Formulario examen final Ingeniería del Control primer cuatrimestre:

1) Demonstrate the equations of the dynamic model using the Lagrange Method.

Encontrar posición y velocidad de x: (centro de masas), (coordenadas punto p).

Energías cinéticas: Lineal ($1/2\, m v^2 = 1/2\, m(\dot{x}^2 + \dot{y}^2)$), rotacional ($1/2\, m v^2 = 1/2(\Sigma I_n)$ parte angular ^{2}).

Energía potencial: mgh, donde h = altura del centro de masas.

Lagrangiano del sistema: $L = T - V$

Ecuaciones de Lagrange: $d/dt(\partial L/\partial \dot{q}_1) - \partial L/\partial q_1$, $d/dt(\partial L/\partial \dot{q}_2) - \partial L/\partial q_2$; ($= u_1$, $= u_2$), ($= 0$, $= u$).

2) Compute the state space representation of the dynamics of the manipulator in which $x = (x_1, x_2, x_3, x_4)^T = (q_1, q_2, \dot{q}_1, \dot{q}_2)^T$. Take the coordinates of point p as output variables.

```
close all; clear; clc;
syms parametros, qn, Xn, g, u, ...
eq1; eq2; (ecuaciones de Lagrange)
S = solve(eq1 == , eq2 == , dd_q1, dd_q2, 'Real', true);
xdot = [x3; x4; S.dd_q1; S.dd_q2];
xdot = subs(xdot, [q1, q2, dd_q1, dd_q2], [x1, x2, x3, x4])
```

$x(t) = [x_1\ x_2\ x_3\ x_4] = [q_1\ q_2\ \dot{q}_1\ \dot{q}_2]$

$\dot{x}(t) = [\dot{x}_1\ \dot{x}_2\ \dot{x}_3\ \dot{x}_4] = [\dot{q}_1\ \dot{q}_2\ \ddot{q}_1\ \ddot{q}_2]$

Modelo dinámico: $B(q)\ddot{q} + C(q, \dot{q}) + N(q) = u$

$B(q) = \begin{bmatrix} \text{Depende de } \ddot{q}_1 & \text{Depende de } \ddot{q}_2 \\ \text{Depende de } \ddot{q}_1 & \text{Depende de } \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} \text{Ec. de Lagrange 1} \\ \text{Ec. de Lagrange 2} \end{bmatrix}$

$C(q, \dot{q}) = \begin{bmatrix} \text{Ec. de Lagrange 1} \\ \text{Ec. de Lagrange 2} \end{bmatrix}$

$N(q) = \begin{bmatrix} \text{Ec. de Lagrange 1} \\ \text{Ec. de Lagrange 2} \end{bmatrix}$

3) Compute the relation between the configuration variables q and the position of the end effector $p = (p_1, p_2)^T$.

```
syms parametros, qn, p1, p2, g, u
px; py; (en función de q)
q = [px; py];
P_bid = [p1; p2];
r1 = subs(P_bid, p1, px);
P = subs(r1, p2, py)
```

Después de obtener los valores de $x_p(t)$ e $y_p(t)$, vemos como los movimientos realizados por el brazo son lineales, por lo que podemos encontrar la relación entre las variables de entrada q_1, q_2 y la posición del tip del brazo. Siendo p1 el movimiento en el eje X y p_2 el movimiento en el eje y, lo que genera la siguiente matriz:

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} \Rightarrow p = \begin{bmatrix} p_1(q) \\ p_2(q) \end{bmatrix}$$

4) Compute the relation between the joint velocities $\dot{q}$ and the velocity of the end effector $\dot{p}$.

```
syms parametros, qn(t)
P = [px; py]; (en función de q)
v1 = diff(P, t);
v2 = subs(v1, diff(q1(t)), d_q1);
d_P = subs(v2, diff(q2(t)), d_q2)
```

Para hallar esta relación, derivamos la matriz P que contiene las coordenadas del punto p respecto del tiempo, y sustituyendo después por dichos valores.

