Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Puebla

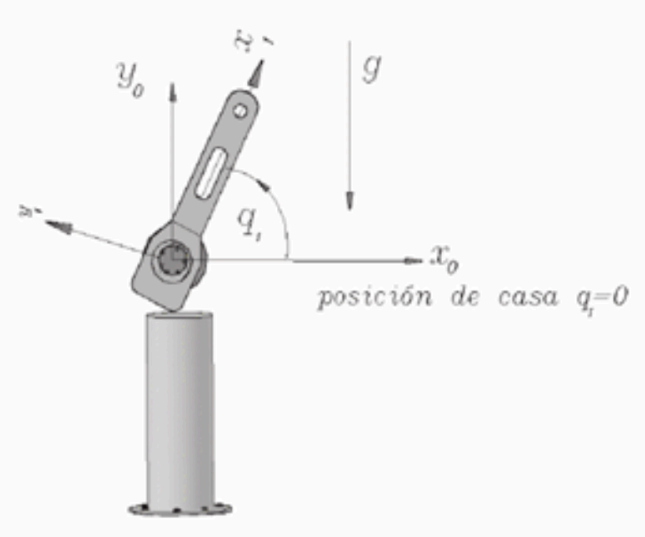
Fundamentación de Robótica (Gpo 101)

Examen final. Torques y fuerzas

Alumno
Victor Manuel Vázquez Morales A01736352

Fecha de entrega
Martes 12 de Marzo 2024

Ejercicio 1. Robot péndulo (1gdl)



Para nuestro primer ejercicio, obtendremos los torques y fuerzas (entre otras cosas más) para un robot péndulo de 1gdl que, como su nombre lo sugiere, se trata de un robot con una sola articulación.

Como bien sabemos, obtener el torque y las fuerzas que actúan sobre el robot requieren de una serie de pasos y cálculos previos que, en conjunto, nos conducirán a obtener lo que deseamos:

1. Declaración de variables simbólicas: De manera general, recordemos que es importante declarar las variables simbólicas que usaremos a lo largo de la codificación de este robot:

```
%Declaración de variables simbólicas
    %theta1     %Velocidad angular     #Aceleración angular
syms th1(t)      th1p(t)                th1pp(t)
syms m1 Ixx1 Iyy1 Izz1 %Masas y matrices de Inercia
syms  l1  lc1    %l=longitud de eslabon y lc=distancia al centro de masa del
eslabón
syms pi g a cero
```

De igual manera, es importante declarar el vector de coordenadas, velocidades y aceleraciones articulares, así como el vector de configuración del robot:

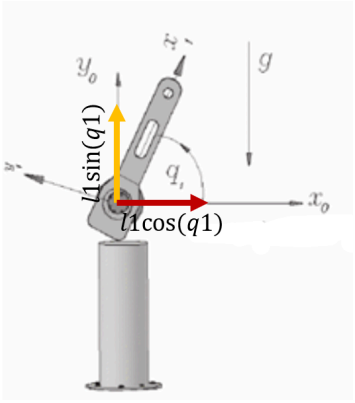
```
%Creamos el vector de coordenadas articulares
Q= [th1];
%Creamos el vector de velocidades articulares
Qp= [th1p];
%Creamos el vector de aceleraciones articulares
Qpp= [th1pp];
%Configuración del robot, 0 para junta rotacional, 1 para junta prismática
RP=[0];
%Número de grado de libertad del robot
GDL= size(RP,2);
GDL_str= num2str(GDL);
```

Estas variables serán de suma importancia más adelante, pues serán necesarias para realizar algunos de los cálculos.

2. Declaración de las matrices de rotación y traslación: Recordemos que obtener las matrices de transformación homogéneas es de suma importancia para predecir y conocer de manera general el comportamiento de nuestro robot, ya sea en una articulación específica o

sobre el efector final. Para obtener las matrices de transformación, declaramos las matrices de rotación y traslación para cada una de las articulaciones de nuestro robot (una, en este caso):

Observemos que, en este caso, tenemos una traslación partiendo del marco de referencia inercial al efector final, pues se traslada l1sin(q1) en el *eje y* y se traslada l1cos(q1) para el *eje x*. Por otro lado, sabemos que la articulación está rotando sobre el eje z:




```
%Articulación 1
%Posición de la articulación 1 respecto a 0
P(:,:,1)= [l1*cos(th1);l1*sin(th1);0];
%Matriz de rotación de la junta 1 respecto a 0
R(:,:,1)= [cos(th1) -sin(th1)  0;
           sin(th1)  cos(th1)  0;
           0         0         1];
```

