Curso de Sistemas y Control en Mecatrónica — Guía de Estudio Video 07

Módulo del Curso: Resumen Ejecutivo y Cierre Integrador del Curso de Sistemas y Control en Mecatrónica
Enfoque de Ingeniería: Integración Completa desde Transformada Z, Espacio de Estados y Kalman hasta Optimización H2, Modos Deslizantes, Electrohidráulica y Co-Diseño Pareto
Plataforma Oficial: `cursosmecatronica.com`

```
x̂[k+1] = Φx̂[k] + Γu[k] + K_k(y[k] - Cx̂[k]) | u[k] = -K_lqr x̂[k] + u_ff + u_smc | Pareto: min(Masa, Energía, Error)
```

1. Síntesis de Conceptos Clave y Reglas de Diseño

- Síntesis de modelado, señales continuas/discretas, transformadas de Laplace y Z, y análisis estructural en espacio de estados.
- Consolidación de diseño clásico y frecuencial (Root Locus, Bode, Nyquist, Loop Shaping) con observadores estocásticos de Kalman y filtrado DSP.
- Unificación de control óptimo H2/LQG, control no lineal y adaptativo (Lyapunov, Backstepping, SMC, MRAC) e inteligencia computacional (Difuso, RBF, ANFIS).
- Aplicación industrial en el control avanzado de ejes electrohidráulicos y la optimización concurrente planta-controlador sobre la frontera de Pareto.

2. Parámetros Nominales de Sistemas y Control (Video 07)

Parámetro de Ingeniería	Símbolo	Valor Nominal	Función en el Sistema de Control Mecatrónico
Periodo de Muestreo Digital	`Ts [ms]`	0.5 ms	Intervalo determinista de discretización ZOH y ejecución de Kalman/LQR
Frecuencia Natural y Amortiguamiento	`ωn / ζ`	150 rad/s / 0.72	Ubicación de polos dominantes en lazo cerrado
Márgenes de Fase y Ganancia	`PM / GM`	60° / 12.5 dB	Márgenes de estabilidad robusta en respuesta en frecuencia

Parámetro de Ingeniería	Símbolo	Valor Nominal	Función en el Sistema de Control Mecatrónico
Covarianzas del Filtro de Kalman	`Q / R`	Q=0.00001, R=0.0005	Matrices de incertidumbre de proceso y de ruido de sensores

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: Síntesis Parte 1: Modelado, Transformadas Z y Laplace, Espacio de Estados, Frecuencia y Kalman

Arquitectura Matemática desde la Dinámica Lineal y Discreta hasta Observadores Estocásticos y Procesamiento DSP

Turno 1 (Masculina): En este curso dominamos el rol del control y el modelado matemático en mecatrónica.

Turno 2 (Femenina): Unificamos señales continuas y discretas mediante transformadas de Laplace, Z y espacio de estados.

Turno 3 (Masculina): Analizamos controlabilidad, observabilidad, respuesta temporal, lugar de las raíces y respuesta en frecuencia.

Turno 4 (Femenina): E implementamos observadores dinámicos con Filtro de Kalman y procesamiento digital de señales en tiempo real.

Diapositiva 02: Síntesis Parte 2: Optimización H2, Control Adaptativo/No Lineal, Redes/Difusos y Eje Electrohidráulico

Control Robusto, Inteligencia Computacional, Actuación Electrohidráulica de Alta Precisión y Co-Diseño Óptimo Pareto

Turno 1 (Masculina): Diseñamos controladores óptimos vía norma H dos, reguladores cuadráticos y leyes basadas en Lyapunov.

Turno 2 (Femenina): Aplicamos linealización exacta, modos deslizantes, control adaptativo, lógica difusa y redes neuronales.

Turno 3 (Masculina): Integramos el control avanzado de un eje electrohidráulico y la optimización multiobjetivo de sistemas mecatrónicos.

Turno 4 (Femenina): Felicitaciones por completar el curso. Descarga todos los códigos y guías en cursosmecatronica punto com.

