CURSO COMPLETO: MODELADO DE SISTEMAS EN MECATRÓNICA

Video 01 — Parte 1A: Modelos de Sistemas Básicos: Matemáticos, Mecánicos y Eléctricos — Leyes Constitutivas, Sistemas Mecánicos Traslacionales y Rotacionales, Bloques RLC y Analogías Fuerza-Voltaje

Parte del Curso: Parte 1A — Modelos de Sistemas Básicos: Matemáticos, Mecánicos y Eléctricos
Planta / Sistema Modelado: Sistema Masa-Resorte-Amortiguador (1-GDL y 2-GDL) y Circuito Serie RLC

$$m\cdot(d^2x/dt^2) + c\cdot(dx/dt) + k\cdot x = F(t) \mid L\cdot C\cdot(d^2v_C/dt^2) + R\cdot C\cdot(dv_C/dt) + v_C = v_in(t)$$

1. Síntesis de Conceptos Clave y Fórmulas de Ingeniería

- Relaciones constitutivas traslacionales: Ley de Hooke F_k = k·(x1 - x2), amortiguador viscoso F_c = c·(v1 - v2) y segunda ley de Newton F_m = m·a.
- Resortes equivalentes: en paralelo suman rigideces (k_eq = k1 + k2); en serie suman flexibilidades (1/k_eq = 1/k1 + 1/k2).
- Bloques rotacionales: resorte torsional T = k_θ·θ, amortiguador rotacional T = c_θ·ω y momento polar de inercia T = J·α.
- Bloques eléctricos pasivos (v_R = R·i, v_L = L·di/dt, i_C = C·dv_C/dt), leyes de Kirchhoff (LCK/LVK) y analogía Fuerza-Voltaje de Maxwell.

2. Parámetros Nominales del Modelo Matemático (Video 01)

Parámetro Físico / Canónico	Símbolo	Valor Nominal	Significado en el Modelado Dinámico
Masa / Momento de Inercia Equivalente	`m_eq / J_eq`	2	Almacenador de energía cinética opositor a cambios de velocidad
Amortiguamiento / Resistencia Equivalente	`c_eq / R_eq`	6	Elemento disipador de energía que gobierna la tasa de atenuación
Rigidez / Elasticidad Equivalente	`k_eq / 1/C`	50	Almacenador de energía potencial restauradora
Frecuencia Natural No Amortiguada	`ω_n [rad/s]`	5 rad/s	Distancia radial de los polos al origen en el plano complejo s
Factor de Amortiguamiento Relativo	`ζ [-]`	0.3	Determina el régimen dinámico y el sobreimpulso máximo M_p
Ganancia Estática / Constante de Tiempo	`K / τ [s]`	K = 0.02, τ = 0.1 s	Sensibilidad estacionaria y rapidez del modo exponencial e^{-t/τ}

3. Desarrollo Académico Diapositiva por Diapositiva

Diapositiva 01: ¿Qué es el Modelado de Sistemas en Ingeniería Mecatrónica?

Abstracción Matemática de Plantas Físicas, Parámetros Concentrados y Ecuaciones Diferenciales

- **Turno 1 (Masculina):** Antes de programar un controlador o construir un prototipo costoso, todo ingeniero mecatrónico necesita predecir matemáticamente cómo reaccionará su máquina ante cualquier comando o perturbación externa.
- **Turno 2 (Femenina):** Un modelo de sistema es una representación matemática que vincula las señales de entrada o excitación con las variables de salida mediante leyes físicas expresadas como ecuaciones diferenciales ordinarias.
- **Turno 3 (Masculina):** Para que el modelo sea útil en control en tiempo real adoptamos la hipótesis de parámetros concentrados: agrupamos la masa, la elasticidad, la disipación y la inductancia en bloques funcionales discretos ideales.
- **Turno 4 (Femenina):** Esta simplificación nos permite pasar de ecuaciones en derivadas parciales intratables a sistemas dinámicos de orden finito que podemos simular en milisegundos dentro de un microcontrolador o en Python.

Diapositiva 02: Formulación de Modelos Matemáticos: Los Dos Pilares Universales

Relaciones Constitutivas de cada Elemento y Leyes de Conservación e Interconexión

- **Turno 1 (Masculina):** Sin importar si estamos estudiando un brazo robótico, un convertidor electrónico, una prensa hidráulica o un horno industrial, todo modelo matemático se construye siempre sobre dos pilares universales.
- **Turno 2 (Femenina):** El primer pilar son las ecuaciones constitutivas de cada bloque básico, que relacionan la variable de esfuerzo, como la fuerza o el voltaje, con la variable de flujo, como la velocidad o la corriente eléctrica.
- **Turno 3 (Masculina):** El segundo pilar son las leyes de conservación y equilibrio en los nodos de interconexión: la segunda ley de Newton y el principio de D'Alembert en mecánica, y las leyes de Kirchhoff en electricidad.
- **Turno 4 (Femenina):** En cada dominio físico existen únicamente tres tipos de bloques pasivos: elementos que almacenan energía potencial, elementos que almacenan energía cinética y elementos disipadores que transforman energía en calor.