5) Compute the Jacobian matrix of the relation determined in 3).

```
syms parametros, Xn
p = [px; py] (en función de X)
J = jacobian([p(1), p(2)], [x1; x2])
```

Para este apartado se ha utilizado la herramienta syms de MATLAB para después utilizar la función jacobian() con dichos valores y así poder encontrar también la derivada parcial de p:

$p = \begin{bmatrix} p_1(x) \\ p_2(x) \end{bmatrix} \Rightarrow J(x) \Rightarrow \dot{p} = J(q)\dot{q}$

6) Compute the time derivative of the Jacobian matrix determined in 5).

```
syms parametros, Xn(t)
p = [px; py]; (en función de X)
J = [ ]; (en función de X)
dotJ = diff(J, t)
simp_dotJ = subs(dotJ, diff(x2(t), t), d_x2(t));
final_dotJ = subs(simp_dotJ, diff(x1(t), t), d_x1(t))
```

Después de encontrar la matriz jacobiana, procedemos a calcular su derivada. Para ello he utilizado nuevamente la herramienta syms ajustando las variables respecto al tiempo y utilizando también la función diff() para derivar J con respecto al tiempo. Por último, he utilizado la función subs() para sustituir los valores derivados por unos más legibles.

$J(x) \Rightarrow \dot{J}(x)$

7) MATLAB simulation

Init: Programa que inicia las medidas de los ejes de la gráfica.

f: Define la función del programa, donde se nombran las variables de estado $(q_1, q_2, \dot{q}_1, \dot{q}_2)$ y las de salida (x_p, y_p). Para luego derivarla y obtener la matriz de $\dot{x}(t)$, obteniendo así los valores de $\dot{x}_1, \dot{x}_2, \ddot{x}_1$ y $\ddot{x}_2$. Después, definimos también las matrices B, C y N del enunciado, y con ello podemos obtener los valores de $\ddot{x}_1$ y $\ddot{x}_2$ mediante la inversa de B.

Main: Es el programa principal, definimos los estados iniciales y los parámetros indicados en el enunciado. Después, se realiza el bucle for para observar el controlador, las variables de estado y todas las variables que cambian con respecto al tiempo, al igual que las variables que afectan al controlador, la posición deseada y las matrices A, B, N y u (también se han colocado en la posición deseada la constante de velocidad rotacional que se nos indicó para realizar la simulación de manera adecuada, " y una situación de if para que la posición cambie entre A y B"). Por último, se agrega el controlador a las variables de estado para un funcionamiento correcto y poder graficar los resultados obtenidos.

Draw: Programa que dibuja la simulación de la práctica.

8) TEORÍA

JACOBIAN TRANSPOSE METHOD:

Suponiendo que $pd = (p_1d, p_2d)^T$, una primera ley de regulación en el espacio operativo es $u = J^T(q)K_p(pd - p) - K_D\dot{q}$. Usando las variables del espacio de estados se pueden reescribir como $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = J^T(x_1, x_2)K_p(pd - p) - K_D\begin{pmatrix} x_3 \\ x_4 \end{pmatrix}$ donde K_p y K_D son matrices positivas simétricas y pueden ser elegidas como matrices diagonales. El primer término de una acción de control proporciona el error de posición $(pd - p)$ en el espacio operativo que puede interpretarse como una fuerza/momento aplicado al efector final del manipulador. La matriz jacobiana traspuesta mapea las fuerzas/momentos del efector final en fuerzas/momentos conjuntos $u = J^T F$. El segundo término es una acción amortiguadora proporcional a las velocidades articulares $\dot{q}$.

