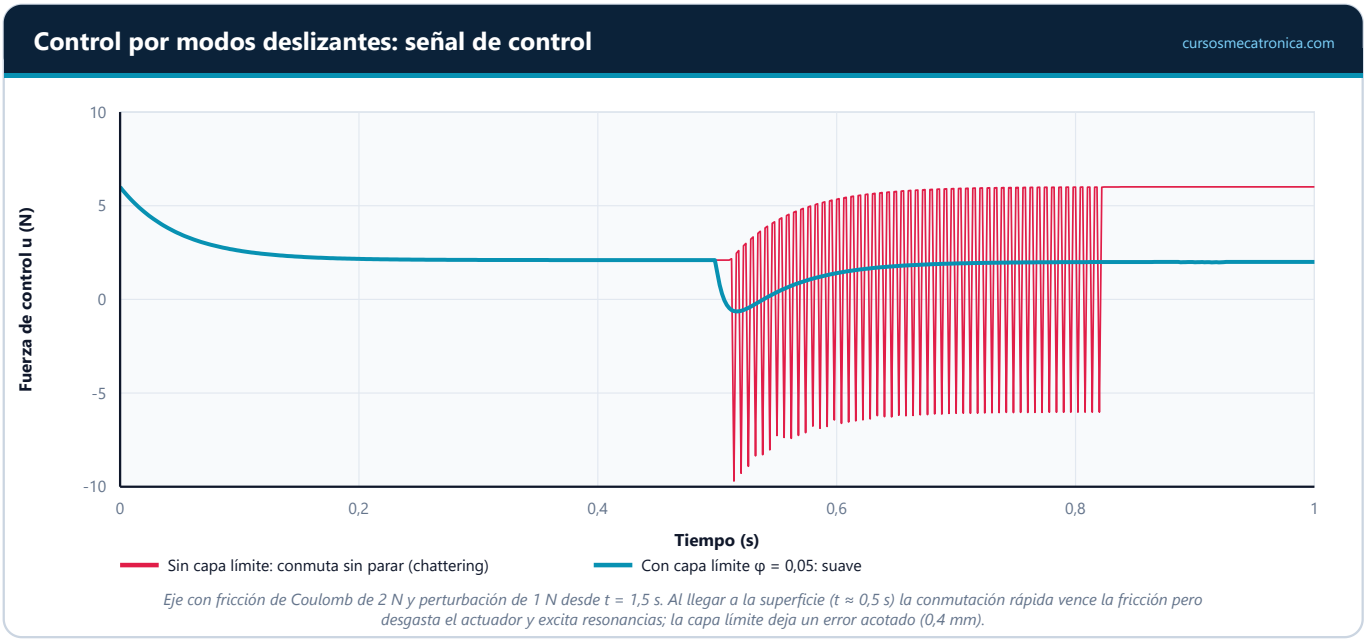
Fricción de Stribeck y stick-slip, zonas muertas, saturación, backlash e histéresis limitan al control lineal: producen ciclos límite, errores estacionarios y windup.

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	$J = \Sigma\,(x^{T}Qx + u^{T}Ru)$	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.

MANUAL DE ESTUDIO · SISTEMAS Y CONTROL EN MECATRÓNICA

Control inteligente, eje electrohidráulico y optimización

Capítulo 06 · En el video: **minuto 52:42** · Duración 10:17

Objetivo. Aplicar técnicas inteligentes y de optimización al control y al diseño de un eje electrohidráulico.

Ideas clave

Lógica difusa

Variables lingüísticas con funciones de pertenencia, reglas SI–ENTONCES e inferencia Mamdani o Takagi-Sugeno. Un PD difuso tiene una superficie de control no lineal; la auto-sintonía difusa ajusta Kp, Ki y Kd según el punto de operación.