Diapositiva 03: Bloques Mecánicos Traslacionales I: Resorte y Amortiguador Viscoso

Ley de Hooke F = k·x, Almacenamiento Potencial y Disipación Viscosa F = c·dx/dt

- **Turno 1 (Masculina):** Comencemos con los sistemas mecánicos de traslación rectilínea. El primer bloque funcional es el resorte elástico ideal, que almacena energía potencial elástica igual a un medio de la rigidez por la deformación al cuadrado.
- **Turno 2 (Femenina):** Según la ley de Hooke, la fuerza restauradora de un resorte lineal es directamente proporcional al desplazamiento relativo entre sus dos extremos: fuerza es igual a la constante de rigidez ka, en newtons por metro, por equis uno menos equis dos.
- **Turno 3 (Masculina):** El segundo bloque es el amortiguador viscoso o pistón hidráulico de fricción fluida, un elemento puramente disipador que no almacena energía sino que la convierte en calor proporcionalmente al cuadrado de la velocidad.
- **Turno 4 (Femenina):** Su ley constitutiva establece que la fuerza de oposición es igual al coeficiente de amortiguamiento viscoso ce, medido en newton segundo por metro, multiplicado por la velocidad relativa entre sus terminales, es decir, la derivada temporal de equis uno menos equis dos.

Diapositiva 04: Bloques Mecánicos Traslacionales II: Masa y Resortes en Serie/Paralelo

Segunda Ley de Newton $F = m \cdot d^2x/dt^2$ y Reducción de Rigideces Equivalentes

- **Turno 1 (Masculina):** El tercer bloque traslacional es la masa inercial eme, que almacena energía cinética igual a un medio de la masa por la velocidad al cuadrado, medida siempre respecto a un marco de referencia inercial fijo en tierra.
- **Turno 2 (Femenina):** Por la segunda ley de Newton con masa constante, la resultante de fuerzas sobre el cuerpo es igual a la masa multiplicada por la aceleración lineal, esto es, masa por la segunda derivada de la posición equis respecto al tiempo.
- **Turno 3 (Masculina):** Cuando varios resortes trabajan juntos, podemos reducirlos a un único resorte equivalente. Si están conectados en paralelo, ambos sufren exactamente el mismo alargamiento y sus fuerzas se suman, por lo que la rigidez equivalente es ka uno más ka dos.
- **Turno 4 (Femenina):** En cambio, cuando dos resortes se conectan en serie, ambos soportan la misma fuerza axial pero sus estiramientos se suman; por tanto, el inverso de la rigidez equivalente es igual a uno sobre ka uno más uno sobre ka dos.

Diapositiva 05: Diagrama de Cuerpo Libre y Sistema Masa-Resorte-Amortiguador

Deducción Paso a Paso de la Ecuación Diferencial Canónica de Segundo Orden

- **Turno 1 (Masculina):** Para obtener el modelo matemático de un mecanismo traslacional aplicamos el método del diagrama de cuerpo libre: aislamos la masa eme, definimos el sentido positivo de su coordenada equis y dibujamos todas las fuerzas que actúan sobre ella.
- **Turno 2 (Femenina):** Si desplazamos la masa hacia la derecha con velocidad positiva bajo una fuerza externa efe de te, el resorte tira hacia la izquierda con fuerza menos ka por equis, y el amortiguador frena hacia la izquierda con fuerza menos ce por equis punto.
- **Turno 3 (Masculina):** Aplicando la segunda ley de Newton, la suma de fuerzas es igual a masa por aceleración: efe de te menos ce por la derivada de equis menos ka por equis es igual a masa por la segunda derivada de equis respecto al tiempo.
- **Turno 4 (Femenina):** Reordenando todos los términos del sistema a la izquierda y la excitación a la derecha, obtenemos la ecuación diferencial fundamental de la mecatrónica: masa por equis dos puntos, más amortiguamiento por equis punto, más rigidez por equis, igual a la fuerza de entrada.

