Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Puebla

Fundamentación de Robótica (Gpo 101)

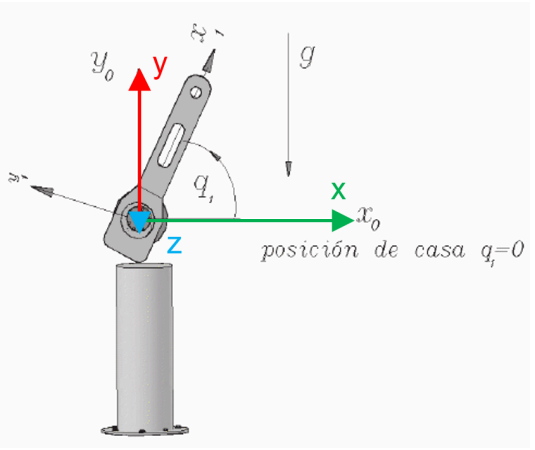
Examen Final 6 (Torque y fuerzas)

Alumno
José Diego Tomé Guardado A01733345

Fecha de entrega
Martes 12 de Marzo de 2024

Robot Péndulo 1GDL

El robot péndulo es un robot manipulador que basicamente consiste en que cuenta con un eslabón que rota alrededor del eje z fijo. Observemos gracias al diagrama y al marco de referencia inercial que tenemos los ejes definidos de una forma en la que el eje z es el que se encuentra para el movimiento rotacional.



A continuación primeramente vamos a definir las variables que usaremos a lo largo del programa, como lo son los ángulos, velocidades, aceletaciones de las articulaciones así como también la masa y matriz de inercia. Longitud del eslabon y la distancia al centro de masa que será muy importante para nuestro análisis de la altura y la energía potencial.

```
%----------------Robot Pendúlo 1GDL--------------------------
%Limpieza de pantalla
tic
%Declaración de variables simbólicas
syms th1(t) t  %Angulos de cada articulación
syms th1p(t)  %Velocidades de cada articulación
syms th1pp(t) %Aceleraciones de cada articulación
syms m1  Ixx1 Iyy1 Izz1     %Masas y matrices de Inercia
syms l1 lc1   %l=longitud de eslabones y lc=distancia al centro de masa de
cada eslabón
syms pi g a cero
```

Despues continuaremos creando el vector de coordenadas articulares del sistema, las cuales explican la posicion de las articulaciones del robot. También tenemos las velocidades articulares que puede representar una velocidad angular de th1p y por último las aceleraciones articulares para explicar la aceleracion angular.

```
%Creamos el vector de coordenadas articulares
 Q= [th1];
disp('Coordenadas generalizadas');
pretty (Q)
```

```
Coordenadas generalizadas
th1(t)
```

```
Creamos el vector de velocidades articulares
Qp= [th1p];
disp('Velocidades generalizadas');
pretty (Qp)
```

```
Velocidades generalizadas
th1p(t)
```

```
%Creamos el vector de aceleraciones articulares
Qpp= [th1pp]; %se utiliza este formato para simplificar la impresion de
resultados
disp('Aceleraciones generalizadas');
pretty (Qpp)
```

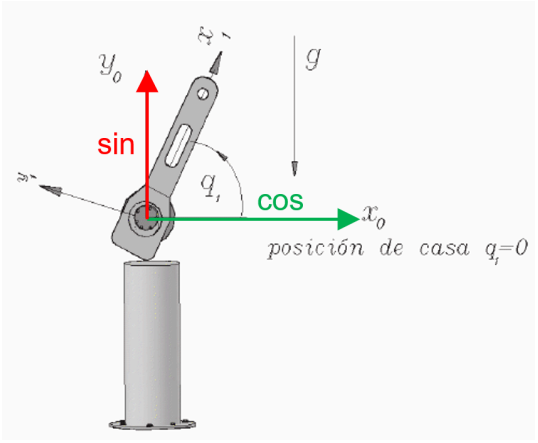
```
Aceleraciones generalizadas
th1pp(t)
```