3. Cálculo de velocidades lineales y angulares: Ahora que hemos declarado las matrices de traslación y rotación, procederemos a calcular las velocidades lineales y angulares. Para ello, antes que nada debemos obtener las matrices de transformación homogénea realizando multiplicación de matrices y la concatenación de rotación, traslación, vector de ceros y factor de escala:

```
%Creamos un vector de ceros
Vector_Zeros= zeros(1, 3);
%Inicializamos las matrices de transformación Homogénea locales
A(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
...
for i = 1:GDL
   i_str= num2str(i);
   A(:,:,i)=simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);
   %... Cálculo de matrices de transformación homogénea.
end
```

Posteriormente, haciendo uso de las matrices de transformación, se realiza el cálculo del jacobiano lineal y angular haciendo uso de las matrices de transformación homogénea:

```
%Calculamos el jacobiano lineal de forma analítica
Jv_a(:,GDL)=PO(:,:,GDL);
Jw_a(:,GDL)=PO(:,:,GDL);

for k= 1:GDL
   ... %Se aplican fórmulas para obtener el jacobiano.

end
%Obtenemos SubMatrices de Jacobianos
```

```
Jv_a= simplify (Jv_a); %Jacobiano lineal
Jw_a= simplify (Jw_a); %Jacobiano angular
```

Ahora que contamos con el jacobiano lineal y angular, podemos obtener la velocidad lineal y angular multiplicando el jacobiano correspondiente con el vector de coordenadas articulares y el vector de velocidades articulares (ambos declarados previamente):

```
disp('Velocidad lineal obtenida mediante el Jacobiano lineal');
V=simplify (Jv_a*Qp);
pretty(V);
%Velocidad angular
disp('Velocidad angular obtenida mediante el Jacobiano angular');
W=simplify (Jw_a*Qp);
pretty(W);
```

Resultados de velocidad angular y lineal:

```
Velocidad lineal obtenida mediante el Jacobiano lineal
/ -l1 sin(th1(t)) th1p(t) \
|                         |
|  l1 cos(th1(t)) th1p(t) |
|                         |
\           0             /

Velocidad angular obtenida mediante el Jacobiano angular
/    0    \
|         |
|    0    |
|         |
\ th1p(t) /
```

El resultado de velocidades es el esperado, pues indica una traslación tanto en x como en y con respecto al marco de referencia inicial y, además, hace sentido que no haya velocidad lineal en el eje z debido a que la articulación está rotando sobre este eje. Esta es la misma razón por la que únicamente hay velocidad angular en el eje z.

4. Cálculo de energía cinética: Hasta el momento, hemos calculado la velocidad lineal y angular sobre el efector final, que de hecho es la misma que las de la primera y única articulación o eslabón, pues se trata de un robot de 1gdl. Dicho esto, procederemos a calcular la energía cinética total para este robot, que vendría siendo la suma de la energía cinética lineal y angular:

$$Ec_{Total} = \frac{1}{2}I\omega^2 + \frac{1}{2}mv^2$$

Observemos que, tal y como se mencionó previamente, esta ecuación considera la velocidad angular y la velocidad lineal, así como otras variables como es la inercia *I* y la masa del eslabón *m*. Si lo notas, en el inicio de este documento, justo en la declaración de variables simbólicas, contamos con variables relacionadas con estos parámetros físicos:

1. *m1* : Masa del primer y único eslabón.
2. *Ixx1:* Momento de inercia en el eje x para el eslabón 1.
3. *Iyy1* : Momento de inercia en el eje y para el eslabón 1.
4. *Izz1 :* Momento de inercia en el eje y para el eslabón 1.