Diapositiva 06: Sistemas Mecánicos de Dos Grados de Libertad (2-GDL Acoplados)

Modelo de Suspensión de Cuarto de Vehículo y Ecuaciones Diferenciales Simultáneas

- **Turno 1 (Masculina):** En máquinas reales, como una suspensión automotriz activa o un pórtico industrial con carga flexible, suelen existir dos o más masas interconectadas, dando lugar a sistemas de múltiples grados de libertad.
- **Turno 2 (Femenina):** Necesitamos una coordenada independiente y una ecuación diferencial de segundo orden por cada masa que pueda moverse libremente: equis uno para la masa primaria eme uno, y equis dos para la masa secundaria eme dos.
- **Turno 3 (Masculina):** El resorte ka dos y el amortiguador ce dos situados entre ambas masas ejercen sobre eme uno una fuerza proporcional a la diferencia equis uno menos equis dos, y sobre eme dos una fuerza igual pero de signo opuesto, proporcional a equis dos menos equis uno.
- **Turno 4 (Femenina):** Estos términos cruzados acoplan matemáticamente ambas ecuaciones diferenciales, haciendo que cualquier movimiento de la primera masa transfiera energía hacia la segunda a través de los modos propios de vibración del sistema.

Diapositiva 07: Bloques Funcionales de Sistemas Mecánicos Rotacionales

Resorte Torsional T = k_θ·θ, Amortiguador Rotacional T = c_θ·ω y Momento de Inercia T = J·α

- **Turno 1 (Masculina):** En ejes de motores, articulaciones robóticas y transmisiones rotativas, la fuerza lineal se reemplaza por el par o torque te en newton metro, y el desplazamiento lineal por el ángulo de giro teta en radianes.
- **Turno 2 (Femenina):** El primer bloque rotacional es el resorte torsional o elasticidad de un eje de transmisión, cuyo par restaurador es igual a la rigidez torsional ka sub teta, en newton metro por radián, multiplicada por la torsión angular relativa teta uno menos teta dos.
- **Turno 3 (Masculina):** El segundo bloque es el amortiguador rotacional viscoso presente en chumaceras y películas de aceite lubricante, cuyo par de frenado es igual a ce sub teta por la velocidad angular relativa omega uno menos omega dos.
- **Turno 4 (Femenina):** Y el tercer bloque es el momento polar de inercia jota del rotor, en kilogramo por metro cuadrado: al aplicar la ecuación de Newton-Euler obtenemos que jota por la segunda derivada de teta, más ce sub teta por la primera derivada, más ka sub teta por teta, es igual al par aplicado.

Diapositiva 08: Bloques Funcionales de Sistemas Eléctricos: Resistor, Inductor y Capacitor

Relaciones Constitutivas v_R = R·i, v_L = L·di/dt e i_C = C·dv_C/dt

- **Turno 1 (Masculina):** Pasemos ahora al dominio eléctrico, donde la variable de esfuerzo es la diferencia de potencial o voltaje ve en voltios, y la variable de flujo es la intensidad de corriente eléctrica i en amperios.
- **Turno 2 (Femenina):** El elemento disipador es el resistor óhmico erre, gobernado por la ley de Ohm: la caída de tensión ve sub erre es igual a la resistencia en ohmios multiplicada por la corriente i, disipando potencia en forma de calor igual a i cuadrado por erre.
- **Turno 3 (Masculina):** El elemento que almacena energía en un campo magnético, como el devanado de un motor o solenoide, es el inductor ele en henrios: la tensión entre sus bornes es igual a la inductancia ele multiplicada por la derivada de la corriente respecto al tiempo.
- **Turno 4 (Femenina):** Y el elemento que almacena energía potencial en un campo electrostático es el capacitor ce en faradios, cuya corriente es igual a la capacitancia ce multiplicada por la derivada de la tensión del capacitor respecto al tiempo.

Diapositiva 09: Leyes de Kirchhoff y Modelado de Redes RC y RL de Primer Orden

Ley de Corrientes en Nodos (LCK) y Ley de Voltajes en Mallas (LVK)

- **Turno 1 (Masculina):** Para interconectar los bloques eléctricos utilizamos las dos leyes de Kirchhoff: la ley de corrientes establece que la suma algebraica de corrientes que entran a un nodo es igual a la suma de las que salen.
- **Turno 2 (Femenina):** Por su parte, la ley de voltajes de Kirchhoff establece que en cualquier lazo o malla cerrada la suma de las tensiones de las fuentes es igual a la suma de las caídas de tensión en los elementos pasivos.
- **Turno 3 (Masculina):** Apliquemos esto a un filtro pasabajos erre ce alimentado por una tensión de entrada ve sub in: la caída en la resistencia es erre por i, pero como está en serie con el capacitor, esa corriente es ce por la derivada del voltaje del capacitor.
- **Turno 4 (Femenina):** Sustituyendo obtenemos de inmediato el modelo diferencial de primer orden del circuito erre ce: erre por ce por la derivada del voltaje del capacitor respecto al tiempo, más el voltaje del capacitor, es igual a la tensión de entrada.