Tenemos la configuración del robot donde gracias al dibujo del robot podemos saber que se trata de una junta rotacional y también que solo cuenta con una articulación por lo que pondremos solo un 0. De esta manera `GDL` se calcula como el número de columnas en nuestra matriz `RP` lo qué representa nuestro número de grados de libertad qué es de dos guardándolo en esta variable.

```
%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);
```

- **Análisis por articulación**

Haremos ahora un análisis de las articulaciones del robot, por lo que ya sabemos solo tenemos una entonces declararemos la posición conforme al siguiente diagrama, donde observamos que el coseno se efectua en el eje x y el seno se efectua en el eje y. Además se multiplica por l1 debido a que es la longitud del eslabon. Posteriormente tenemos la matriz de rotación definida en z debido a que nuestro marco de referencia inercial tenemos al eje z efectuando la rotación de nuestra articulación.




```
%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];

%Creamos un vector de ceros
Vector_Zeros= zeros(1, 3);
```

Vamos a inicializar las matrices de transformación homogénea local y global donde se le da la asignación con la letra `A` en la última posición `GDL` de esta matrices tridimensionales qué tiene de concatenación a la matriz de rotación, la matriz de posición y una fila de ceros seguida de un 1 para poder indicar toda la matriz de transformación homogénea completa

```
%Inicializamos las matrices de transformación Homogénea locales
A(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
%Inicializamos las matrices de transformación Homogénea globales
T(:,:,GDL)=simplify([R(:,:,GDL) P(:,:,GDL); Vector_Zeros 1]);
```

Se representarán los vectores de posición donde se copia la matriz de posición `P` y son vistos desde el marco de referencia inercial, también las matrices de rotación, pero copia la matriz de rotación `R` e igualmente vistas desde el marco de referencia inercial.

```
%Inicializamos las posiciones vistas desde el marco de referencia inercial
PO(:,:,GDL)= P(:,:,GDL);
%Inicializamos las matrices de rotación vistas desde el marco de referencia
inercial
RO(:,:,GDL)= R(:,:,GDL);
```

- **Velocidades Lineales y angulares**

Para comenzar con las matrices de transformación tendremos primeramente un ciclo qué itera sobre cada grado de libertad, calculando la matriz de transformación homogenea local concatenando la rotacion, posicion y un vector de zeros. Con el uso de try y catch empezamos a formar la matriz de transformación global pero como no hay una matriz de transformación previa con el catch se asigna la local a la global quedando la misma para las dos.

```
for i = 1:GDL
   i_str= num2str(i);
  %disp(strcat('Matriz de Transformación local A', i_str));
   A(:,:,i)=simplify([R(:,:,i) P(:,:,i); Vector_Zeros 1]);
  %pretty (A(:,:,i));
  %Globales
   try
      T(:,:,i)= T(:,:,i-1)*A(:,:,i);
   catch
      T(:,:,i)= A(:,:,i);
   end
%     disp(strcat('Matriz de Transformación global T', i_str));
   T(:,:,i)= simplify(T(:,:,i));
%     pretty(T(:,:,i))
   RO(:,:,i)= T(1:3,1:3,i);
   PO(:,:,i)= T(1:3,4,i);
   %pretty(RO(:,:,i));
   %pretty(PO(:,:,i));
end
```

Tenemos definido el cálculo del jacobiano lineal de forma analítica donde primeramente se asigna la posición final del extremo del robot al jacobiano angular y lineal. Continuando con el bucle en el que primero se comprueba si la articulación es de junta rotacional o prismatica, en este caso la juntas rotacional se calcula con el producto cruz entre el eje de rotación y la diferencia de la posición de la articulación y el extremo final. En este caso al no tener una articulación previa se asignan se hace el producto cruz con el eje rotacional z.

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

for k= 1:GDL
   if RP(k)==0
      %Para las juntas rotacionales
       try
           Jv_a(:,k)= cross(RO(:,3,k-1), PO(:,:,GDL)-PO(:,:,k-1));
           Jw_a(:,k)= RO(:,3,k-1);
       catch
           Jv_a(:,k)= cross([0,0,1], PO(:,:,GDL));%Matriz de rotación de 0
con respecto a 0 es la Matriz Identidad, la posición previa tambien será 0
           Jw_a(:,k)=[0,0,1];%Si no hay matriz de rotación previa se obtiene
la Matriz identidad
        end
```

De la siguiente manera obtenemos las submatrices de los jacobianos para posteriormente calcular la velocidad angular y lineal donde solamente es con una multiplicación con las velocidades articulares QP.