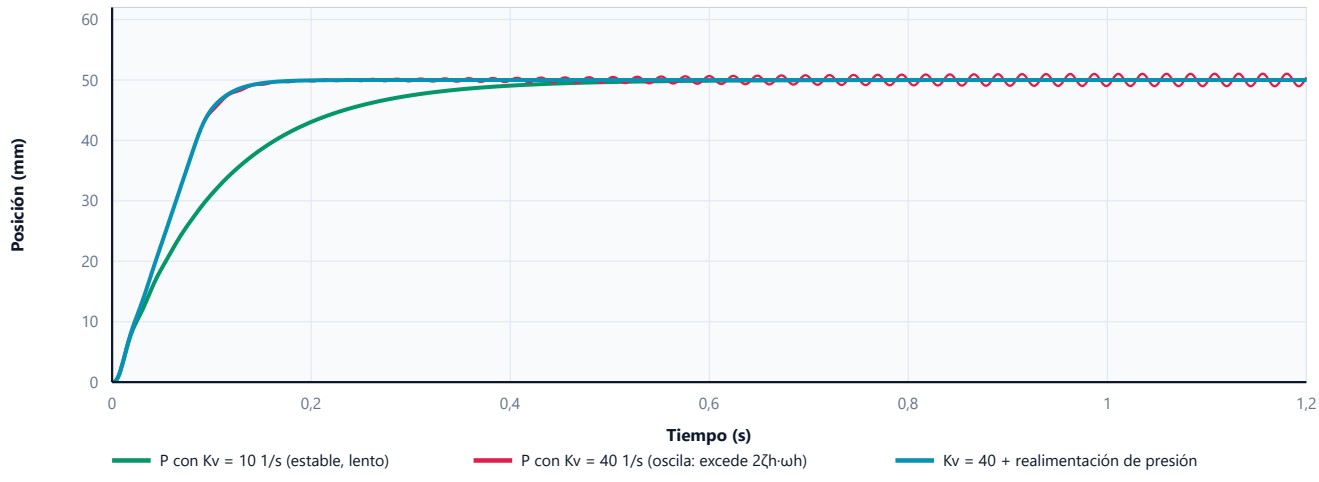
Redes neuronales y ANFIS

Un perceptrón multicapa o una red RBF aproxima cualquier función continua: sirven para identificar plantas (modelos NARX), control por modelo inverso y compensación en línea. ANFIS une la interpretabilidad difusa con el aprendizaje: cinco capas equivalentes a un Takagi-Sugeno.

Eje electrohidráulico

Servoválvula (caudal por orificio, raíz de la diferencia de presión), cilindro y continuidad con compresibilidad del aceite (β_e). La rigidez del aceite y la masa forman una resonancia $\omega_h = \sqrt{(4\beta_e A^2/(V \cdot m))}$ con muy poco amortiguamiento ($\zeta_h \approx 0{,}05–0{,}15$).

Eje electrohidráulico: escalón de 50 mm (ωh = 41 Hz, ζh = 0,055) cursosmecatronica.com



La resonancia hidráulica casi no tiene amortiguamiento: sin ayuda, el lazo de posición debe ser lento. Restar la presión de carga agrega amortiguamiento activo.