Diapositiva 10: Modelado del Circuito RLC y Analogía Electro-Mecánica Fuerza-Voltaje

Ecuación Diferencial de Segundo Orden $LC \cdot d^2v_C/dt^2 + RC \cdot dv_C/dt + v_C = v_{in}(t)$

- **Turno 1 (Masculina):** Si añadimos un inductor ele en serie con la resistencia erre y el capacitor ce, la ley de voltajes de Kirchhoff suma las tres caídas: ele por la derivada de la corriente, más erre por la corriente, más la tensión en el capacitor es igual al voltaje de entrada.
- **Turno 2 (Femenina):** Como la corriente es igual a ce por la derivada del voltaje en el capacitor, su derivada temporal es ce por la segunda derivada de dicho voltaje; al sustituir llegamos a la ecuación diferencial de segundo orden: ele por ce por ve dos puntos, más erre por ce por ve punto, más ve, igual al voltaje de entrada.
- **Turno 3 (Masculina):** Observemos la extraordinaria simetría matemática con el sistema mecánico masa-resorte-amortiguador: en la analogía fuerza-voltaje de Maxwell, la inductancia ele equivale a la masa eme, la resistencia erre equivale al amortiguamiento ce, y el inverso de la capacitancia equivale a la rigidez ka.
- **Turno 4 (Femenina):** Gracias a esta identidad estructural, todas las herramientas analíticas que desarrollemos para resolver un circuito eléctrico se aplican de forma exacta e inmediata a cualquier mecanismo traslacional o rotacional.