Una segunda ley de regulación en el espacio operacional es $u = J^T(q)[K_p(pd - p) - K_D\dot{p}]$ en el que la única diferencia con respecto al anterior es que la acción amortiguadora proporcional a la velocidad del efector final $\dot{p}$. Ya que $\dot{p} = J(q)\dot{q} = J(x_1, x_2)\begin{pmatrix} x_3 \\ x_4 \end{pmatrix}$ y usando las variables del espacio de estado, esta ley de regulación se puede reescribir como: $\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = J^T(x_1, x_2)K_p(pd - p) - J^T(x_1, x_2)K_D J(x_1, x_2)\begin{pmatrix} x_3 \\ x_4 \end{pmatrix}$

FEEDBACK LINEARIZATION METHOD:

Usando las variables de configuración, la derivada temporal de la matriz jacobiana puede reescribirse como $\dot{J}(x) \Rightarrow \dot{J}(q)$. Se toman las variables de salida $y_1 = q_1$, $y_2 = q_2$, tenemos $\ddot{y} = -B^{-1}(q)C(q, \dot{q})\dot{q} + B^{-1}(q)u$. Con esta relación podemos ver que $A(q) = B^{-1}(q)$, $b(q) = -B^{-1}(q)C(q, \dot{q})\dot{q}$. Dado que $A^{-1}(q) = B(q)$, la transformación de linealización y desacoplamiento es $u = A^{-1}(q)(v - b(q)) = B(q)v - B(q)(-B^{-1}(q)C(q, \dot{q})\dot{q}) =$
$= B(q)v + C(q, \dot{q})\dot{q}$, donde $v = (v_1, v_2)$ es la nueva entrada de control (añadir $+ N(q)$ en caso de que haya gravedad).

Suponiendo que $pd = (p_1d, p_2d)^T$, una ley de regulación en el espacio operacional es: $v = J^{-1}(q)(\ddot{p}d + K_D(\dot{p}d - \dot{p}) + K_p(pd - p) - \dot{J}(q, \dot{q})\dot{q})$, y siendo $\dot{p} = J(q)\dot{q}$ se puede reescribir como: $v = J^{-1}(q)(\ddot{p}d + K_D(\dot{p}d - J(q)\dot{q}) + K_p(pd - p) - \dot{J}(q, \dot{q})\dot{q})$. En términos de variables de estado, esta ley de regulación se puede expresar como: $v = J^{-1}(x_1, x_2)(\ddot{p}d + K_D(\dot{p}d - J(x_1, x_2)\begin{pmatrix} x_3 \\ x_4 \end{pmatrix}) + K_p(pd - p) - \dot{J}(x)\begin{pmatrix} x_3 \\ x_4 \end{pmatrix})$. Como de costumbre, se resuelve la tarea de seguimiento generando una posición deseada en movimiento de la punta que en este caso: $pd = c + r\begin{pmatrix} \cos t \\ \sin t \end{pmatrix}$, $\dot{p}d = r\begin{pmatrix} -\sin t \\ \cos t \end{pmatrix}$, $\ddot{p}d = r\begin{pmatrix} -\cos t \\ -\sin t \end{pmatrix}$.

9) PROGRAMAS MATLAB

Main:

```
Init;
% Estado inicial (x = [0;0;0;0]; u = [0;0]; v = [0;0];
% Velocidad angular (w = ;)
% Parámetros
% Radio y centro del círculo (tracking task)(r = ; c = [ , ]; ).
% Contadores (frame_counter = 0; dt = 0.01; t = 0; )
for t = 0 : dt : 50
    q1, q2, q̇1, q̇2, u1, u2 = x(1,2,3,4); u(1,2);
    % Coordenadas del punto p(q) (p = [;];).
    % Matriz jacobiana (q) (J = [,;,];).
    % Controlador PD (Kp = [,;,]; Kd = [,;,];).
    pd = c + r · [cos(w · t); sen(wt)];                      } JTM
    v = transpose(J)Kp(pd - p) - Kd[d_q1; d_q2];             }
    % Derivada de J (dotJ = [,;,];).
    % Posición deseada (func = rem(int32(wt), 2);).
    if func == 0 : Posición A (pd = [;];).
    else : Posición B (pd = [;];) end
    pd_d = [0;0]; pd_dd = [0;0];                              } FLM
    v = inv(J)(pd_dd + Kd(pd_d - J[x(3); x(4)]) + Kp(pd - p)
    - dotJ[x(3); x(4)])
    % Matrices (q) (B = [,;,]; C = [;]; N = [;];)
    u = Bv + C + N
    x = x + f(x,u)dt;
    x = x + dt(0'25f(x,u) + 0'75(f(x + dt 2/3 f(x,u), u)));
    frame_counter = frame_counter + 1;
    if frame_counter == 10
        plot(t, x(1), 'k--', t, x(2), 'r--.', t, u(1), 'g--', t, u(2), 'm--.')
        legend('q1', 'q2', 'xp', 'yp')
        draw(x, pd)
        pause(dt)
        frame_counter = 0;
    end
end
```