Para implementar esto en código, comenzaremos por declarar vectores de posición respecto al centro de masa para el eslabón. Estos vectores los consideraremos en cálculos más adelante:

```
%Distancia del origen del eslabón a su centro de masa
%Vectores de posición respecto al centro de masa
P01=subs(P(:,:,1)/2, l1, lc1);%La función subs sustituye l1 por lc1 en
                              %la expresión P(:,:,1)/2
```

De igual manera, declararemos matrices de inercia, las cuales incluyen los momentos de inercia para cada eje sobre el eslabón 1:

```
%Creamos matrices de inercia para cada eslabón
I1=[Ixx1 0 0;
   0 Iyy1 0;
   0 0 Izz1];
```

Posterior a esto, crearemos nuevas variables con el fin de extraer las velocidades lineales y angulares en cada uno de los ejes. Es importante mencionar que algunas de estas variables, especialmente *V* y *W*, nos servirán para realizar los cálculos de energía cinética:

```
%Extraemos las velocidades lineales en cada eje
V=V(t); Vx= V(1,1); Vy= V(2,1); Vz= V(3,1);
%Extraemos las velocidades angular en cada ángulo de Euler
W=W(t); W_pitch= W(1,1); W_roll= W(2,1); W_yaw= W(3,1);
```

Finalmente, contamos con todas las variables necesarias para realizar el cálculo de energía cinética. Observemos que, la siguiente sección de código, únicamente es la implementación de la fórmula previamente mencionada. Sin embargo, resulta interesante mencionar que la velocidad angular tiene un efecto sobre la velocidad lineal, por lo que consideramos este cálculo en la variable *V1_Total:*

```
%Calculamos la energía cinética para cada el eslabón %%%%%%
%Eslabón 1
V1_Total= V+cross(W,P01);
K1= (1/2*m1*(V1_Total))'*((V1_Total)) + (1/2*W)'*(I1*W);
%disp('Energía Cinética en el Eslabón 3');
K1= simplify (K1);
disp('Energía cinética total: ');
K_Total= simplify (K1);
pretty (K_Total);
```

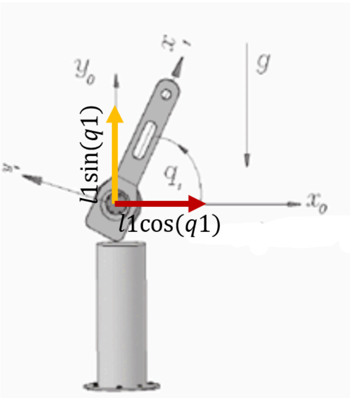
Resultados de energía cinética: :

```
Energía cinética total:
             2           2      ______              __           2             2
Izz1 |th1p(t)|    |th1p(t)|  cos(th1(t) - th1(t)) m1 (l1 |lc1|  + 2 lc1 |l1| ) (2 l1 + lc1)
--------------- + ------------------------------------------------------------------------
       2                                      8 l1 lc1
```

5. Cálculo de energía potencial y lagrangiano: Para realizar el cálculo de torques y fuerzas (que es nuestro objetivo final) necesitamos contar con el lagrangiano. Para llegar a este valor, antes debemos realizar el cálculo de energía potencial, el cuál viene dado por la siguiente ecuación:

$$E_p = mgh$$

Notemos, de igual manera, que previamente hemos declarado simbólicamente a *m1* y, como es de esperarse, *g* viene siendo el valor de la gravedad que, de igual manera, fue declarado simbólicamente al inicio del código. Lo interesante aquí está en la variable de altura *h1*. Para declarar esta variable, debemos analizar nuevamente nuestro robot:



Observemos que, en este caso, la altura está sobre el eje *y* y, de hecho, el mismo esquema del robot nos está indicando que la gravedad está actuando sobre este eje. Dicho esto, procedemos a hacer la codificación para la energía potencial:

```
%Obtenemos las alturas respecto a la gravedad
h1= P01(2); %Tomo la altura paralela al eje y
U1=m1*g*h1;
%Calculamos la energía potencial total
U_Total= U1;
```

Resultados de energía potencial y modelo de energía (Energía cinética + Energía potencial):

```
Energía cinética total:
              2             2     ______           __          2             2
Izz1 |th1p(t)|     |th1p(t)|  cos(th1(t) - th1(t)) m1 (l1 |lc1|  + 2 lc1 |l1| ) (2 l1 + lc1)
--------------- + -------------------------------------------------------------------------
       2                                          8 l1 lc1

Energia potencial total:
g lc1 m1 sin(th1(t))
--------------------
         2
```

Si bien el modelo de energía puede ser complicado de analizar debido a las variables que este considera, notemos que para el caso de la energía potencial total (que es de igual forma la del eslabón 1), está considera como su altura a lc1sin(th1(t)), lo cual es correcto ya que acorde a las identidades trigonométricas, esta es la componente para el eje y.