4. Firmware C++/Arduino: Integrador RK4 y Filtro de Tustin (simulacion_hil_y_adquisicion_video_01.ino)

```
/*
 * ========================================================================
 * CURSO COMPLETO: MODELADO DE SISTEMAS EN MECATRÓNICA
 * VIDEO 01: Video 01 — Parte 1A: Modelos de Sistemas Básicos: Matemáticos, Mecánicos y Eléctricos
 * Archivo: simulacion_hil_y_adquisicion_video_01.ino
 * ------------------------------------------------------------------------
 * Descripción Técnica:
 *   Firmware en tiempo real (Ts = 2 ms -> 500 Hz) para simulación Hardware-in-
 *   the-Loop (HIL), integración numérica Runge-Kutta de 4.º orden (RK4) de la
 *   planta mecatrónica "Sistema Masa-Resorte-Amortiguador (1-GDL y 2-GDL) y Circuito Serie RLC",
 *   filtro discreto por Transformada Bilineal de Tustin y adquisición por ADC
 *   para identificación experimental de parámetros (K, tau, wn, zeta).
 *
 * Ecuación Modelo:
 *   m·(d²x/dt²) + c·(dx/dt) + k·x = F(t)   |   L·C·(d²v_C/dt²) + R·C·(dv_C/dt) + v_C = v_in(t)
 * ========================================================================
 */

#include <Arduino.h>

// --- Definición de Pines de Hardware ---
const uint8_t PIN_ADC_PLANTA    = A0;  // Entrada analógica desde sensor real (0..5 V)
const uint8_t PIN_ADC_PERTURB   = A1;  // Perturbación externa / carga medida
const uint8_t PIN_PWM_ACTUADOR  = 9;   // Salida PWM hacia driver / puente H / válvula
const uint8_t PIN_SYNC_SCOPE    = 8;   // Pulso de sincronismo para osciloscopio

// --- Parámetros Físicos del Modelo (Video 01) ---
const float M_EQ       = 2.0000f;   // Masa / Inercia equivalente
const float C_EQ       = 6.0000f;   // Amortiguamiento / Resistencia equivalente
const float K_EQ       = 50.0000f;   // Rigidez / Elasticidad equivalente
const float WN_RAD_S   = 5.000f;  // Frecuencia natural no amortiguada [rad/s]
const float ZETA_REL   = 0.300f; // Factor de amortiguamiento relativo [-]
const float K_GAIN     = 0.020f;   // Ganancia estática de la planta
const float TAU_SEC    = 0.100f; // Constante de tiempo equivalente [s]

// --- Configuración del Reloj de Muestreo en Tiempo Real ---
const float TS_SEC               = 0.002f; // Periodo de integración Ts = 2.0 ms (500 Hz)
const unsigned long TS_MICROS    = 2000UL;
unsigned long prevMicros         = 0;
unsigned long stepCounter        = 0;

// --- Vector de Estado del Integrador RK4: x1 = y(t), x2 = dy(t)/dt ---
float x1_pos = 0.0f;
float x2_vel = 0.0f;
float u_cmd  = 0.0f;

// --- Filtro Digital de 1.er Orden Discretizado por Transformada de Tustin ---
// G_f(s) = K / (tau*s + 1)  -->  y[k] = b0*u[k] + b1*u[k-1] - a1*y[k-1]
float b0_tustin = 0.0f;
float b1_tustin = 0.0f;
float a1_tustin = 0.0f;
float u_prev1   = 0.0f;
float y_tustin  = 0.0f;

// --- Derivadas del Espacio de Estados Continuo: dx1/dt = x2, dx2/dt = f(x1,x2,u) ---
inline void evaluarDerivadasEstado(float x1, float x2, float u, float &dx1, float &dx2) {
  dx1 = x2;
  // Forma canónica de 2.º orden: x2' = K*wn^2*u - 2*zeta*wn*x2 - wn^2*x1
  dx2 = (K_GAIN * WN_RAD_S * WN_RAD_S * u)
      - (2.0f * ZETA_REL * WN_RAD_S * x2)
      - (WN_RAD_S * WN_RAD_S * x1);
}

// --- Paso de Integración Numérica Runge-Kutta de 4.º Orden (RK4) ---
void integrarPasoRK4(float u_in, float dt) {
  float k1_x1, k1_x2, k2_x1, k2_x2, k3_x1, k3_x2, k4_x1, k4_x2;
```

```
  evaluarDerivadasEstado(x1_pos, x2_vel, u_in, k1_x1, k1_x2);
  evaluarDerivadasEstado(x1_pos + 0.5f * dt * k1_x1, x2_vel + 0.5f * dt * k1_x2, u_in, k2_x1, k2_x2);
  evaluarDerivadasEstado(x1_pos + 0.5f * dt * k2_x1, x2_vel + 0.5f * dt * k2_x2, u_in, k3_x1, k3_x2);
  evaluarDerivadasEstado(x1_pos + dt * k3_x1,        x2_vel + dt * k3_x2,        u_in, k4_x1, k4_x2);

  x1_pos += (dt / 6.0f) * (k1_x1 + 2.0f * k2_x1 + 2.0f * k3_x1 + k4_x1);
  x2_vel += (dt / 6.0f) * (k1_x2 + 2.0f * k2_x2 + 2.0f * k3_x2 + k4_x2);
}

// --- Inicialización de Coeficientes Discretos por Regla Trapezoidal (Tustin) ---
void inicializarFiltroTustin() {
  float denom = 2.0f * TAU_SEC + TS_SEC;
  b0_tustin   = (K_GAIN * TS_SEC) / denom;
  b1_tustin   = b0_tustin;
  a1_tustin   = (TS_SEC - 2.0f * TAU_SEC) / denom;
}

void setup() {
  Serial.begin(115200);
  pinMode(PIN_PWM_ACTUADOR, OUTPUT);
  pinMode(PIN_SYNC_SCOPE, OUTPUT);
  inicializarFiltroTustin();
  prevMicros = micros();

  Serial.println(F("# ============================================================"));
  Serial.println(F("# CURSO MODELADO DE SISTEMAS EN MECATRONICA - VIDEO 01"));
  Serial.println(F("# Columnas CSV: t_s,u_cmd,y_rk4_2do,dy_rk4,y_tustin_1ro,adc_V"));
  Serial.println(F("# ============================================================"));
}

void loop() {
  unsigned long nowMicros = micros();
  if (nowMicros - prevMicros >= TS_MICROS) {
    prevMicros += TS_MICROS;
    float t_sec = stepCounter * TS_SEC;

    // 1. Generar entrada de prueba: Escalón unitario + barrido senoidal suave
    if (t_sec < 0.2f) {
      u_cmd = 0.0f;
    } else if (t_sec < 3.0f) {
      u_cmd = 1.0f; // Respuesta al escalón unitario u_s(t)
      digitalWrite(PIN_SYNC_SCOPE, HIGH);
    } else {
      // Excitación armónica para medir respuesta en frecuencia G(jw)
      u_cmd = 1.0f + 0.5f * sinf(WN_RAD_S * (t_sec - 3.0f));
      digitalWrite(PIN_SYNC_SCOPE, LOW);
    }

    // 2. Actualizar gemelo digital de 2.º orden por RK4
    integrarPasoRK4(u_cmd, TS_SEC);

    // 3. Actualizar modelo de 1.er orden por Transformada de Tustin
    y_tustin = b0_tustin * u_cmd + b1_tustin * u_prev1 - a1_tustin * y_tustin;
    u_prev1  = u_cmd;

    // 4. Leer planta física real en el ADC y comandar salida PWM
    float adcVolt = analogRead(PIN_ADC_PLANTA) * (5.0f / 1023.0f);
    int pwmVal    = constrain((int)(x1_pos * 127.0f + 128.0f), 0, 255);
    analogWrite(PIN_PWM_ACTUADOR, pwmVal);

    // 5. Transmitir telemetría cada 5 pasos (10 ms -> 100 Hz)
    if (stepCounter % 5 == 0) {
      Serial.print(t_sec, 3);      Serial.print(',');
      Serial.print(u_cmd, 4);      Serial.print(',');
      Serial.print(x1_pos, 4);     Serial.print(',');
      Serial.print(x2_vel, 4);     Serial.print(',');
      Serial.print(y_tustin, 4);   Serial.print(',');
      Serial.println(adcVolt, 3);
```

```
    }
    stepCounter++;
  }
}
```