```
%Obtenemos SubMatrices de Jacobianos
Jv_a= simplify (Jv_a);
Jw_a= simplify (Jw_a);

%Obtenemos vectores de Velocidades Lineales y Angulares
disp('Velocidad lineal obtenida mediante el Jacobiano lineal');
V=simplify (Jv_a*Qp);
pretty(V)
disp('Velocidad angular obtenida mediante el Jacobiano angular');
W=simplify (Jw_a*Qp);
pretty(W)
```

```
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) /
```

Ya obtenidas las velocidades podemos hacer observar en la velocidad lineal que si existe un movimiento tanto en el eje x como en el eje y, pero el movimiento en z se encuentra en cero debido a que esto hace hincapié en la posición del robot la cual podemos ver que en x tendremos la derivada `l1cos(th1)` que será `-l1 sin(th1)` y multiplicándose por la velocidad de la articulación `th1p(t)`.

De la misma manera para y donde tenemos en su posición `l1 sin(th1)` que la derivada nos `l1 cos(th1)` y multiplicada por la velocidad de la articulación. Con la velocidad angular se observa que tendremos cero en los ejes x,y pero encontramos una velocidad angular en el eje z que es la velocidad de la articulación indicando también que el sistema completo tiene solo la rotación en este eje z.

- **Cálculo de energía cinética**

Se obtendrá la distancia del origen del eslabón hasta el centro de masa, calculando la posición del centro de masa del eslabon con respecto al sistema de referencia inercial. Incluyéndose la matriz de inercia la cual se encarga de poder describir como la masa se distribuye por los ejes de referencia, se representan con Ixx los momentos principales de inercia lo que es la resistencia a la rotación alrededor de los ejes principales.

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

De la misma manera tendremos la extracción de las velocidades lineales y angulares en cada eje y posteriormente calcularemos el jacobiano específico del eslabón 1 ya teniendo en cuenta esto podremos hacer el mismo cálculo de las velocidades angulares y lineales de este eslabon, que en este caso resultan las mismas qué sacamos al inicio debido a que este robot solo cuenta con 1GDL.

```
%Eslabon1
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 1');
K1= simplify (K1);
pretty (K1)
K_Total= simplify (K1);
pretty (K_Total)
%K_Total= simplify (K1+K2);
```

```
Energía Cinética en el Eslabón 1
              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


K_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
```

- **Energía potencial**

Considerando el sistema de referencia dentro del robot, puede observarse que la altura se encuentra posicionada sobre el eje *y0*. Por lo tanto gracias también a dibujar el eje con rojo podemos darnos cuenta que se trata de un robot de 1 solo grado de libertad, la altura dependerá únicamente de la rotación del péndulo considerando la variación de *y0.* Tomamos la coordenada paralela al eje z teniendo sentido porque la energía potencial gravitatoria depende de la altura vertical del centro de masa del objeto tomando de referencia un nivel cero qué en este caso es el origen de nuestro sistema de coordenadas.

La energía potencial en este caso se calculará como el producto de la masa con la aceleración de la gravedad y la altura qué tomamos antes. En este caso solo contamos con un eslabón por lo que la energía potencial total será la del eslabón 1.

```
%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= U1x
```

```
U_Total =

(g*lc1*m1*sin(th1(t)))/2
```

- **Lagrangiano**

Para el cálculo del lagrangiano tendremos una resta entre la energía cinética total y la energía potencial total de todo el sistema. Con ello tendremos una correcta formulacion de las ecuaciones de movimiento de todo el sistema debido a que se utiliza para poder derivar las ecuaciones de Lagrange.

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

- **Modelo de energía**

El modelo de energía definido como H es la suma de la energía cinética total con la energía potencial total del sistema, proporciona una representación de la energía del sistema y además se utiliza para poder analizar el comportamiento de nuestro sistema en distintas condiciones.

