

Propósito: Resolver cualquier ejercicio de examen de forma sistemática con los scripts de la carpeta **TOOLKIT_EXAMEN_ROBOTICA** sin perder tiempo en derivaciones manuales.

1. MAPA DE FUNCIONES DEL TOOLKIT

Archivo	¿Qué hace?	Sintaxis básica en 1 línea
dh_matrix.m	Matriz DH estándar $(^{i-1}A_i)$	<code>A = dh_matrix(theta, d, a, alpha);</code>
compactar_expresion.m	Saca factor común y agrupa trig.	<code>x = compactar_expresion(x_raw, [cos(q1), sin(q1)]);</code>
calcular_jacobiano_y_singularidades.m	Calcula J , $\det(J)$ y singularidades	<code>[J, det_J, sing] = calcular_jacobiano_y_singularidades(X_op, q);</code>
desacoplo_muneca_pci.m	PCI planar por desacoplo de muñeca	<code>[xw, yw, q] = desacoplo_muneca_pci(x, y, phi, L3, 'RPR', L2);</code>
calcular_inercia_com.m	CoM, vector local s_{iii} y tensor Steiner	<code>[m, r_c, s_ii, I] = calcular_inercia_com(segmentos, O_i);</code>
euler_lagrange_dinamica.m	Deduce $M(q)$, $C(q, \dot{q})$, $G(q)$	<code>[M, C, G] = euler_lagrange_dinamica(K, U, q, qd);</code>
sintonizar_pid.m	Ganancias PID ($\xi=1$), polo rápido	<code>[Kp, Kd, Ki, Kp_i, Kd_i, Ki_i] = sintonizar_pid(J, wn, 1.0, Kt);</code>
trayectoria_cubica.m	Spline cúbico $q(t)$, $\dot{q}(t)$, $\ddot{q}(t)$	<code>[q, qd, qdd] = trayectoria_cubica(t0, tf, q0, qf, 0, 0, t);</code>
trayectoria_trapezoidal.m	Perfil velocidad trapezoidal	<code>[q, qd, qdd, ta] = trayectoria_trapezoidal(t0, tf, q0, qf, vmax, amax, t);</code>
PLANTILLA_MAESTRA_EXAMEN.m	Script todo-en-uno listo para examen	Abrir, rellenar tabla DH y pulsar Ctrl+Enter

2. RECETARIO PASO A PASO SEGÚN PREGUNTA DE EXAMEN

Caso A: Te piden "Parámetros D-H, Matriz Resultante y PCD"

```
syms q1 q2 q3 L1 L2 L3 real; PI = sym(pi); A1 = dh_matrix(q1 + PI/2, 0, 0, PI/2); A2 = dh_matrix(0, L2 + q2, 0, -PI/2); A3 = dh_matrix(q3 - PI/2, 0, L3, 0); T03 = simplify(A1 * A2 * A3); % Extraer PCD y compactar: x = compactar_expresion(T03(1,4), [cos(q1), sin(q1)]); y = compactar_expresion(T03(2,4), [sin(q1), cos(q1)]); z = simplify(T03(3,4)); phi = simplify(atan2(T03(2,1), T03(1,1))); % Orientacion del extremo % Comprobacion obligatoria HOME: T_HOME = simplify(subs(T03, [q1, q2, q3], [0, 0, 0]));
```

Caso B: Te piden "Jacobiano y Singularidades"

```
% 1. Definir vector operacional y articular X_op = [x; y; phi]; % o [x; y; z] si es 3D q_vec = [q1; q2; q3]; % 2. Llamada directa [J, det_J, sing] = calcular_jacobiano_y_singularidades(X_op, q_vec); % Si det(J) = -(L2 + q2) -> Singularidad en q2 = -L2 (muñeca coincide con la base)
```

Caso C: Te piden "Problema Cinemático Inverso (PCI)"

```
% 1. Desacoplo de la muñeca (xw, yw): xw = x - L3*cos(phi); yw = y - L3*sin(phi); % 2. Resolver segun la cinematica del robot: % Si es R-P-R coaxial: q1 = atan2(yw, xw); q2 = sqrt(xw^2 + yw^2) - L2; q3 = phi - q1; % Si es R-R-R planar: % [xw, yw, q_sol] = desacoplo_muneca_pci(x, y, phi, L3, 'RRR', [L1, L2]);
```