Ahora bien, realizar el cálculo del lagrangiano es bastante sencillo, pues únicamente basta con realizar la resta entre la energía cinética total menos la energía potencial total:

```
%Obtenemos el Lagrangiano
Lagrangiano= simplify (K_Total-U_Total);
```

Resultados de Lagrangiano:

```
Lagrangiano:
                2                                             2         ______          __             2            2
Izz1 |th1p(t)|     g lc1 m1 sin(th1(t))    |th1p(t)|  cos(th1(t) - th1(t)) m1 (l1 |lc1|  + 2 lc1 |l1| ) (2 l1 + lc1)
--------------- - -------------------- + ------------------------------------------------------------------------
       2                   2                                           8 l1 lc1
```

6. Cálculo de ecuaciones de movimiento (par gravitacional, torques , fuerzas, etc): Para obtener las ecuaciones de movimiento del robot (matriz de inercia, fuerzas centrípetas y coriolis y el par gravitacional) es necesario implementar una serie de ecuaciones dentro de Matlab:

1. **Cálculo de torque**. En este caso, contamos con una única articulación. Para realizar el cálculo del torque para esta articulación es necesario implementar la siguiente fórmula:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = 0$$

```
Qd=[th1p(t); th1pp(t)];
dQ1=[diff(diff(Lagrangiano,th1p), th1),...
diff(diff(Lagrangiano,th1p), th1p)];
t1= dQ1*Qd- diff(Lagrangiano, th1);
```

2. Ahora que contamos con el torque, procedemos a calcular la **matriz de inercia**, así como su derivada, pues nos serán útiles más adelante:

```
%Matriz de Inercia
%Extraemos coeficientes de aceleraciones
M=[diff(t1, th1pp)];
rank(M);
M=M(t);
%Fuerzas Centrípetas y de Coriolis
%Definimos Mp
Mp= diff(M, th1p);
```

3. Posteriormente, declaramos la **energía cinética** en su forma matricial y realizamos el cálculo de su **derivada**:

```
%Definimos la energía cinética en su forma matricial
k=1/2*transpose(Qp)*M*Qp;
%Definimos dk
dk=[diff(k, th1)];
```

4.Ahora contamos con todas las variables que son necesarias para obtener las **fuerzas centrípetas y de Coriolis**:

```
%Fuerzas centrípetas y de Coriolis
C= Mp*Qp-dk;
```

5. Finalmente, podemos calcular el par gravitacional del robot. Para hacer esto, utilizamos la variable de torque que calculamos previamente y anulamos a cada una de sus variables, es

decir, las velocidades y aceleraciones se sustituyen por 0. Esto lo realizamos debido a que en el par gravitacional, las velocidades y aceleraciones no actúan:

```
%Par Gravitacional
%se sustituyen las velocidades y aceleraciones por 0
r=cero;
a1=subs(t1, th1p, r);
%Torque gravitacional en el motor 1
G1=a1;
%Par gravitacional
G=G1;
```