5. Script Python: Polos, Respuesta Temporal y Bode (modelado_laplace_y_bode_video_01.py)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
================================================================================
CURSO COMPLETO: MODELADO DE SISTEMAS EN MECATRÓNICA
VIDEO 01: Video 01 — Parte 1A: Modelos de Sistemas Básicos: Matemáticos, Mecánicos y Eléctricos
Archivo: modelado_laplace_y_bode_video_01.py
--------------------------------------------------------------------------------
Descripción:
  Script integral en Python para simulación de ecuaciones diferenciales (RK4),
  análisis de sistemas de 1.er y 2.º orden, cálculo exacto de polos y ceros
  en el plano complejo s, especificaciones transitorias (tr, tp, Mp, ts),
  evaluación fasorial G(jw), trazado de Diagramas de Bode y cálculo de
  Márgenes de Ganancia (MG) y de Fase (MF).
================================================================================
"""

import math
from typing import Dict, List, Tuple

# Parámetros nominales del Video 01: Sistema Masa-Resorte-Amortiguador (1-GDL y 2-GDL) y Circuito Serie RLC
M_EQ     = 2
C_EQ     = 6
K_EQ     = 50
WN_RAD   = 5
ZETA     = 0.3
K_STATIC = 0.02
TAU_SEC  = 0.1


def metricas_primer_orden(K: float, tau: float, escalon_A: float = 1.0) -> Dict[str, float]:
    """Calcula las medidas de desempeño exactas de G(s) = K / (tau*s + 1)."""
    y_ss = K * escalon_A
    y_1tau = y_ss * (1.0 - math.exp(-1.0))
    t_d = tau * math.log(2.0)          # 50% del valor final
    t_r = tau * math.log(9.0)          # 10% al 90% (~2.197 * tau)
    t_s_5pct = 3.0 * tau               # Banda del +-5%
    t_s_2pct = 4.0 * tau               # Banda del +-2%
    omega_c = 1.0 / tau                # Frecuencia de esquina (-3 dB, -45 deg)
    return {
        "y_ss": round(y_ss, 4),
        "y_en_1tau_63pct": round(y_1tau, 4),
        "tiempo_retardo_td_s": round(t_d, 4),
        "tiempo_subida_tr_s": round(t_r, 4),
        "tiempo_establecimiento_5pct_s": round(t_s_5pct, 4),
        "tiempo_establecimiento_2pct_s": round(t_s_2pct, 4),
        "frecuencia_esquina_wc_rads": round(omega_c, 4),
    }


def metricas_segundo_orden(K: float, wn: float, zeta: float) -> Dict[str, object]:
    """Calcula polos en el plano s y métricas (wd, tr, tp, Mp, ts, wr, Mr, w_BW)."""
    sigma = zeta * wn
    if zeta < 1.0:
        wd = wn * math.sqrt(1.0 - zeta ** 2)
        polos = f"-{sigma:.3f} +- j*{wd:.3f}"
        regimen = "Subamortiguado (0 < zeta < 1)" if zeta > 0 else "No amortiguado"
        tr = (math.pi - math.acos(zeta)) / wd
        tp = math.pi / wd
        mp = math.exp(-math.pi * zeta / math.sqrt(1.0 - zeta ** 2))
    elif abs(zeta - 1.0) < 1e-6:
        wd = 0.0
        polos = f"-{sigma:.3f} (Raíz real doble)"
        regimen = "Críticamente amortiguado (zeta = 1)"
        tr = 3.35 / wn
        tp = float("inf")
        mp = 0.0
```

```
    else:
        wd = 0.0
        r1 = -sigma + wn * math.sqrt(zeta ** 2 - 1.0)
        r2 = -sigma - wn * math.sqrt(zeta ** 2 - 1.0)
        polos = f"s1 = {r1:.3f}, s2 = {r2:.3f}"
        regimen = "Sobreamortiguado (zeta > 1)"
        tr = 2.2 / abs(r1)
        tp = float("inf")
        mp = 0.0

    ts_2pct = 4.0 / sigma if sigma > 0 else float("inf")