```
%Modelo de Energía
H= simplify (K_Total+U_Total)
 pretty (H)
```

```
Modelo de energía =
              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
```

- **Matriz de inercia**

```
%Definimos un vector columna de derivadas con respecto al tiempo

%En este vector agrego las velocidades y aceleraciones
 %Derivadas respecto al tiempo
Qd=[th1p(t); th1pp(t)];

%Obtenemos las derivadas de la velocidad en la primera coordenada
%generalizada
dQ1=[diff(diff(Lagrangiano,th1p), th1),...
    diff(diff(Lagrangiano,th1p), th1p)];%Derivamos con respecto a la primera
velocidad generalizada th1p para las 3 velocidades articulaciones

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


%Matriz de Inercia
%Extraemos coeficientes de aceleraciones
%derivamos torque uno con respecto a la aceleracion
M=[diff(t1, th1pp)];
rank (M);
M=M(t);

%Definimos Mp
disp('Matriz de inercia =');
Mp=[diff(M,th1p)]*Qp%Se deriva parcialmente en el tiempo respecto a todas las
variables
```

Primeramente se crea un vector columna que se llama Qd el cual contiene las derivadas temporales de las variables que van a definir el estado del sistema incluyendo las velocidades y aceleraciones variables del sistema, en dQ1 se calculan las segundas derivadas temporales

de la función lagrangiana que se toma con respecto a la primera coordenada generalizada th1 y la velocidad.

También tendremos el cálculo del torque para la primera articulación del sistema realizando una multiplicación de las derivadas de la función lagrangiana con respecto a variables de posición y velocidad por las velocidades y aceleraciones actuales y a su vez se resta la derivada de la función lagrangiana con respecto a la posición th1. Finalmente con la matriz de inercia se extrae el coeficiente de aceleración del torque 1 con respecto a la aceleración de th1pp.

```
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 |l1|

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

- **Modelo de las fuerzas centripetas y de Coriolis**

Tenemos ahora el cálculo de las derivadas de la energia cinetica con respecto a nuestras coordenadas generalizadas y se almacenan en dk. Con la ayuda de la matriz Mp y las derivadas de dk se calcula como la multiplicación de Mp y Qp y con la resta de las derivadas.

```
%Definimos la energía cinética en su forma matricial
k=1/2*transpose(Qp)*M*Qp;

%Definimos dk
dk=[diff(k, th1)];

%Fuerzas centrípetas y de Coriolis
disp('Fuerzas centrípetas y de Coriolis =');
C= Mp*Qp-dk;
pretty(C)
```

```
Fuerzas centrípetas y de Coriolis =
              /
            2 |
thlp(t)     | 2 Izz1 |thlp(t)| dirac'(thlp(t)) + 6 Izz1 dirac(thlp(t))
              \

                                                 __                                                                   __                         \
                         |thlp(t)| #2 m1 dirac'(thlp(t)) #1 (2 l1 + lc1)   dirac(thlp(t)) #2 sign(thlp(t)) m1 #1 (2 l1 + lc1) 3 |
   sign(thlp(t)) + ---------------------------------------------- + --------------------------------------------------- |
                                                      2 l1 lc1                                                          2 l1 lc1                  /

where

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

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

- **Modelo par gravitacional**

Se calcula el par gravitacional en cada articulación sustituyendo las velocidades y aceleraciones por cero, lo que permite obtener el efecto de la gravedad sobre el sistema. Estos términos representan el efecto de la gravedad en cada articulación del robot. Son importantes para calcular la carga gravitacional que el sistema debe superar durante su funcionamiento.

```
%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;

% Vector de par gravitacional
disp('Modelo de par gravitacional =');
G=[G1];
pretty(G)
```

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

                                          __               \
     #2  |cero| dirac(cero) m1 #1  (2 l1 + lc1) |   g lc1 m1 cos(th1(t))
   + ---------------------------------------- | + --------------------
                      2 l1 lc1                 /            2

where

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

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