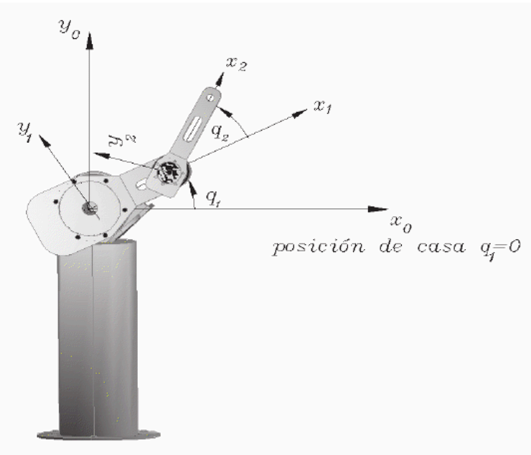
Resultados de ecuaciones de movimiento:

```
Matriz de inercia:
                                                                       2 __
                 2                                 #2 sign(th1p(t))   m1 #1 (2 l1 + lc1)
Izz1 sign(th1p(t))  + 2 Izz1 |th1p(t)| dirac(th1p(t)) + ------------------------------------
                                                                   4 l1 lc1

                                         __
     |th1p(t)| dirac(th1p(t)) #2 m1 #1 (2 l1 + lc1)
   + ---------------------------------------------
                      2 l1 lc1

where

                2            2
   #1 == l1 |lc1|  + 2 lc1 |l11|

                 ______
   #2 == cos(th1(t) - th1(t))
```

```
Fuerzas centripetas y coriolis:
        /
      2 |                                                                      __
th1p(t) | 2 Izz1 |th1p(t)| dirac'(th1p(t)) + 6 Izz1 dirac(th1p(t)) sign(th1p(t)) + |th1p(t)| #2 m1 dirac'(th1p(t)) #1 (2 l1 + lc1)
        \                                                                        -----------------------------------------------
                                                                                                 2 l1 lc1

                                  __                 \
     dirac(th1p(t)) #2 sign(th1p(t)) m1 #1 (2 l1 + lc1) 3 |
   + -------------------------------------------------- |
                        2 l1 lc1                        /

where

                2            2
   #1 == l1 |lc1|  + 2 lc1 |l11|

               ______
   #2 == cos(th1(t) - th1(t))
```

```
Par gravitacional
        /                                                                      __            2                                                    __                 \
        |               2                                                  #2 m1 sign(cero)  #1 (2 l1 + lc1)   #2 |cero| dirac(cero) m1 #1 (2 l1 + lc1) |
th1pp(t) | Izz1 sign(cero)  + 2 Izz1 |cero| dirac(cero) + ----------------------------------- + ----------------------------------------- |
        \                                                                          4 l1 lc1                                   2 l1 lc1                     /

     g lc1 m1 cos(th1(t))
   + --------------------
              2

where

                 2              2
   #1 == l1 |lc1|  + 2 lc1 |l1|

              ______
   #2 == cos(th1(t) - th1(t))
```

Ejercicio 2. Robot articular (2gdl)



Realizaremos el mismo análisis aplicado al robot del ejercicio 1, con el fin de obtener las ecuaciones de movimiento correspondientes a esta configuración:

1. Declaración de variables simbólicas:

```
    %theta           %Velocidad angular      #Aceleración angular
syms th1(t) th2(t)   th1p(t) th2p(t)         th1pp(t) th2pp(t)
syms m1 m2 Ixx1 Iyy1 Izz1 Ixx2 Iyy2 Izz2 %Masas y matrices de Inercia
syms l1 lc1 l2 lc2 %l=longitud de eslabon y lc=distancia al centro de masa
del eslabón
syms pi g a cero
```

De igual manera, declaramos el vector de coordenadas, velocidades y aceleraciones articulares, así como el vector de configuración del robot:

```
%Creamos el vector de coordenadas articulares
Q= [th1 th2];
%Creamos el vector de velocidades articulares
Qp= [th1p th2p];
%Creamos el vector de aceleraciones articulares
Qpp= [th1pp th2pp];
%Configuración del robot, 0 para junta rotacional, 1 para junta prismática
RP=[0 0];
```

Estas variables serán de suma importancia más adelante, pues serán necesarias para realizar algunos de los cálculos.

2. Declaración de las matrices de rotación y traslación: Observemos que este robot se conforma de dos juntas rotacionales y es de hecho una versión ligeramente más compleja que la del robot analizado anteriormente. Este robot cuenta tiene las siguientes rotaciones y traslaciones:

```
%Articulación 1
%Posición de la articulación 1 respecto a 0
P(:,:,1)= [l1*cos(th1);l1*sin(th1);0];
%Matriz de rotación de la junta 1 respecto a 0
R(:,:,1)= [cos(th1) -sin(th1)  0;
           sin(th1)  cos(th1)  0;
           0         0         1];
%Articulación 2
%Posición de la articulación 1 respecto a 0
P(:,:,2)= [l2*cos(th2);l2*sin(th2);0];
%Matriz de rotación de la junta 1 respecto a 0
R(:,:,2)= [cos(th2) -sin(th2)  0;
           sin(th2)  cos(th2)  0;
           0         0         1];
```

3. Cálculo de velocidades lineales y angulares: Ahora que hemos declarado las matrices de traslación y rotación, procederemos a calcular las velocidades lineales y angulares.