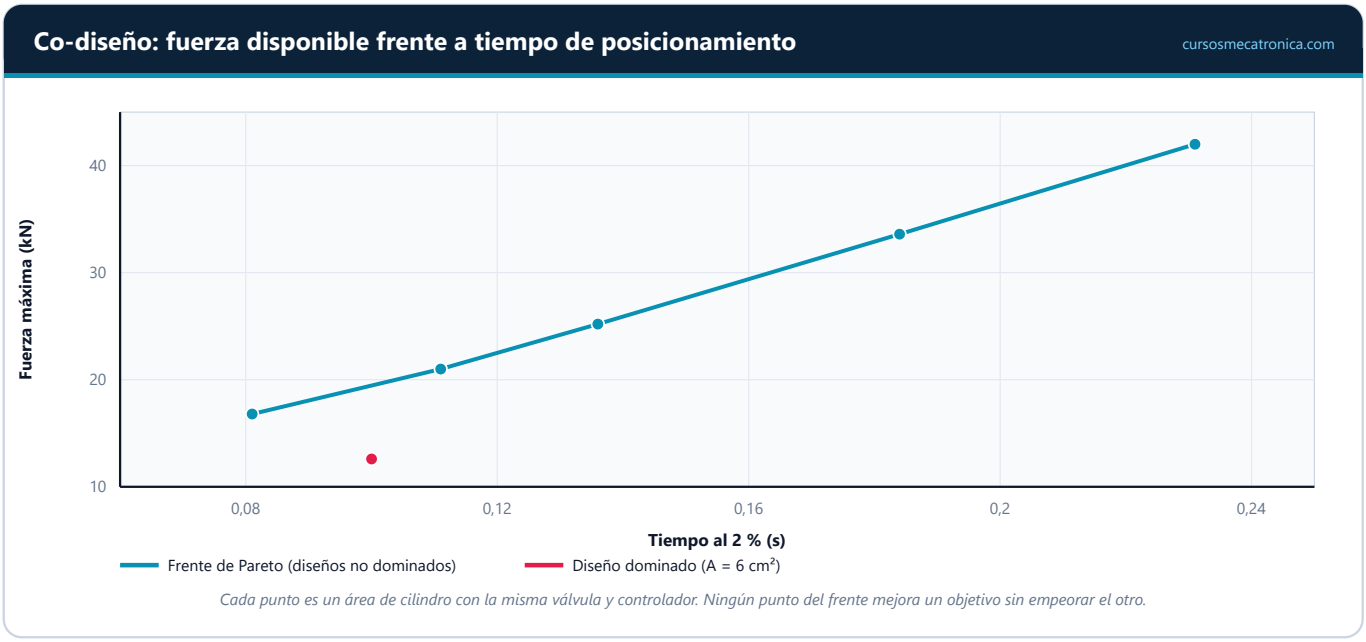
Respuestas simuladas con eje_electrohidraulico.py.

Control del eje

Con un P de posición la ganancia de velocidad debe ser menor que $2\zeta_h\omega_h$. Restar la presión de carga agrega amortiguamiento activo; además se compensa la zona muerta del carrete y se pueden usar backstepping, SMC o control adaptativo.

Optimización y co-diseño

Se buscan las variables de diseño p (mecánicas, eléctricas y del control) que minimizan objetivos J(p) con restricciones g(p) ≤ 0. Con varios objetivos en conflicto (masa, energía, ancho de banda) se obtiene un frente de Pareto. Métodos: SQP, algoritmos genéticos (NSGA-II) y enjambre de partículas (PSO).



Optimización multiobjetivo del eje.

Fórmulas y relaciones

Relación	Expresión	Cómo leerla
Caudal de la servoválvula	$Q_L = K_q \cdot x_v \cdot \sqrt{((P_s - \mathrm{sgn}(x_v) P_L)/P_s)}$	No lineal.
Continuidad	$(V/4\beta_e) \cdot \dot{P}_L = Q_L - A \cdot v - C_{tl} \cdot P_L$	Compresibilidad y fugas.
Frecuencia hidráulica	$\omega_h = \sqrt{(4\beta_e A^2/(V \cdot m))}$	41 Hz en el ejemplo.
Límite del lazo P	$K_v = K_p K_q / A < 2\zeta_h \omega_h$	Estabilidad sin amortiguamiento activo.
Takagi-Sugeno	$u = \Sigma\, w_i \cdot u_i \,/\, \Sigma\, w_i$	Promedio ponderado de las reglas.
Dominancia de Pareto	a domina a b si es mejor o igual en todo y mejor en algo	Define el frente.

Ejemplo resuelto: límite de ganancia de un eje hidráulico

1. A = 12 cm², m = 200 kg, V = 6·10⁻⁴ m³, β_e = 1,4 GPa: $\omega_h = 259$ rad/s (41 Hz), $\zeta_h \approx 0{,}055$.
2. Límite: $K_v < 2 \cdot 0{,}055 \cdot 259 = 28{,}7$ 1/s.
3. K_v = 10: estable, se asienta en 0,39 s. K_v = 40: oscila y no se asienta.
4. K_v = 40 restando la presión de carga: estable y se asienta en 0,14 s.

eje_electrohidraulico.py simula el modelo no lineal, compara los controladores y obtiene el frente de Pareto fuerza–rapidez; eje_hidraulico_plc.st muestra la implementación en un PLC con supervisión.