Caso D: Te piden "Dinámica de Eslabón Compuesto / Newton-Euler (s_ii, CoM, Inercia)"

```
% Definir los tramos de la manivela/barra: seg(1).masa = rho * L2B/2; seg(1).com = [L2B/4; 0; 0]; seg(1).I_prop = ...; seg(2).masa = rho * L2A; seg(2).com = [L2B/2; L2A/2; 0]; seg(2).I_prop = ...; seg(3).masa = rho * L2B/2; seg(3).com = [3*L2B/4; L2A; 0]; seg(3).I_prop = ...; O_joint3 = [L2B; L2A; 0]; % Origen del sistema {S2} [m2, r_com, s22, I22] = calcular_inercia_com(seg, O_joint3); % s22 da el vector local exacto desde Joint 3 hasta el centro de masas C2.
```

Caso E: Te piden "Dinámica Euler-Lagrange / Ecuaciones de Movimiento"

```
syms q1 q2 qd1 qd2 real; % Introducir K(q, qd) y U(q): K = 0.5 * M11(q2) * qd1^2 + 0.5 * M22 * qd2^2; U = m2 * g * lc2 * sin(q2); [M, C, G] = euler_lagrange_dinamica(K, U, [q1; q2], [qd1; qd2]); % Despejar aceleraciones directas (para bloque Simulink ModeloDinamico_R.m): % qdd = M \ (tau - C*qd - B_m*qd - G);
```

Caso F: Te piden "Diseño / Sintonización de Controlador PID"

```
% Sintonizacion para planta J*s^2 con amortiguamiento critico (xi=1.0) y po = 4*wn: % [Kp, Kd, Ki, Kp_i, Kd_i, Ki_i] = sintonizar_pid(J_nom, wn, xi, Kt); [Kp, Kd, Ki, Kp_i, Kd_i, Ki_i] = sintonizar_pid(2.525, 3.0, 1.0, 10.0); % Si el motor entrega par directamente: usar [Kp, Kd, Ki] (N*m) % Si la señal de control es intensidad de corriente i (A): usar [Kp_i, Kd_i, Ki_i] (A) % Añadir feedforward de gravedad en eje 2: i2_total = i2_PID + (m2*g*lc2*cos(q2)) / Kt2
```

Caso G: Te piden "Generación de Trayectorias (Cúbica / Trapezoidal)"

```
% Trayectoria cubica entre q0 y qf en tiempo T (v0=vf=0): t = linspace(0, 2.0, 100); [q, qd, qdd, a_coef] = trayectoria_cubica(0, 2.0, 0, pi/2, 0, 0, t); % Trayectoria trapezoidal con vmax y amax: [q, qd, qdd, ta] = trayectoria_trapezoidal(0, 2.0, 0, pi/2, 1.2, 2.5, t);
```

3. LOS 4 COMANDOS CLAVE QUE SALVAN EL EXAMEN

Objetivo	Comando de MATLAB	Efecto
Evitar números gigantes con $\backslash (\pi \backslash)$	<code>PI = sym(pi);</code>	Mantiene expresiones exactas sin decimales infinitos.
Saber la orientación $\backslash (\phi \backslash)$ sin dudar	<code>phi = atan2(T(2,1), T(1,1));</code>	Extrae el ángulo exacto de rotación del extremo en el plano.
Factorizar y agrupar eslabones	<code>collect(x, [cos(q1), sin(q1)])</code>	Convierte $\backslash (L_2 \backslash \cos(q_1) + q_2 \backslash \cos(q_1) \backslash)$ en $\backslash ((L_2 + q_2) \backslash \cos(q_1) \backslash)$.
Calcular el Jacobiano al instante	<code>J = jacobian([x; y; phi], [q1; q2; q3])</code>	Calcula la matriz de derivadas parciales en 1 línea.