- Obtenemos las matrices de transformación:

```
%Creamos un vector de ceros
Vector_Zeros= zeros(1, 3);
%Inicializamos las matrices de transformación Homogénea locales
A(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
...
for i = 1:GDL
   i_str= num2str(i);
   A(:,:,i)=simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);
   %... Cálculo de matrices de transformación homogénea.
end
```

- Realizamos el cálculo del jacobiano lineal y angular:

```
%Calculamos el jacobiano lineal de forma analítica
Jv_a(:,GDL)=PO(:,:,GDL);
Jw_a(:,GDL)=PO(:,:,GDL);

for k= 1:GDL
   ... %Se aplican fórmulas para obtener el jacobiano.

end
%Obtenemos SubMatrices de Jacobianos
Jv_a= simplify (Jv_a); %Jacobiano lineal
Jw_a= simplify (Jw_a); %Jacobiano angular
```

Ahora que contamos con el jacobiano lineal y angular, podemos obtener la velocidad lineal y angular multiplicando el jacobiano correspondiente con el vector de velocidades articulares:

```
disp('Velocidad lineal obtenida mediante el Jacobiano lineal');
V=simplify (Jv_a*Qp);
pretty(V);
%Velocidad angular
disp('Velocidad angular obtenida mediante el Jacobiano angular');
W=simplify (Jw_a*Qp);
pretty(W);
```

Resultados de velocidad angular y lineal:

```
Velocidad lineal obtenida mediante el Jacobiano lineal
/ - th1p(t) (l1 sin(th1(t)) + l2 #1) - l2 #1 th2p(t) \
|                                                    |
|  th1p(t) (l1 cos(th1(t)) + l2 #2) + l2 #2 th2p(t)  |
|                                                    |
\                         0                          /

where

   #1 == sin(th1(t) + th2(t))

   #2 == cos(th1(t) + th2(t))


Velocidad angular obtenida mediante el Jacobiano angular
/         0         \
|                   |
|         0         |
|                   |
\ th1p(t) + th2p(t) /
```

Si bien analizar a detalle los vectores de velocidades puede resultar complejo, podemos observar que la velocidad lineal en z es nula, debido a que en este eje únicamente existen movimientos rotacionales.

4. Cálculo de energía cinética: Realizaremos el cálculo de la energía cinética para cada una de los eslabones, para ello, seguiremos el procedimiento del ejercicio 1:

- Declaramos los vectores de posición respecto al centro de masa

```
%Distancia del origen del eslabón a su centro de masa
%Vectores de posición respecto al centro de masa
P01=subs(P(:,:,1)/2, l1, lc1);%La función subs sustituye l1 por lc1 en
P12=subs(P(:,:,2)/2, l2, lc2);%la expresión P(:,:,1)/2
```

- Declaramos las matrices de inercia:

```
%Creamos matrices de inercia para cada eslabón
I1=[Ixx1 0 0;
   0 Iyy1 0;
   0 0 Izz1];

I2=[Ixx2 0 0;
   0 Iyy2 0;
   0 0 Izz2];
```

Procedemos a extraer las velocidades lineales y angulares en cada uno de los ejes. Es importante mencionar que algunas de estas variables, especialmente V y W, nos servirán para realizar los cálculos de energía cinética:

```
%Extraemos las velocidades lineales en cada eje
V=V(t); Vx= V(1,1); Vy= V(2,1); Vz= V(3,1);
%Extraemos las velocidades angular en cada ángulo de Euler
W=W(t); W_pitch= W(1,1); W_roll= W(2,1); W_yaw= W(3,1);
```

Ahora bien, es importante mencionar que contamos con la velocidad del eslabón 2, pues la misma que la del efector final. Sin embargo, debemos realizar el cálculo de la velocidad para el eslabón 1:

```
%Calculamos las velocidades para el eslabón1 %
Jv_a1(:,GDL-1)=PO(:,:,GDL-1);
Jw_a1(:,GDL-1)=PO(:,:,GDL-1);
for k= 1:GDL-1
   ...
end
%Obtenemos SubMatrices de Jacobianos
Jv_a1= simplify (Jv_a1);
Jw_a1= simplify (Jw_a1);

%Matriz de Jacobiano Completa
Jac1= [Jv_a1;
     Jw_a1];
Jacobiano1= simplify(Jac1);

%Velocidad lineal
Qp=Qp(t);
V1=simplify (Jv_a1*Qp(1));

%Velocidad angular
W1=simplify (Jw_a1*Qp(1));
```