    # Resonancia y Ancho de Banda en frecuencia
    if zeta < 1.0 / math.sqrt(2.0):
        wr = wn * math.sqrt(1.0 - 2.0 * (zeta ** 2))
        mr = 1.0 / (2.0 * zeta * math.sqrt(1.0 - zeta ** 2))
        mr_db = 20.0 * math.log10(mr)
    else:
        wr = 0.0
        mr = 1.0
        mr_db = 0.0

    w_bw = wn * math.sqrt((1.0 - 2.0 * zeta ** 2) + math.sqrt(4.0 * zeta ** 4 - 4.0 * zeta ** 2 + 2.0))

    return {
        "regimen": regimen,
        "polos_plano_s": polos,
        "atenuacion_sigma": round(sigma, 4),
        "frecuencia_amortiguada_wd_rads": round(wd, 4),
        "tiempo_subida_tr_s": round(tr, 4),
        "tiempo_pico_tp_s": round(tp, 4) if math.isfinite(tp) else None,
        "sobreimpulso_Mp_pct": round(mp * 100.0, 2),
        "tiempo_establecimiento_ts_2pct_s": round(ts_2pct, 4),
        "frecuencia_resonancia_wr_rads": round(wr, 4),
        "pico_resonancia_Mr_dB": round(mr_db, 2),
        "ancho_banda_wBW_rads": round(w_bw, 4),
    }


def evaluar_respuesta_frecuencia_bode(
    K: float, wn: float, zeta: float, frecuencias_rads: List[float]
) -> List[Dict[str, float]]:
    """Evalúa G(jw) = K / ((1 - u^2) + j*(2*zeta*u)) con u = w / wn."""
    tabla_bode = []
    for w in frecuencias_rads:
        u = w / wn
        parte_real = 1.0 - (u ** 2)
        parte_imag = 2.0 * zeta * u
        mag_abs = K / math.sqrt(parte_real ** 2 + parte_imag ** 2)
        mag_db = 20.0 * math.log10(max(mag_abs, 1e-12))
        fase_deg = -math.degrees(math.atan2(parte_imag, parte_real))
        tabla_bode.append({
            "w_rads": round(w, 3),
            "u_norm": round(u, 3),
            "mag_abs": round(mag_abs, 4),
            "mag_dB": round(mag_db, 2),
            "fase_deg": round(fase_deg, 2),
        })
    return tabla_bode


def identificar_segundo_orden_desde_escalon(
    u_escalon: float, y_estacionario: float, y_pico_max: float, t_pico_s: float
) -> Dict[str, float]:
    """Identifica experimentalmente (K, Mp, zeta, wn) a partir de la curva medida."""
    K_id = y_estacionario / u_escalon
    mp_id = max(0.0, (y_pico_max - y_estacionario) / y_estacionario)
    if mp_id > 0.0:
```

```
        ln_mp = abs(math.log(mp_id))
        zeta_id = ln_mp / math.sqrt(math.pi ** 2 + ln_mp ** 2)
    else:
        zeta_id = 1.0
    wn_id = math.pi / (t_pico_s * math.sqrt(max(1e-6, 1.0 - zeta_id ** 2)))
    return {
        "K_identificada": round(K_id, 4),
        "Mp_medido": round(mp_id, 4),
        "zeta_identificado": round(zeta_id, 4),
        "wn_identificada_rads": round(wn_id, 4),
    }


if __name__ == "__main__":
    print("=" * 76)
    print(" CURSO: MODELADO DE SISTEMAS EN MECATRÓNICA — VIDEO 01")
    print(" Planta:", "Sistema Masa-Resorte-Amortiguador (1-GDL y 2-GDL) y Circuito Serie RLC")
    print("=" * 76)

    m1 = metricas_primer_orden(K_STATIC, TAU_SEC)
    print("\n[1] Análisis de Sistema de Primer Orden (K, tau):", m1)

    m2 = metricas_segundo_orden(K_STATIC, WN_RAD, ZETA)
    print("\n[2] Análisis de Sistema de Segundo Orden (K, wn, zeta):", m2)

    freqs = [0.1 * WN_RAD, 0.5 * WN_RAD, WN_RAD, 2.0 * WN_RAD, 10.0 * WN_RAD]
    bode_pts = evaluar_respuesta_frecuencia_bode(K_STATIC, WN_RAD, ZETA, freqs)
    print("\n[3] Tabla de Respuesta en Frecuencia y Diagrama de Bode G(jw):")
    for row in bode_pts:
        print(f"    w = {row['w_rads']:7.2f} rad/s | |G| = {row['mag_abs']:6.4f} ({row['mag_dB']:6.2f} dB) | fase
= {row['fase_deg']:7.2f} deg")

    id_res = identificar_segundo_orden_desde_escalon(1.0, 1.0, 1.254, 0.343)
    print("\n[4] Ejemplo de Identificación Experimental desde Escalón:", id_res)
```

6. Script MATLAB/Octave: Funciones de Transferencia y Estabilidad (analisis_control_mecatronico_video_01.m)