Init:

```
close all; clear; clc;
figure / hold / grid
xmin; xmax; ymin; ymax;
axis([xmin xmax ymin ymax]);
axis('square');
```

f:

```
function xdot = f(x,u)
    % Parámetros
    q1, q2, q̇1, q̇2, u1, u2 = x(1,2,3,4); u(1,2);
    % Matrices (q) (B = [,;,]; C = [;]; N = [;];)
    dd_q = inv(B)(-C + u - N);
    xdot = [d_q1; d_q2; dd_q(1); dd_q(2)];
end
```

Draw:

```
function draw(x, pd)
    % Parámetros
    clf() / hold on / axis square / axis([xmin xmax ymin ymax])
    q1 = x(1); q2 = x(2);
    arm1 = [0, long; 0, 0; 1, 1];
    arm2 = [0, long; 0, 0; 1, 1];
    tr = [1, 0, long_eje; 0, 1, 0; 0, 0, 1];
    rot = [cos(α), -sen(α), 0; sen(α), cos(α), 0; 0, 0, 1];
    result = tr · rot · arm2 (2 brazos)
    result = rot · arm1 (1 brazo)
    plot(pd(1), pd(2), '+r')
    plot(0, 0, 'o r')
    circle = nsidedpoly(1000, "Center", c, "Radius", r);      } TT
    plot(circle, "FaceColor", "w", "FaceAlpha", 0)             }
    plot(result(1,:), result(2,:), "black", "LineWidth", 1)
    plot(arm1(1,:), arm1(2,:), "green", "LineWidth", 1)      } PP
```

10) Compute the relation between the position of the end effector p and the configuration variables q. Compute the values of the configuration variables that correspond to $p = p_A$ and $p = p_B$.

```
syms parametros, pn(t)
p1 = ; p2 = ; (en función de q)
q1 = ; q2 = ; (en función de q)
q = [q1; q2]
```

Para hallar la siguiente relación, he pasado las ecuaciones referentes a las coordenadas igualadas a q_1 y q_2 en vez de p_1 y p_2.

11) Compute the relation between the velocity of the end effector $\dot{p}$ and the joint velocities $\dot{q}$. Suppose that the robot is ready to execute the tracking task, that is, the position of the end effector is $p = p_A$. Compute the initial joint velocities to execute the task.

```
syms parametros, pn(t)
q1 = ; q2 = ; (en función de q)
d_q1 = diff(q1, t);
d_q2 = diff(q2, t);
d_q1_1 = subs(d_q1, diff(p1, t), d_p1);
d_q1_Final = subs(d_q1_1, diff(p2, t), d_p2);
d_q2_Final = subs(d_q2, diff(p2, t), d_p2);
d_q = [d_q1_Final; d_q2_Final];
```

Para hallar la siguiente relación he partido de las ecuaciones de las coordenadas de p igualadas a q calculadas anteriormente. Después, derivamos estas ecuaciones en función del tiempo y las sustituimos por valores más legibles.