Finalmente, podemos realizar el cálculo de la energía cinética total:

```
%Eslabón 1
V1_Total= V1+cross(W1,P01);
K1= (1/2*m1*(V1_Total))'*((V1_Total)) + (1/2*W1)'*(I1*W1);
%disp('Energía Cinética en el Eslabón 3');
K1= simplify (K1);
%Eslabón 2
V2_Total= V+cross(W,P12);
K2= (1/2*m2*(V2_Total))'*((V2_Total)) + (1/2*W)'*(I2*W);
%disp('Energía Cinética en el Eslabón 3');
K2= simplify (K2);
disp('Energía cinética total: ');
K_Total= simplify (K1+K2);
```

```
pretty (K_Total);
```

Resultados de energía cinética: :

```
Energía cinética total:
          /
Izz1 #5   | __ /                                                              lc2 sin(th2(t)) #1 \
------- + | m2 | th1p(t) (l1 sin(th1(t)) + l2 sin(#4)) + l2 sin(#4) th2p(t) + ------------------ |
   2      \    \                                                                       2          /

   /                                                                        \ \             /                   \
   | _______          __        ______ __    _______         __       ______  ___ | |             | _______   _______ |
   | th1p(t) (sin(#3) l2 + sin(th1(t)) l1) + th2p(t) sin(#3) l2 + sin(th2(t)) lc2 #2 | |             | th1p(t)   th2p(t) |
   | _______          __        ______ __    _______         __   ------------------ | |/2 + Izz2 #1 | ------- + ------- |
   \                                                                       2         / /             \    2         2    /

     /
     | __ /                                                            lc2 cos(th2(t)) #1 \
   + | m2 | th1p(t) (l1 cos(th1(t)) + l2 cos(#4)) + l2 cos(#4) th2p(t) + ------------------ |
     \    \                                                                     2          /

   /                                                                        \ \
   | _______          __        ______ __    _______         __       ______  ___ | |
   | th1p(t) (cos(#3) l2 + cos(th1(t)) l1) + th2p(t) cos(#3) l2 + cos(th2(t)) lc2 #2 | |
   | _______          __        ______ __    _______         __   ------------------ | |/2
   \                                                                       2         / /

                                          2          2
            ______          __
     #5 cos(th1(t) - th1(t)) m1 (l1 |lc1|  + 2 lc1 |l1| ) (2 l1 + lc1)
   + ------------------------------------------------------------------
                                 8 l1 lc1
```

5. Cálculo de energía potencial y lagrangiano: Ahora que hemos calculado la energía cinética, procedemos a realizar el cálculo de la energía potencial:

```
%Obtenemos las alturas respecto a la gravedad
h1= P01(2); %Tomo la altura paralela al eje y
h2= P12(2); %Tomo la altura paralela al eje y
U1=m1*g*h1;
U2=m2*g*h2;
%Calculamos la energía potencial total
U_Total= U1 + U2;
```

Resultados de energía potencial:

```
Energia potencial total:
g lc1 m1 sin(th1(t))   g lc2 m2 sin(th2(t))
-------------------- + --------------------
          2                      2
```

Ahora bien, realizar el cálculo del lagrangiano es bastante sencillo, pues únicamente basta con realizar la resta entre la energía cinética total menos la energía potencial total:

```
%Obtenemos el Lagrangiano
Lagrangiano= simplify (K_Total-U_Total);
```

Resultados de Lagrangiano:

```
Lagrangiano:
          /
Izz1 #5   | __ /                                                              lc2 sin(th2(t)) #1 \
------- + | m2 | th1p(t) (l1 sin(th1(t)) + l2 sin(#4)) + l2 sin(#4) th2p(t) + ------------------ |
   2      \    \                                                                       2         /

   /                                                                  ______  ___    \ \             / _______   _______ \
   | _______          __       ______  __    _______         __   sin(th2(t)) lc2 #2 | |             | th1p(t)   th2p(t) |
   | th1p(t) (sin(#3) l2 + sin(th1(t)) l1) + th2p(t) sin(#3) l2 + ------------------ | |/2 + Izz2 #1 | ------- + ------- |
   \                                                                       2         / /             \    2         2    /

     /
     | __ /                                                              lc2 cos(th2(t)) #1 \
   + | m2 | th1p(t) (l1 cos(th1(t)) + l2 cos(#4)) + l2 cos(#4) th2p(t) + ------------------ |
     \    \                                                                       2         /

   /                                                                  ______  ___    \ \
   | _______          __       ______  __    _______         __   cos(th2(t)) lc2 #2 | |     g lc1 m1 sin(th1(t))
   | th1p(t) (cos(#3) l2 + cos(th1(t)) l1) + th2p(t) cos(#3) l2 + ------------------ | |/2 - --------------------
   \                                                                       2         / /               2

                                                                2             2
                                   ______           __
     g lc2 m2 sin(th2(t))   #5 cos(th1(t) - th1(t)) m1 (l1 |lc1|  + 2 lc1 |l1| ) (2 l1 + lc1)
   - -------------------- + -----------------------------------------------------------------
               2                                         8 l1 lc1
```