4. Firmware C++/MCU: Filtro de Kalman, Biquad DSP, LQR (H2) y Modos Deslizantes (control_sistemas_mecatronica_video_07.ino)

```
/*
 * ================================================================================
 * CURSO: SISTEMAS Y CONTROL EN MECATRÓNICA (GEMINI — cursosmecatronica.com)
 * VIDEO 07: Video 07 — Resumen Ejecutivo y Cierre del Curso de Sistemas y Control en Mecatrónica
 * Enfoque: Integración Completa desde Transformada Z, Espacio de Estados y Kalman hasta Optimización
H2, Modos Deslizantes, Electrohidráulica y Co-Diseño Pareto
 * Ecuación Clave: x̂[k+1] = Φx̂[k] + Γu[k] + K_k(y[k] - Cx̂[k])  |  u[k] = -K_lqr x̂[k] + u_ff + u_smc  |
Pareto: min(Masa, Energía, Error)
 * ================================================================================
 */
#include <Arduino.h>

// Parámetros de tiempo real y control en espacio de estados discreto
static const float TS_SEC = 0.0005f;
static const uint32_t TS_US = 500UL;

// Estado estimado por Filtro de Kalman [posición x1, velocidad x2, perturbación d]
static float x_hat[3] = {0.0f, 0.0f, 0.0f};
static float P_cov[2][2] = {{1.0f, 0.0f}, {0.0f, 1.0f}};
static const float Q_PROC = 0.00001f;
static const float R_MEAS = 0.0005f;

// Ganancias óptimas LQR (H2) + Superficie de Modos Deslizantes (SMC)
static const float K_LQR_POS = 67.50f;
static const float K_LQR_VEL = 9.36f;
static const float LAMBDA_SMC = 52.50f;
static const float K_SW_SMC = 2.80f;
static const float PHI_BOUND = 0.05f;

// Filtro digital IIR Biquad Notch (supresión de resonancia mecánica/hidráulica)
struct BiquadFilter {
  float b0, b1, b2, a1, a2;
  float w1, w2;
};
static BiquadFilter notchFilter = {0.94f, -1.78f, 0.94f, -1.78f, 0.88f, 0.0f, 0.0f};

static inline float stepBiquad(BiquadFilter &f, float x) {
  float w0 = x - f.a1 * f.w1 - f.a2 * f.w2;
  float y  = f.b0 * w0 + f.b1 * f.w1 + f.b2 * f.w2;
  f.w2 = f.w1;
  f.w1 = w0;
  return y;
}

static inline float satBoundary(float s, float phi) {
  if (s > phi) return 1.0f;
  if (s < -phi) return -1.0f;
  return s / phi;
}
```

```
void kalmanObserverStep(float u_prev, float y_meas) {
  // 1. Predicción temporal: x_hat^- = Phi * x_hat + Gamma * u
  const float b_fric = 1.25f;
  const float k_gain = 18.5f;
  float x1_pred = x_hat[0] + TS_SEC * x_hat[1];
  float x2_pred = x_hat[1] + TS_SEC * (-b_fric * x_hat[1] + k_gain * u_prev - x_hat[2]);

  // Propagación de covarianza P^- = Phi * P * Phi^T + Q
  P_cov[0][0] += TS_SEC * (2.0f * P_cov[0][1] + TS_SEC * P_cov[1][1]) + Q_PROC;
  P_cov[0][1] += TS_SEC * P_cov[1][1];
  P_cov[1][0]  = P_cov[0][1];
  P_cov[1][1] += Q_PROC * 10.0f;

  // 2. Actualización por medición (Innovación y Ganancia de Kalman K_k)
  float innov = y_meas - x1_pred;
  float S_inv = 1.0f / (P_cov[0][0] + R_MEAS);
  float K0 = P_cov[0][0] * S_inv;
  float K1 = P_cov[1][0] * S_inv;

  x_hat[0] = x1_pred + K0 * innov;
  x_hat[1] = x2_pred + K1 * innov;
  x_hat[2] -= 0.08f * innov; // Estimación adaptativa de carga externa

  float p00 = P_cov[0][0], p01 = P_cov[0][1];
  P_cov[0][0] -= K0 * p00;
  P_cov[0][1] -= K0 * p01;
  P_cov[1][0] -= K1 * p00;
  P_cov[1][1] -= K1 * p01;
}

void setup() {
  Serial.begin(115200);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
}

void loop() {
  static uint32_t lastTickUs = 0;
  static float u_cmd = 0.0f;
  uint32_t nowUs = micros();

  if ((uint32_t)(nowUs - lastTickUs) >= TS_US) {
    lastTickUs += TS_US;

    // Lectura de referencia y posición + filtrado DSP Biquad
    float ref_pos = (analogRead(A0) - 512) * 0.01f;
    float raw_pos = (analogRead(A1) - 512) * 0.01f;
    float y_clean = stepBiquad(notchFilter, raw_pos);

    // Estimación óptima de estados con Filtro de Kalman
    kalmanObserverStep(u_cmd, y_clean);

    // Ley de control híbrida: LQR (H2) + Modo Deslizante Suavizado (SMC)
    float e_pos = x_hat[0] - ref_pos;
    float e_vel = x_hat[1];
```

```
    float s_surf = e_vel + LAMBDA_SMC * e_pos;

    float u_lqr = -K_LQR_POS * e_pos - K_LQR_VEL * e_vel;
    float u_smc = -K_SW_SMC * satBoundary(s_surf, PHI_BOUND);
    u_cmd = constrain(u_lqr + u_smc + (x_hat[2] / 18.5f), -12.0f, 12.0f);

    int pwmOut = constrain((int)(fabsf(u_cmd) * (255.0f / 12.0f)), 0, 255);
    digitalWrite(10, (u_cmd >= 0.0f) ? HIGH : LOW);
    analogWrite(9, pwmOut);
  }
}
```