Técnicas inteligentes

Técnica	Fortaleza	Limitación
Lógica difusa	Usa el conocimiento del operador; no lineal y legible	Ajuste manual de reglas
Red neuronal (MLP, RBF)	Aprende de datos; aproxima cualquier función	Caja negra; necesita datos
ANFIS	Aprende y sigue siendo interpretable	Crece rápido con el número de entradas
Algoritmo genético / PSO	Optimiza sin derivadas, problemas no convexos	Muchas evaluaciones del modelo

Código del capítulo

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

eje_electrohidraulico.py — modelo no lineal de servoválvula y cilindro, amortiguamiento por presión y frente de Pareto. *(extracto: primeras 60 de 77 líneas)*

```
"""
eje_electrohidraulico.py
Eje electrohidráulico con servoválvula y cilindro (capítulo 06).

Modelo no lineal:
    caudal:      QL = Kq·xv·raíz((Ps − sign(xv)·PL)/Ps)
    presión:     (V/(4·beta))·PL' = QL − A·v − Ctl·PL
    movimiento:  m·v' = A·PL − b·v
Se calcula la frecuencia natural hidráulica wh = raíz(4·beta·A²/(V·m)) y su
amortiguamiento, y se comparan dos controladores de posición:
  a) proporcional en posición: con zeta_h ≈ 0,1 la ganancia debe ser baja;
  b) proporcional + realimentación de presión de carga (amortiguamiento activo).
Al final se hace una optimización multiobjetivo sencilla: barrido del área del
cilindro y búsqueda del frente de Pareto (tiempo de respuesta frente a caudal).

Uso:  python eje_electrohidraulico.py
"""
import math

PS, BETA = 210e5, 1.4e9          # Pa
A, M, B = 1.2e-3, 200.0, 2000.0  # m², kg, N·s/m
V = 2 * A * 0.25                 # volumen total para una carrera de 0,5 m
KQ, CTL = 0.6, 2e-12              # (m³/s) por metro de carrete (≈ 36 L/min con 1 mm), fuga (m³/s)/Pa
```

```
def simular(kp, kpl=0.0, a=A, t_fin=1.0, dt=2e-5, ref=0.05):
    x = v = pl = 0.0
    t, osc_max, t98 = 0.0, 0.0, None
    vol = 2 * a * 0.25
    q_max = 0.0
    while t < t_fin:
        xv = kp * (ref - x) - kpl * pl
        xv = max(-1e-3, min(1e-3, xv))           # carrera máxima del carrete
        signo = 1 if xv >= 0 else -1
        ql = KQ * xv * math.sqrt(max(PS - signo * pl, 0) / PS)
        dpl = (4 * BETA / vol) * (ql - a * v - CTL * pl)
        dv = (a * pl - B * v) / M
        pl += dpl * dt
        v += dv * dt
        x += v * dt
        t += dt
        q_max = max(q_max, abs(ql))
        osc_max = max(osc_max, x - ref)
        if t98 is None and abs(x - ref) < 0.02 * ref and t > 0.01:
            t98 = t
        elif t98 is not None and abs(x - ref) >= 0.02 * ref:
            t98 = None
    return osc_max / ref * 100, t98, q_max


if __name__ == '__main__':
    wh = math.sqrt(4 * BETA * A * A / (V * M))
    zh = B / (4 * A) * math.sqrt(V / (BETA * M)) + CTL / A * math.sqrt(BETA * M / V)
    print(f'Frecuencia natural hidráulica wh = {wh:.1f} rad/s ({wh / (2 * math.pi):.1f} Hz), zeta_h ≈
{zh:.3f}')
    print(f'Ganancia de velocidad Kv = Kp·Kq/A; límite de estabilidad sin presión: Kv < 2·zeta_h·wh = {2 * zh
* wh:.1f} 1/s')
    for kp in (0.02, 0.08):
        print(f'   Kp = {kp}: Kv = {kp * KQ / A:.1f} 1/s')
    print('\n   controlador                    sobreimpulso   t al 2 %')
    casos = [('P, Kp = 0,02', 0.02, 0.0), ('P, Kp = 0,08', 0.08, 0.0),
             ('P + presión, Kp = 0,08', 0.08, 3e-11)]
```

eje_hidraulico_plc.st — bloque de función con rampa, realimentación de presión, zona muerta y supervisión.

difuso_y_optimizacion.py — PD difuso Takagi-Sugeno, auto-sintonía y optimización por enjambre de partículas.