6. Cálculo de ecuaciones de movimiento (par gravitacional, torques , fuerzas, etc): Para obtener las ecuaciones de movimiento del robot (matriz de inercia, fuerzas centrípetas y coriolis y el par gravitacional) es necesario implementar una serie de ecuaciones dentro de Matlab:

1. **Cálculo de torque**. En este caso, contamos con dos articulaciones. Para realizar el cálculo del torque para cada articulación realizamos lo siguiente:

```
%Derivadas respecto al tiempo
Qd=[th1p(t);  th2p(t); th1pp(t); th2pp(t)];

%Obtenemos las derivadas de la velocidad en la primera coordenada
%generalizada

dQ1=[diff(diff(Lagrangiano,th1p), th1), diff(diff(Lagrangiano, th1p),
th2),...
   diff(diff(Lagrangiano,th1p), th1p), diff(diff(Lagrangiano, th1p), th2p)]

%Definimos el torque 1
t1= dQ1*Qd- diff(Lagrangiano, th1);

%Obtenemos las derivadas de la velocidad en la segunda coordenada
%generalizada

dQ2=[diff(diff(Lagrangiano,th2p), th1), diff(diff(Lagrangiano, th2p),
th2),...
   diff(diff(Lagrangiano,th2p), th1p), diff(diff(Lagrangiano, th2p), th2p)]

%Definimos el torque 2
t2= dQ2*Qd- diff(Lagrangiano, th2);
```

2. Ahora que contamos con el torque, procedemos a calcular la **matriz de inercia**, así como su derivada, pues nos serán útiles más adelante:

```
%Matriz de Inercia
%Extraemos coeficientes de aceleraciones
M=[diff(t1, th1pp), diff(t1, th2pp);
  diff(t2, th1pp), diff(t2, th2pp)];
rank(M);
M=M(t);
%Fuerzas Centrípetas y de Coriolis
%Definimos Mp
%Derivamos parcialmente en tiempo respecto a todas las variables:
M11 = [diff(M(1,1), th1), diff(M(1,1),th2)]*Qp;
M12 = [diff(M(1,2), th1), diff(M(1,2),th2)]*Qp;
M21 = [diff(M(2,1), th1), diff(M(2,1),th2)]*Qp;
M22 = [diff(M(2,2), th1), diff(M(2,2),th2)]*Qp;
Mp= [M11, M12;
   M21, M22];
```

3. Posteriormente, declaramos la **energía cinética** en su forma matricial y realizamos el cálculo de su **derivada**:

```
%Definimos la energía cinética en su forma matricial
k=1/2*transpose(Qp)*M*Qp;
%Definimos dk
dk=[diff(k, th1); diff(k, th2)];
```

4.Ahora contamos con todas las variables que son necesarias para obtener las **fuerzas centrípetas y de Coriolis**:

```
%Fuerzas centrípetas y de Coriolis
C= Mp*Qp-dk;
```

5. Finalmente, podemos calcular el par gravitacional del robot. Para hacer esto, utilizamos la variable de torque que calculamos previamente y anulamos a cada una de sus variables, es decir, las velocidades y aceleraciones se sustituyen por 0. Esto lo realizamos debido a que en el par gravitacional, las velocidades y aceleraciones no actúan:

```
%Par Gravitacional
%se sustituyen las velocidades y aceleraciones por 0
r=cero;al de
a1=subs(t1, th1p, r);
a2=subs(a1, th2p, r);
a3=subs(a2, th1pp, r);
a4=subs(a3, th2pp, r);
%Torque gravitacional en el motor 1
G1=a4;
b1 = subs(t2, th1p, r);
b2 = subs(b1, th2p, r);
```

```
b3 = subs(b2, th1pp, r);
b4 = subs(b3, th2pp, r);
G2 = b4;
%Par gravitacional
G=[G1;G2];
```

Para este caso, el resultado arrojado por MATLAB es bastante largo y complejo, por lo que no se incluye como parte del reporte.