5. Simulación Python: Espacio de Estados, Riccati LQR/H2 y Observador de Kalman (simulacion_sistemas_y_control_video_07.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO: SISTEMAS Y CONTROL EN MECATRÓNICA (GEMINI — cursosmecatronica.com)
VIDEO 07: Video 07 — Resumen Ejecutivo y Cierre del Curso de Sistemas y Control en Mecatrónica
Simulación de Espacio de Estados, LQR/H2, Filtro de Kalman, Eje Electrohidráulico
y Optimización Multiobjetivo de Pareto.
================================================================================
"""
import math

TS = 0.0005
WN = 150
ZETA = 0.72
Q_KALMAN = 0.00001
R_KALMAN = 0.0005

def simulate_state_space_lqr_kalman(steps=400, ref=1.0):
    # Estado real [pos, vel] y estado estimado por Filtro de Kalman
    x = [0.0, 0.0]
    x_hat = [0.0, 0.0]
    p00, p01, p11 = 1.0, 0.0, 1.0

    # Ganancias LQR óptimas (síntesis H2)
    k1 = WN * 0.55
    k2 = 2.0 * ZETA * math.sqrt(WN)
    history = []

    for k in range(steps):
        y_meas = x[0]  # Medición del sensor
        # Actualización de Kalman
        innov = y_meas - x_hat[0]
        s_cov = p00 + R_KALMAN
        k_kal0 = p00 / s_cov
        k_kal1 = p01 / s_cov
        x_hat[0] += k_kal0 * innov
```

```
        x_hat[1] += k_kal1 * innov

        # Ley de control LQR + compensación no lineal de fricción
        e0 = x_hat[0] - ref
        e1 = x_hat[1]
        u = max(-24.0, min(24.0, -k1 * e0 - k2 * e1))

        # Integración de la planta mecatrónica / electrohidráulica
        acc = -2.0 * ZETA * WN * x[1] - (WN ** 2) * (x[0] - ref) * 0.05 + 15.0 * u
        x[0] += TS * x[1]
        x[1] += TS * acc