Errores frecuentes

- Subir la ganancia de un eje hidráulico sin considerar ω_h.
- Dar escalones de referencia que excitan la resonancia: usar rampas.
- Entrenar una red con datos que no cubren todo el rango de operación.
- Optimizar un solo objetivo y descubrir después que otro quedó inaceptable.

Autoevaluación

1. ¿Qué limita la ganancia de un eje hidráulico?
2. ¿Qué aporta realimentar la presión de carga?
3. ¿Qué tipo de inferencia usa un promedio ponderado de consecuentes?
4. ¿Qué es un frente de Pareto?

Respuestas. 1. La resonancia hidráulica poco amortiguada. **2.** Amortiguamiento activo. **3.** Takagi-Sugeno. **4.** El conjunto de diseños que no pueden mejorar un objetivo sin empeorar otro.

Resumen y cierre

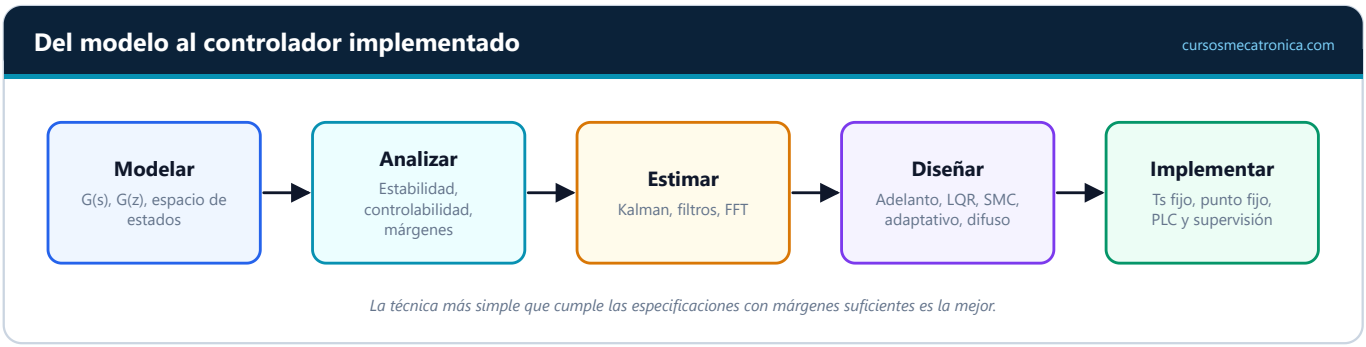
Cierre · En el video: **minuto 1:02:59** · Duración 00:57

Objetivo. Integrar las herramientas del curso en un proceso de diseño completo.

Ideas clave

Lo esencial

Modelar en continuo y discreto, verificar controlabilidad, observabilidad y estabilidad, diseñar con el lugar de las raíces o en frecuencia, estimar con Kalman, y recurrir al control óptimo, no lineal, adaptativo o inteligente cuando el problema lo pide.



Síntesis del curso.

Cómo seguir

Implementa un lazo completo en un microcontrolador con periodo fijo, mide sus márgenes y compáralos con el diseño. Para la parte de sensores y actuadores, revisa los cursos de Sensores y de Sistemas de Actuación.

Relaciones clave

Tema	Relación	Capítulo
Mapeo s → z	$z = e^{sT_s}$	01
Controlabilidad	$\text{rango}[B\ AB\ \ldots] = n$	02
Nyquist	$Z = N + P$	03
Kalman	$K = P^{-}H^{T}(HP^{-}H^{T} + R)^{-1}$	04
LQR	$u = -Kx$ (Riccati)	05
Hidráulica	$\omega_h = \sqrt{(4\beta_e A^2/(Vm))}$	06

Autoevaluación

1. ¿Qué revisarías antes de diseñar un observador?
2. ¿Qué técnica usarías para una carga que cambia mucho durante la operación?

Respuestas. 1. Que el sistema sea observable con los sensores elegidos. **2.** Control adaptativo (o robusto).