```
% ========================================================================
% CURSO COMPLETO: MODELADO DE SISTEMAS EN MECATRÓNICA
% VIDEO 01: Video 01 – Parte 1A: Modelos de Sistemas Básicos: Matemáticos, Mecánicos y Eléctricos
% Archivo: analisis_control_mecatronico_video_01.m
% ------------------------------------------------------------------------
% Descripción:
%   Script para MATLAB / GNU Octave que construye las funciones de
%   transferencia en Laplace con tf(), calcula polos, ceros y factor de
%   amortiguamiento con damp(), simula la respuesta al escalón step() y al
%   impulso impulse(), cierra el lazo de realimentación con feedback(),
%   traza el Diagrama de Bode con márgenes de estabilidad margin() y el
%   Lugar Geométrico de las Raíces rlocus().
% ========================================================================

clear; clc; close all;
fprintf('========================================================================\n');
fprintf(' CURSO: MODELADO DE SISTEMAS EN MECATRONICA - VIDEO 01\n');
fprintf(' Planta: Sistema Masa-Resorte-Amortiguador (1-GDL y 2-GDL) y Circuito Serie RLC\n');
fprintf('========================================================================\n\n');

%% 1. Definicion de Parametros Fisicos y Canónicos
m_eq   = 2;
c_eq   = 6;
k_eq   = 50;
wn     = 5;
zeta   = 0.3;
K_gain = 0.02;
tau    = 0.1;

s = tf('s');

%% 2. Funcion de Transferencia de Primer Orden: G1(s) = K / (tau*s + 1)
G1 = K_gain / (tau * s + 1);
fprintf('--- Sistema de Primer Orden G1(s) ---\n');
disp(G1);

%% 3. Funcion de Transferencia de Segundo Orden: G2(s) = K*wn^2 / (s^2 + 2*zeta*wn*s + wn^2)
num_G2 = [1.0];
den_G2 = [2.0, 6.0, 50.0];
G2 = tf(num_G2, den_G2);
fprintf('--- Sistema Principal del Video 01: G2(s) ---\n');
disp(G2);

fprintf('--- Polos, Factor de Amortiguamiento (zeta) y Frecuencia Natural (wn) ---\n');
damp(G2);
polos_G2 = pole(G2);
ceros_G2 = zero(G2);

%% 4. Reduccion de Diagramas de Bloques: Serie, Paralelo y Lazo Cerrado T(s)
Kp_ctrl = 4.0;
Ki_ctrl = 8.0;
Gc = Kp_ctrl + Ki_ctrl / s;          % Controlador PI en paralelo
L_abierto = series(Gc, G2);          % Sistemas en serie L(s) = Gc(s)*G2(s)
H_sensor  = 1.0;                     % Realimentacion unitaria H(s) = 1
T_cerrado = feedback(L_abierto, H_sensor, -1); % T(s) = G(s) / (1 + G(s)H(s))

fprintf('--- Funcion de Transferencia en Lazo Cerrado T(s) ---\n');
disp(T_cerrado);

%% 5. Graficas en el Dominio del Tiempo (Escalon e Impulso) y Plano Complejo s
figure('Name', 'Video 01 - Respuestas Dinamicas y Mapa de Polos', 'Color', 'w', 'Position', [80 80 1100 720]);

subplot(2,2,1);
step(G1, 'b--', G2, 'r-', T_cerrado, 'g-.');
grid on;
title('Respuesta al Escalon Unitario u_s(t)');
legend('1.er Orden G_1(s)', 'Planta G_2(s)', 'Lazo Cerrado T(s)', 'Location', 'best');
```

```
subplot(2,2,2);
impulse(G2, 'r-', T_cerrado, 'g-.');
grid on;
title('Respuesta al Impulso Unitario \delta(t)');
legend('Planta G_2(s)', 'Lazo Cerrado T(s)', 'Location', 'best');

subplot(2,2,3);
pzmap(G2, 'r', T_cerrado, 'g');
grid on;
title('Ubicacion de Polos (x) y Ceros (o) en el Plano Complejo s');

subplot(2,2,4);
rlocus(L_abierto);
grid on;
title('Lugar Geometrico de las Raices (Root Locus)');

%% 6. Respuesta en Frecuencia: Diagrama de Bode y Margenes de Estabilidad (MG, MF)
figure('Name', 'Video 01 - Diagrama de Bode y Estabilidad Relativa', 'Color', 'w', 'Position', [140 100 980 680]);
margin(L_abierto);
grid on;

[Gm, Pm, Wcg, Wcp] = margin(L_abierto);
Gm_dB = 20 * log10(Gm);
fprintf('\n=== ESPECIFICACIONES DE ESTABILIDAD EN FRECUENCIA (VIDEO 01) ===\n');
fprintf('  Margen de Ganancia (MG)            : %.2f dB (en w_pc = %.3f rad/s)\n', Gm_dB, Wcp);
fprintf('  Margen de Fase (MF)                : %.2f deg (en w_gc = %.3f rad/s)\n', Pm, Wcg);
fprintf('  Amortiguamiento equivalente (MF/100): %.3f\n', Pm / 100.0);
```