        # Predicción de Kalman
        x_hat[0] += TS * x_hat[1]
        x_hat[1] += TS * (-2.0 * ZETA * WN * x_hat[1] + 15.0 * u)
        p00 += TS * (2.0 * p01 + TS * p11) + Q_KALMAN
        p01 += TS * p11
        p11 += Q_KALMAN * 5.0

        if k % 50 == 0:
            history.append((round(k * TS, 4), round(x[0], 4), round(x_hat[0], 4), round(u, 3)))
    return history

if __name__ == "__main__":
    print("=== Simulación Sistemas y Control en Mecatrónica - Video 07 ===")
    for t_s, pos, pos_est, u_v in simulate_state_space_lqr_kalman():
        print(f"t={t_s:6.3f}s | Pos={pos:7.4f} | Kalman={pos_est:7.4f} | u={u_v:7.3f}V")
```

6. Programa PLC IEC 61131-3 (ST): Control Avanzado de Eje Electrohidráulico (plc_eje_electrohidraulico_control_video_07.st)

```
(* ===============================================================================
 * CURSO: SISTEMAS Y CONTROL EN MECATRÓNICA (GEMINI — cursosmecatronica.com)
 * VIDEO 07: Video 07 — Resumen Ejecutivo y Cierre del Curso de Sistemas y Control en Mecatrónica
 * Norma IEC 61131-3 (Structured Text): Observador de Estado Discreto,
 * Control en Cascada de Eje Electrohidráulico con Amortiguamiento Activo de Presión
 * =============================================================================== *)
PROGRAM PLC_Sistemas_Y_Control_V07
VAR_INPUT
    rRefPos_mm      : REAL := 100.0;   (* Referencia de posición [mm] *)
    rMeasPos_mm     : REAL := 0.0;     (* Sensor magnetoestrictivo / encoder [mm] *)
    rPresionA_bar   : REAL := 70.0;    (* Presión cámara A [bar] *)
    rPresionB_bar   : REAL := 70.0;    (* Presión cámara B [bar] *)
    bEnableAxis     : BOOL := TRUE;    (* Habilitación de seguridad *)
END_VAR
VAR_OUTPUT
    rValveCmd_V     : REAL := 0.0;     (* Comando a servoválvula [-10..+10 V] *)
    rPosEstimada_mm : REAL := 0.0;     (* Posición filtrada por observador *)
    rVelEstimada_mms: REAL := 0.0;     (* Velocidad estimada por Kalman [mm/s] *)
END_VAR
VAR
    rTs             : REAL := 0.0005;
```

```
    rErrorPos       : REAL := 0.0;
    rPresionCarga   : REAL := 0.0;
    rPresionAnt     : REAL := 0.0;
    rDerivPresion   : REAL := 0.0;
    rInnov          : REAL := 0.0;
    rKp_Cascada     : REAL := 4.50;
    rKv_Cascada     : REAL := 0.18;
    rKd_Presion     : REAL := 0.035;   (* Amortiguamiento hidromecánico activo *)
END_VAR

IF NOT bEnableAxis THEN
    rValveCmd_V := 0.0;
    RETURN;
END_IF;

(* 1. Observador de estado alfa-beta (Kalman estacionario) *)
rInnov := rMeasPos_mm - rPosEstimada_mm;
rPosEstimada_mm := rPosEstimada_mm + rTs * rVelEstimada_mms + 0.35 * rInnov;
rVelEstimada_mms := rVelEstimada_mms + (0.06 / rTs) * rInnov;

(* 2. Retroalimentación activa de presión de carga PL = PA - PB *)
rPresionCarga := rPresionA_bar - rPresionB_bar;
rDerivPresion := (rPresionCarga - rPresionAnt) / rTs;
rPresionAnt   := rPresionCarga;

(* 3. Ley de control avanzada con amortiguamiento activo e inversión *)
rErrorPos := rRefPos_mm - rPosEstimada_mm;
rValveCmd_V := (rKp_Cascada * rErrorPos) - (rKv_Cascada * rVelEstimada_mms) - (rKd_Presion *
rDerivPresion);

(* Saturación física de servoválvula proporcional [-10V, +10V] *)
IF rValveCmd_V > 10.0 THEN
    rValveCmd_V := 10.0;
ELSIF rValveCmd_V < -10.0 THEN
    rValveCmd_V := -10.0;
END_IF;

END_PROGRAM
```